Convex Cost Multicommodity Flow Problems: Applications and Algorithms

Athanasios Migdalas

School of Engineering
Dept. of Mathematics, Physics and Computational Science
University Campus, 54124 Thessaloniki

samig@gen.auth.gr

May 11, 2013

The general form of the problems studied in nonlinear optimization may be stated as :

$$\begin{aligned} \textbf{(NLP)} \quad \min \quad & f(\mathbf{x}) && (1)\\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} && (2)\\ & \mathbf{h}(\mathbf{x}) = \mathbf{0} && (3)\\ & \mathbf{x} \in \mathcal{S}, && (4) \end{aligned}$$

where $\mathcal{S} \subseteq \mathbb{R}^n$, $f : \mathbf{x} \in \mathcal{S} \rightarrow f(\mathbf{x}) \in \mathbb{R}$, $\mathbf{g} : \mathbf{x} \in \mathcal{S} \rightarrow \mathbf{g}(\mathbf{x}) \in \mathbb{R}^\ell$, and $\mathbf{h} : \mathbf{x} \in \mathcal{S} \rightarrow \mathbf{h}(\mathbf{x}) \in \mathbb{R}^m$.

alternatively

$$\begin{aligned} \textbf{(P)} \quad \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in \mathcal{X}, \end{aligned}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ is the feasible region, and $f(\cdot)$ is the objective function of the problem. A point $\mathbf{x} \in \mathbb{R}^n$ is called feasible point to **P** if $\mathbf{x} \in \mathcal{X}$, otherwise $\mathbf{x}$ is infeasible.

Definition 1

A point $\mathbf{x}^*$ *is a local minimum point of* $\mathbf{P}$ *if*

1. $\mathbf{x}^* \in \mathcal{X}$, *i.e., it is feasible, and*
2. *there exists a neighborhood* $\mathcal{N}_\epsilon(\mathbf{x}^*) = \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}^*\| \leq \epsilon\}$, $\epsilon > 0$, *such that* $f(\mathbf{x}^*) \leq f(\mathbf{x}),\ \forall \mathbf{x} \in \mathcal{N}_\epsilon(\mathbf{x}^*) \cap \mathcal{X}$.

A point $\mathbf{x}^*$ *is a global minimum point of* $\mathbf{P}$ *if*

1. $\mathbf{x}^* \in \mathcal{X}$, *i.e., it is feasible, and*
2. $f(\mathbf{x}^*) \leq f(\mathbf{x}),\ \forall \mathbf{x} \in \mathcal{X}$.

The point $\mathbf{x}^*$ *is often referred to as a local (or global) optimal point or optimal solution.*

Definition 2

The function $f(\mathbf{x})$ defined on $\mathcal{X} \subseteq \mathbb{R}^n$ is said to be lower semicontinuous in $\mathcal{X}$ if the sets $\mathcal{L}(b) = \{\mathbf{x} \in \mathcal{X} | f(\mathbf{x}) \leq b\}$ are closed relative to $\mathcal{X}$ for any $b \in \mathbb{R}$.

Theorem 1

If $\mathcal{X}$ is a nonempty compact set and the function $f(\mathbf{x})$ is lower semicontinuous in $\mathcal{X}$, then $\mathbf{P}$ has a solution $\mathbf{x}^* \in \mathcal{X}$.

Definition 3

The set $\mathcal{X} \subset \mathbb{R}^n$ *is convex if for all* $\mathbf{x}^1, \mathbf{x}^2 \in \mathcal{X}$ *it satisfies the inclusion*

$$\alpha \mathbf{x}^1 + (1-\alpha)\mathbf{x}^2 \quad \in \quad \mathcal{X}$$

for all $\alpha \in [0,1]$*, that is, the convex combination of any two points in* $\mathcal{X}$ *is also in* $\mathcal{X}$*. The function* $f(\mathbf{x})$ *is convex in the convex set* $\mathcal{X}$ *if for all* $\mathbf{x}^1, \mathbf{x}^2 \in \mathcal{X}$ *and* $\alpha \in (0,1)$ *it satisfies the inequality*

$$f(\alpha \mathbf{x}^1 + (1-\alpha)\mathbf{x}^2) \quad \leq \quad \alpha f(\mathbf{x}^1) + (1-\alpha) f(\mathbf{x}^2).$$

The function is strictly convex if strict inequality holds in the above relation. Moreover, the function $-f(\mathbf{x})$ *is (strictly) concave if* $f(\mathbf{x})$ *is (strictly) convex.*

Theorem 2

If $\mathcal{X}$ and $f(\mathbf{x})$ are convex in $\mathbf{P}$, then any local minimum point is also global. Moreover, if $f(\mathbf{x})$ is strictly convex the minimum point is unique.

Definition 4

A polyhedron $\mathcal{X}$ in $\mathbb{R}^n$ is the intersection of a finite number of closed halfspaces, i.e., $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{a}_i^T \mathbf{x} \leq b_i, i = 1, \ldots, m\}$, where $\mathbf{a}_i \in \mathbb{R}^n$ are constant vectors and b_i are real numbers. A bounded polyhedron is called a polytope.

Definition 5

The point $\mathbf{x}$ is an extreme point or a vertex of the polyhedron $\mathcal{X} \subset \mathbb{R}^n$ if $\mathbf{x} = \alpha \mathbf{x}^1 + (1 - \alpha)\mathbf{x}^2$ with $\mathbf{x}^1, \mathbf{x}^2 \in \mathcal{X}$ and $\alpha \in (0, 1)$ implies that $\mathbf{x} = \mathbf{x}^1 = \mathbf{x}^2$. That is, an extreme point cannot be written as a convex combination of two distinct points in $\mathcal{X}$.

Definition 6

The nonzero vector $\mathbf{d} \in \mathbb{R}^n$ *is called a direction of the polyhedron* $\mathcal{X}$ *if for every* $\mathbf{x} \in \mathcal{X}$, $\mathbf{x} + \alpha\mathbf{d} \in \mathcal{X}$ *for all* $\alpha \geq 0$. *The direction* $\mathbf{d}$ *of* $\mathcal{X}$ *is an extreme direction of* $\mathcal{X}$ *if* $\mathbf{d} = \alpha\mathbf{d}^1 + \beta\mathbf{d}^2$ *for* $\alpha, \beta > 0$ *implies* $\mathbf{d}^1 = \gamma\mathbf{d}^2$ *for some* $\gamma > 0$. *That is, an extreme direction cannot be written as a positive linear combination of two distinct directions in* $\mathcal{X}$.

Theorem 3

The number of extreme points or vertices of a polyhedron is finite. The number of extreme directions of a polyhedron is finite.

Theorem 4

A polytope is the convex hull of its vertices, that is, if $\bar{\mathbf{x}}^1, \bar{\mathbf{x}}^2, \ldots, \bar{\mathbf{x}}^K$ *are the extreme points of a polytope* $\mathcal{X} \subset \mathbb{R}^n$*, then*

$$\mathcal{X} = \left\{ \mathbf{x} \in \mathbb{R}^n | \mathbf{x} = \sum_{k=1}^{K} \alpha_k \bar{\mathbf{x}}^k, \ \sum_{k=1}^{K} \alpha_k = 1, \ \alpha_k \geq 0 \text{ for } k = 1, \ldots, K \right\}$$

In particular, since $\mathcal{X} \subset \mathbb{R}^n$*, then at most* $n+1$ *extreme points are needed in order to represent any point* $\mathbf{x} \in \mathcal{X}$*.*
Moreover, if $\mathbf{X} \subset \mathbb{R}^n$ *is an unbounded polyhedron with* K *extreme points* $\bar{\mathbf{x}}^1, \bar{\mathbf{x}}^2, \ldots, \bar{\mathbf{x}}^K$ *and* L *extreme directions* $\bar{\mathbf{d}}^1, \bar{\mathbf{d}}^2, \ldots, \bar{\mathbf{d}}^L$*, then* $\mathbf{x} \in \mathcal{X}$ *if and only if*

$$\begin{aligned} \mathbf{x} &= \sum_{k=1}^{K} \alpha_k \bar{\mathbf{x}}^k + \sum_{\ell=1}^{L} \beta_\ell \bar{\mathbf{d}}^\ell \\ & \sum_{k=1}^{K} \alpha_k = 1 \\ & \alpha_k \geq 0, \ \forall k \\ & \beta_\ell \geq 0, \ \forall \ell \end{aligned}$$

