

Decomposition Methods in Optimization and Applications in Practical Problems

Lecture Notes
Politecnico di Milano
March 2007
UNDER CONSTRUCTION

Horst W. Hamacher¹

March 28, 2007

¹Fachbereich Mathematik, hamacher@mathematik.uni-kl.de. The research has been partially supported by Deutsche Forschungsgemeinschaft (DFG) grant HA 1737/7 “Algorithmik großer und komplexer Netzwerke”, the Rheinland-Pfalz cluster of excellence “Dependable adaptive systems and mathematical modeling” and the Politecnico di Milano.

Contents

1	Consecutive Ones Matrices as Tools for Modeling Practical Problems	5
1.1	C1 Matrices and Matrix Decompositions	5
1.2	Applications of C1 Decomposition	8
2	Solving Linear and Integer Programs with C1 Constraint Matrices	9
2.1	Polynomial Algorithm for KC1 with $K = 1$	10
2.1.1	Solving 1C1 Column Problems by a Network Flow Approach	10
2.1.2	Solving 1C1 (Row) Problems by a Network Flow Approach	14
2.2	Linear Programs and Semi-Simultaneous Flows	17
2.2.1	Definition of se-sim flows	17
2.2.2	Feasible se-sim flows	19
2.2.3	Bounds on se-sim flows	20
2.2.4	Optimal se-sim flows	21
2.3	KC1 Projects	23
3	Constrained Decomposition and Complexity of Decomposition Cardinality Problem	25
3.1	Two Network Flow Formulations for the Constrained Decomposition Time Problem	25
3.2	Decomposition Cardinality is NP-hard	31
3.3	Multicriteria Optimization and the Box Method	35
3.3.1	Finding Representative Sets by Box Method	38
4	Stop Location Problem	45
4.1	Covering and Access Problems	45
4.2	Covering Stop Location	48
4.2.1	Planar CovStopLoc	50
4.2.2	Network CovStopLoc	51
4.3	Accessibility	52
4.3.1	Planar AccessStopLoc	52
4.3.2	Accessibility Problem in Network Environment	58

Chapter 1

Consecutive Ones Matrices as Tools for Modeling Practical Problems

1.1 C1 Matrices and Matrix Decompositions

Denotations used in the following

$$\begin{aligned}\mathcal{M} &:= \{1, \dots, M\} \quad \text{with } \mathcal{M} \in \mathbb{N} \\ \mathcal{N} &:= \{1, \dots, N\} \quad \text{with } \mathcal{N} \in \mathbb{N}\end{aligned}$$

$Mat_{M,N}(R)$... set of M matrices with entries in R (by default $R = \mathbb{Z}$).
 $\mathbb{B} := \{0, 1\}$.

Definition 1.1. $Y \in Mat_{M,N}(\mathbb{B})$ is a consecutive ones (C1) matrix in rows if the ones occur in each row in a single block, i.e., if for all $m \in M$

$$1 \leq i < j \leq N, y_{mi} = 1 = y_{mj} \Rightarrow y_{mk} = 1 \quad \forall k \in N : i \leq k \leq j. \quad (1.1)$$

(Note that the cases $y_{mn} = 0 \quad \forall n \in N$, and $y_{mi} = 1$ for exactly one $i \in N$ are allowed.) Y is a C1 matrix in columns iff for all $n \in N$

$$1 \leq i < j \leq M, y_{in} = y_{jn} = 1 \Rightarrow y_{kn} = 1 \quad \forall k \in M : 1 \leq k \leq j \quad (1.2)$$

By default we use the denotation "C1 matrix" for "C1 matrix in rows". In contrast to our denotation, "C1 matrix" is in the literature often a binary matrix which can be transformed by column permutations into one satisfying 1.1. We call them here *weak C1 matrices*.

Definition 1.2. The interval denotation of a C1 matrix Y is

$$Y = \begin{pmatrix} [\ell_1, r_1) \\ \vdots \\ [\ell_M, r_M) \end{pmatrix} \quad (1.3)$$

where $\ell_m \in N$, $r_m \in N_r := N \cup \{n+1\}$, $\ell_m \leq r_m$,

$$[\ell_m, r_m) := \{n \in N : \ell_m \leq n < r_m\} \quad (1.4)$$

and

$$y_{mn} = 1 \Leftrightarrow n \in [\ell_m, r_m) \quad (1.5)$$

If $\ell_m = r_m$ then row m is the zero row. In this case, the interval denotation is not unique.

Corresponding a C1 matrix Y in column has??? the interval denotation

$$Y = \left((\ell_1, r_1), \dots, (\ell_N, r_N) \right) \quad (1.6)$$

Example 1.3.

a)

$$y = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} [2, 5) \\ [1, 3) \\ [4, 5) \\ [1, 4) \end{pmatrix}$$

b)

$$y = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} [2, 5) \\ [1, 1) \\ [4, 2) \end{pmatrix}$$

c)

$$y = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} = ([1, 3), [4, 5), [2, 5), [1, 2))$$

Definition 1.4. Given matrix $A \in \text{Mat}_{M,N}(\mathbb{Z}_+)$

$$A = \sum_{t \in T} \alpha_t y_t \quad (1.7)$$

is a C1 decomposition if

- T is an index set of C1 matrices
- $\alpha_t \geq 0 \forall t \in T$
- y_t is C1 matrix $\forall t \in T$.

By definition, every C1 decomposition is a decomposition into C1 row matrices. If we decompose into C1 column matrices, we say so explicitly.

Example 1.5.

$$\begin{aligned} A &= \begin{pmatrix} 2 & 5 & 3 \\ 3 & 5 & 2 \end{pmatrix} \\ &= 2 \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} + 1 \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} + 2 \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} \\ &= 2 \begin{pmatrix} [1, 4] \\ [1, 4] \end{pmatrix} + 1 \cdot \begin{pmatrix} [2, 4] \\ [1, 3] \end{pmatrix} + 2 \cdot \begin{pmatrix} [2, 3] \\ [2, 3] \end{pmatrix} \end{aligned}$$

Definition 1.6. *Objective functions in C1 decompositions:*
Decomposition Time

$$DT(\alpha) = \sum_{t \in T} \alpha_t \quad (1.8)$$

Decomposition Cardinality

$$DC(\alpha) = \#\{t : \alpha_t > 0\} \quad (1.9)$$

Decomposition Sequence

$$DS(\alpha) = \sum_{t \in T} \alpha_t + \sum_{i=1}^{DC(\alpha)-1} C(i, \sigma(i+1)) \quad (1.10)$$

where $\{t_1, \dots, t_{DC(\alpha)}\} = \{t : \alpha_t > 0\}$, σ is a permutation of $\{1, \dots, DC(\alpha)\}$ matrices and c_{ij} is the cost of moving between C1 matrices y_i and y_j .

In 1.10 there are various possibilities in defining the cost c_{ij} , for instance

- $c_{ij} = 1 \forall i, j \Rightarrow DS(\alpha) = DT(\alpha) + DC(\alpha) - 1$
- $c_{ij} =$ movement cost between C1 matrices

$$y^i = ([\ell_k, r_m])_{m \in M} \text{ and } y^j = ([\ell_m^j, r_m^j])_{m \in M}$$

defined by

$$c_{ij} := \max_{m \in M} \{ \max\{|\ell_m^i - \ell_m^j|, |r_m^i - r_m^j|\} \} \quad (1.11)$$

Example 1.7. (Nussbaum 2006)

The following example shows that the three objective functions of Definition 1.6 are in general contradictory, i.e. decomposition α may be optimal for one

objective, but suboptimal for the others. (In (1.10) c_{ij} is completed?? by (1.11)

$$\begin{array}{rcl}
 A & = & \begin{pmatrix} 8 & 5 & 6 \\ 5 & 3 & 6 \end{pmatrix} & DT & DC & DS \\
 & = & 3 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + 1 \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} + 3 \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} + 2 \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} & \mathbf{9} & 4 & 15 \\
 & = & 5 \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} + 3 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} + 6 \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} & 14 & \mathbf{3} & 17 \\
 & = & 2 \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} + 3 \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} + 1 \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} + 4 \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} & 10 & 4 & \mathbf{14}
 \end{array}$$

1.2 Applications of C1 Decomposition

see transparencies at

http://optimierung.mathematik.uni-kl.de/hamacher/2007_03_12_Milano.pdf

References:

Radiation Therapy:

Ahuja, R.K. and Hamacher, H.W.: "A Network Flow Algorithm to Minimize Beam-On Time for Unconstrained Multileaf Collimator Problems in Cancer Radiation Therapy" *Networks* 2005, 36-41

Baatar, D., Ehrgott, M., Hamacher, H.W. and Woeginger, G.: "Decomposition of Integer Matrices and Multileaf Collimator Sequencing", *Discrete Applied Mathematics*, 152 (2005) 6-34.

Boland, N., Hamacher, H.W. and Lenzen, F.: "Minimizing Beam-On Time in Cancer Radiation Treatment Using Multileaf Collimators", *NETWORKS* (2004, 226-240

Semi-Simultaneous Flows:

Engau, A. and Hamacher, H.W.: "Semi-Simultaneous Flows and Binary Constrained (Integer) Linear Programs", *Reports in Wirtschaftsmathematik, Technische Universität Kaiserslautern*, No. 99 (2006)

Stop Location:

Poetranto, D.R., Hamacher, H.W., Horn, S., and Schöbel, A.: "Stop Location Design in Public Transportation Networks: Covering and Accessibility Objectives", *Reports in Wirtschaftsmathematik, Technische Universität Kaiserslautern, Department of Mathematics*, No. 97 (2006)

Schöbel, A., Hamacher, H.W., Liebers, A. and Wagner, D.: "The continuous stop location problem in public transportation networks", *Report in Wirtschaftsmathematik, Technische Universität Kaiserslautern, Department of Mathematics*, No. 81 (2002)

Chapter 2

Solving Linear and Integer Programs with C1 Constraint Matrices

Engau, Alexander and Hamacher, Horst W.: "Semi-Simultaneous Flows and Binary Constrained (Integer) Linear Programs", Reports in Wirtschaftsmathematik 99, Fachbereich Mathematik, Technische Universität Kaiserslautern (2006)

In this section, we consider linear programs and integer linear programs with a coefficient matrix A consisting of $K \geq 1$ C1 matrices (KC1, for short).

Definition 2.1. Let $K \in \mathbb{N}$ be a given positive integer, let $A^k \in \mathbb{B}^{M \times N_k}$, $k = 1, \dots, K$ be C1 matrices, let $b \in \mathbb{Z}_+^M$ be a nonnegative integer vector and let $c^k \in \mathbb{Z}^{N_k}$, $k = 1, \dots, K$ be integer vectors. Then the K consecutive ones integer program (KC1-IP) is defined by

$$\begin{aligned} & \text{minimize} && \sum_{k=1}^K c^k \text{T} x^k \\ & \text{subject to} && \sum_{k=1}^K A^k x^k = b, \\ & && x^k \geq 0 \text{ and integer for all } k = 1, \dots, K. \end{aligned} \tag{2.1}$$

The relevance of KC1 is indicated by the following result.

Proposition 2.2. Any integer program with binary coefficient matrix and integer data is equivalent to KC1-IP for some $K \leq \lceil \frac{N}{2} \rceil$.

Proof: Given KC1-IP, we use $A = (A^1 \dots A^K) \in \mathbb{B}^{M \times N}$, $c = (c^1 \dots c^K) \in \mathbb{Z}^N$ and $x = (x^1 \dots x^K)$ with $N = \sum_{k=1}^K N_k$ to obtain the equivalent IP $\min\{c \text{T} x : Ax = b, x \geq 0 \text{ integer}\}$ with binary coefficient matrix A and integer data.

Conversely, given any IP with $A \in \mathbb{B}^{M \times N}$, we can always find some partition $A = (A^1 \dots A^K)$ so that each A^k , $k = 1 \dots, K$ is C1. Trivially, we can

choose $K = N$ so that the matrix A is partitioned into its N column vectors, $A = (a^1 \dots a^N)$, with $a^k \in \mathbb{B}^M$ being C1 for all $k = 1, \dots, N$. In fact, K can be chosen so that $K \leq \lceil \frac{N}{2} \rceil$, as any matrix consisting of only two binary column vectors is necessarily C1 as well. \square

To find in Proposition 2.2 the smallest K such that a binary IP is equivalent to KC1-IP we can solve a decomposition cardinality (DC) problem. Although we will show later that the general DC problem is NP hard, we will give a polynomial algorithm for the DC problem applied to binary matrices.

We can formulate KC1 also in terms of C1 matrices in columns.

Definition 2.3. *Let $K \in \mathbb{N}$ be a given positive integer, let $A^k \in \mathbb{B}^{M_k \times N}$, $k = 1, \dots, K$ be C1 matrices in columns, let $b^1, \dots, b^K \in \mathbb{Z}_+^M$ be nonnegative integer vectors with $b^k \in \mathbb{Z}^{M_k}$ for $k = 1, \dots, K$, and let $c \in \mathbb{Z}^N$. Then the K consecutive ones integer program in columns (KC1-IP-Column) is defined by*

$$\begin{aligned} & \text{minimize } c^T x \\ & \text{subject to } A^k x = b^k, \quad \forall k = 1, \dots, K \\ & \quad \quad \quad x \geq 0 \text{ and integer.} \end{aligned} \tag{2.2}$$

Obviously, Proposition 2.2 can be reformulated using KC1 problems in columns (instead of rows).

2.1 Polynomial Algorithm for KC1 with $K = 1$

We first show 1C1-column is equivalent to a network flow problem. As application, we show that the beam-on time radiation problem and the decomposition cardinality problem of binary matrices into C1 matrices in columns is solvable in polynomial time. We then address the 1C1 (row) problem and use duality to reduce the latter problem to a 1C1-column problem and the subsequent application of complementary slackness computations.

2.1.1 Solving 1C1 Column Problems by a Network Flow Approach

The following transformation of a 1C1-IP-Column problem to a network flow problem can, for instance, be found in Ahuja et al. (1993).

Theorem 2.4. *KC1-IP-Column is equivalent to a network flow problem and thus solvable in polynomial time.*

Proof: Consider the 1C1-IP-Column

$$\begin{aligned} & \text{minimize } c^T x \\ & \text{subject to } Ax = b, \\ & \quad \quad \quad x \geq 0 \text{ and integer.} \end{aligned} \tag{2.3}$$

where $A = ([l_1, r_1), \dots, [l_N, r_N))$ is a C1 matrix in columns. Add to the extended $M \times (N + 1)$ matrix (A, b) a row of zeros as $(M + 1)^{st}$ row. The set of solutions of $Ax = b$ does not change if we subtract iteratively for $m = M, \dots, 1$ the $(m + 1)^{th}$ row from the m^{th} row and maintain the first row. After this transformation, A has turned into the node-arc incidence matrix of a network flow problem with $M + 1$ nodes and N arcs (i, j) with $i = l_n$ and $j = r_n, n = 1, \dots, N$. Hence 1C1-IP-Column has turned into a network flow problem. \square

Example 2.5. For

$$(A | b) = \left(\begin{array}{cccccc|c} 1 & 0 & 1 & 0 & 1 & 0 & 2 \\ 1 & 1 & 0 & 0 & 1 & 0 & 3 \\ 1 & 1 & 0 & 1 & 0 & 0 & 5 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array} \right) \text{ and } c = (3 \ 2 \ 1 \ 8 \ -2 \ 4),$$

the matrix is extended by a row of zeroes,

$$(A | b) = \left(\begin{array}{cccccc|c} 1 & 0 & 1 & 0 & 1 & 0 & 2 \\ 1 & 1 & 0 & 0 & 1 & 0 & 3 \\ 1 & 1 & 0 & 1 & 0 & 0 & 5 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \text{ and}$$

and then rows are subtracted from each other from bottom to top as in the proof of Theorem 2.4

$$(\tilde{A} | \tilde{b}) = \left(\begin{array}{cccccc|c} 1 & 0 & 1 & 0 & 1 & 0 & 2 \\ 0 & 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 & 2 \\ -1 & -1 & 0 & 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & -1 & 0 & -1 & -1 \end{array} \right).$$

\tilde{A} is the node-arc matrix of the graph of Figure ??? and \tilde{b} is the vector of supplies and demands, where – by construction – its components add up to zero (a necessary condition for the flow problem to be feasible). Hence 1C1-IP has turned into the equivalent network flow problem with costs c on the arcs (see Figure 2.1). From the minimal cost flow we get the optimal solution $\mathbf{x} = (\mathbf{0}, \mathbf{5}, \mathbf{4}, \mathbf{0}, \mathbf{0}, \mathbf{1})$ of 1C1-Column.

Theorem 2.6. (Ahuja-Hamacher, 2005)

Let $A \in \text{Mat}_{M,N}(\mathbb{Z}_+)$ and let T be the set of all $M \times N$ C1 matrices. Then the integer, unconstrained decomposition time problem DT_{MC}

Then the integer, unconstrained decomposition time problem DT_{MC}

$$\begin{aligned} \min \quad & DT(\alpha) = \sum_{t \in T} \alpha_t \\ \text{s.t.} \quad & \sum_{t \in T} \alpha_t \cdot y_t = A \\ & \alpha_t \geq 0 \quad \text{integer} \end{aligned} \quad (2.4)$$

is solvable in $O(M, N)$ time.

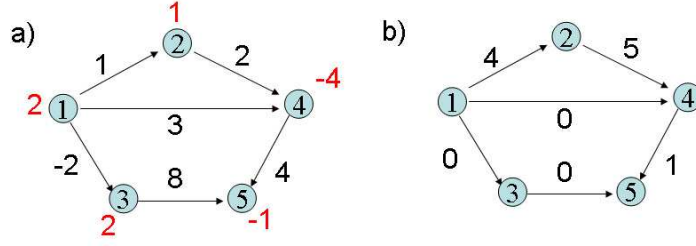


Figure 2.1: a) Flow Graph of Example 2.5 and b) minimal cost flow.