Definition 7

Let $f(\mathbf{x})$ be defined on the nonempty set $\mathcal{X} \subseteq \mathbb{R}^n$ and suppose that $\bar{\mathbf{x}} + \alpha\mathbf{d} \in \mathcal{X}$ for $\bar{\mathbf{x}} \in \mathcal{X}$, $\mathbf{d} \neq \mathbf{0}$ a vector in $\mathbb{R}^n$, and $\alpha > 0$ sufficiently small. The directional derivative of $f(\mathbf{x})$ at $\bar{\mathbf{x}}$ in the direction $\mathbf{d}$ is given by the following limit if it exists:

$$\lim_{\alpha \to 0^+} \frac{f(\bar{\mathbf{x}} + \alpha\mathbf{d}) - f(\bar{\mathbf{x}})}{\alpha}$$

Theorem 5

Let $\mathcal{X}$ be nonempty and convex set and assume that $f(\mathbf{x})$ is convex, then the limit in the definition of the directional derivative exists.

Theorem 6

If $f(\mathbf{x})$ has the gradient $\nabla f(\bar{\mathbf{x}})$ at $\bar{\mathbf{x}} \in \mathbb{R}^n$ and $\mathbf{d} \neq \mathbf{0}$ is a vector in $\mathbb{R}^n$, then the directional derivative of $f(\mathbf{x})$ at $\bar{\mathbf{x}}$ in the direction $\mathbf{d}$ is $\nabla f(\bar{\mathbf{x}})^T \mathbf{d}$.

Definition 8

Let $f(\mathbf{x})$ have the gradient $\nabla f(\bar{\mathbf{x}})$ at $\bar{\mathbf{x}}$. Then the vector $\mathbf{d} \neq \mathbf{0}$ is descent direction of $f(\mathbf{x})$ at $\bar{\mathbf{x}}$ if $\nabla f(\bar{\mathbf{x}})^T \mathbf{d} < 0$. The set $\mathcal{D}(\bar{\mathbf{x}}) = \{\mathbf{d} \in \mathbb{R}^n | \nabla f(\bar{\mathbf{x}})^T \mathbf{d} < 0\}$ is the set of descent directions of $f(\cdot)$ at $\bar{\mathbf{x}}$.

Definition 9

Let $\bar{\mathbf{x}}$ be feasible to $\mathbf{P}$. Then the vector $\mathbf{d} \neq \mathbf{0}$ is a feasible direction from $\bar{\mathbf{x}}$ if there exists a $\bar{\alpha} > 0$ such that $\bar{\mathbf{x}} + \alpha \mathbf{d}$ is feasible to $\mathbf{P}$ for all α satisfying $0 \leq \alpha \leq \bar{\alpha}$. The set of all feasible directions from $\bar{\mathbf{x}}$ will be denoted by $\mathcal{F}(\bar{\mathbf{x}})$.

Theorem 7

If $f(\mathbf{x})$ is differentiable at $\bar{\mathbf{x}}$, then it is continuous and has the gradient $\nabla f(\bar{\mathbf{x}})$ at $\bar{\mathbf{x}}$. Moreover, if $f(\mathbf{x})$ is differentiable on the segment joining $\mathbf{x}^1$ and $\mathbf{x}^2$, then the function $\phi(\alpha) = f(\alpha\mathbf{x}^1 + (1-\alpha)\mathbf{x}^2)$, defined for $\alpha \in [0,1]$, is differentiable in $[0,1]$ and has the derivative $\phi'(\alpha) = (\mathbf{x}^1 - \mathbf{x}^2)^T \nabla f(\alpha\mathbf{x}^1 + (1-\alpha)\mathbf{x}^2)$, that is, $\phi'(\alpha)$ is the directional derivative of $f(\mathbf{x})$ at $\alpha\mathbf{x}^1 + (1-\alpha)\mathbf{x}^2$ in the direction $\mathbf{x}^1 - \mathbf{x}^2$. If $\phi'(0) < 0$, i.e., $(\mathbf{x}^1 - \mathbf{x}^2)^T \nabla f(\mathbf{x}^2) < 0$, then there is a $\alpha \in (0,1)$ such that $\phi(\alpha) < \phi(0)$, or equivalently $f(\alpha\mathbf{x}^1 + (1-\alpha)\mathbf{x}^2) < f(\mathbf{x}^2)$, for all $\alpha \in (0, \bar{\alpha})$.

Theorem 8

Let $f(\mathbf{x})$ be a differentiable function on a nonempty open set $\mathcal{X} \subseteq \mathbb{R}^n$. Then $f(\mathbf{x})$ is convex if and only if for any $\mathbf{y} \in \mathcal{X}$ the inequality

$$f(\mathbf{x}) \quad \geq \quad f(\mathbf{y}) + \nabla f(\mathbf{y})^T(\mathbf{x} - \mathbf{y}), \ \forall \mathbf{x} \in \mathcal{X}$$

holds.

Theorem 8

Let $f(\mathbf{x})$ be a differentiable function on a nonempty open set $\mathcal{X} \subseteq \mathbb{R}^n$. Then $f(\mathbf{x})$ is convex if and only if for any $\mathbf{y} \in \mathcal{X}$ the inequality

$$f(\mathbf{x}) \quad \geq \quad f(\mathbf{y}) + \nabla f(\mathbf{y})^T(\mathbf{x} - \mathbf{y}), \ \forall \mathbf{x} \in \mathcal{X}$$

holds.

Theorem 9

Let $f(\mathbf{x})$ be a differential convex function defined on the convex set $\mathcal{X}$. Then $\mathbf{x}^*$ is a global optimal solution to $\mathbf{P}$ if and only if

$$\nabla f(\mathbf{x}^*)^T(\mathbf{x} - \mathbf{x}^*) \quad \geq \quad 0, \ \forall \mathbf{x} \in \mathcal{X}.$$

Moreover, if $\mathcal{X}$ is open then $\mathbf{x}^*$ is an optimal solution if and only if $\nabla f(\mathbf{x}^*) = \mathbf{0}$.

Definition 10

The differentiable function $f(\mathbf{x})$ defined on a nonempty open set $\mathcal{X} \subseteq \mathbb{R}^n$ is said to be pseudoconvex if

$$\begin{aligned} \nabla f(\mathbf{x}^1)^T(\mathbf{x}^2 - \mathbf{x}^1) \geq 0 \quad &\Rightarrow \quad f(\mathbf{x}^2) \geq f(\mathbf{x}^1), \text{ or equivalently,} \\ f(\mathbf{x}^2) < f(\mathbf{x}^1) \quad &\Rightarrow \quad \nabla f(\mathbf{x}^1)^T(\mathbf{x}^2 - \mathbf{x}^1) < 0, \end{aligned}$$

for all $\mathbf{x}^1, \mathbf{x}^2 \in \mathcal{X}$. Moreover, the function $-f(\cdot)$ is then called pseudoconcave.

Theorem 10

Let $f(\mathbf{x})$ be a differential concave function defined on the convex set $\mathcal{X}$. If $\bar{\mathbf{x}}$ is a local optimal solution to $\mathbf{P}$ then

$$\nabla f(\bar{\mathbf{x}})^T(\mathbf{x} - \bar{\mathbf{x}}) \quad \geq \quad 0, \ \forall \mathbf{x} \in \mathcal{X}. \tag{5}$$

Moreover, if $\mathcal{X}$ is a nonempty polytope, then there exists an extreme point $\bar{\mathbf{x}} \in \mathcal{X}$ which is an optimal solution of $\mathbf{P}$.