Proof: We first show that DT_{MC} can be solved in $O(N)$ time for each single row $A_m \in Mat_{1,N}(\mathbb{Z}_+)$. Let \tilde{T} be the set of C1 vectors $\mathbf{y}_t \in Mat_{1,N}(\mathbb{B})$. \tilde{T} contains the $\binom{N}{2}$ vectors

$$\begin{array}{cccc}
 (1, 0, \dots, 0, 0), & (1, 1, 0, \dots, 0) & , \dots, & (1, 1, \dots, 1, 0), & (1, 1, \dots, 1, 1), \\
 & (0, 1, 0, \dots, 0) & , \dots, & (0, 1, \dots, 1, 0), & (0, 1, \dots, 1, 1), \\
 & \vdots & & \vdots & \vdots \\
 & & & (0, 0, \dots, 1, 0), & (0, 0, \dots, 1, 1), \\
 & & & (0, 0, \dots, 0, 1) &
 \end{array}$$

Then

$$\begin{aligned}
 \sum_{t \in \tilde{T}} \alpha_t \mathbf{y}_t = A_m &\Leftrightarrow \sum_{t \in \tilde{T}} \alpha_t \mathbf{y}_t^T = \mathbf{A}_m^T \\
 &\Leftrightarrow \left(\begin{array}{cccccccccccc}
 1 & 1 & \dots & 1 & 1 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 \\
 0 & 1 & \dots & 1 & 1 & 1 & \dots & 1 & 1 & \dots & 0 & 0 & 0 \\
 0 & 0 & \dots & 1 & 1 & 0 & \dots & 1 & 1 & \dots & 0 & 0 & 0 \\
 \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots \\
 0 & 0 & \dots & 1 & 1 & 0 & \dots & 1 & 1 & \dots & 1 & 1 & 0 \\
 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 1 & \dots & 0 & 1 & 1
 \end{array} \right) \alpha = A_m.
 \end{aligned}$$

The latter system of linear equations has an $N \times \frac{1}{2}N(N+1)$ coefficient matrix which has the C1 in columns property and DT is thus by Theorem 2.4 equivalent to a network flow problem.

The resulting flow network has $(N+1)$ nodes, $\frac{1}{2}N(N+1)$ arcs (i, j) with $1 \leq i < j \leq N+1$ and cost $c_{ij} = 1$ for all arcs (i, j) . According to the proof of Theorem 2.4 the supply/demand in each node i is

$$b_i := \begin{cases} a_i - a_{i-1} & \text{for } i = N+1, N, \dots, 2 \\ a_1 & \text{for } i = 1 \end{cases} \quad (2.5)$$

An example of such a network is shown in Figure 2.2 (??)

ADD FIG. 2.2 HERE

An optimal flow can be found in linear time by the shortest augmenting path algorithm. In each iteration this path is found as a single arc from the left most supply node i (i.e. $b_i > 0$) to the left most demand node j (i.e. $b_j < 0$)

and the flow on this path is $\min\{b_i, \dots, b_j\}$. This can be done in constant time (and the network does not have to be constructed to find this shortest path). In each iteration the demand in node i changes to $b_i = 0$ or the supply in node j changes to $b_j = 0$ (or both) such that we need at most N iterations to find an optimal flow.

ADD FIG. 2.3 HERE

If d_t^m is the flow on arc $\ell_t = (i, j)$, then we set

$$\mathbf{y}_t^m = \begin{pmatrix} & i & & j \\ & \vdots & & \vdots \\ 0 & \dots & 0 & 1 & \dots & 1 & 0 & \dots \end{pmatrix} = [i, j]$$

such that $A_{m \cdot} = \sum_{t \in \tilde{T}} \alpha_t^m \cdot \mathbf{y}_t^m$.

Hence the minimum decomposition time of row $A_{m \cdot}$ is $DT_m = \sum_{c \in \tilde{T}} \alpha_t^m$ can be found in $O(N)$ time for each row $A_{m \cdot}$ of A .

The optimal DT decomposition time $DT(\alpha)$ of A cannot be smaller than the decomposition time of any of the rows $A_{m \cdot}$.

$$\max_{m \in M} DT_m \tag{2.6}$$

is a lower bound for the optimal value $DT(\alpha)$ of 2.4. This bound is attained by piecing the row decompositions suitably together (see Example 2.7). \square

Example 2.7. *Let*

$$A = \begin{pmatrix} 2 & 7 & 6 & 1 \\ 8 & 2 & 0 & 3 \\ 1 & 4 & 1 & 3 \end{pmatrix}$$

$m = 1$:

$$\begin{aligned} A_1 &= (2 \ 7 \ 6 \ 1) \Rightarrow \mathbf{b}_1 = (2 \ 5 \ -1 \ -5 \ -1) \\ &\Rightarrow (2 \ 7 \ 6 \ 1) = 1 \cdot (1 \ 1 \ 0 \ 0) + 1 \cdot (1 \ 1 \ 1 \ 0) + 4 \cdot (0 \ 1 \ 1 \ 0) + 1 \cdot (0 \ 1 \ 1 \ 1) \end{aligned}$$

and $DT_1 = 7$

$m = 2$:

$$\begin{aligned} A_2 &= (8 \ 2 \ 0 \ 3) \Rightarrow \mathbf{b}_2 = (8 \ -6 \ -2 \ 3 \ -2) \\ &\Rightarrow (8 \ 2 \ 0 \ 3) = 6 \cdot (1 \ 0 \ 0 \ 0) + 2 \cdot (1 \ 1 \ 0 \ 0) + 3 \cdot (0 \ 0 \ 0 \ 1) \end{aligned}$$

and $DT_2 = 11$

$m = 3$:

$$\begin{aligned} A_3 &= (1 \ 4 \ 1 \ 3) \Rightarrow \mathbf{b}_3 = (1 \ 3 \ -3 \ 2 \ -3) \\ &\Rightarrow (1 \ 4 \ 1 \ 3) = 1 \cdot (1 \ 1 \ 0 \ 0) + 2 \cdot (0 \ 1 \ 0 \ 0) + 1 \cdot (0 \ 1 \ 1 \ 1) + 2 \cdot (0 \ 0 \ 0 \ 1) \end{aligned}$$

and $DT_3 = 6$

The bound (2.6) is $\max\{DT_1, DT_2, DT_3\} = 11$. Hence the minimum DT decomposition of A is

$$\begin{aligned} A = \begin{pmatrix} 2 & 7 & 6 & 1 \\ 8 & 2 & 0 & 3 \\ 1 & 4 & 1 & 3 \end{pmatrix} &= 2 \cdot \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} + 2 \cdot \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &+ 1 \cdot \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} + 1 \cdot \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \\ &+ 1 \cdot \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + 4 \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{aligned}$$

The procedure resulting from the proof of Theorem 2.6 can immediately be applied to minimizing the beam-on-time in radiation therapy. This procedure is known as sweep procedure (Bortfeld & Boyer, 1993), but it was not known since recently, that this procedure yields an optimal decomposition time.

If the starting matrix A is itself binary, then $\alpha_t \in \{0, 1\}$ for all $t \in T$ and the decomposition time objective $DT_{UC}(\alpha)$ and the decomposition cardinality objective $DC_{UC}(\alpha)$ coincide such that the latter problem is polynomially solvable. We have thus shown the following result.

Corollary 2.8. *Let $A \in \text{Mat}_{M,N}(\mathbb{B})$. Then the unconstrained decomposition cardinality problem ($T = \text{set of all C1 matrices}$)*

$$\begin{aligned} \min \quad & DC(\alpha) = |\{\alpha_t : \alpha_t > 0\}| \\ \text{s.t.} \quad & \sum_{t \in T} \alpha_t y_t = A \\ & \alpha_t \geq 0 \end{aligned}$$

is solvable in $O(M \cdot N)$ time.

Exercise 2.9. (= **Assignment 1**) *Minimize the decomposition time $DT(\alpha)$ for the matrix*

$$A = \begin{pmatrix} 2 & 1 & 3 & 4 & 7 & 1 \\ 4 & 0 & 0 & 3 & 8 & 2 \\ 1 & 3 & 0 & 0 & 7 & 8 \\ 9 & 3 & 2 & 4 & 0 & 7 \end{pmatrix}$$

2.1.2 Solving 1C1 (Row) Problems by a Network Flow Approach

Let

$$A = \begin{pmatrix} [\ell_1 & , & r_1) \\ & \vdots & \\ [\ell_M & , & r_M) \end{pmatrix}$$

be an $M \times N$ C1 matrix in rows.

Proposition 2.10. *The 1C1-IP*

$$\begin{aligned} \min \quad & \mathbf{c}\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \text{ integer} \end{aligned} \tag{2.7}$$

is solvable in polynomial time by a network flow algorithm.

Proof: The polynomial solvability follows immediately by observing that A is a totally unimodular matrix (see, for instance integer programming lectures). We show in the following how to use network flows by considering the dual of (2.7)

$$\begin{aligned} \max \quad & \mathbf{b}^T \boldsymbol{\pi} \\ \text{s.t.} \quad & A^T \boldsymbol{\pi} + I\boldsymbol{\alpha} = \mathbf{c}^T \\ & \boldsymbol{\pi} \leq \mathbf{0} \\ & \boldsymbol{\alpha} \geq \mathbf{0} \end{aligned} \tag{2.8}$$

Here A^T and I are $N \times M$ and $N \times N$ C1 matrices in columns, respectively. We can thus use the row operations as in the proof of Theorem (2.4) to rewrite the dual 2.8 as network flow problem

$$\begin{aligned} \max \quad & \mathbf{b}^T \boldsymbol{\pi} \quad (\Leftrightarrow -\min(-\mathbf{b}^T)\boldsymbol{\pi}) \\ \text{s.t.} \quad & A^T \boldsymbol{\pi} + \tilde{B}\boldsymbol{\alpha} = \tilde{\mathbf{c}}^T \\ & \boldsymbol{\pi} \leq \mathbf{0} \\ & \boldsymbol{\alpha} \geq \mathbf{0} \end{aligned} \tag{2.9}$$

where \tilde{A} and \tilde{B} is an $(N+1) \times M$ and $(N+1) \times N$ node-arc incidence matrix and $\tilde{c}_n := c_n - c_{n-1}$ for $n = 2, \dots, N+1$ and $\tilde{c}_1 := c_n$.

The resulting flow network has $N+1$ nodes $1, \dots, N, N+1$ and M π -arcs $e_m = (\ell_m, r_m), m \in M$ and N α -arcs $f_n = (n, n+1), n \in N$. The π -arcs have unit profits b_m , the α -arcs cost 0. The node supplies are given by $\tilde{c}_n, n = 1, \dots, N+1$ (see Figure 2.4 ?? and Example 2.11)

FIGURE 2.4 TO BE ADDED

The network flow problem can be solved in polynomial time. If the problem is unbounded, then the primal 1C1-IP has no feasible solution.

Otherwise an optimal solution or the primal 1C1-IP is computed by complementary slackness or by using the potential method from network optimization. \square

Example 2.11.

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} [2, 4) \\ [4, 6) \\ [2, 3) \\ [3, 6) \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 1 \\ 5 \\ 3 \\ 6 \end{pmatrix}, \quad \mathbf{c} = (2, 3, -1, 4, -2)$$

$$(A^T | \mathbf{c}^T) = \left(\begin{array}{cccc|c} 0 & 0 & 1 & 0 & 2 \\ 1 & 0 & 0 & 0 & 3 \\ 1 & 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & 1 & 4 \\ 0 & 1 & 0 & 1 & -2 \end{array} \right) \Rightarrow (\tilde{\mathbf{A}}^T | \tilde{\mathbf{c}}^T) = \left(\begin{array}{cccc|c} 0 & 0 & 1 & 0 & 2 \\ 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & -4 \\ -1 & 1 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & -6 \\ 0 & -1 & 0 & -1 & -2 \end{array} \right)$$

$$\mathbf{I} = \left(\begin{array}{cccc} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{array} \right) \Rightarrow \tilde{B} = \left(\begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & -1 \end{array} \right)$$

ADD FIG. 2.5 A AND B

The definition of the incremental network for flows (note that none of the arcs have a capacity, but $\alpha(e) \geq 0 \forall \alpha$ -arcs) implies that there does not exist a positive (recall that we **maximize** the objective function) cycle and thus (π, α) is an optimal flow.

Using π as dual solution in $A^T \pi \leq \mathbf{c}^T$ the second and fourth inequality hold strictly such that the complementary slackness conditions imply $x_2 = x_4 = 0$. The remaining system

$$(A_{.1}, A_{.3}, A_{.5}) = \begin{pmatrix} x_1 \\ x_3 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \\ x_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 5 \\ 3 \\ 6 \end{pmatrix}$$

has the unique solution

$$\begin{pmatrix} x_1 \\ x_3 \\ x_5 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 5 \end{pmatrix}$$

The resulting feasible solution $\mathbf{x}^T = (3 \ 0 \ 1 \ 0 \ 5)$ has the primal objective value

$$\mathbf{c}\mathbf{x} = (2, 3, -1, 4, -2) \begin{pmatrix} 3 \\ 0 \\ 1 \\ 0 \\ 5 \end{pmatrix} = 6 - 1 - 10 = -5 = \mathbf{b}^T \pi$$

and is thus optimal

Exercise 2.12. (=Assignment 2)

Solve the 1C1 Row Problem

$$\begin{aligned} \min \quad & \mathbf{c}\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \text{ integer} \end{aligned} \tag{2.10}$$

using the network flow approach applied to the data

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 2 \\ 1 \\ 3 \\ 4 \end{pmatrix}, \quad \mathbf{c} = (2, 1, 3, 7, 4)$$

2.2 Linear Programs and Semi-Simultaneous Flows

In this section we return to the general $KC1 - IP$ and $KC1 - LP$ given by $K \in \mathbb{N}$, K C1 matrices $A^k \in \mathbb{B}^{m \times n_k}$, $b \in \mathbb{Z}_+^m$, and $c^k \in \mathbb{Z}^{n_k}$, $k = 1, \dots, K$ as

$$\begin{aligned} & \text{minimize} \quad \sum_{k=1}^K c^k \mathbf{T} x^k \\ & \text{subject to} \quad \sum_{k=1}^K A^k x^k = b, \\ & \quad \quad \quad x^k \geq 0 \text{ and integer for all } k = 1, \dots, K. \end{aligned} \tag{2.11}$$

The following example shows that the integrality property which holds for $K = 1$ is already lost for $K = 2$. (Examples were independently found by Schöbel (2004) and Engau (2005))

Example 2.13. Choose $K = n_1 = n_2 = 2$, $m = 3$, $A^1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$, $A^2 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$, $b = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$, and $c^1 = c^2 = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$ with $x^T = (x_1^1 \ x_2^1 \ x_1^2 \ x_2^2)$. Then, the $2C1-IP$ problem

$$\min \left\{ (1 \ 3 \ 1 \ 3)x : \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} x = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, x \geq 0 \text{ integer} \right\} \tag{2.12}$$

has the optimal solution $x_1^1 = x_2^1 = x_1^2 = 0$, $x_2^2 = 1$ with objective 3. Its linear programming relaxation $2C1-LP$, however, yields the improved, fractional solution $x_1^1 = x_2^1 = x_1^2 = 0.5$, $x_2^2 = 0$ with objective 2.5. Clearly, this also implies that the combined matrix $A = (A^1 \ A^2)$ cannot be TU , which is verified easily.

2.2.1 Definition of se-sim flows

We concentrate in the following on the linear, continuous problem $KC1-LP$ and its dual $KC1-LPD$ given by

$$\max \left\{ b^T \pi : A^k \mathbf{T} \pi \leq c^k \text{ for all } k = 1, \dots, K, \pi \in \mathbb{R}^m \right\}. \tag{2.13}$$

Since the matrices A^k in (2.11) are row C1, the matrices $A^k \mathbf{T}$ in 2.13 are column C1. Thus, after introducing nonnegative slack variables $\alpha^k \in \mathbb{R}_+^{n_k}$, we obtain the equality constraints

$$A^k \mathbf{T} \pi + I_{n_k} \alpha^k = c^k$$

which now can be transformed into K systems of flow conservation constraints in K underlying networks G^k , $k = 1, \dots, K$ as in Section 2.1.2. A solution of problem (2.13) thus corresponds to vectors $\pi \in \mathbb{R}^m$, $\alpha^1 \in \mathbb{R}_+^{n_1}, \dots, \alpha^K \in \mathbb{R}_+^{n_K}$ so that for each $k = 1, \dots, K$, the pair (α^k, π) establishes a feasible flow in network G^k with π maximizing the objective $b^T \pi$. Since each such pair consists of an individual flow α^k and the common flow π that has to be chosen simultaneously for all K networks, we call the collection of all these flows a *semi-simultaneous network flow* and the associated problem the *semi-simultaneous network flow problem*. More general, we give the following definition.

Definition 2.14. A semi-simultaneous network (se-sim network) is a collection $G = \{G^k = (V^k, D^k \cup E^k) : k = 1, \dots, K\}$ of K individual networks G^k . The node sets are $V^k = \{1^k, 2^k, \dots, n_k^k, (n_k + 1)^k\}$ and the arc sets are partitioned into the sets $D^k = \{e_1^k, e_2^k, \dots, e_{n_k}^k\} = \{(1^k, 2^k), (2^k, 3^k), \dots, (n_k^k, (n_k + 1)^k)\}$ of individual arcs and $E^k = E = \{e_1, e_2, \dots, e_m\}$ of common arcs.

If $G = \{G^k : k = 1, \dots, K\}$ is a se-sim network, let f^k be an individual flow in network G^k , $k = 1, \dots, K$.

The collection $f = \{f^k : k = 1, \dots, K\}$ of K flows f^k is called semi-simultaneous flow (se-sim flow), if the flow values on common arcs coincide, i.e.,

$$f^{k_1}(e_i) = f^{k_2}(e_i) \text{ for all } i = 1, \dots, m \text{ and } k_1, k_2 = 1, \dots, K. \quad (2.14)$$

For each individual network G^k , the flow f^k restricted to single and common arcs, $f^k|_{D^k} =: \alpha^k$ and $f^k|_{E^k} =: \pi$, is called the individual and common network flow associated with f^k , respectively.