Definition 11

The Kuhn-Tucker constraint qualification holds at $\bar{\mathbf{x}}$ *if* $\mathcal{D}(\bar{\mathbf{x}}) = \mathcal{D}^{\mathcal{L}}(\bar{\mathbf{x}})$.

Theorem 11

In **NLP**, *let* $\mathcal{S}$ *be a nonempty set with at least one interior point, assume that* $f, \mathbf{g}$ *and* $\mathbf{h}$ *are continuously differentiable. If* $\bar{\mathbf{x}}$ *is a local minimum point and some constraint qualification holds at* $\bar{\mathbf{x}}$, *then there exist* $\boldsymbol{\lambda} \in \mathbb{R}^{\ell}$ *and* $\boldsymbol{\mu} \in \mathbb{R}^{m}$ *such that*

$$-\nabla f(\bar{\mathbf{x}}) = \boldsymbol{\lambda}^T \mathbf{g}(\bar{\mathbf{x}}) + \boldsymbol{\mu}^T \mathbf{h}(\bar{\mathbf{x}}) \quad (6)$$

$$\boldsymbol{\lambda}^T \mathbf{g}(\bar{\mathbf{x}}) = 0 \quad (7)$$

$$\boldsymbol{\lambda} \geq \mathbf{0} \quad (8)$$

Theorem 12

Assume that the conditions of Theorem 11 on $\mathcal{S}, \mathbf{g}$ and $\mathbf{h}$ are valid. Assume further that f and the component functions $g_i(\cdot)$ of $\mathbf{g}(\cdot)$ are pseudoconvex and that $\mathbf{h}$ is linear, that is $\mathbf{h}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}$. Then, any feasible point $\bar{\mathbf{x}}$ in **NLP** *which satisfies the KKT-conditions (6)-(8) is a global minimum point to* **NLP**.

Sufficient optimality conditions for **NLP** can also be stated in terms of the Lagrangian function. For the problem **NLP**, define the Lagrangian function

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}) + \boldsymbol{\mu}^T \mathbf{h}(\mathbf{x}) \tag{9}$$

for $\mathbf{x} \in \mathcal{S}$ and $\boldsymbol{\lambda} \geq \mathbf{0}$, and consider the Lagrangian subproblem

$$\Theta(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \min_{\mathbf{x} \in \mathcal{S}} f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}) + \boldsymbol{\mu}^T \mathbf{h}(\mathbf{x}), \tag{10}$$

defined for $\boldsymbol{\lambda} \geq \mathbf{0}$.

Theorem 13

Let $\bar{\mathbf{x}}$ *be a solution to (10) for a given* $(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}})$. *If* $\bar{\mathbf{x}}$ *is a feasible point to* **NLP** *and satisfies the conditions*

$$f(\bar{\mathbf{x}}) + \bar{\boldsymbol{\lambda}}^T \mathbf{g}(\bar{\mathbf{x}}) + \bar{\boldsymbol{\mu}}^T \mathbf{h}(\bar{\mathbf{x}}) = \Theta(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}}) \quad (11)$$

$$\bar{\boldsymbol{\lambda}}^T \mathbf{g}(\bar{\mathbf{x}}) = 0 \quad (12)$$

$$\bar{\boldsymbol{\lambda}} \geq \mathbf{0}, \quad (13)$$

then $\bar{\mathbf{x}}$ *is an optimal solution to* **NLP**.

Definition 12

The situation $\mathbf{x}^*$ *is a Nash equilibrium of the game* $\mathcal{G} = [\mathcal{N}, \{\mathcal{X}\}_{i\in\mathcal{N}}, \{f_i\}_{i\in\mathcal{N}}]$ *if*

$$f_i(\mathbf{x}^*) \quad = \quad \min_{\mathbf{x}_i \in \mathcal{X}_i} f_i(\mathbf{x}^*_{i-}, \mathbf{x}_i, \mathbf{x}^*_{i+}), \ \forall i \in \mathcal{N} \tag{14}$$

The game $\mathcal{G}$ is a team game if $f_i(\cdot) = f(\cdot), \ \forall i \in \mathcal{N}$, that is, if all players share the same loss function. In such a case, a Nash equilibrium of the game is obtainable by solving the following problem

$$\begin{aligned} \textbf{(CPP)} \quad \min \quad & f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \\ \text{s.t.} \quad & \mathbf{x}_i \in \mathcal{X}_i, \ \forall i \in \mathcal{N}, \end{aligned}$$

which is an optimization problem defined over a Cartesian product of sets.

(GBP) min $g(\mathbf{y}, \mathbf{x}^*)$ (15)

s.t. $\mathbf{y} \in \mathcal{Y}$, (16)

where $\mathbf{x}^*$ is such that

$$f_i(\mathbf{y}, \mathbf{x}^*) = \min_{\mathbf{x}_i \in \mathcal{X}_i} f_i(\mathbf{y}, \mathbf{x}^*_{i-}, \mathbf{x}, \mathbf{x}^*_{i+}), \ \forall i \in \mathcal{N}. \quad (17)$$

(BP) min $g(\mathbf{y}, \mathbf{x}^*)$ (18)

s.t. $\mathbf{y} \in \mathcal{Y}$, (19)

where $\mathbf{x}^*$ is such that

$$f(\mathbf{y}, \mathbf{x}^*) = \min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{y}, \mathbf{x}). \quad (20)$$

Traffic Assignment and Routing

Wardrop's equilibrium principle

$$
\begin{aligned}
h_{pk} > 0 \quad &\Rightarrow \quad c_{pk} = \pi_k, \ \forall p \in \mathcal{P}_k, \\
h_{pk} = 0 \quad &\Rightarrow \quad c_{pk} \geq \pi_k, \ \forall p \in \mathcal{P}_k,
\end{aligned}
$$

User equilibrium conditions for fixed demand

$$
\begin{aligned}
h_{pk}(c_{pk} - \pi_k) &= 0, \ \forall p \in \mathcal{P}_k, && (21) \\
c_{pk} - \pi_k &\geq 0, \ \forall p \in \mathcal{P}_k, && (22) \\
\sum_{p \in \mathcal{P}_k} h_{pk} &= r_k, && (23) \\
h_{pk} &\geq 0, \ \forall p \in \mathcal{P}_k && (24) \\
\pi_k &\geq 0, && (25)
\end{aligned}
$$

Theorem 14

Assume that the network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ is strongly connected with respect to the pairs in $\mathcal{K}$, that the demand matrix $\mathbf{R}$ *is nonnegative, and that the travel time function $s_a(\cdot)$ is positive, strictly monotone increasing and continuously differentiable. Then, conditions (21)-(25) are the Karush-Kuhn-Tucker optimality conditions of the convex optimization problem*

$$\textbf{(FTAP)} \quad \min \quad \sum_{a \in \mathcal{A}} \int_0^{x_a} s_a(t)dt, \tag{26}$$

$$s.t. \quad \sum_{p \in \mathcal{P}_k} h_{pk} = r_k, \ \forall k, \tag{27}$$

$$\sum_k \sum_{p \in \mathcal{P}_k} \delta_{kap} h_{pk} = x_a, \ \forall a \in \mathcal{A}, \tag{28}$$

$$h_{pk} \geq 0, \ \forall p \in \mathcal{P}_k, \ \forall k \tag{29}$$

The system optimum problem

$$
\begin{aligned}
\textbf{(SFTAP)} \quad \min \quad & \sum_{a \in \mathcal{A}} s_a(x_a) x_a, \\
\text{s.t.} \quad & \sum_{p \in \mathcal{P}_k} h_{pk} = r_k, \ \forall k, \\
& \sum_{k} \sum_{p \in \mathcal{P}_k} \delta_{kap} h_{pk} = x_a, \ \forall a \in \mathcal{A}, \\
& h_{pk} \geq 0, \ \forall p \in \mathcal{P}_k, \ \forall k
\end{aligned}
$$