Example 2.15. TO DO: ADD OTHER EXAMPLE PRESENTED IN MILAN. SEE EXAMPLE 2C1 IN DECOMPOSITION NOTES

Consider the linear programming dual 2C1-LPD of Example 2.13 with $\pi^T = (\pi_1 \ \pi_2 \ \pi_3)$, $\alpha^1 = (\alpha_1^1 \ \alpha_2^1)$ and $\alpha^2 = (\alpha_1^2 \ \alpha_2^2)$,

$$\max \left\{ \pi_1 + \pi_2 + \pi_3 : \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \pi + \alpha^1 = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \pi + \alpha^2 = \begin{pmatrix} 1 \\ 3 \end{pmatrix} \right\}. \quad (2.15)$$

To transform this problem into a semi-simultaneous network flow problem, first append an additional zero row to the constraints, and then subtract its preceding row from all but the first, yielding the two systems of flow conservation constraints

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & -1 & 0 \\ -1 & 0 & -1 \end{pmatrix} \pi + \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \alpha^1 = \begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -1 & -1 \end{pmatrix} \pi + \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \alpha^2 = \begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix}. \quad (2.16)$$

Hence, 2C1-LPD is equivalent to finding a flow $(\alpha^1, \alpha^2, \pi)$ that maximizes $b^T \pi$ subject to flow conservation at each node, flow capacities $\alpha^1, \alpha^2 \geq 0$ and identical partial flow π in both networks (see Figure 2.2). Notice that although the values of π_1, π_2 and π_3 are required to be identical in both parts of the semi-simultaneous network, the corresponding edges connect, in general, different nodes.

It is easy to see that this problem has the optimal fractional solution $\pi_1 = 1.5$, $\pi_2 = -0.5$ and $\pi_3 = 1.5$ with slack variables $\alpha_1^1 = 0$, $\alpha_2^1 = 1$, $\alpha_1^2 = 0$ and $\alpha_2^2 = 0.5$ and objective 2.5 - thus confirming the result of Example 2.13.

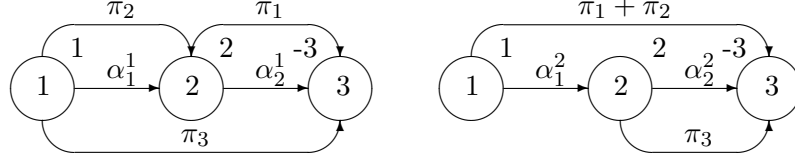


Figure 2.2: Semi-simultaneous flow corresponding to 2C1-LPD of Example 2.15 with common flow π and individual flows α^1 and α^2 .

Definition 2.16. Let $G = \{G^k : k = 1, \dots, K\}$ be a given se-sim network with benefits $b \geq 0$ and flow conservation right-hand-sides c^k for each individual network G^k . Then the se-sim flow problem (SE-SIM-FLOP) is defined as

$$\text{maximize } b(f) = \sum_{i=1}^m b(e_i) f(e_i) \quad (2.17)$$

subject to

$$\sum_{e \in \delta_+^k(i^k)} f^k(e) - \sum_{e \in \delta_-^k(i^k)} f^k(e) = \tilde{c}_i^k \quad i^k = 1^k, \dots, (n_k + 1)^k, \quad k = 1, \dots, K, \quad (2.17a)$$

$$f^{k_1}(e_i) = f^{k_2}(e_i) =: \pi_i \quad i = 1, \dots, m, \quad k_1, k_2 = 1, \dots, K, \quad (2.17b)$$

$$f^k(e_j^k) =: \alpha_j^k \geq 0 \quad j = 1, \dots, n_k, \quad k = 1, \dots, K. \quad (2.17c)$$

For fixed $k = 1, \dots, K$, the associated subproblem (where the flows π_i^k on the common arcs do not have to coincide) is called individual network flow problem (IN-FLOP).

2.2.2 Feasible se-sim flows

The goal of this section is to establish that the feasibility of IN-FLOP is necessary and sufficient for the feasibility of SE-SIM-FLOP. The first part of this statement is obvious.

Proposition 2.17. Given a feasible se-sim flow $f = \{f^k : k = 1, \dots, K\}$ for SE-SIM-FLOP, then each of the flows f^k is feasible for its associated individual network flow problems IN-FLOP.

To prove the converse we first show the following results.

Lemma 2.18. Any feasible se-sim network flow $f = \{f^k = (\alpha^k, \pi) : k = 1, \dots, K\}$ for SE-SIM-FLOP is uniquely determined by the common flow π .

Proof: The individual flows α^k in networks G^k correspond to the slack variables in the LPR dual and thus are uniquely determined by the dual variable or common flow vector π . \square

Lemma 2.19. *Let SE-SIM-FLOP be given and let $k \in \{1, \dots, K\}$ be fixed. For the k^{th} IN-FLOP, let $f^k = (\alpha^k, \pi^k)$ be a feasible flow and let $\pi \leq \pi^k$. Then the flow determined by π according to Proposition 2.18 is also feasible for IN-FLOP.*

Proof: This result follows from the special chain structure in the individual networks G^k : since the common flow along all common arcs is unrestricted, we can reduce the flow along some common arc e_i by the respective nonnegative flow difference $\pi^k - \pi$ while maintaining the flow balance constraints by increasing the flow along the individual arcs connecting its tail and head nodes g_i^k and h_i^k by the same amount. \square

Using this result, we are now able to show that individual feasibility for all IN-FLOPs is sufficient for the feasibility of SE-SIM-FLOP.

Proposition 2.20. *Let SE-SIM-FLOP be given and assume that for each $k = 1, \dots, K$, there exists a feasible individual flow $f^k = (\alpha^k, \pi^k)$ for the associated IN-FLOP. Then there exists a feasible se-sim network flow for SE-SIM-FLOP.*

Proof: Let $\{f^k = (\alpha^k, \pi^k) : k = 1, \dots, K\}$ be a collection of feasible flows for the IN-FLOPs, and define a common flow $\pi = \pi^1 \wedge \pi^2 \wedge \dots \wedge \pi^K$ as the componentwise minimum among all flows π^k , i.e., $\pi_i = \min\{\pi_i^k : k = 1, \dots, K\}$. Clearly $\pi \leq \pi^k$ for all $k = 1, \dots, K$, and hence, by Lemma 2.19, the common flow π determines feasible flows (β^k, π) in all individual networks G^k . Hence, the collection $f = \{(\beta^k, \pi) : k = 1, \dots, K\}$ of all such flows establishes a feasible se-sim network flow for SE-SIM-FLOP. \square

By combining Propositions 2.17 and 2.20, we conclude that se-sim feasibility is equivalent to individual feasibility for each individual network.

Theorem 2.21. *SE-SIM-FLOP is feasible if and only if all IN-FLOPs are feasible.*

2.2.3 Bounds on se-sim flows

Since each feasible se-sim flow defines a feasible flow for each IN-FLOP, we immediately obtain the following upper bound on the optimal objective value for SE-SIM-FLOP.

Proposition 2.22. *Let f be any feasible flow for SE-SIM-FLOP, and let $\{\hat{f}^k = (\hat{\alpha}^k, \hat{\pi}^k) : k = 1, \dots, K\}$ be a collection of optimal flows for the associated IN-FLOPs. Then $b(f) \leq \min\{b(\hat{f}^k) = b^T \hat{\pi}^k : k = 1, \dots, K\}$.*

It also follows that, if one IN-FLOP is bounded, then SE-SIM-FLOP must be bounded. Hence, we get the following result.

Corollary 2.23. *SE-SIM-FLOP has an optimal solution if it is feasible and if at least one associated IN-FLOP is bounded.*

The criterion of Corollary 2.23 can be used to check the feasibility of KC in strongly polynomial time. Unfortunately, this is, however, not a necessary criterion, since the following example demonstrates, that the converse of Proposition 2.23 is, in general, not true.

Example 2.24. Consider the two networks in Figure 2.3 with zero right hand sides for the flow conservation constraints at all nodes and uniform benefits $b_1 = b_2 = b_3 = 1$ for all common arcs. Then every flow of the form $f^1 =$

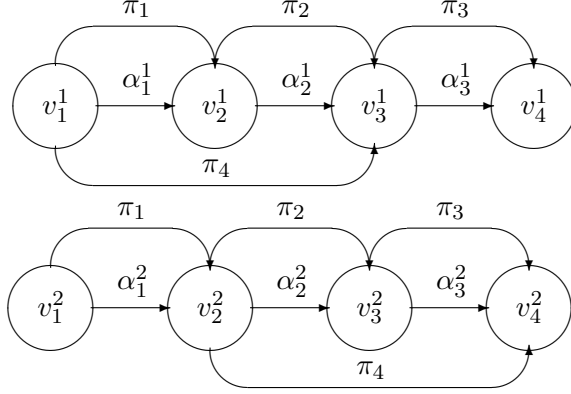


Figure 2.3: Illustration of Example 2.24 (semi-simultaneous boundedness)

$(\alpha^1, \pi^1) = \left((0 \ 0 \ 0)^T, (\rho \ \rho \ 0 \ -\rho)^T \right)$ with $\rho \in \mathbb{R}$ is feasible for the first network and may achieve arbitrary benefit ρ . Similarly, for the second network, every flow $f^2 = (\alpha^2, \pi^2) = \left((0 \ 0 \ 0)^T, (0 \ \rho \ \rho \ -\rho)^T \right)$ with $\rho \in \mathbb{R}$ is feasible with benefit ρ , so that both IN-FLOPs are unbounded. However, it is easily verified that the optimal se-sim flow is the zero flow with zero benefit.

2.2.4 Optimal se-sim flows

In order for a given flow to be optimal, we can state the following sufficient condition.

Proposition 2.25. Let SE-SIM-FLOP be given and let $\{ \hat{f}^k = (\hat{\alpha}^k, \hat{\pi}^k) : k = 1, \dots, K \}$ be a collection of optimal flows for the associated IN-FLOPs. If there exists an index $j = 1, \dots, K$ such that $\hat{\pi}^j \leq \hat{\pi}^k$ for all $k = 1, \dots, K$, then $\pi = \hat{\pi}^j$ determines an optimal se-sim network flow for SE-SIM-FLOP.

Proof: Let $\hat{\pi}^j \leq \hat{\pi}^k$ for all $k = 1, \dots, K$, or equivalently, $\hat{\pi}^1 \wedge \hat{\pi}^2 \wedge \dots \wedge \hat{\pi}^K = \hat{\pi}^j$. Since all benefits are assumed to be nonnegative, it follows that $b^T \hat{\pi}^j \leq b^T \hat{\pi}^k$, and hence $b(f) \leq b^T \hat{\pi}^j$ for all feasible se-sim flows f by Proposition 2.22. As was shown in the proof of Proposition 2.20, the common flow $\pi = \hat{\pi}^1 \wedge \hat{\pi}^2 \wedge \dots \wedge \hat{\pi}^K = \hat{\pi}^j$ determines a feasible se-sim flow for SE-SIM-FLOP with maximal benefit $b^T \pi = b^T \hat{\pi}^j$, and hence the se-sim flow determined by $\pi = \hat{\pi}^j$ is optimal. \square

Another sufficient optimality condition which follows immediately from known results of classic network flow theory is based on se-sim residual flows.

Proposition 2.26. *Let SE-SIM-FLOP be given and let f and \bar{f} be two given feasible se-sim network flows with $b(f) < b(\bar{f})$. Then the difference flow $\tilde{f} = \bar{f} - f$ defines a se-sim flow with positive residual benefit $b(\tilde{f}) = b(\bar{f}) - b(f) > 0$ in the semi-simultaneous residual network $G_f = \{G_f^k : k = 1, \dots, K\}$, where G_f^k denotes the residual network of network G^k induced by the flow f^k .*

Definition 2.27. *Let SE-SIM-FLOP be given and let f be a feasible se-sim flow. The se-sim flow \tilde{f} is called an improvement (flow) for f if \tilde{f} is feasible in the se-sim residual network G_f with positive residual benefit $b(\tilde{f}) > 0$.*

The next result shows that improvement flows take over the role of negative cycles in the theory of classical network flow theory.

Theorem 2.28. *Let SE-SIM-FLOP be given and \hat{f} be a feasible se-sim flow. Then \hat{f} is optimal if and only if there does not exist an improvement.*

Proof: If \hat{f} is optimal, then $b(\hat{f}) \geq b(f)$ for all other flows f and thus, by definition and Proposition 2.26, there cannot exist an improvement.

Conversely, for any other se-sim flow f Proposition 2.26 implies that the flow $\tilde{f} = \hat{f} - f$ is a feasible se-sim flow in the residual network $G_{\hat{f}}$ which - by the assumption of the theorem - has a residual benefit $b(\tilde{f}) = b(\hat{f}) - b(f) = b(\hat{f} - f) \geq 0$. Hence, the se-sim flow \hat{f} is optimal. \square

We have thus shown the validity of a generic procedure for the se-sim network flow problem which can be interpreted as generalization of cycle canceling algorithm of classical network flow theory.

Procedure 2.29. *Solving SE-SIM-FLOP:*

1. *Given SE-SIM-FLOP, find a feasible se-sim flow f .
IF no feasible flow exists, STOP - the problem is infeasible.*
2. *Given feasible se-sim flow f , find an improvement \tilde{f} .
IF no further improvement exists, STOP - f is optimal.*
3. *Update $f := f + \tilde{f}$ and repeat step 2.*

Exercise 2.30. (=Assignment 3) *Solve the 2C1 Row Problem*

$$\begin{aligned} \min \quad & \mathbf{c}^1 \mathbf{x}^1 + \mathbf{c}^2 \mathbf{x}^2 \\ \text{s.t.} \quad & A^1 \mathbf{x}^1 + A^2 \mathbf{x}^2 = \mathbf{b} \\ & \mathbf{x}^1, \mathbf{x}^2 \geq \mathbf{0} \text{ integer} \end{aligned} \tag{2.18}$$

using the se-sim network flow approach applied to the data

$$A = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}, \quad \mathbf{c} = (\mathbf{2}, \mathbf{1}, \mathbf{3}, \mathbf{1}, \mathbf{2}, \mathbf{1})$$

2.3 KC1 Projects

Various additional research questions are motivated by the results of this paper.

Project 2.31. *The first project involves implementation.*

- *Implement the SeSim Flow Algorithm*
- *Compare with standard software*

Project 2.32. *Are there any special cases of KC1 ($K > 1$) for which SeSim Flow can be solved in polynomial time?*

Project 2.33. *Can the sufficient feasibility criterion in Corollary 2.23 be extended to a necessary and sufficient criterion.*

Project 2.34. *(Posed by Sandro Bosio, Milan, 2007-03-19) According to Corollary 2.8 the decomposition cardinality problem DC can be solved for binary matrices in polynomial time. Can one combine this result with transformation results of weak C1 matrices to solve the cardinality problem if column permutation is also allowed?*

Project 2.35. *Instead of starting with coefficient matrices which have the row C1 property, one may want to consider a binary matrix $A \in \mathbb{B}^{m \times n}$ into matrices that are column consecutive one, $A = (A^1 \dots A^L)$ with $A^l \in \mathbb{B}^{m_l \times n}$, $l = 1, \dots, L$ and $\sum_{l=1}^L m_l = m$. The resulting LP is equivalent to*

$$\begin{aligned} & \text{minimize } \sum_{l=1}^L c^T x \\ & \text{subject to } A^l x = b^l \text{ for all } l = 1, \dots, L, \\ & \quad x \in \mathbb{R}_+^n. \end{aligned} \tag{2.19}$$

and the equality constraints $A^l x = b^l$ can directly be transformed into L systems of flow conservation constraints in L underlying networks G^l , $l = 1, \dots, L$. A solution of problem (2.19) then corresponds to a vector $x \in \mathbb{Z}_+^n$ so that for each $l = 1, \dots, L$, the vector x establishes a feasible flow in each of the networks G^l while minimizing the objective $c^T x$. In contrast to the semi-simultaneous flows considered in this paper, in this approach there are no individual flows. We can therefore call x a simultaneous flow.

Which results can you derive for simultaneous flows?

Project 2.36. *Another interesting topic is to generalize the concept of se-sim flows to more general se-sim networks. In our paper, the se-sim networks were defined by the underlying KC1 linear program. This implies that the individual arcs in each of the individual networks have a Hamiltonian path structure. If we allow an arbitrary set of individual arcs, an interesting question is to characterize structures which allow a duplication of the feasibility, optimality and improvement results of Section 2.2.*

Project 2.37. *Even more general, the topic of this paper motivates research on simultaneous graph theory problems defined on a collection $G = \{G^k = (V^k, D^k \cup E^k) : k = 1, \dots, K\}$ of directed or undirected individual graphs G^k . Feasibility and optimality characterizations of (semi-)simultaneous shortest path, spanning tree, matching, etc. are interesting in their own right, but may also be of practical interest in the same way as the (semi-)simultaneous network flow theory developed in this paper.*

Chapter 3

Constrained Decomposition and Complexity of Decomposition Cardinality Problem

As we have seen in section 2.1.2, the decomposition time problem

$$DT(\alpha) = \sum_{t \in T} \alpha_t \quad (3.1)$$

can be solved in linear time if T is the set of all C1 matrices. This is the *unconstrained DT problem*. In the first section of this chapter, we will consider *constrained decomposition problems*, where only a subset of T with certain properties is allowed. We then show that the unconstrained decomposition cardinality problem is NP hard.

3.1 Two Network Flow Formulations for the Constrained Decomposition Time Problem

The content of this section is based on the paper Boland, N., Hamacher, H.W. and Lenzen, F.: "Minimizing Beam-On Time in Cancer Radiation Treatment Using Multileaf Collimators", NETWORKS (2004), 226-240

Constrained decomposition problems are mainly motivated by their applications in intensity modulated radiation therapy.

Example 3.1. *In the modeling of modulation of radiation using multileaf collimators, one often has to consider interleaf motion constraints, i.e., exclude that the left leaf in one row of leaves is to the right of the right leaf in an adjacent row and vice versa (otherwise crashes between the two leaf points or significant radiation leakage in areas which ought to be blocked).*

Another constraint which may have to be enforced for reasons of physics is that the opening defined by the block of ones in the C1 matrix has to have a certain minimal width (width constraint).

We will concentrate in the following on constrained decomposition time problems involving interleaf motion constraints.

Definition 3.2. A C1 matrix $Y = ([l_i, r_i])_{i=1}^M$ is called (interleaf motion) constrained C1 matrix if for all $m = 1, \dots, M - 1$

$$l_{m+1} \leq r_m, \quad \text{and} \quad (3.2)$$

$$l_m \leq r_{m+1}. \quad (3.3)$$

Given an $M \times N$ matrix A of non-negative integers, the (interleaf motion) constraint decomposition time (DT) problem is

$$\begin{aligned} \min \quad DT(\alpha) &= \sum_{t \in T'} \alpha_t \\ \text{s.t.} \quad &\sum_{t \in T'} \alpha_t \cdot y_t = A \\ &\alpha_t \geq 0 \quad \text{integer} \end{aligned} \quad (3.4)$$

where T' is the set of all constrained C1 matrices.

We first show that constrained C1 matrices can be represented as paths in a suitably chosen network, the C1 matrix graph $G_{C1} = (V_{C1}, E_{C1})$. This network is introduced first.

G_{C1} is a layered digraph, consisting of M layers which correspond to the M rows of a C1 matrix. In each layer $m = 1, \dots, M$ there are $\frac{1}{2}N(N+1)$ nodes, denoted by (m, l, r) , which represent potential positions l and r of the left and right leaf in row m , respectively. Here $l \in \{1, \dots, N\}$, $r \in \{1, \dots, N, N+1\}$ and $l \leq r$. Two dummy nodes D and D' that act as start and end nodes are added, so

$$V_{C1} := \{(m, l, r) : m = 1, \dots, M, l = 1, \dots, N, r = 1, \dots, N+1, l \leq r\} \cup \{D, D'\}. \quad (3.5)$$

The arc set E_{C1} contains all edges from D to the first layer 1 and from the last layer m to D' , i.e., E_{C1} contains

$$E_+(D) := \{(D, (1, l, r)) : (1, l, r) \in V_{C1}\} \quad (3.6)$$

$$E_-(D') := \{((M, l, r), D') : (M, l, r) \in V_{C1}\}. \quad (3.7)$$

For $m = 1, \dots, M - 1$, we define the arcs between layers by

$$E_+(i) := \{((m, l, r), (m+1, l', r')) : l' \leq r, r' \geq l\}. \quad (3.8)$$

The latter arcs reflect the exclusion of interleaf motion. In the case that interleaf constraints are unnecessary, we simply omit the restrictions $l' \leq r$ and $r' \geq l$ in (3.8), and include arcs from all nodes in one layer to all nodes in the next. Finally we add the return arc (D', D) , so the set of all arcs is given by

$$E_{C1} := E_+(D) \cup E_-(D') \cup \{(D', D)\} \cup \bigcup_{i=1}^{M-1} E_+(i). \quad (3.9)$$

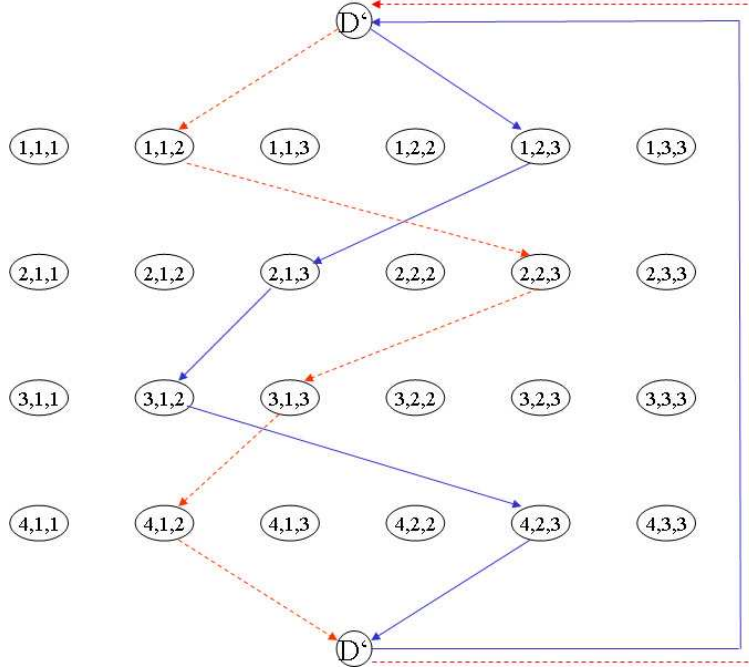


Figure 3.1: A $C1$ matrix graph with complete node set V_{C1} and some of the arcs of E_{C1} , including two cycles C_1 and C_2 (straight and dotted arcs).

An example of a $C1$ matrix graph G_{C1} is shown in Figure 3.1. The next lemma states some properties of G_{C1} . The proof of this lemma is an immediate consequence of the definition of G_{C1} .

Lemma 3.3.

- (1) $G_{C1} \setminus \{(D', D)\}$ is an acyclic digraph.
- (2) Every cycle – and thus every (D, D') path – in G_{C1} corresponds to a constrained $C1$ matrix and vice versa.

Figure 3.2 shows the $C1$ matrix and the leaf configuration corresponding to the cycle $C = (D, (112), (223), (313), (412), D', D)$ in the $C1$ matrix graph of Figure 3.1.

Next, we will give a compact linear programming formulation for the constrained decomposition time problem based on this network model. In fact, the MLC problem can be formulated as a (polynomially solvable) network flow problem with side constraints.

The key observation is that due to the equivalence between $C1$ matrices and cycles in G_{C1} , $\sum_{t \in \mathcal{T}} \alpha_t$ can be interpreted as the value $x_{(D', D)}$ of a circulation x which is a composite of flows α_t on cycles C_t corresponding to constrained $C1$ matrices Y_t .

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{array}{c} \begin{array}{ccc} & 1 & 2 & 3 \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{array} \end{array}$$

Figure 3.2: Constrained C_1 matrix and leaf configuration corresponding to cycle $C = (D, (112), (223), (313), (412), D', D)$ in G_{C_1} of Figure 3.1.

If we consider, for instance, the two cycles

$$C_1 = (D, (102), (213), (303), (402), D', D)$$

and

$$C_2 = (D, (113), (203), (302), (413), D', D)$$

of the C_1 matrix graph in Figure 3.1 and $x(C_1) = 3$, $x(C_2) = 2$, (i.e., x is a circulation formed by taking 3 units of flow on cycle C_1 and adding 2 units of flow on cycle C_2), we get a flow corresponding to the intensity matrix

$$3 \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} + 2 \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 2 & 5 \\ 5 & 3 \\ 3 & 2 \end{pmatrix} = A.$$

Since by part (1) of Lemma 3.3 all cycles contain arc (D', D) we have

$$x_{(D', D)} = x(C_1) + x(C_2) = 5,$$

the flow on arc (D', D) represents the flow value which is to be minimized. Since, conversely, every circulation can be decomposed into cycles containing arc (D', D) (see Ahuja et al. (1993), Theorem 3.5 and Property 3.6, or Hamacher and Klamroth (2001)), the MLC problem can be solved as a network flow problem in G_{C_1} , in fact as a minimum cost circulation problem, with respect to arc costs given by

$$c_e := \begin{cases} 1 & \text{if } e = (D', D) \\ 0 & \text{if } e \in E_{C_1} \setminus \{(D', D)\}, \end{cases}$$

for each $e \in E_{C_1}$.

We have, however, to enforce the side constraint that the values a_{ij} of the matrix A are generated by the circulation, i.e., we have the side constraints

$$\sum_{l=0}^{n-1} \sum_{r=n+1}^{N+1} \sum_{e \in E_{-}(m, l, r)} x_e = a_{ij} \quad (3.10)$$

for all $m = 1, \dots, M$ and all $n = 1, \dots, N$. (Here, we write $E_{-}(q)$ to denote the set of edges in E_{C_1} entering node q , i.e., $\{(p, q) \in E_{C_1}\}$. Later we will find

it helpful to use $E_+(q)$ to denote the set of edges in E_{C_1} leaving node q , i.e., $\{(q, p) \in E_{C_1}\}$.)

The network circulation formulation, together with the side constraints (3.10), provide a linear programming formulation for the constrained decomposition time problem. Note that its size, in terms of numbers of variables and constraints, is polynomial in n and m . We have thus proved the following result.

Theorem 3.4. *The constrained decomposition time problem can be solved in polynomial time as a min cost network flow problem with side constraints (3.10).*

Next, we give a second linear programming formulation that is even closer to a pure network flow formulation. Rather than stating (3.10) as an algebraic constraint, the C1 matrix graph is expanded to a network $\hat{G}_{C_1} = (\hat{V}_{C_1}, \hat{E}_{C_1})$ defined as follows (see Figure 3.3).

Each node $(m, l, r) \in N_{C_1} := V_{C_1} \setminus \{D, D'\}$ is split into two nodes $(m, l, r)^1$ and $(m, l, r)^2$. The idea is that flow enters row m via a node of the form $(m, l, r)^1$ and leaves row m via a node of the form $(m, l, r)^2$. In between, we have the flow go through arcs whose end nodes define the m^{th} interval $[l, r)$ representing the m^{th} row of the constrained C1 matrix. Thus we introduce new nodes of the form (m, n) .

Hence

$$\begin{aligned} \hat{V}_{C_1} &:= \{(m, l, r)^1, (m, l, r)^2 : (m, l, r) \in N_{C_1}\} \\ &\cup \{(m, n) : i = 1, \dots, M, j = 1, \dots, N\} \cup \{D, D'\}. \end{aligned} \quad (3.11)$$

Note that $|N_{C_1}| = \frac{1}{2}MN(N+1)$, and thus $|\hat{V}_{C_1}| = 2|N_{C_1}| + M(N+1) + 2 = O(MN^2)$.

The arc set \hat{E}_{C_1} consists of the following subsets. Let

$$\begin{aligned} \hat{E}_{C_1}^{\text{old}} &:= \{((m, l, r)^2, (m+1, l', r')^1) : ((m, l, r), (m+1, l', r')) \in E_{C_1}\} \\ &\cup \{(D, (1, l, r)^1) : (1, l, r)^1 \in \hat{V}_{C_1}\} \\ &\cup \{((M, l, r)^2, D') : (M, l, r)^2 \in \hat{V}_{C_1}\} \\ &\cup \{(D', D)\}. \end{aligned} \quad (3.12)$$

This set contains the edges in E_{C_1} connected to appropriate nodes in \hat{V}_{C_1} . In addition we have new arcs of two types. The first collection of new edges is

$$\begin{aligned} \hat{E}_{C_1}^1 &= \{((m, l, r)^1, (m, l)) : (m, l, r)^1 \in \hat{V}_{C_1}\} \\ &\cup \{((m, r), (m, l, r)^2) : (m, l, r)^2 \in \hat{V}_{C_1}\}. \end{aligned}$$

All arcs in $\hat{E}_{C_1}^{\text{old}} \cup \hat{E}_{C_1}^1$ have

$$\text{lower capacities } \underline{u}_e = 0 \text{ and upper capacities } \bar{u}_e = \infty. \quad (3.13)$$

The second set of new arcs is

$$\hat{E}_{C_1}^2 := \{((m, n), (m, n + 1)) : m = 1, \dots, M, n = 1, \dots, N + 1\}.$$

These are called the *decomposition arcs*, since the flow on these arcs has to be equal to the values a_{ij} of the matrix A to be decomposed. Therefore, the lower and upper capacity on these arcs are

$$\underline{u}_e = \bar{u}_e = a_{ij} \quad \forall e = ((m, n), (m, n + 1)) \in \hat{E}_{C_1}^2. \quad (3.14)$$

An illustration of the extended C1 matrix network, $\hat{G}_{C_1} = (\hat{V}_{C_1}, \hat{E}_{C_1})$, is given in Figure 3.3. It is not hard to see that the extended C1 matrix network warrants its name, and that a result similar to Lemma 3.3 for the C1 matrix network holds for the extended network. This result applies only to cycles with a particular property. We say that a cycle in \hat{G}_{C_1} has the *leaf position matching property* if, for any $(m, l, r) \in \hat{V}_{C_1}$, arc $((m, l, r)^1, (m, l))$ is in the cycle if and only if arc $((m, r - 1), (m, l, r)^2)$ is in the cycle.

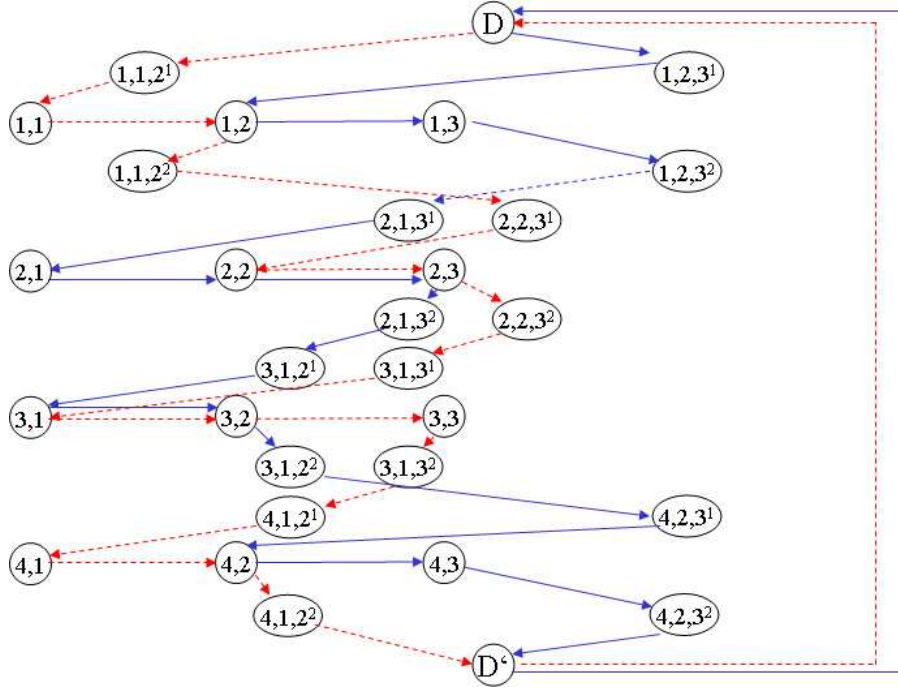


Figure 3.3: An extended C1 matrix graph. Only nodes and arcs of the extended cycles C_1 and C_2 corresponding to the ones in Figure 3.1 are shown.

Lemma 3.5.

- (1) $\hat{G}_{C_1} \setminus \{(D', D)\}$ is an acyclic digraph.
- (2) Every cycle in \hat{G}_{C_1} with the leaf position matching property corresponds to a C1 matrix without interleaved motion and vice versa.

We now establish that the constrained decomposition time problem is equivalent to a constrained network flow problem with simpler constraints than in the first model.

Theorem 3.6. *The decomposition time problem is equivalent to the network flow problem*

$$\min x_{(D',D)}$$

subject to x being a circulation in \hat{G}_{C1} satisfying the lower and upper capacity limits \underline{u} and \bar{u} defined by (3.13) - (3.14), and satisfying

$$x_{((i,l,r)^1,(i,l))} = x_{((i,r-1),(i,l,r)^2)} \quad \forall (i,l,r) \in \mathcal{N}_{C1}. \quad (3.15)$$

(We say that a circulation x in \hat{G}_{C1} is leaf position matched if x satisfies (3.15).)

For the proof of this theorem and algorithmic approaches to solve the second alternative network flow formulation of the decomposition time problem we refer to Boland, Hamacher, and Lenzen (2004). It is important to note, that the flow problem can be solved by iteratively choosing cycles with the leaf position matching property, thus proving that the constrained decomposition time problem always has an integer solution. While this idea is important to establish integer solvability, it is, however, not known whether there exists a (strongly) polynomial algorithm to solve the constraint decomposition time problem based on the network flow approach – or any other idea.

Exercise 3.7. (=Assignment 4) *Set up the two network flow formulations for finding the minimal constraint decomposition time for*

$$A = \begin{pmatrix} 2 & 4 & 7 & 0 & 1 \\ 1 & 0 & 3 & 5 & 0 \\ 2 & 0 & 1 & 3 & 2 \end{pmatrix}$$

Solve the constraint DT using these networks by inspection. Try in particular a cycle canceling approach in the second network, i.e. find iteratively improving cycles which maintain the edge matching property. Any suggestion for combinatorial algorithms to solve the constrained network flow problems (both in the first and the second network are welcome)!

3.2 Decomposition Cardinality is NP-hard

While the decomposition time problem is solvable in linear time, the (unconstrained) decomposition cardinality problem $\min\{DC(\alpha) : A = \sum_{k \in \mathcal{K}} \alpha_k Y^k\}$ turns out to be NP-hard. This was proved by Burkard (2002) for matrices with at least two rows using a reduction from subset sum. In the following we will strengthen his result using the result from

Baatar, D., Ehrgott, M., Hamacher, H.W. and Woeginger, G.: "Decomposition of Integer Matrices and Multileaf Collimator Sequencing", Discrete Applied Mathematics, 152 (2005) 6 - 34.

Theorem 3.8. *The C1 decomposition cardinality problem is strongly NP-hard, even for matrices with a single row.*

Proof: The decision version of the C1 decomposition cardinality problem is as follows:

C1 Decomposition-Cardinality (DC)

Input: Matrix $A = (a_1, \dots, a_N)$, $K \in \mathbb{N}$

Output: Does there exist a decomposition α of A into at most K C1 (row) matrices?

We reduce the following well-known strongly NP-complete problem (see Garey and Johnson (1979)) to (DC).

Three Partitioning (3-PART)

Input: $B, Q \in \mathbb{N}$; $b_1, \dots, b_{3Q} \in \mathbb{N}$ with $\sum_{j=1}^{3Q} b_j = QB$ and $\frac{B}{4} < b_j < \frac{B}{2}$

Output: Does there exist a partitioning of $\{b_1, \dots, b_{3Q}\}$ into triples T_1, \dots, T_Q such that $\sum_{b \in T_q} b = B$ for all $q = 1, \dots, Q$?

Given an instance of 3-PART we define the following instance of DC

$$\begin{aligned} N &:= 4Q, \\ a_n &:= \begin{cases} \sum_{j=1}^n b_j, & \text{if } n \leq 3Q \\ (4Q - n + 1)B, & \text{if } n > 3Q, \end{cases} \\ K &:= 3Q. \end{aligned} \tag{3.16}$$

Claim: DC has YES output \iff 3-PART has YES output.