Wardrop's user equilibrium for elastic demand

principle for both route flows and demands are mathematically stated as follows:

$$
\begin{aligned}
h_{pk} > 0 &\Rightarrow c_{pk} = \pi_k, \ \forall p \in \mathcal{P}_k, \\
h_{pk} = 0 &\Rightarrow c_{pk} \geq \pi_k, \ \forall p \in \mathcal{P}_k, \\
r_k > 0 &\Rightarrow r_k = g_k(\boldsymbol{\pi}), \\
r_k = 0 &\Rightarrow g_k(\boldsymbol{\pi}) \leq 0,
\end{aligned}
$$

User's equilibrium conditions for elastic demand

$$h_{pk}(c_{pk} - \pi_k) = 0, \ \forall p \in \mathcal{P}_k, \quad (30)$$

$$c_{pk} - \pi_k \geq 0, \forall p \in \mathcal{P}_k, \quad (31)$$

$$\sum_{p \in \mathcal{P}_k} h_{pk} = g_k(\pi), \quad (32)$$

$$h_{pk} \geq 0, \ \forall p \in \mathcal{P}_k, \quad (33)$$

$$\pi_k \geq 0, \quad (34)$$

Theorem 15

Assume that the network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ is strongly connected with respect to the pairs in $\mathcal{K}$, and that the travel time function s_a is positive, monotone increasing and continuous differentiable. Then, conditions (30)-(34) are the Karush-Kuhn-Tucker optimality conditions of the following convex optimization problem:

$$\textbf{(ETAP)} \quad \min \quad \sum_{a \in \mathcal{A}} \int_0^{x_a} s_a(t)dt - \sum_k \int_0^{r_k} g_k^{-1}(t)dt, \tag{35}$$

$$s.t. \quad \sum_{p \in \mathcal{P}_k} h_{pk} = r_k, \ \forall k, \tag{36}$$

$$\sum_k \sum_{p \in \mathcal{P}_k} \delta_{kap} h_{pk} = x_a, \ \forall a \in \mathcal{A}, \tag{37}$$

$$h_{pk} \geq 0, \ \forall p \in \mathcal{P}_k, \forall k, \tag{38}$$

$$r_k \geq 0, \ \forall k. \tag{39}$$

User optimum flow assignment

$$\textbf{(AFTAP)} \quad \min \quad \sum_{a\in\mathcal{A}} \int_0^{x_a} s_a(t)dt, \tag{40}$$

$$\text{s.t.} \quad \sum_{a\in\mathcal{A}^+(i)} x_a^k - \sum_{a\in\mathcal{A}^-(i)} x_a^k = r_i^k, \ \forall i \in \mathcal{N}, \ \forall k \in \mathcal{K}, \tag{41}$$

$$x_a = \sum_{k\in\mathcal{K}} x_a^k, \ \forall a \in \mathcal{A}, \tag{42}$$

$$x_a^k \geq 0, \ \forall a \in \mathcal{A}, \ \forall k \in \mathcal{K}, \tag{43}$$

where x_a^k denotes the portion of flow from origin $o(k)$ to destination $o(k)$ that streams through link a, and

$$r_i^k = \begin{cases} r_k, & \text{if } i = o(k), \\ -r_k, & \text{if } i = d(k), \\ 0, & \text{otherwise.} \end{cases}$$

System optimum flow assignment

$$\textbf{(SAFTAP)} \quad \min \quad \sum_{a \in \mathcal{A}} s_a(x_a) x_a,$$

$$\text{s.t.} \quad \sum_{a \in \mathcal{A}^+(i)} x_a^k - \sum_{a \in \mathcal{A}^-(i)} x_a^k = r_i^k, \ \forall i \in \mathcal{N}, \ \forall k \in \mathcal{K},$$

$$x_a = \sum_{k \in \mathcal{K}} x_a^k, \ \forall a \in \mathcal{A},$$

$$x_a^k \geq 0, \ \forall a \in \mathcal{A}, \ \forall k \in \mathcal{K},$$

User optimim flow assignment for elastic demand

$$\textbf{(AETAP)} \quad \min \quad \sum_{a\in\mathcal{A}} \int_0^{x_a} s_a(t)dt - \sum_k \int_0^{r_k} g_k^{-1}(t)dt,$$

$$\text{s.t.} \quad \sum_{a\in\mathcal{A}^+(i)} x_a^k - \sum_{a\in\mathcal{A}^-(i)} x_a^k = r_i^k, \ \forall i \in \mathcal{N}, \ \forall k \in \mathcal{K},$$

$$x_a = \sum_{k\in\mathcal{K}} x_a^k, \ \forall a \in \mathcal{A},$$

$$x_a^k \geq 0, \ \forall a \in \mathcal{A}, \ \forall k \in \mathcal{K},$$

where

$$r_i^k = \begin{cases} r_k, & \text{if } i = o(k), \\ -r_k, & \text{if } i = d(k), \\ 0, & \text{otherwise,} \end{cases}$$

and $r_k \geq 0$.

Capacity Assignment and Network Design

Example 1

Consider the network in Figure (a), where the link costs s_{ij} are linear increasing functions of the flow x_{ij} for all links (i,j) of the network, and assume that there are 6 units of flow to be routed from node 1 to node 2. The total delay on each link (i,j) for the user equilibrium model **FTAP** *is then given by:*

$$\begin{aligned} f_{13}(x_{13}) &= 5x_{32}^2 \\ f_{32}(x_{32}) &= 50x_{32} + 0.5x_{32}^2 \\ f_{14}(x_{14}) &= 50x_{14} + 0.5x_{14}^2 \\ f_{42}(x_{42}) &= 5x_{42}^2 \end{aligned}$$

Braess' Paradox

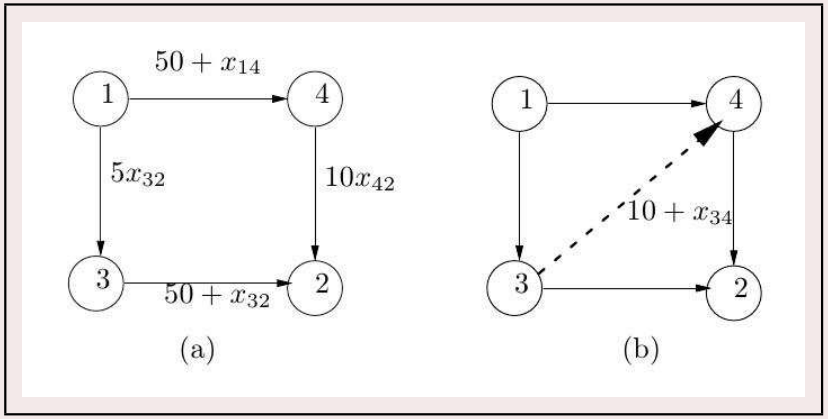


Figure 1: Illustration of Braess' paradox in network design

The network design problem from the system's perspective

$$\textbf{(SNDP)} \quad \min \quad \sum_{a \in \mathcal{A}} \{s_a(x_a, y_a)x_a + g_a(y_a)\}$$
$$\text{s.t.} \quad (27) - (29) \text{ or, equivalently, } (41) - (43)$$
$$y_a \in \mathcal{Y}_a, \ \forall a \in \mathcal{A}$$

It should be noted that in data communication networks, explicit capacity constraints on the flow are often present , that is,

$$x_a \leq u_a + y_a, \ \forall a \in \mathcal{A}$$

should be included in the above formulation.