“ \Leftarrow ” For $j = 1, \dots, 3Q$ let $q \in \{1, \dots, Q\}$ be such that $b_j \in T_q$. A feasible output for DC is given by intervals $[j, 3Q + q + 1)$, $j = 1, \dots, 3Q$ and $\alpha_j = b_j$ (see Figures 3.4 and 3.5).

“ \Rightarrow ” By the definition of a_n , A cannot have a decomposition with cardinality smaller than $3Q$ since $b_j > 0$, $j = 1, \dots, 3Q$ and thus $a_{n+1} > a_n$, $n = 1, \dots, 3Q - 1$. Hence we have exactly $K = 3Q$ C1 matrices in the decomposition of A . Consider a solution of DC given by intervals $I_q = [l_q, r_q)$ and coefficients α_q , $q = 1, \dots, 3Q$, such that the sum of the interval lengths is maximized. We derive the following properties.

1. For all $p, q \in \{1, \dots, 3Q\}$ $l_q \neq r_p$. Otherwise we can replace I_p and I_q by $I'_p := I_p \cup I_q$ with $\alpha'_p := \min\{\alpha_p, \alpha_q\}$ and $I'_q := I_q$ with $\alpha'_q := \alpha_q - \alpha'_p$ to get a C1 decomposition with larger interval lengths.
2. Without loss of generality $l_q = q$ for all $q = 1, \dots, 3Q$. This follows since $a_1 < a_2 < \dots < a_{3Q}$ and some interval has to start in q .
3. $r_q > 3Q$ for all $q = 1, \dots, 3Q$. Otherwise, we have a contradiction to 1 and 2 with $l_p = r_q$ for some $p = 1, \dots, 3Q$.
4. $r_q \neq 3Q + 1$ for all $q = 1, \dots, 3Q$. Otherwise some $l_q = 3Q + 1$ would be needed since $a_{3Q} = a_{3Q+1}$. This would contradict 2.

Hence all intervals end in the set $\{3Q + 2, \dots, 4Q + 1\}$. Define triples T_1, \dots, T_Q by

$$b_j \in T_q \Leftrightarrow r_j = 3Q + q + 1.$$

By definition of a_{3Q+j} , the sum of the $b_j \in T_q$ equals B . This is obviously true for $j = Q$, since $a_{3Q+Q} = a_{4Q} = B$. For $j = Q - 1, \dots, 1$ this follows by an inductive argument.

□

$(b_1, \dots, b_{3Q}) = (9, 11, 10, 11, 9, 12, 14, 9, 11)$		
$A = (9, 20, 30, 41, 50, 62, 76, 85, 96, 96, 64, 32)$		
$T_1 = \{b_1, b_2, b_6\}$	$Y^1 = Y([1, 11))$	$\alpha_1 = 9$
	$Y^2 = Y([2, 11))$	$\alpha_2 = 11$
	$Y^3 = Y([6, 11))$	$\alpha_3 = 12$
$T_2 = \{b_3, b_4, b_9\}$	$Y^4 = Y([3, 12))$	$\alpha_4 = 10$
	$Y^5 = Y([4, 12))$	$\alpha_5 = 11$
	$Y^6 = Y([9, 12))$	$\alpha_6 = 11$
$T_3 = \{b_5, b_7, b_8\}$	$Y^7 = Y([5, 13))$	$\alpha_7 = 9$
	$Y^8 = Y([7, 13))$	$\alpha_8 = 14$
	$Y^9 = Y([8, 13))$	$\alpha_9 = 9$

Figure 3.4: $3\text{-PART} \times DC$ where $B = 32, Q = 3, N = 12, K = 9$, and A are computed according to (3.16).

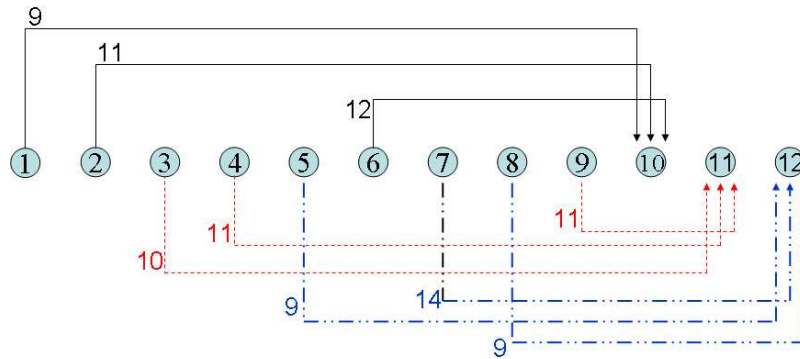


Figure 3.5: $C1$ Decomposition defined by the output of 3-Part in Figure 3.2

An obvious consequence of the preceding result is that also the constraint decomposition time problem is NP hard. Currently, the best way to solve

the DC problem is based on Kalinowski (2005). He shows, how to solve the following problem in polynomial time: Find among all optimal solutions of the constrained decomposition time problem one, which has the smallest decomposition cardinality. We know from Example 1.7 that this approach does not provide an optimal solution for DC. Nußbaum (2006) used in this diploma thesis Kalinowski's idea and solved the DC problem for varying decomposition times starting from minimal DT and increasing by units of 1. The best of these results is an optimal solution of DC.

In some cases, DC can, however, be solved in polynomial time. We have seen one example in Corollary 2.8. The next results describes a more general situation.

Proposition 3.9. *If A is a positive integer multiple of a binary matrix then the C1 unconstrained decomposition cardinality problem can be solved in polynomial time.*

Proof: For binary matrices, this result was stated in Corollary 2.8.

Let A be an integer multiple of a binary matrix B , i.e., $A = pB$. Then from any decomposition of B , multiplying by p , we get a decomposition of A with the same cardinality. Therefore, if B yields an optimal solution of the DC problem for A then the decomposition algorithm for minimizing $DT(\alpha) = DC(\alpha)$ for B finds in polynomial time a decomposition of B and consequently a decomposition of A . We complete the proof by showing that for any decomposition of A there exists a decomposition of B with the same or a smaller cardinality. Consider any decomposition $A = \sum_{k=1}^K \alpha_k Y^k$. Observe that, if $\alpha_k = p$ for all $k = 1, \dots, K$ then we have a decomposition of B with the same cardinality. If not, we can assume without loss of generality that

$$A = \sum_{k=1}^{k_0} \alpha_k Y^k + p \sum_{k=k_0+1}^K Y^k$$

where $\alpha_k < p$ for all $k = 1, \dots, k_0$. Let A' and B' be matrices defined as

$$\begin{aligned} A' &:= A - p \sum_{k=k_0+1}^K Y^k = \sum_{k=1}^{k_0} \alpha_k Y^k \\ B' &:= B - \sum_{k=k_0+1}^K Y^k. \end{aligned}$$

Then $A' = pB'$ since B is binary and $A = pB$. Consider any optimal DT decomposition of B'

$$B' := \sum_{k=1}^{k_1} \bar{Y}^k.$$

Note that for B' , $DT = DC = k_1$. Since in the unconstrained case the decomposition time of $A' = pB'$ is by equation (2.6) p times the decomposition time of

B , we get $pk_1 \leq \sum_{k=1}^{k_0} \alpha_k$, which implies that $k_1 < k_0$ since by our assumption $\alpha_k < p$ for all $k = 1, \dots, k_0$. Therefore, the decomposition of B

$$B = \sum_{k=k_0+1}^K Y^k + \sum_{k=1}^{k_1} \bar{Y}^k.$$

has smaller cardinality than the decomposition of A . \square

3.3 Multicriteria Optimization and the Box Method

Reference:

Hamacher, H.W., Pedersen, C.R., and Ruzika, S.: "Multiple Objective Minimum Cost Flow Problems: A Review", *European Journal of Operational Research* 176 (2007), 1404-1422.

Motivated by the contradictory nature of the decomposition objective functions shown in Example 1.7, we will briefly introduce basic concepts and sketch some solution approaches of multicriteria optimization. Note that these methods are also crucial in tackling the intensity problem in cancer radiation modeling (see Hamacher and Küfer (2002)).

Definition 3.10. A multiple objective linear program (MOLP) is given by

$$\begin{aligned} \min \quad & \begin{pmatrix} f^1(x) \\ \dots \\ f^p(x) \end{pmatrix} = Cx \\ \text{s.t.} \quad & x \in \mathcal{X} \end{aligned} \tag{MOLP}$$

where $f^i(x) = c^i x, i = 1, \dots, p$, i.e., $C = (c^1, \dots, c^p)^T$ with rows c^1, \dots, c^p is a $p \times m$ linear objective matrix, $x \in \mathbb{R}^m$ the vector of variables, and \mathcal{X} the set of feasible solutions, subsequently called the decision space. For MOLPs, this set is a polyhedron, i.e.,

$$\mathcal{X} := \mathcal{P} := \{x \in \mathbb{R}^m : Ax = b, x \geq 0\}$$

where A is an $n \times m$ constraint matrix and $b \in \mathbb{R}^n$ the right-hand side vector. We refer to

$$\mathcal{Y} := C\mathcal{X} := \{y := Cx \in \mathbb{R}^p : x \in \mathcal{X}\}.$$

as the objective space. By adding integrality requirements to the variables in \mathcal{X} we obtain a multiple objective linear integer program (MOLIP).

Example 3.11. We consider a MOLP with two objectives (bicriteria linear program) and two variables.

$$\begin{pmatrix} 3x_1 & + & x_2 \\ -x_1 & - & 2x_2 \end{pmatrix}$$

subject to

$$\begin{aligned} & x_2 \leq 3 \\ 3x_1 - x_2 & \leq 6 \\ x_1, x_2 & \geq 0 \end{aligned}$$

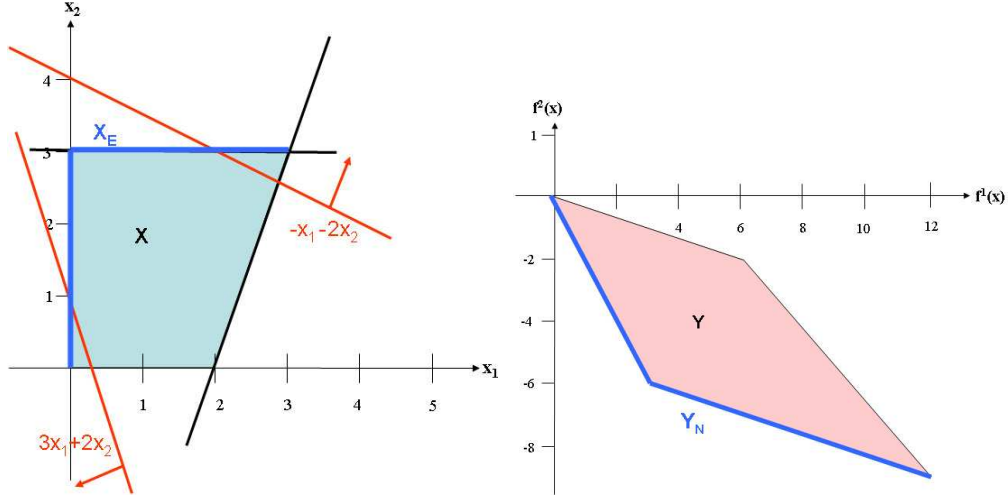


Figure 3.6: MOLP: Decision space with two objectives and objective space

For both MOLP and MOLIP we assume that the objective functions are conflicting, that is, we exclude the existence of an *ideal solution* $x \in \mathcal{X}$ which minimizes all p objectives simultaneously.

Minimizing a vector-valued objective function needs some more explanation since there is no canonical ordering defined in \mathbb{R}^p for $p \geq 2$. Throughout this article we use the Pareto concept of optimality for MOLPs and MOLIPs which is based on the following two binary relations $<$ and \leq . Let $y^1, y^2 \in \mathbb{R}^p$. Then

$$\begin{aligned} y^1 \leq y^2 & \Leftrightarrow y_k^1 \leq y_k^2 \quad \forall k = 1, \dots, p \\ y^1 \leq y^2 & :\Leftrightarrow y_k^1 \leq y_k^2 \quad \forall k = 1, \dots, p \quad \text{and} \quad y^1 \neq y^2 \\ y^1 < y^2 & :\Leftrightarrow y_k^1 < y_k^2 \quad \forall k = 1, \dots, p. \end{aligned}$$

Definition 3.12. A point $y^2 \in \mathbb{R}^p$ is called dominated by $y^1 \in \mathbb{R}^p$ if $y^1 \leq y^2$. The Pareto cone is defined as $\mathbb{R}_{\geq}^p := \{y \in \mathbb{R}^p : y \geq 0\}$.

The efficient or Pareto set, \mathcal{X}_E , and the weakly efficient or weakly Pareto set, \mathcal{X}_{wE} , are defined as

$$\begin{aligned} \mathcal{X}_E & := \{x \in \mathcal{X} : \text{there exists no } \bar{x} \in \mathcal{X} : C\bar{x} \leq Cx\} \\ \mathcal{X}_{wE} & := \{x \in \mathcal{X} : \text{there exists no } \bar{x} \in \mathcal{X} : C\bar{x} < Cx\}. \end{aligned}$$

The images of these sets under the vector-valued linear mapping C

$$\mathcal{Y}_N := C\mathcal{X}_E \quad \text{and} \quad \mathcal{Y}_{wN} := C\mathcal{X}_{wE}$$

are called the nondominated set and the weakly nondominated set, respectively.

In Figure 3.7 examples of nondominated and dominated objective vectors are indicated by crosses and dots, respectively. Following the terminology of Steuer (1986) we use the denotation

$$Z^{\geq} := \text{conv}(\mathcal{Y}_N) + \mathbb{R}_{\leq}^p$$

(the shaded grey area of Figure 3.7) where conv is the convex hull operator.

Definition 3.13. For MOLP and MOLIP the efficient frontier is defined as the set $\{y \in \text{conv}(\mathcal{Y}_N) : \text{conv}(\mathcal{Y}_N) \cap (y + (-\mathbb{R}_{\leq}^p)) = y\}$.

For MOLP the efficient frontier is identical with the set \mathcal{Y}_N . In the case of $p = 2$ objectives, the efficient frontier of MOLP is known to be piecewise linear and convex.

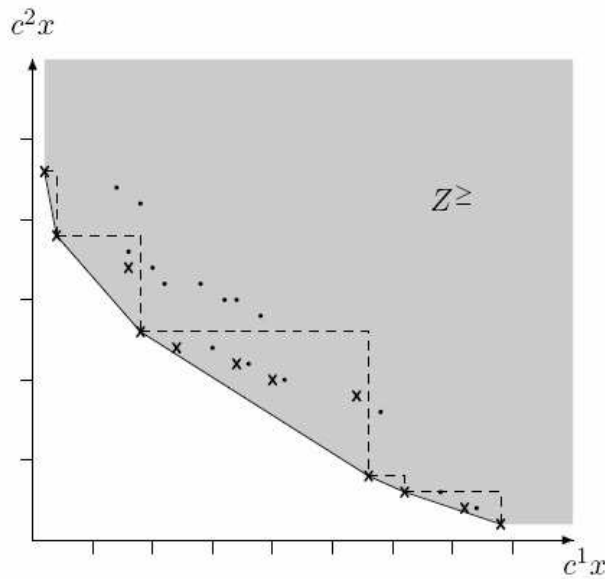


Figure 3.7: Z^{\geq} for a biobjective linear integer program with nondominated (cross) and dominated (dot) objective vectors. Unsupported nondominated objective vectors are located in its interior.

Definition 3.14. The breakpoints of the efficient frontier are the extreme nondominated points their preimages in the decision space are the extreme efficient solutions. For MOLIP the efficient frontier is the nondominated set of its continuous relaxation. If a nondominated objective vector is on the efficient frontier it is called a supported nondominated objective vector. Otherwise it is an unsupported nondominated objective vector. We shall denote the corresponding solutions in decision space supported efficient solutions and unsupported efficient solutions, respectively.

It is important to notice that for MOLP only supported efficient solutions exist, whereas for MOLIP unsupported efficient solutions may exist even if the constraint matrix for the considered MOLIP instance is totally unimodular.

Exercise 3.15. (=Assignment 5) Design and draw an example of MOLIP (similar to Example 3.11) with two variables which has unsupported, nondominated objective vectors.

It is well known, that each supported efficient solution for MOLP can be found as an optimal solution to the *weighted sum problem*, $\min \{\lambda Cx : x \in \mathcal{X}\}$, for a certain $\lambda \in]0, 1[^p$ (Geoffrion Geoffrion (1968) and Isermann Isermann (1974)). This is also true for the extreme efficient solutions. Unfortunately, the problem of finding all of these solutions is *intractable*, i.e. the number is, in general, exponential in the size of the problem. We therefore suggest to compute a *representative set* instead which should satisfy certain requirements.

We restrict ourselves to bicriteria problems to discuss the computation of such representative sets. For continuous multicriteria problems they can be computed, for instance, by the *sandwich algorithm* Burkard et al. (1989). In the framework of this text, we will, however, discuss a method which applies to integer or discrete optimization problems, the box method.

3.3.1 Finding Representative Sets by Box Method

Reference: Hamacher, H.W., Pedersen, C.R., and Ruzika, S.: "Finding Representative Systems for Discrete Bicriteria Optimization Problems by Box Algorithms", Operations Research Letters (in print).

The representative set should satisfy certain quality attributes including the number of solutions in the set (which should be small enough to be handled in a data base by a decision maker) and accuracy (which measures the closeness of any given non-dominated solution by some representing solution)

The representation computed by the *box method* consists of a collection of nondominated points, i.e. $Rep \subseteq \mathcal{Y}_N$, where each point is associated with a rectangle (or *box*) and represents all nondominated points within this box. The geometrical features of the rectangles can thus be used to measure the preceding quality attributes.

At any stage of the algorithm, the collection of boxes contains the complete nondominated set. Hence, the rectangles bound the nondominated set, and thus also show regions containing no nondominated points. Representing points are generated with a modification of the ϵ -constraint method. This method is capable of generating any nondominated point with appropriate choices of parameters. Furthermore, optimal solutions of the modified ϵ -constraint problem are known to be nondominated.