If the functions $g_a(y_a)$ are convex nonlinear or linear, then the presence of two blocks of variables encourages the application of a so-called primal or Benders decomposition in which, the $|\mathcal{A}|$ capacity assignment subproblems

capacity assignment subproblems

$$\begin{aligned} f_a(x_a) \quad = \quad & \min_{y_a} s_a(x_a, y_a)x_a + g_a(y_a) \\ & \text{s.t. } y_a \in \mathcal{Y} \end{aligned}$$

are solved and a so-called master problem is formed

Benders Master Problem

$$\begin{aligned} \textbf{(BMP)} \quad & \min \quad \sum_{a \in \mathcal{A}} f_a(x_a) \\ & \text{s.t.} \quad (27) - (29) \text{ or, equivalently, } (41) - (43) \end{aligned}$$

The bilevel modeling

$$\textbf{(SNDP)} \quad \min_{\mathbf{y}} \quad \sum_{a\in\mathcal{A}}\{s_a(x_a^*, y_a)v_a + g_a(y_a)\}$$

$$\text{s.t.} \quad y_a \in \mathcal{Y}_a,\ \forall a \in \mathcal{A}$$

$$\text{where } \mathbf{x}^* = [x_a]_{a\in\mathcal{A}} \text{ solves}$$

$$\min_{\mathbf{x}} \sum_{a\in\mathcal{A}} \int_0^{x_a} s_a(t, y_a)dt$$

$$\text{s.t. } (27) - (29) \text{ or, equivalently,} (41) - (43)$$

The transportation problem with costumer competition for the offered service level

Assumptions

- A producer supplies a product to a set of customers.
- Customers themselves have to transport the product in order to satisfy their demand
- The producer tries to provide its customers the best, at his opinion, level of service at minimum cost
- The level of service provided is measured by the service delay encountered.
- The costumers make decisions based on the minimization of their individual total.
- Customer competition can be expressed in terms of a noncompetitive Nash game.

Symbols

- n is the number of customers
- m is the number of warehouses
- r_j is the demand of customer $j, (j = 1, \ldots, n)$
- q_i is the capacity of warehouse $i, (i = 1, \ldots, n)$
- x_{ij} is the quantity purchased by customer i at facility j
- $x_i = \sum_{j=1}^{n} x_{ij}$ is the total amount serviced by warehouse i
- $d_i(x_i)$ is the delay faced by customer j at facility i for each unit served, example $d_i(x_i) = \frac{1}{(q_i - x_i)}$
- $c_{ij}(x_i) = \alpha_i p_i + \beta_{ij} t_{ij} + \mu_i d_i(x_i)$ is the perceived unit cost

The producer's convex optimization problem

$$(\mathbf{SO}-\mathbf{TP})\ \min \quad \sum_{i=1}^{m} d_i(x_i)x_i + \sum_{i=1}^{m} p_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}x_{ij} \quad (44)$$

$$\text{s.t} \quad \sum_{i=1}^{m} x_{ij} = r_j,\ \forall j \quad (45)$$

$$x_i \leq q_i,\ \forall i \quad (46)$$

$$x_i - \sum_{j=1}^{n} x_{ij} = 0,\ \forall i \quad (47)$$

$$x_{ij} \geq 0,\ \forall i,\ \forall j \quad (48)$$

In this "system optimum" in order to take place a transaction between customer j and facility i the following must hold:

$$x_{ij} > 0 \quad \Rightarrow \quad d_i(x_i) + x_i \frac{\partial d_i(x_i)}{\partial x_i} + p_i + t_{ij} = \tilde{c}_{ij} \quad \forall i, j \qquad (49)$$

$$x_{ij} = 0 \quad \Rightarrow \quad d_i(x_i) + x_i \frac{\partial d_i(x_i)}{\partial x_i} + p_i + t_{ij} \geq \tilde{c}_{ij} \quad \forall i, j \qquad (50)$$

where $\tilde{c}(x_{ij})$ is the total unit cost faced at i

Problem (44)-(48) can be expressed as a nonlinear complementarity problem:

$$\left[d_i(x_i) + x_i \frac{\partial d_i(x_i)}{\partial x_i} + p_i + t_{ij} - \tilde{c}_{ij}\right] x_{ij} = 0, \quad \forall i, \ \forall j \quad (51)$$

$$d_i(x_i) + x_i \frac{\partial d_i(x_i)}{\partial x_i} + p_i + t_{ij} - \tilde{c}_{ij} \geq 0, \quad \forall i, \ \forall j \quad (52)$$

$$\sum_{j=1}^{n} x_{ij} = x_i \leq q_i, \quad \forall i \quad (53)$$

$$\sum_{i=1}^{m} x_{ij} = r_j, \quad \forall j \quad (54)$$

$$x_{ij} \geq 0, \ \forall i, \quad \forall j \quad (55)$$

Customers' equilibrium conditions

$$x_{ij} > 0 \quad \Rightarrow \quad d_i(x_i) + p_i + t_{ij} = \hat{c}_{ij} \quad \forall i, j \tag{56}$$

$$x_{ij} = 0 \quad \Rightarrow \quad d_i(x_i) + p_i + t_{ij} \geq \hat{c}_{ij} \quad \forall i, j \tag{57}$$

where $\hat{c}_{ij}$ is the maximum unit cost at which customer j is willing to buy.

The corresponding nonlinear complementarity problem:

$$
\begin{aligned}
[d_i(x_i) + p_i + t_{ij} - \hat{c}_{ij}]\, x_{ij} &= 0, \quad \forall i, \ \forall j && (58)\\
d_i(x_i) + p_i + t_{ij} - \hat{c}_{ij} &\geq 0, \quad \forall i, \ \forall j && (59)\\
x_i &\leq q_i, \forall i && (60)\\
x_i - \sum_{j=1}^{n} x_{ij} &= 0, \quad \forall i && (61)\\
\sum_{i=1}^{m} x_{ij} &= r_j, \quad \forall j && (62)\\
x_{ij} &\geq 0, \ \forall i, \quad \forall j && (63)
\end{aligned}
$$

Customers Problem

$$(\mathbf{UO-TP}) \quad \min \quad \sum_{i=1}^{m} \int_0^{x_i} d_i(t)\, dt + \sum_{i=1}^{m} p_i x_i + \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} x_{ij} \tag{64}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} x_{ij} = r_j, \ \forall j \tag{65}$$

$$x_i \leq q_i, \ \forall i \tag{66}$$

$$x_i - \sum_{j=1}^{n} x_{ij} = 0, \ \forall i \tag{67}$$

$$x_{ij} \geq 0, \ \forall i, \ \forall j \tag{68}$$

where $\hat{c}_{ij} = w_j - \pi_i$, and w_j and π_i are the Lagrangian multipliers corresponding to constraints (66) and (65) respectively.

The Stochastic Transportation Problem

- The stochastic transportation is an extension of the classical transportation problem, where the demand r_j of the costumer j is uncertain, and thus, it could be treated as a continuous random variable with probability density function $\varphi_j(r_j)$.
- Moreover, it is assumed that variables r_j are independent, that is, the demand of the costumer ℓ does not affect the demand of costumer j, for $\ell \neq j$.
- Let t_{ij} be the unit transportation cost from supply point i to demand point j and
- $x_j = \sum_{i=1}^{m} x_{ij}$ the amount of the product dispatched to demand point j.

- Since the demand is not known in advance, when a particular quantity x_j of product is transferred to one of the destinations the following cases may be faced:
  1. $x_j < r_j$, i.e., the product is under-supplied. This shortage creates a penalty $g_j(r_j - x_j)$ caused by the lost sales.
  2. $x_j > r_j$, i.e., the product is oversupplied. The result of this surplus is a penalty which is equal to $\varpi_j(x_j - r_j)$.
  3. $x_j = r_j$, which does not entail any additional cost.

Thus, due to randomness of r_j the expected cost of not covering the demand of customer j is

$$g_j \int_j^{\infty} (r_j - x_j)\varphi_j(r_j)\ dr_j, \tag{69}$$

while

$$\varpi_j \int_0^{x_j} (x_j - r_j)\varphi_j(r_j)\ dr_j \tag{70}$$

is the cost of the oversupplied demand.
Assuming further that each supply center has a production capacity of q_i units, the problem can be stated as follows:

The stochastic transportation problem

$$\textbf{(STP)} \quad \min \quad \sum_{j=1}^{n} \left\{ g_j \int_j^{\infty} (r_j - x_j)\varphi_j(r_j)\, dr_j + \right.$$

$$\left. + \varpi_j \int_0^{x_j} (x_j - r_j)\varphi_j(r_j)\, dr_j \right\} +$$

$$+ \sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}x_{ij} \tag{71}$$

$$\text{s.t} \quad \sum_{j=1}^{n} x_{ij} \leq q_i, \ \forall i \tag{72}$$

$$\sum_{i=1}^{m} x_{ij} = x_j, \ \forall j \tag{73}$$

$$x_{ij} \geq 0, \ \forall i, \forall j, \tag{74}$$

Production-transportation problem in the presence of an intermediary

Assumptions

- There exist $|M|$ supply points
- and $|N|$ demand points for a product.
- The supply points are controlled by $|K|$ producers (M_k refers to production facilities owned by the k^{th} producer)
- the demand points are owned by a single buyer.
- The buyer's demand is a vector $\mathbf{r} = [r_j]$.
- The buyer is addressed to an intermediary in order to satisfy this demand.
- The intermediary is able to contact the $|K|$ producers to obtain the product and meet the buyer's demand.

Assumptions

- Each producer k faces a different production function and charges a price for the product based on a pricing function $o_k(\mathbf{s}_k)$, where $\mathbf{s}_k = [x_i]_{i \in M_k}$ is a vector in $\mathbb{R}^{|M_k|}$ containing the quantities produced by facility i owned by producer k .
- Function $p_k(\cdot)$ is a convex (non-decreasing) differentiable function with
- Function $o_k(\cdot)$ is a convex (non-decreasing) differentiable function with $o(\mathbf{0}) = 0$.
- If we assume further that the transportation cost $t_{ij}x_{ij}$, where t_{ij} denotes the unit transportation cost and x_{ij} represents the quantity transported from production facility i to demand point j is charged to the customer, then the problem of the intermediary can be stated as follows:

Production-transportation problem in the presence of an intermediary

$$\textbf{[ITP]} \quad \min \quad \sum_{k=1}^{|K|} o_k(\mathbf{s}_k) + \sum_{i=1}^{|M|} \sum_{j=1}^{|N|} t_{ij} x_{ij} \tag{75}$$

$$\text{s.t} \quad \sum_{j=1}^{|N|} x_{ij} = x_i, \ \forall i \tag{76}$$

$$\sum_{i=1}^{|M|} x_{ij} = r_j, \ \forall j \tag{77}$$

$$x_{ij} \geq 0, \ \forall i, \forall j \tag{78}$$

- We extend the problem to the case where at each demand point j there exist $|L|$ buyers.
- In this case the demand faced by the intermediary is a vector $\mathbf{r} = [r_j^\ell]$, where r_j^ℓ is the demand of buyer ℓ at demand point j .
- Let $x_i = \sum_{\ell=1}^{|L|} x_i^\ell$.
- Functions $o_k(\cdot)$ and $p_k(\cdot)$ are considered to be convex (non decreasing) and differentiable.
- The problem of the intermediary is to choose the quantities to be ordered from each manufacturer in order to meet the customers' demand at minimum cost.
- The problem can be mathematically modeled as:

Production-transportation problem in the presence of an intermediary and $|L|$ buyers

$$\textbf{(MITP)} \quad \min \quad \sum_{k=1}^{|K|} o_k(\mathbf{s}_k) + \sum_{i=1}^{|M|}\sum_{j=1}^{|N|} t_{ij}\left(\sum_{\ell=1}^{|L|} x_{ij}\right) \tag{79}$$

$$\text{s.t} \quad \sum_{j=1}^{|N|}\sum_{\ell=1}^{|L|} x_{ij}^{\ell} = x_i, \ \forall i \tag{80}$$

$$\sum_{i=1}^{|M|} x_{ij}^{\ell} = r_j^{\ell}, \ \forall j, \ \forall \ell \tag{81}$$

$$x_{ij}^{\ell} \geq 0, \ \forall i, \forall j \tag{82}$$

The Frank-Wolfe Algorithm

Consider the problem of minimizing a differentiable function over a polyhedral set, i.e.,'

$$\textbf{(P1)} \quad \begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in \mathcal{X}, \end{aligned}$$

where $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^n \ : \ \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ is a nonempty polyhedron.

- The algorithm is based on the linearization of the objective function.
- That is, given an iteration point $\mathbf{x}^k \in \mathcal{X}$, the algorithm approximates the objective with a first order Taylor expansion at $\mathbf{x}^k$, resulting in the linear programming subproblem

$$\textbf{(FW-SUB}^k\textbf{)} \quad \begin{aligned} \min \quad & \nabla f(\mathbf{x}^k)^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{x} \in \mathcal{X}, \end{aligned}$$

where the constant terms have been dropped from the objective function.

- The solution $\bar{\mathbf{x}}^k$ of this subproblem is used in the construction of the search direction of descent $\mathbf{d}^k = \bar{\mathbf{x}}^k - \mathbf{x}^k$.
- A line search on the interval $[0, 1]$ furnishes the next iterate $\mathbf{x}^{k+1}$, that is, $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{d}^k$, where $\alpha_k \in \arg\min_{\alpha \in [0,1]} f(\mathbf{x}^k + \alpha \mathbf{d}^k)$, and the process is repeated.

- If f in **P1** is convex, then an interesting aspect of the application of the Frank-Wolfe algorithm to **P1** is the generation of a lower bound on $f(\mathbf{x}^*)$ at each iteration point $\mathbf{x}^k$, namely $f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\bar{\mathbf{x}}^k - \mathbf{x}^k)$.
- However, these lower bounds are not monotonically increasing. Hence, at iteration k, the current lower bound is defined as $\mathrm{lbd}^k = \max\{\mathrm{lbd}^{k-1}, f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T(\bar{\mathbf{x}}^k - \mathbf{x}^k)\}$, where lbd^{k-1} is the incumbent lower bound, initially set to $-\infty$.
- In practice, the algorithm can be terminated once $f(\mathbf{x}^k) - \mathrm{lbd}^k \leq \epsilon_1$ or $\frac{f(\mathbf{x}^k) - \mathrm{lbd}^k}{f(\mathbf{x}^k)} \leq \epsilon_2$ for suitably chosen $\epsilon_1 > 0$ and $\epsilon_2 > 0$.

We observe further that due to the linearity of the objective function in **FW-SUB**^{k}, the subproblem separates into n problems, one for each factor in the Cartesian product when the algorithm is applied to the team game **CPP**:

$$\left.\begin{array}{ll} \min & \nabla_i f(\mathbf{x}^k)^T \mathbf{x}_i \\ \text{s.t.} & \mathbf{x}_i \in \mathcal{X}_i \end{array}\right\} \forall i \in \mathcal{N}, \tag{83}$$

Convergence of the F-W algorithm

Assumptions for the global convergence of the approach

(A1) $f(\mathbf{x})$ is continuously differentiable on $\mathcal{X}$,

(A2) $f(\mathbf{x})$ is pseudoconvex on $\mathcal{X}$, and

(A3) $\mathcal{X}$ is closed and bounded (i.e. a polytope).

- Assumption **A2** is essential for the verification of the global optimality of any accumulation point of the sequence $\{\mathbf{x}^k\}$.
- Assumption **A3** ensures that **P1** and **FW-SUB**^{k} have finite optima. It can be replaced by :
  - the coercivity assumption $\lim_{\|\mathbf{x}\|\to\infty} f(\mathbf{x}) = \infty$ on $\mathcal{X}$,
  - and the assumption that $\nabla f(\mathbf{x}^k)^T\mathbf{x}$ is bounded from below on $\mathcal{X}$ for all $\mathbf{x}^k \in \mathcal{X}$.

Theorem 16

Under the posed assumptions, the Frank-Wolfe algorithm either terminates finitely with an optimal solution of **P1** *or it generates an infinite sequence* $\{\mathbf{x}^k\}$ *of feasible points in* $\mathcal{X}$ *such that any of its accumulation points is an optimal solution of* **P1**.