The lexicographic ϵ -constraint method

The scalarization that we will use intensively in the remainder of this paper is a lexicographic variant of the well-known ϵ -constraint scalarization (see, e.g., Chankong and Haimes (1983)). We refer to the following mathematical program

as the *lexicographical ε -constraint problem*.

$$\begin{aligned} \text{lex min } & \begin{pmatrix} f_2(x) \\ f_1(x) \end{pmatrix} \\ \text{s.t. } & f_1(x) \leq \varepsilon \\ & x \in \mathcal{X}. \end{aligned} \tag{P_\varepsilon}$$

In P_ε the vector $(f_2(x), f_1(x))^T$ is lexicographically minimized, and its feasible set is the feasible set of the bicriterion optimization problem \mathcal{X} with an additional constraint bounding the f_1 -values from above. Since the lexicographical ordering is complete (i.e., any two points are comparable) a lexicographical objective function is as easy to optimize as a single criterion objective.

The lexicographical ε -constraint problem has, however, several desirable properties which are very useful in constructing a representation of the nondominated set. In particular, the next result shows that the lexicographic objective function yields (strongly) efficient solutions - opposed to weakly efficient ones in the classical ε -constraint approach.

Proposition 3.16. *Let $\hat{x} \in \mathcal{X}$ be an optimal solution of P_ε . Then $\hat{x} \in \mathcal{X}_E$.*

Proof:

Let $x' \in \mathcal{X}$ with $f(x') \leq f(\hat{x})$. In particular, $f_1(x') \leq f_1(\hat{x}) \leq \varepsilon$, such that x' is feasible for P_ε . The definition of the lexicographical ordering and the optimality of \hat{x} for P_ε excludes that $f(x') \leq f(\hat{x})$, hence $\hat{x} \in \mathcal{X}_E$. \square

Conversely, any efficient solution can be obtained with the lexicographic ε -constraint scalarization with an adequate choice of ε .

Proposition 3.17. *Let $\hat{x} \in \mathcal{X}_E$. Then $\varepsilon := f_1(\hat{x})$, yields \hat{x} as an optimal solution for P_ε .*

Proof:

The proof follows directly from the same result of the classical ε -constraint scalarization, see, e.g., Chankong and Haimes (1983). \square

In particular, the one-to-one correspondence between solutions of P_ε and \mathcal{X}_E implies that we get supported as well as unsupported Pareto solutions. This is not achievable by the weighted sum method which is by far the most utilized scalarization in literature.

An additional advantage of using the lexicographic ε -constraint method is the fact that we maintain the original objective functions. This is an appealing feature, especially if the original objective functions possess structural properties like linearity or convexity. To the best of our knowledge, all scalarization techniques (for a list of the most common scalarizations we refer to Chapter 17 in Figueira et al. (2005)) which preserve linearity of the objective functions and which are capable of generating all nondominated points for discrete problems, modify the constraint set by at least one constraint. Therefore, the additional

constraint induced by the lexicographic ε -constraint method can be viewed as smallest possible modification.

Initialization and update of box method

Before explaining the details, we outline the major ideas of the box algorithm. Initially, we compute the two lexicographical optimal solutions. These points determine a rectangle (the *starting box*) containing the complete nondominated set. In the following we discard rectangular pieces of the starting box based on additional information obtained by iteratively solving P_ε with adequately chosen values for ε . This yields a collection of rectangles or *boxes* containing the nondominated set in each stage of the algorithm. Furthermore, for each box, we know a nondominated point representing the set of nondominated points inside the box. The box algorithm stops based on a predetermined accuracy, which is measured by the biggest of the remaining boxes.

More detailed, the box algorithm works as follows. Initially, we determine both lexicographic minima of the bicriterion optimization problem

$$z^1 := \begin{pmatrix} z_1^1 \\ z_2^1 \end{pmatrix} := \text{lex} \min_{x \in \mathcal{X}} \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix} \text{ and } z^2 := \begin{pmatrix} z_1^2 \\ z_2^2 \end{pmatrix} := \text{lex} \min_{x \in \mathcal{X}} \begin{pmatrix} f_2(x) \\ f_1(x) \end{pmatrix}.$$

Obviously, \mathcal{Y}_N is a subset of $R := R(z^1, z^2)$, the rectangle with upper left and lower right vertex z^1 and z^2 , respectively. This is the *starting box* of our algorithms. In general, we will use in the following the notation $R(y^1, y^2)$ to denote the rectangle having $y^1 = \begin{pmatrix} y_1^1 \\ y_2^1 \end{pmatrix} \in \mathbb{Z}^2$ as upper left and $y^2 = \begin{pmatrix} y_1^2 \\ y_2^2 \end{pmatrix} \in \mathbb{Z}^2$ as lower right vertex. Moreover, let $a(R(y^1, y^2)) := (y_1^2 - y_1^1) \cdot (y_2^1 - y_2^2)$ denote the area of the rectangle $R(y^1, y^2)$.

Initially, the starting box $R(z^1, z^2)$ contains the complete nondominated set \mathcal{Y}_N . Whenever additional nondominated points become known during the execution of the algorithm, one of the boxes will be split up into several smaller rectangles – while maintaining the inclusion property of \mathcal{Y}_N . We will stop when the area of the largest of these boxes is smaller than some given accuracy $\Delta > 0$. We refer to such a collection of boxes as a Δ -*representation* of \mathcal{Y}_N . The lower right corner point of each of the rectangles is a *representing point* and will be added to the representing system.

In the following, we consider a box with $a(R(y^1, y^2)) > \Delta$, such that the representation has to be locally updated in $R(y^1, y^2)$. Consider P_ε with $\varepsilon := \lfloor \frac{y_1^1 + y_1^2}{2} \rfloor$. Let $x^* \in \mathcal{X}$ be optimal for P_ε and let $z^* := (f_1(x^*), f_2(x^*)) = f(x^*)$. Due to Theorem 3.16, z^* is nondominated. Using the point z^* and ε , we divide $R(y^1, y^2)$ into five rectangles

$$R_1 := R(y^1, z^*) \quad R_2 := R(p^1, p^4) \quad R_3 := R(p^2, p^6) \quad R_4 := R(p^3, p^7) \quad R_5 := R(p^5, y^2)$$

(see Figure 3.8), where

$$p^1 := \begin{pmatrix} z_1^* \\ y_2^1 \end{pmatrix}, \quad p^2 := \begin{pmatrix} \varepsilon + 1 \\ y_2^1 \end{pmatrix}, \quad p^3 := \begin{pmatrix} y_1^1 \\ z_2^* \end{pmatrix}, \quad p^4 := \begin{pmatrix} \varepsilon \\ z_2^* \end{pmatrix},$$

$$p^5 := \begin{pmatrix} \varepsilon + 1 \\ z_2^* - 1 \end{pmatrix}, \quad p^6 := \begin{pmatrix} y_1^2 \\ z_2^* \end{pmatrix} \quad \text{and} \quad p^7 := \begin{pmatrix} \varepsilon \\ y_2^2 \end{pmatrix}.$$

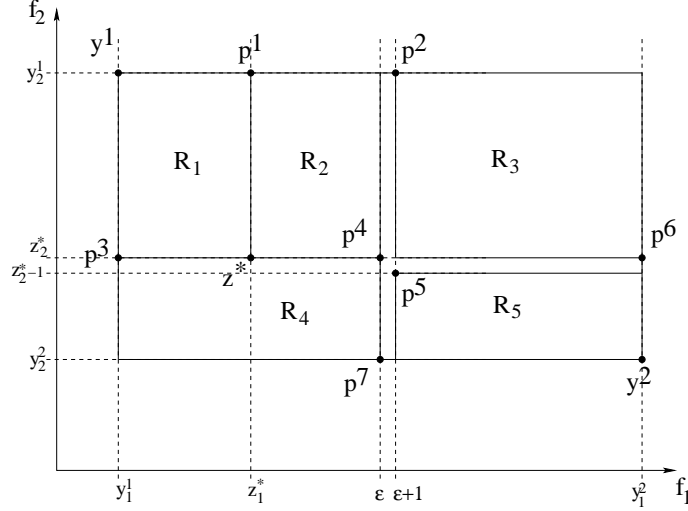


Figure 3.8: *Updating of a rectangle $R(y^1, y^2)$.*

Note that the points p^2 and p^5 have first coordinate one unit to the right of the ε -constraint. Furthermore, the point p^5 has second coordinate one smaller than the second objective value of z^* . This is due to the fact that no feasible nondominated point can have non-integral coordinates. In general, $p^i \notin \mathcal{Y}$, $i = 1, \dots, 7$, i.e., the points p^i are not necessarily feasible criterion vectors. The rectangles R_1, R_2 , and R_4 can degenerate to a line, in which case they are considered to have zero area.

Proposition 3.18. *R_2, R_3 , and R_4 can be eliminated, since*

- a) $(R_2 \cup R_3) \cap \mathcal{Y}_N \subseteq \{y^1, z^*\}$, and
- b) $R_4 \cap \mathcal{Y}_N \subseteq \{z^*\}$.

Proof:

- a) z^* is dominating any feasible point in $R_2 \cup R_3$.
- b) Any feasible point in R_4 contradicts the optimality of z^* for P_ε . □

Due to the previous result, there does not exist nondominated points in R_2, R_3, R_4 , other than z^* (and possibly y^1). Consequently, we can formulate the following corollary.

Corollary 3.19.

$$\mathcal{Y}_N \cap R(y^1, y^2) \subseteq R_1 \cup R_5$$

It is easy to see that the following result, comparing the size of rectangles R_1 and R_5 to the size of rectangle $R(y^1, y^2)$, holds true.

Proposition 3.20.

$$a(R_1) + a(R_5) \leq \frac{1}{2}a(R(y^1, y^2))$$

After having obtained z^* as a new nondominated point, the representation is updated associating z^* with R_1 and y^2 with R_5 . By Proposition 3.20 we see that computing z^* has locally improved the representation by a factor of 2.

Corollary 3.21. *After computing z^* and adding it to the representation, the representation's accuracy improved in $R(y^1, y^2)$ by a factor of 2.*

The a posteriori algorithm

In this subsection we iteratively apply the updating procedure from Section 3.3.1 to place ε -constraints in a rectangle having area bigger than Δ . We assume that Δ is given as input in our algorithm. Since Δ is the measure of the size of our boxes, we may also leave its determination to the decision maker or compute it as a given percentage of the starting box.

Since the value of the ε -constraint is chosen *after* the given rectangle is identified, this method is called an *a posteriori* algorithm. A pseudo code description is given in Algorithm 1.

Algorithm 1 A posteriori algorithm.

Input: A discrete bicriterion optimization problem, $\Delta > 0$

Output: A representation $Rep \subseteq \mathcal{Y}_N$ with accuracy Δ .

- 1: $S \leftarrow \emptyset$.
 - 2: $Rep \leftarrow \emptyset$.
 - 3: Compute the lexicographic minima z^1 and z^2 , and $a(R(z^1, z^2))$.
 - 4: $Rep \leftarrow \{z^1, z^2\}$.
 - 5: $S \leftarrow \{R(z^1, z^2)\}$.
 - 6: **while** $S \neq \emptyset$ **do**
 - 7: Choose the largest rectangle $R(y^1, y^2) \in S$.
 - 8: $S \leftarrow S \setminus \{R(y^1, y^2)\}$.
 - 9: Solve P_ε with $\varepsilon := \lfloor y_1^1 + \frac{y_1^2 - y_1^1}{2} \rfloor$ and obtain optimal solution $z^* \in \mathcal{Y}_N$.
 - 10: $Rep \leftarrow Rep \cup \{z^*\}$.
 - 11: **if** $a(R(y^1, z^*)) > \Delta$ **then**
 - 12: $S \leftarrow S \cup \{R(y^1, z^*)\}$.
 - 13: **end if**
 - 14: **if** $a(R(y^1, y^2)) > \Delta$ **then**
 - 15: $S \leftarrow S \cup \{R(y^1, y^2)\}$.
 - 16: **end if**
 - 17: **end while**
-

Note that the a posteriori algorithm is flexible in the sense that it always chooses the largest rectangle, and the representation is hence only refined where it is needed.

The following theorem establishes finiteness and the complexity of the a posteriori algorithm.

Theorem 3.22. *In finitely many steps, the a posteriori algorithm yields a Δ -representation of \mathcal{Y}_N in which all representing points are nondominated. More specifically, the algorithm performs at most $\mathcal{O}(\frac{A}{\Delta})$ many iterations, where A is the area of the starting box $R(z^1, z^2)$.*

Proof:

As long as the representation is not a Δ -representation, the algorithm processes in each iteration the largest rectangle that does not meet the accuracy of Δ . Due to Corollary 3.21, the resulting rectangle(s) have, after the update, an area which is at most half the size of the original area. The algorithm is therefore finite. Due to Theorem 3.16, all representing points are nondominated.

More specifically, the algorithm produces after $2^k - 1$ iterations no more than 2^k rectangles, each of which have an area of at most $\frac{A}{2^k}$. Since $k^* = \lceil \log_2(\frac{A}{\Delta}) \rceil$ is the smallest integer satisfying $\frac{A}{2^{k^*}} \leq \Delta$, an upper bound on the number of iterations to obtain a Δ -representation is $2^{k^*} - 1 = 2^{\lceil \log_2(\frac{A}{\Delta}) \rceil} - 1$, which is $\mathcal{O}(\frac{A}{\Delta})$. \square

A direct consequence of Theorem 3.22 is the following result.

Corollary 3.23. *The a posteriori algorithm runs in $\mathcal{O}(T_1 + \frac{a(R(y^1, y^2))}{\Delta} T_2)$ where T_1 and T_2 is the time needed to compute a lexicographic minimum and to evaluate P_ε , respectively.*

Observe that T_1 and T_2 depend on the specific problem under consideration.

Exercise 3.24. (=Assignment 6) *Develop an a priori version of the box method: Compute in advance the necessary number of iterations to obtain an accuracy Δ .*

Chapter 4

Stop Location Problem

References:

Poetranto, D.R., Hamacher, H.W., Horn, S., and Schöbel, A.: "Stop Location Design in Public Transportation Networks: Covering and Accessibility Objectives", Reports in Wirtschaftsmathematik, Technische Universität Kaiserslautern, Department of Mathematics, No. 97 (2006)

Schöbel, A., Hamacher, H.W., Liebers, A. and Wagner, D.: "The continuous stop location problem in public transportation networks", Report in Wirtschaftsmathematik, Technische Universität Kaiserslautern, Department of Mathematics, No. 81 (2002)

In *StopLoc* we consider the location of new stops along the edges of an existing public transportation network. Examples of StopLoc include the location of bus stops along some given bus routes or of railway stations along the tracks in a railway system. In order to measure the "convenience" of the location decision for potential customers in given demand facilities, two objectives are proposed. In the first one, we give an upper bound on reaching a closest station from any of the demand facilities and minimize the number of stations. In the second objective, we fix the number of new stations and minimize the sum of the distances between demand facilities and stations. The resulting two problems *CovStopLoc* and *AccessStopLoc* are solved by a reduction to a classical set covering and a restricted location problem, respectively. We implement the general ideas in two different environments - the plane, where demand facilities are represented by coordinates and in networks, where they are nodes of a graph.

4.1 Covering and Access Problems

In this section, we formally introduce the Covering and Access Stop Location Problem in planar and network environments.

Definition 4.1. A public transportation network (PTN) is an undirected, connected graph $G = (V, E)$ which has an embedding in the plane. The edges $e = (v_i, v_j) \in E$ between nodes v_i and v_j are part of public transportation lines and are assumed to be straight lines in the plane. The nodes can be existing

stations V_{stop} , or breakpoints V_{break} , i.e.

$$V = V_{stop} \cup V_{break}. \quad (4.1)$$

Each edge e has an associated length l_e which is a real number representing physical length or travel time. A point $s \in G$ of the graph $G = (V, E)$ is either a node or strictly contained in an edge $e = (v_i, v_j)$. We use the denotation $s = (e, t)$ where $0 \leq t \leq 1$ and where the distance between s and v_i is tl_e and between s and v_j is $(1 - t)l_e$.

The potential users of the public transportation system are represented by a finite set $\mathcal{P} = \{p_1, \dots, p_M\}$ of demand facilities. For each demand facility $p \in \mathcal{P}$ and point $s \in G$ we assume that a distance $d(p, s) \in \mathbb{R}$ is known. If $\mathcal{S} \subseteq G$ is a subset of points, then

$$d(p, \mathcal{S}) := \min_{s \in \mathcal{S}} d(p, s). \quad (4.2)$$

In this paper, we consider two versions of stop location problems.

In the *Covering Stop Location Problem (CovStopLoc)* a real number r is given as *radius*. It is assumed that customers will not use the public transportation system if the distance to a station is larger than r . We want to find a set of locations for stations, which is a subset $\mathcal{S} \subseteq G = (V, E)$ with minimal cardinality, such that every customer can reach at least one of the new stations with a distance of at most r . Hence we want to solve:

$$\begin{aligned} & \text{minimize } |\mathcal{S}| \\ & \text{subject to } d(p, \mathcal{S}) \leq r \quad \forall p \in \mathcal{P} \\ & \mathcal{S} \subseteq G. \end{aligned} \quad (\text{CovStopLoc})$$

In the *Accessibility Stop Location Problem (AccessStopLoc)* a positive integer k is given which is the maximal number of stations which can be built. We want to find in AccessStopLoc a subset $\mathcal{S} \subseteq G = (V, E)$ with $|\mathcal{S}| \leq k$ such that its overall distance to the demand facilities is minimized, where we measure the overall distance by the sum of the distances from \mathcal{S} to all customers. The resulting problem is:

$$\begin{aligned} & \text{minimize } f(\mathcal{S}) := \sum_{p \in \mathcal{P}} d(p, \mathcal{S}) \\ & \text{subject to } \mathcal{S} \subseteq G \\ & |\mathcal{S}| \leq k \end{aligned} \quad (\text{AccessStopLoc})$$

Both versions of the stop location problem include the distance $d(p, s)$ between demand facilities and points in the PTN in their formulations. We have two different environments, where distances are defined.