Example 2

Consider the following instance of **CPP**:

$$
\begin{aligned}
\min \quad & (x_1 - 2x_4)^2 + (3x_2 - x_3)^2 + (x_1 - 2)^2 + \\
& (x_3 - 2)^2 + (x_2 - 5)^2 + (x_4 - 3)^2 \\
\text{s.t.} \quad & 3x_1 + 2x_2 \leq 6 \\
& 5x_3 + 2x_4 \leq 10 \\
& x_j \geq 0, \ j = 1, 2, 3, 4
\end{aligned}
$$

Its optimal solution is $(1.407, 0.890, 1.583, 1.042)$ and the corresponding optimal objective function value is 22.89.

If the algorithm is initialized with the origin, it must approach the optimal solution by following directions which are based on the alternate generation of the two extreme points $(\bar{x}_1^1, \bar{x}_2^1) = (2, 0)$ and $(\bar{x}_1^2, \bar{x}_2^2) = (0, 3)$ of the first simplex and the alternate generation of the two extreme points $(\bar{x}_3^1, \bar{x}_4^1) = (2, 0)$ and $(\bar{x}_3^2, \bar{x}_4^2) = (0, 5)$ of the second simplex, zig-zagging thus towards the optimal solution. This phenomenon is illustrated in Figure 2. The jamming starts as the steplengths α_k become smaller for each new iteration. Table 1 shows a few iterations of the algorithm. Note the rapid decrease of the objective function value in early iterations and the small decimal changes in later iterations. ■

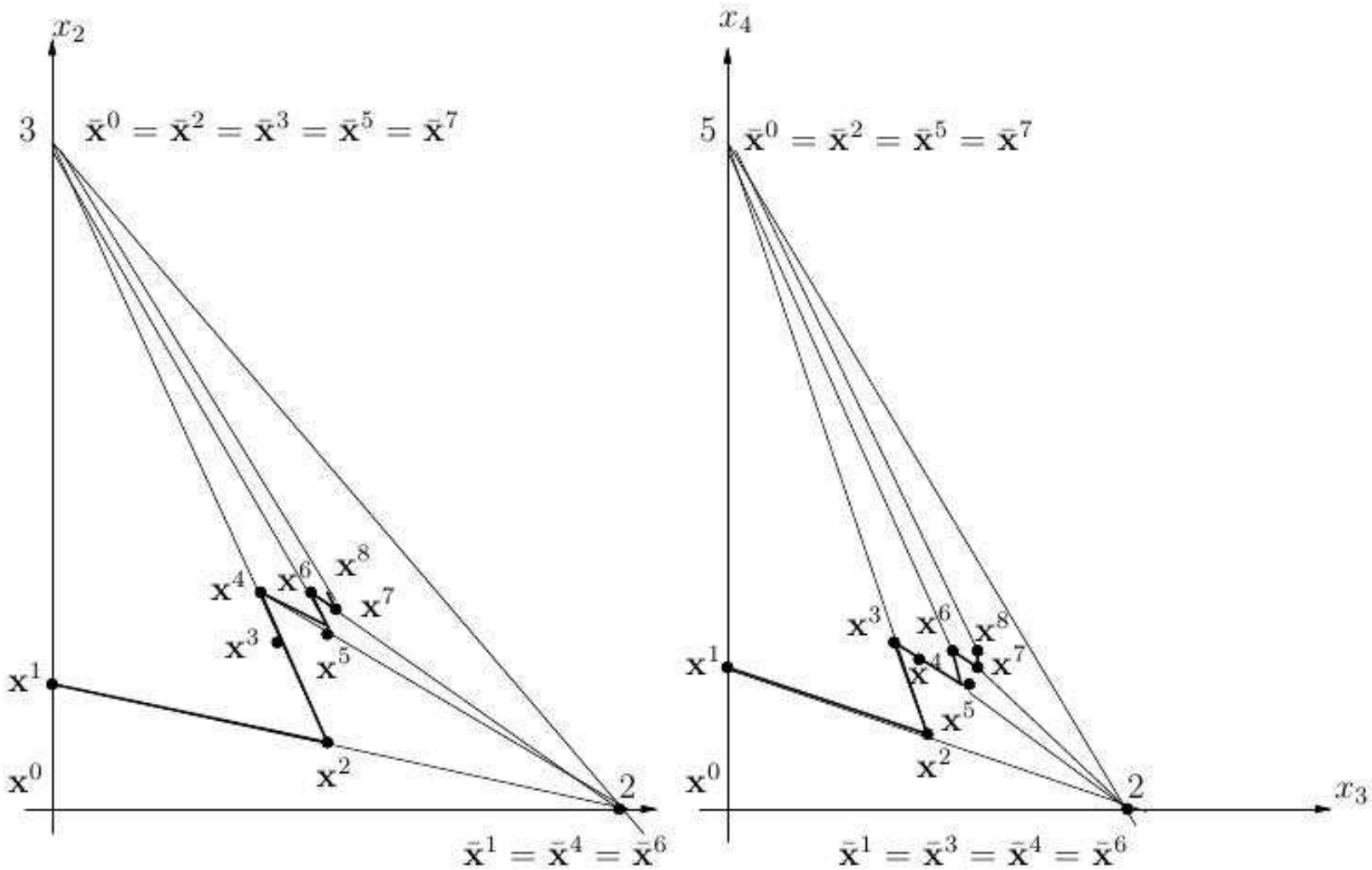


Figure 2: Movements of the Frank-Wolfe algorithm in the subproblem spaces

Table 1: Frank-Wolfe iterations

k	$\mathbf{x}^k$	$f(\mathbf{x}^k)$	$\bar{\mathbf{x}}_1^k$	$\bar{\mathbf{x}}_2^k$	lbd^k	$\mathbf{d}^k$	α_k
1	(0.000,0.000, 0.000,0.000)	42.000	(0.000,3.000)	(0.000,5.000)	-18.000	(0.000,3.000, 0.000,5.000)	0.139
2	(0.000,0.419, 0.000,0.695)	37.814	(2.000,0.000)	(2.000,0.000)	11.209	(2.000,-4.186, 2.000,-0.698)	0.432
3	(0.763,0.562, 0.763,0.936)	29.102					
33	(1.284,0.826, 1.428,1.015)	23.861	(0.000,3.000)	(2.000,0.000)	22.290	(-1.284,2.174, 0.572,-1.015)	0.0179
34	(1.261,0.865, 1.438,0.997)	23.847					

Feasible Direction Improvements

- The basic idea of improving the Frank-Wolfe algorithm is to not let the generated directions be based so heavily on the extreme points of the feasible region.
- This can be done basically in two different ways;
  1. either by avoiding the complete linearization of the objective function or
  2. by enriching the Frank-Wolfe subproblems with some nonlinear information.
- In both cases the original problem **P** is replaced iteratively by a sequence of easier (sub-) problems obtained by replacing the original objective function by a new one which may depend on the current iteration point.

Partial Linearization

- In this approach the original function $f(\cdot)$ should only be partially linearized, i.e,
- if $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i) + e(\mathbf{x})$, where $f_i(\cdot)$ are strictly convex functions and $e(\cdot)$ is not additively separable, then only $e(\cdot)$ needs to be linearized.
- Moreover, if $f(\cdot)$ does not have the necessary form, such a form can be enforced with the introduction of a second function $\varphi(\cdot)$, which may be assumed strictly convex and additively separable.
- Then, the original objective function $f(\cdot)$ is replaced by the equivalent $\varphi(\cdot) + [f(\cdot) - \varphi(\cdot)]$ and the "error" $e(\cdot) = f(\cdot) - \varphi(\cdot)$ is linearized.