In the *planar environment* we use the given straight line planar embedding of the PTN $G = (V, E)$. Here, any point s of the graph G can be interpreted as

a point $s = (s_1, s_2)$ in the Euclidean plane \mathbb{R}^2 . If we additionally assume that for each $m = 1, \dots, M$ the demand facility $p_m = (p_{m1}, p_{m2})$ is also a point in the Euclidean plane \mathbb{R}^2 , then $d(p, s)$ can be any distance function in \mathbb{R}^2 , for instance, the rectangular distance

$$d(p, s) := l_1(p, s) := |p_1 - s_1| + |p_2 - s_2|,$$

or the Euclidean distance (see Figures 4.1 and 4.2)

$$d(p, s) := l_2(p, s) := \sqrt{(p_1 - s_1)^2 + (p_2 - s_2)^2}.$$

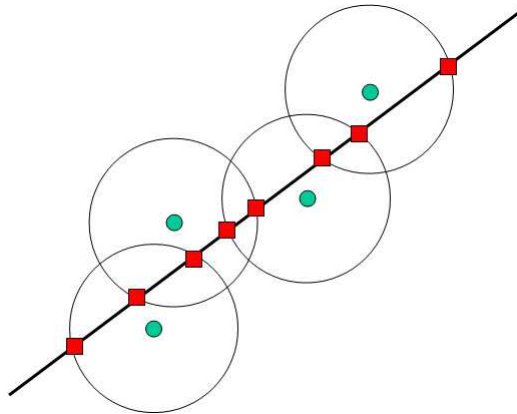


Figure 4.1: Stop location in a graph with a single line segment (full circles: demand point, full square: candidate locations)

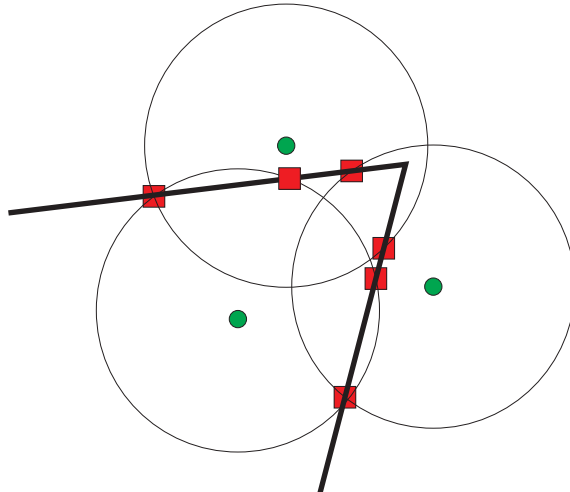


Figure 4.2: Stop location in a graph with two line segments (full circles: demand point, full square: candidate locations)

In the *network environment*, the PTN is part of a larger network, the *street and public transportation network (SPTN)* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = V \cup \mathcal{P}$ and $E \subseteq \mathcal{E}$. The edges in $\mathcal{E} \setminus E$ represent streets which can be used by potential

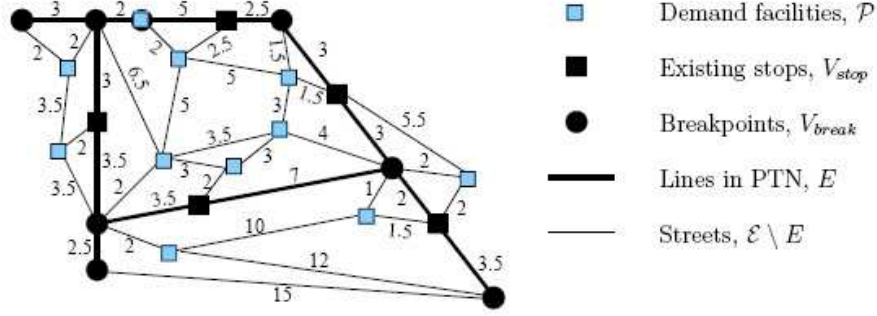


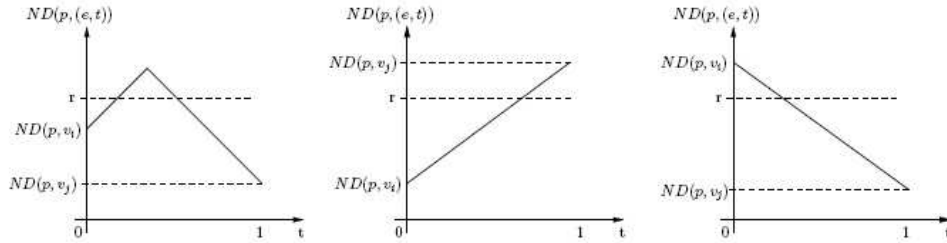
Figure 4.3: Street and Public Transportation Network.

passengers to reach one of the new stations. An example of a SPTN is given in Figure 4.3.

The distance $d(p, s)$ is the shortest path distance in \mathcal{G} , if both p and s are nodes of \mathcal{G} , i.e. $p \in \mathcal{P}$ and $s \in V$. We denote this network distance in the following with $ND(p, s)$. If $s = (e, t)$ is a point of the PTN inside the edge $e = (v_i, v_j) \in E$ (i.e., with $0 < t < 1$) and $p \in \mathcal{P}$, then

$$ND(p, s) := \min \{ND(p, v_i) + tl_e, ND(p, v_j) + (1 - t)l_e\}. \quad (4.3)$$

The distance $ND(p, s)$ as a function of t if we fix $p \in \mathcal{P}$ and move $s = (e, t)$ along an edge e is illustrated in Figure 4.4.

Figure 4.4: Visualization of the network distance $ND(p, s)$

Having defined two different versions of StopLoc and two environments, we can define four types of StopLoc. Planar CovStopLoc, Network CovStopLoc, Planar AccessStopLoc and, Network AccessStopLoc.

4.2 Covering Stop Location

In this section we present a unifying framework for stop location problems with covering objective function both for networks and for a planar environment. We assume that a *radius* $r \in \mathbb{R}$ is given.

Definition 4.2. A demand facility $p \in \mathcal{P}$ is covered by a point s in the PTN $G = (V, E)$ if $d(s, p) \leq r$. For a set \mathcal{S} of points in G we define

$$\text{cover}(\mathcal{S}) = \{p \in \mathcal{P} : d(s, p) \leq r \text{ for some } s \in \mathcal{S}\}.$$

If $\mathcal{S} = \{s\}$ is a singleton we use the denotation $\text{cover}(s)$ instead of $\text{cover}(\{s\})$. \mathcal{S} is called a cover of \mathcal{P} if all $p \in \mathcal{P}$ are covered by some $s \in \mathcal{S}$, i.e., for all $p \in \mathcal{P}$ there exists some $s \in \mathcal{S}$ such that $d(s, p) \leq r$.

The *Covering Stop Location Problem (CovStopLoc)* is thus the problem of finding a cover with smallest cardinality.

If a finite set of candidates $\mathcal{S}_{\text{cand}} \subseteq G$ of points in the PTN is allowed for new stops (we call this property the *finite dominating set, FDS, property*), the problem is the well-known *Location Covering Problem*. In this case it can be formulated as integer program

$$\min \sum_{s \in \mathcal{S}_{\text{cand}}} x_s \quad (4.4)$$

$$\text{s.t. } Ax \geq 1, \quad \forall p \in \mathcal{P} \quad (4.5)$$

$$x_s \in \{0, 1\}, \quad \forall s \in \mathcal{S}_{\text{cand}} \quad (4.6)$$

with the binary matrix $A = (a_{ps})_{\substack{p \in \mathcal{P} \\ s \in \mathcal{S}_{\text{cand}}}}$ where

$$a_{ps} = \begin{cases} 1 & \text{if } d(p, s) \leq r \\ 0 & \text{else} \end{cases}, \quad x_s = \begin{cases} 1 & \text{if } s \in \mathcal{S} \\ 0 & \text{else,} \end{cases}$$

The formulation describes a *set-covering problem*.

Example 4.3. *The cover matrices of the examples in Figures 4.1 and 4.2 are*

$$A_1 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad A_2 = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

A_1 is obviously C1, whereas A_2 is (not even weak).

In order to show that every CovStopLoc can be written as set-covering problem, we need to establish the FDS property. We do so by a unifying approach handling both the planar and the network case.

Definition 4.4. *Let*

$$D_m := \{s \in \text{PTN} : d(p_m, s) \leq r, m \in \mathcal{M}\}, \quad (4.7)$$

where d is a norm in the planar case and the network distance in the network environment. Define an equivalence relation on the set of points in G by

$$s_1 \approx s_2 \iff \text{cover}(s_1) = \text{cover}(s_2).$$

The closures of the equivalence classes form the set $\mathcal{Z} \subseteq \text{PTN}$ of cells of the PTN.

Each cell $z \in \mathcal{Z}$ is of the form $Z = \bigcap_{m \in \mathcal{M}'} D_m$ for some subset $\mathcal{M}' \subseteq \mathcal{M}$. Note that an equivalence class may consist of one or several (open, half-closed or closed) intervals which are possibly generated to single points.

Lemma 4.5. *Let $s \in PTN$ be a point in the interior of some cell $Z \in \mathcal{Z}$. Then $cover(s) \subseteq cover(\tilde{s})$ for any point \tilde{s} on the boundary of Z .*

Proof: Let $s \in Z = \bigcap_{m \in \mathcal{M}'} D_m$ for some $\mathcal{M}' \subseteq \mathcal{M}$. Then $m \in cover(s)$ for all $m \in \mathcal{M}'$. Since each $D_m, m \in \mathcal{M}'$ is a closed set we conclude that $m \in cover(\tilde{s})$ for all $m \in \mathcal{M}'$ thus proving the lemma. \square

Since the boundary of each cell $Z = \bigcap_{m \in \mathcal{M}'} D_m$ is defined by the boundary of some D_m with $m \in \mathcal{M}'$, the lemma motivates the following definition of a candidate set.

The lemma motivates the following definition of the candidate set.

Definition 4.6.

$$\mathcal{S}_{cand} := \{\delta D_m : m \in \mathcal{M}\}.$$

Theorem 4.7. *If CovStopLoc is feasible, then there exists an optimal solution $\mathcal{S} \subseteq \mathcal{S}_{cand}$.*

Proof: Let \mathcal{S} be optimal and $\mathcal{S} \not\subseteq \mathcal{S}_{cand}$. Take any $s \in \mathcal{S} \setminus \mathcal{S}_{cand}$. $s \in Z$ for some cell $Z \in \mathcal{Z}$. Due to Lemma 4.5 there exists $\tilde{s} \in \delta Z$ (i.e. $\tilde{s} \in \mathcal{S}_{cand}$) with $cover(s) \subseteq cover(\tilde{s})$. Hence,

$$\bar{\mathcal{S}} = \mathcal{S} \setminus \{s\} \cup \{\tilde{s}\}.$$

covers all demand points and satisfies $|\bar{\mathcal{S}}| \leq |\mathcal{S}|$. Moreover, it contains strictly less stops which are not in \mathcal{S}_{cand} . Iterating this process until all stops in $\mathcal{S} \setminus \mathcal{S}_{cand}$ are replaced by stops in \mathcal{S}_{cand} finishes the proof. \square

4.2.1 Planar CovStopLoc

In the planar environment, \mathcal{S}_{cand} consists of at most $\mathcal{O}(M|E|)$ candidates, and hence satisfies the FDS property with a polynomial number of candidates. Details can be found in Schöbel et al. (2002); Schöbel (2006). In general, Planar-CovStopLoc is, however, NP hard.

Theorem 4.8. *CSLP is NL hard.*

Proof: The following problem is known to be NP-complete (see D. S. Johnson: "The NP-completeness column: An ongoing guide", J. of Algorithms 3 (1982), 182-195)

Geometric covering by discs (GeoCovDisc)

Given: finite set P of integer-coordinate points in $\mathbb{R}^2, q \in \mathbb{N}$.

$K \in \mathbb{N}$ with $K \leq |P|$. Question: Can the points of P be covered by at most K discs of parameter q ?

In order to show GeoCovDisc ∞ CSLP we construct an instance of CSLP from an instance of GeoCovDisc

- $r := \frac{1}{2}q$, keep P and K

- to define E
 - add the line segment between p_1 and p_2 as an edge
 - add a sufficiently large piece of the bisector of p_1 and p_2 as another edge
- $T = U_e$ is the set of all points on edges
- V is the union of P with the set of crossing points of edges

Claim:

P can be covered by $\leq K$ discs of diameter q

$\Leftrightarrow \exists S \subseteq T$ with $|S| \leq K$ covering P .

Proof: " \Leftarrow " trivial, since any set of K covering stops corresponds to a set of discs with diameter q .

" \Rightarrow " Suppose P is covered by a collection C of at most K discs of diameter q . For any $C \in C$

- a) if C contains no point of $P \rightarrow$ disregard C
- b) if $C \cap P = \{p\} \rightarrow$ replace C by disc C' with center point p ($p \in T!$) and radius r .
- c) if $C \cap P = A$ with $|A| \geq 2 \rightarrow$ replace C by a disc C' with center point p and radius r where r is smallest possible.
 Since C covers $A \Rightarrow r \leq q$, and center point lies on intersection of at least 2 bisectors (see location theory) we get $p \in T$.
 Hier Bild
 (a), (b), (c) $\Rightarrow P$ can be covered by $\leq K$ discs of radius r , all with center points in T .

□

But there are polynomially solvable special cases.

Corollary 4.9. *If the PTN consists just of a single track, the planar CovStopLoc can be solved in polynomial time.*

Proof: If the PTN consists just of a single track, the location covering problem 4.5 has a coefficient matrix which is C1. Hence the resulting integer program can be solved in polynomial time by 2.10. □

Exercise 4.10. =Assignment 7) Solve the planar CovLocStop problems of Example 4.3 using the methods of Chapter 2.

4.2.2 Network CovStopLoc

For the case of the network environment, the candidate set is

$$\mathcal{S}_{cand} := \{s \in G : \exists p_m \in \mathcal{P} \text{ with } ND(p_m, s) = r\} \quad (4.8)$$

The following results shows that its cardinality is the same as in the planar case.

Theorem 4.11. *The candidate set \mathcal{S}_{cand} for the network case satisfies*

$$|\mathcal{S}_{cand}| \leq 2M |E|$$

Proof: Consider a demand facility $p \in \mathcal{P}$ and a point $s = (e, t)$ on edge $e \in E$. From the definition of network distance given by (4.3), it follows that if we move s along e , then $ND(p, s) = ND(p, (e, t))$ is continuous, concave, and piecewise linear in t with at most two linear pieces with slopes l_e and $-l_e$ respectively (see Figure 4.4). Thus, given $r \in \mathbb{R}$, there exist at most two points on edge e that have distance exactly r from p , and the result follows. \square

4.3 Accessibility

The *Accessibility Stop Location Problem* (*AccessStopLoc*) at the beginning of this section is

$$\begin{aligned} & \text{minimize } f(\mathcal{S}) := \sum_{p \in \mathcal{P}} d(p, \mathcal{S}) \\ & \text{subject to } \mathcal{S} \subseteq G \\ & \quad |\mathcal{S}| \leq k \end{aligned} \tag{AccessStopLoc}$$

Recall that $\mathcal{S} \subseteq G$ indicates that \mathcal{S} is a subset of the points of the PTN. Thus, *AccessStopLoc* is a *restricted k -median problem*, where the (at most) k new locations are required to be points of the PTN.

We consider two problems, namely completely redesigning all stops and opening additional stops. In the first problem, we assume that the customers will be served only by the new stops, thus we look for $\mathcal{S}^* \subseteq G$ that minimizes the objective function:

$$\min f(\mathcal{S}) = \sum_{p \in \mathcal{P}} d(p, \mathcal{S}) \tag{4.9}$$

In the second model, we assume that some stations do already exist in a subset $V_{stop} \subseteq V$ and the customers can go to both the existing and the new stops, whichever is closer. For this model, the objective function is:

$$\min f'(\mathcal{S}) = \sum_{p \in \mathcal{P}} d(p, V_{stop} \cup \mathcal{S}) \tag{4.10}$$

where $d(p, V_{stop} \cup \mathcal{S}) := \min_{s \in V_{stop} \cup \mathcal{S}} d(p, s)$ is the distance between a demand facility p and the set $V_{stop} \cup \mathcal{S}$ of existing and new stops.

4.3.1 Planar AccessStopLoc

We show in the following how to deal with these problems for the rectilinear distance

$$d(p, s) := l_1(p, s) := |p_1 - s_1| + |p_2 - s_2|.$$

The arguments can, however, also be applied to a larger class of distance function, so-called *polyhedral gauges*.

We begin with the special case $k = 1$, i.e., of establishing only a single new stop. Moreover, we first assume that all demand facilities are served by the new stop, i.e., there is so far no stop in the PTN. In this case the objective (4.9) is formulated as

$$\min f(s) = \sum_{m \in \mathcal{M}} |p_{m1} - s_1| + |p_{m2} - s_2|. \quad (4.11)$$

Definition 4.12. *The set of construction lines \mathcal{L} is the set of the $2M$ vertical and horizontal lines passing through the M demand points. The set of (rectangular) cells, denoted \mathcal{Z} is the planar arrangement of the construction lines.*

By definition of the cells, the objective function f is linear in each of these cells (see, e.g. Hamacher (1995)). We are now in the position of introducing a finite candidate set for the problem.

Definition 4.13. $\mathcal{S}_{cand} := V_{break} \cup \mathcal{C}$ where

$$\mathcal{C} := \{e \cap l : e \in E, l \in \mathcal{L}, \text{ and } |e \cap l| = 1\}.$$

The candidate set \mathcal{S}_{cand} is illustrated in Figure 4.5.

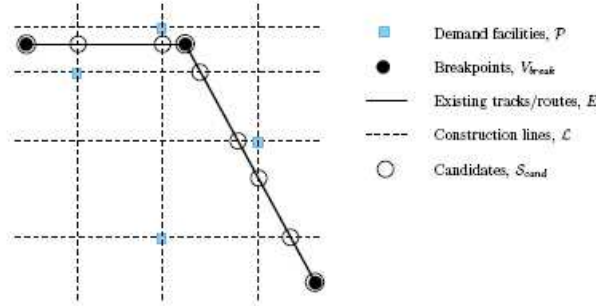


Figure 4.5: Candidates for AccessStopLocation in planar environment

Theorem 4.14. \mathcal{S}_{cand} is an FDS for AccessStopLoc in the case of a single stop, i.e., there exists an optimal solution $s^* \in \mathcal{S}_{cand}$.

Proof: From (4.11) we know that the objective function f is linear on each rectangular cell $Z \in \mathcal{Z}$. Recall from Section 4.1 that the PTN has a planar representation, where each edge $e \in E$ is a line segment in \mathbb{R}^2 . Thus $e \cap Z$ is either empty or a line segment, and f is linear on $e \cap Z$. Hence, the linear programming problem

$$\min\{f(s) : s \in e \cap Z\}$$

is either infeasible or has an optimal solution at an endpoint of the segment $e \cap Z$. An endpoint of $e \cap Z$ is either a breakpoint, hence in V_{break} , or an intersection point $e \cap l$ for some construction line l not containing e , hence in \mathcal{C} . \square

Note that \mathcal{C} is contained in the set of intersection points $\mathcal{S} \cap \mathcal{L}$ between the tracks and the construction lines, but neglecting cases where a construction line l coincides with a linear segment e of the tracks. Such a situation is illustrated in Figure 4.6. As the proof of Theorem 4.14 shows, the intersection $l \cap e$ need not be considered in this case.

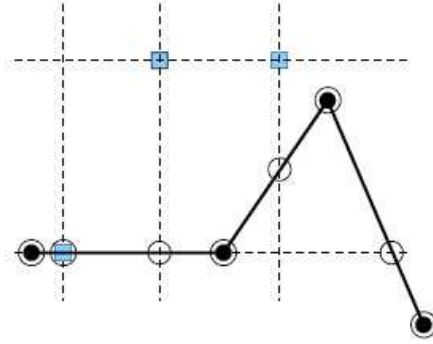


Figure 4.6: Candidates for AccessStopLoc where some edge of the PTN coincides with a segment of some construction line.

The candidates s_1, \dots, s_L can be determined in $\mathcal{O}(|V| + M|E|)$. The effort to compute the objective value for each candidate is $\mathcal{O}(M)$, thus the overall complexity of the algorithm is $\mathcal{O}(M^2|E|)$.

Now we remove the condition that all demand facilities have to be served by the new stop, i.e., we consider a subset $V_{stop} \neq \emptyset$. To distinguish this problem from the former one, we denote it by AccessStopLoc'; its objective function is presented in (4.10). For each demand facility $p_m, m \in \mathcal{M}$, we determine an existing stop $v_m \in V_{stop}$ that is closest to p_m .

$$v_m \in \operatorname{argmin}_{v \in V_{stop}} l_1(p_m, v)$$

$$d_m := l_1(p_m, v_m)$$

For $s \in \mathbb{R}^2$, we have to decide whether p_m is closer to s than to the current closest stop v_m . This is easily done using the (diamond-shaped) circles

$$D_m := \{s \in \mathbb{R}^2 : l_1(p_m, s) \leq d_m\}, m \in \mathcal{M}. \quad (4.12)$$

with center p_m and radius d_m . Customers in demand facility p_m will prefer a new stop in s to the previous stops if and only if s lies inside this circle. Therefore, we only need to consider points that lie in the interior of the set

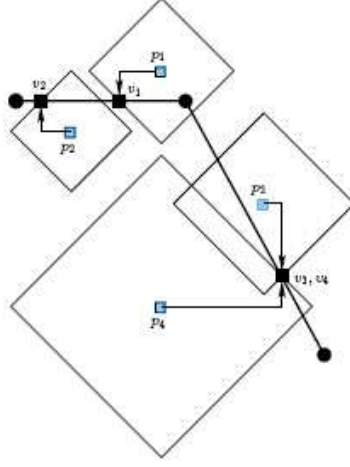


Figure 4.7: Graphical solution idea.

$$D := \bigcup_{m \in \mathcal{M}} D_m, \quad (4.13)$$

i.e., in the interior of at least one circle (see Figure 4.7). We have thus shown.

Lemma 4.15.

- $f'(s) = f'(\emptyset) = \sum_{m \in \mathcal{M}} d_m =: d \quad \forall s \in V_{stop}$ and for all $s \notin \text{int}(D)$.
- $f'(s) < d \Leftrightarrow s \in \text{int}(D)$.

This means that candidates outside $\text{int}(D)$ can never be optimal and need not be considered. Now consider an intersection of some of the sets D_m , i.e. let $\tilde{\mathcal{M}} \subseteq \mathcal{M}$ and $\tilde{D} = \bigcap_{m \in \tilde{\mathcal{M}}} D_m$. For $s \in \tilde{D}$ we obtain

$$f'(s) = \sum_{m \in \tilde{\mathcal{M}}} w_m d(p_m, s) + \sum_{m \in \mathcal{M} \setminus \tilde{\mathcal{M}}} d_m$$

The first part coincides with the objective function of a problem of type AccessStopLoc, but with existing demand points $\tilde{\mathcal{M}}$ instead of \mathcal{M} , while the second part of F is constant on \tilde{D} . Hence, the objective function is (affin) linear on

$$D_m \cap \tilde{D} \quad (4.14)$$

for $m \in \mathcal{M}$.

Definition 4.16. *The set of construction lines is $\mathcal{L}' := \{l \cap \text{int}(D) : l \in \mathcal{L}\}$. The set of cells, denoted \mathcal{Z}' , is the planar arrangement of \mathcal{L}' and the boundaries δD_m of all sets $D_m, m \in \mathcal{M}$.*

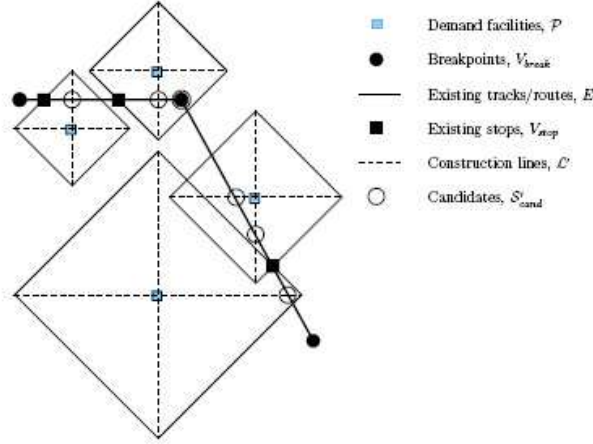


Figure 4.8: Candidates for AccessStopLoc' in planar environment.

The construction lines $l \in \mathcal{L}'$ decompose each set D_m into four quadrants (see Figure 4.8).

Theorem 4.17. *An optimal solution for AccessStopLoc' is contained in the candidate set $\mathcal{S}'_{cand} := \mathcal{S}_{cand} \cap \text{int}(D)$.*

Proof: Using the same arguments as in the proof of Theorem 4.14, the linearity of f' in each cell ensures that the minimum in each of these cells is attained in an end point. Since by definition, the distance to all $p_m \in \mathcal{P}$ defining this cell is strictly decreasing in the inside (while leaving the distances to all other $p_m \in \mathcal{P}$ unchanged), none of the points $\tilde{s} \in G \cap \partial D_{\tilde{m}} \setminus \mathcal{S}_{cand}$ can be optimal (see Figure 4.9). \square

As in AccessStopLoc, the candidate set \mathcal{S}'_{cand} is finite and consists of at most

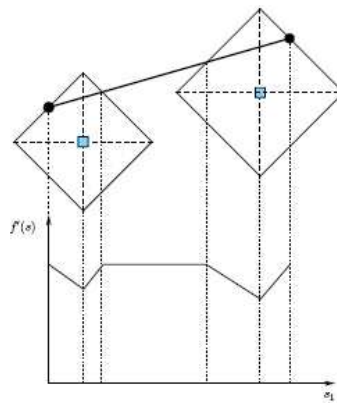


Figure 4.9: $f'(s)$ along one track.

$\mathcal{O}(M|E|)$ candidates. The resulting algorithm for solving single AccessStopLoc', which just compares the objective values of the points in \mathcal{S}'_{cand} has thus the complexity $\mathcal{O}(M^2|E|)$.

It can happen that $\mathcal{S}'_{cand} = \emptyset$, namely if there does not exist any point on the tracks that improves the objective value $f'(v)$, $v \in V_{stop}$ (see Figure 4.10). Thus, in such a situation it is not reasonable to add any station to our PTN.

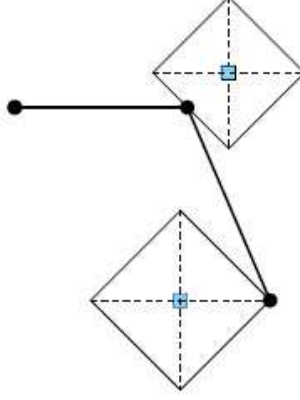


Figure 4.10: An example where the existing PTN can not be improved.

Note that $\mathcal{S}'_{cand} \subseteq \mathcal{S}_{cand}$. Moreover, if $\mathcal{S}'_{cand} = \mathcal{S}_{cand}$, then *AccessStopLoc* and *AccessStopLoc'* are equivalent. Such a situation happens, for example, if all the existing stops are at infinity. In practice, this can be interpreted as follows: if all the existing stops are far away from the set of demand facilities, we do not need to consider them in our planning to open some additional stop and the problem is reduced to the problem of completely redesigning all stops.

Now we are ready to analyze the general stop location problem where we want to establish at most $k > 1$ stops. In what follows we will only consider *AccessStopLoc*, but all the arguments also hold for *AccessStopLoc'*. We know that the optimal solution for the single stop location problem is contained in \mathcal{S}_{cand} . The next theorem shows that this holds also for the general problem.

Theorem 4.18. *For AccessStopLoc, there exists an optimal solution $\mathcal{S} \subseteq \mathcal{S}_{cand}$, $|\mathcal{S}^*| \leq k$*

Proof: Let $\tilde{\mathcal{S}}$ be a solution with $|\tilde{\mathcal{S}}| \leq k$. Then each $\tilde{s} \in \tilde{\mathcal{S}}$ is the optimal solution of a single stop location problem, namely with the demand facilities that are served by \tilde{s} . For the single stop location problem we know from Theorem 4.14 that there exists an optimal solution in \mathcal{S}_{cand} , hence the result follows. \square

However, *AccessStopLoc* is \mathcal{NP} -hard even if we only consider the finite dominating set \mathcal{S}_{cand} .

Theorem 4.19. *AccessStopLoc is \mathcal{NP} -hard.*

Proof: The proof is a reduction of the rectangular k -median problem to the decision version of *AccessStopLoc*.

(Rectangular k -median problem) Given a set $\mathcal{P} = \{p_1, \dots, p_M\} \subset \mathbb{R}^2$ of

points in the plane and positive rational number k . Is there a set $S \subset \mathbb{R}^2$ of k points such that if $l_1(p_m, S)$ is the length of shortest l_1 travel path from p_m to the closest point in S , then $\sum_{m=1}^M l_1(p_m, S) \leq k$? \square

The rectangular k -median problem is known to be \mathcal{NP} -hard (see Megiddo and Supowit (1984)).

It can easily be shown (see Poetranto (2005)) that rectangular k -median is polynomially transformable to an instance of AccessStopLoc as follows:

- Leave \mathcal{P} and k as it is.
- Define G such that it contains sufficiently large parts of the construction lines, where V consists of the end points of the segments and of the intersection points between pairs of the segments. G can now be defined as a grid graph with V as nodes and edges corresponding to the linear pieces between the nodes. \square

Heuristic methods based on a node partitioning scheme and some numerical results are presented in Poetranto (2005).

4.3.2 Accessibility Problem in Network Environment

Finally, we discuss the accessibility problem in the network environment.

In locational analysis, the problem of locating k facilities on a graph in order to minimize the sum of weighted shortest distances to the given customers is commonly referred as k -median problem, for a survey see Drezner and Hamacher (2002). More precisely, in the k -median problem, we have given a graph $G = (V, E)$ with weights $w_v \geq 0$ for all $v \in V$. The goal is to locate a set New of k new facilities (either on the nodes or along the edges of the graph) in order to minimize

$$\sum_{v \in V} w_v d(v, New)$$

The following results have been show by Hakimi (1965) and by Hakimi and Kariv (1979).

Theorem 4.20.

- *The k -median problem has the node optimality property, i.e., there exists an optimal solution in which all new facilities are vertices.*
- *Even in this case, the k -median problem is \mathcal{NP} -hard.*

Exercise 4.21. (=Assignment 8) Prove Theorem 4.20. (Hint: Start with the case of a single new location and then conclude that the result is also correct, if $k > 1$ new facilities have to be located.) Is the node optimality result also true if the median objective function is replaced by the center objective function $\max_{v \in V} w_v d(v, S)$?

Due to the first property, one must only examine all subsets of V containing k vertices to find the optimal solution. However, the second result shows that this is non-trivial. Note that the NP-hardness result even holds for networks with a simple structure, e.g. for planar graphs with maximum vertex degree three.

AccessStop Loc on a network can be considered as a restricted k -median problem: Let G be the SPTN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = \mathcal{P} \cup \mathcal{V}$ and $V = V_{break} \cup V_{stop}$ (see (4.1)), define the following weights:

$$w_v = \begin{cases} 1 & \text{if } v \in \mathcal{P} \\ 0 & \text{otherwise} \end{cases}$$

The goal is to locate k stops that minimize the sum of weighted distances to the other nodes. This is a restricted version of the k -median problem, since in the case of AccessStopLoc we are only allowed to locate the new facilities in the PTN, i.e. on a subset of the complete network SPTN.

Theorem 4.22. $\mathcal{S}_{cand} := V_{break}$ is an FDS for Network-AccessStopLoc in the case of a single stop, i.e., there exists an optimal solution $s^* \in \mathcal{S}_{cand}$.

Proof: The proof of the first property of Theorem 4.20 is based on the fact that the distance function of each demand point is concave along each edge (see Figure 4.4), such that moving a location to the endpoint of its edge never worsens the objective value. Since the endpoints of each feasible edge are also feasible, the proof for the node optimality property of Hakimi (1965) also holds in the restricted case. Hence, there exists an optimal solution $S^* \subseteq V$. Since it is useless to locate a new facility at an already existing one, the result follows. \square

Obviously, the set V_{break} is finite. However, for the case in which $V_{stop} \neq \emptyset$, i.e. AccessStopLoc', we obtain a further reduction using the same arguments as in the planar case: For each demand facility $p_m \in \mathcal{P}$, we determine a closest stop $v_m \in V_{stop}$, that is

$$v_m \in \operatorname{argmin}_{v \in V_{stop}} ND(p_m, v)$$

The objective function is only improved if we locate a new stop in the interior of D , where D is defined as in (4.13) but the radius is measured by the network distance. That means, the set $\mathcal{S} \setminus \operatorname{int}(D)$ can be considered as forbidden region. Hence, we only need to consider the set of breakpoints that lie on $\operatorname{int}(D)$. We obtain the set of candidates $\mathcal{S}'_{cand} := V_{break} \cap \operatorname{int}(D)$ which is sufficient as FDS for AccessStopLoc' in networks. Solving AccessStopLoc and AccessStopLoc' on SPTN can thus be done evaluating the objective values for $v \in \mathcal{S}'$.

Bibliography

- Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows. Theory, Algorithms, and Applications*. Prentice Hall Inc., Englewood Cliffs, NJ.
- Burkard, R. (2002). Open Problem Session, Oberwolfach Conference on Combinatorial Optimization, November 24-29, 2002.
- Burkard, R., Rote, G., Ruhe, G., and Sieber, N. (1989). Algorithmische Untersuchungen zu bikriteriellen kostenminimalen Flüssen in Netzwerken. *Wissenschaftliche Zeitschrift der technischen Hochschule Leipzig*, 13(6):333–341.
- Chankong, V. and Haimes, Y. (1983). *Multiobjective Decision Making Theory and Methodology*. Elsevier Science, New York.
- Drezner, Z. and Hamacher, H. W., editors (2002). *Facility Location: Application and Theory*. Springer, Berlin.
- Engau, A. (2005). *Semi-Simultaneous Flows in Multiple Networks*. Diploma Thesis, University of Kaiserslautern.
- Figueira, J., Greco, S., and Ehrgott, M., editors (2005). *Multiple Criteria Decision Analysis - State of the Art Surveys*. Springer, New York.
- Garey, M. and Johnson, D. (1979). *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco, CA.
- Geoffrion, A. (1968). Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, 22(3):618–630.
- Hakimi, S. L. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operation Research*, 13:462–475.
- Hakimi, S. L. and Kariv, O. (1979). An algorithmic approach to network location problems. part 2 : The p-medians. *SIAM Journal on Applied Mathematics*, 37:539–560.
- Hamacher, H. and Küfer, K.-H. (2002). Inverse radiation therapy planing – A multiple objective optimization approach. *Discrete Applied Mathematics*, 118(1-2):145–161.
- Hamacher, H. W. (1995). *Mathematische Lösungsverfahren für planare Standortprobleme*. Vieweg Verlag.

- Hamacher, H. W. and Klamroth, K. (2001). *Linear and Network Optimization - A Bilingual Textbook*. Mathematics International. Vieweg Verlag.
- Isermann, H. (1974). Proper efficiency and the linear vector maximum problem. *Operations Research*, 22:189–191.
- Kalinowski, T. (2005). A duality based algorithm for multileaf collimator field segmentation with interleaf collision constraint. *Discrete Applied Mathematics*, 152:52–88.
- Megiddo, N. and Supowit, K. J. (1984). On the complexity of some common geometric location problems. *SIAM Journal on Computing*, 13:182–196.
- Nußbaum, M. (2006). Min cardinality $c1$ -decomposition of integer matrices. Master’s thesis, Department of Mathematics, Technical University of Kaiserslautern.
- Poetranto, D. R. (2005). A restricted median location model for stop location design in public transportation networks. In *Operational Research Paripatetic Postgraduate Programme (ORP3)*, pages 297–310, Valencia, Spain. Universidad Politécnic de Valencia.
- Schöbel, A. (2004). Personal communication.
- Schöbel, A. (2006). *Optimization in public transportation*, volume 3 of *Optimization and Its Applications*. Springer, New York.
- Schöbel, A., Hamacher, H. W., Liebers, A., and Wagner, D. (2002). The continuous stop location problem in public transportation networks. Report in Wirtschaftsmathematik 81/2002, Fachbereich Mathematik, Technische Universität Kaiserslautern. http://kluedo.ub.uni-kl.de/volltexte/2002/1338/ps/gelb_81.ps.
- Steuer, R. (1986). *Multiple-Criteria Optimization: Theory, Computation, and Application*. John Wiley & Sons, New York.