In the case of the form of the **CPP**, (83) is replaced by the strictly convex subproblems

$$\left.\begin{array}{ll} \min & f_i(\mathbf{x}_j) + \nabla_i e(\mathbf{x}^k)^T \mathbf{x}_i \\ \text{s.t.} & \mathbf{x}_i \in \mathcal{X}_i \end{array}\right\} \ \forall i \in \mathcal{N}. \tag{84}$$

Letting $\bar{\mathbf{x}}^k = [\bar{\mathbf{x}}_1^k, \ldots, \bar{\mathbf{x}}_n^k]$ denote the point obtained by solving these subproblems, a feasible direction of descent $\mathbf{d}^k = \bar{\mathbf{x}}^k - \mathbf{x}^k$ is formed and the next iterate $\mathbf{x}^{k+1}$ is furnished by a line search on the interval $[0, \bar{\alpha}_k]$, where $\bar{\alpha}_k = \max\{\alpha \geq 0 | \mathbf{x}^k + \alpha \mathbf{d}^k \in \mathcal{X}\}$, that is, $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{d}^k$, where $\alpha_k \in \arg\min_{\alpha \in [0, \bar{\alpha}_k]} f(\mathbf{x}^k + \alpha \mathbf{d}^k)$.

Example 3

Partial linearization is applied to the problem instance of Example 2 by linearizing only the non-separable part of the objective function, i.e., $-4x_1x_4 - 6x_2x_3$, we then obtain the two subproblems:

$$\begin{aligned} (\text{SUB}^1) \quad \min \quad & 2(x_1)^2 + 10(x_2)^2 - (4 + 4\bar{x}_4)x_1 - (10 + 6\bar{x}_3)x_2 \\ \text{s.t.} \quad & 3x_1 + 2x_2 \leq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

and

$$\begin{aligned} (\text{SUB}^2) \quad \min \quad & 2(x_3)^2 + 5(x_4)^2 - (4 + 6\bar{x}_2)x_3 - (6 + 4\bar{x}_1)x_4 \\ \text{s.t.} \quad & 5x_3 + 2x_4 \leq 10 \\ & x_3, x4 \geq 0 \end{aligned}$$

Table 2 lists three iteration of the partial linearization algorithm, while Figure 3 illustrates the movements of the algorithm in the subproblems' simplices. ■

Table 2: Iterations of the partial linearization algorithm

k	$\mathbf{x}^k$	$f(\mathbf{x}^k)$	$\bar{\mathbf{x}}_1^k$	$\bar{\mathbf{x}}_2^k$	$\mathbf{d}^k$	α_k
0	(0.000,0.000, 0.000,0.000)	42.000	(1.000,0.500)	(1.000,0.600)	(1.000,0.500, 1.000,0.600)	1.500
1	(1.500,0.750, 1.500,0.900)	23.625	(1.410,0.885)	(1.556,1.109)	(-0.089,0.135, 0.056,0.209)	1.000
2	(1.410,0.885, 1.556,1.109)	22.913	(1.417,0.875)	(1.582,1.045)	(0.007,-0.010, 0.026,-0.064)	0.880
3	(1.416,0.876, 1.578,1.053)	22.894				

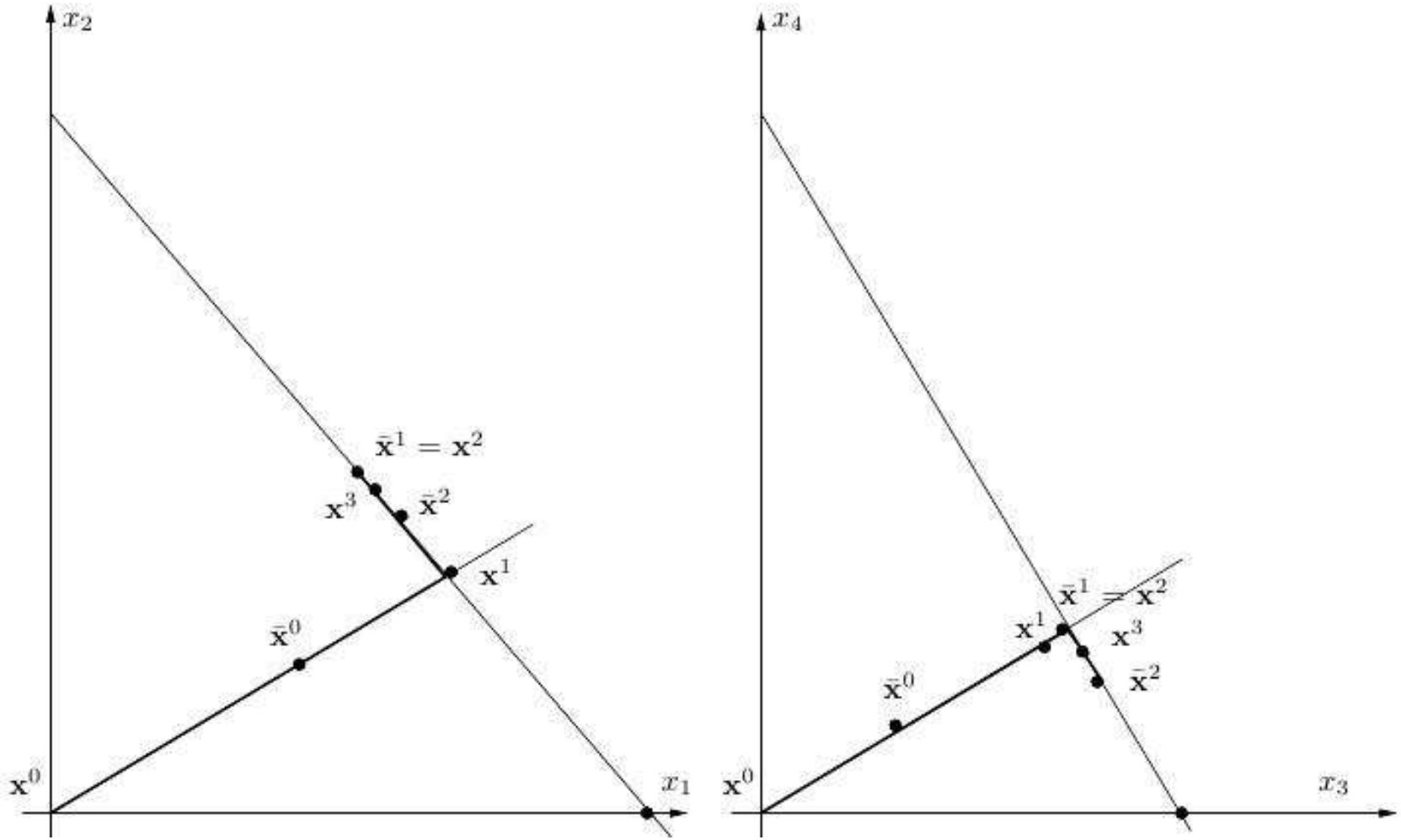


Figure 3: Movements of the partial linearization algorithm in the subproblem spaces

Regularized Frank-Wolfe

In regularized Frank-Wolfe algorithm a regularization function $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ satisfying the following properties:

(A4) ϕ is continuously differentiable on $\mathcal{X} \times \mathcal{X}$,

(A5) ϕ is nonnegative and convex on $\mathcal{X} \times \mathcal{X}$,

(A6) $\phi(\mathbf{x}, \mathbf{y})$ is strictly convex for every fixed $\mathbf{y} \in \mathcal{X}$,

(A7) $\phi(\mathbf{x}, \mathbf{y})$ is strictly convex for every fixed $\mathbf{x} \in \mathcal{X}$, and

(A8)

$$\nabla\phi(\mathbf{x}, \mathbf{y}) = [\nabla_{\mathbf{x}}\phi(\mathbf{x}, \mathbf{y}), \nabla_{\mathbf{y}}\phi(\mathbf{x}, \mathbf{y})] = [\mathbf{0}, \mathbf{0}] \text{ if and only if } \mathbf{x} = \mathbf{y}.$$

Examples of functions that satisfy the stated requirements include

- the proximal point function $\phi(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|^2$,
- the projection function $\phi(\mathbf{x}, \mathbf{y}) = \frac{1}{2}(\mathbf{x} - \mathbf{y})^T \mathbf{D}(\mathbf{x} - \mathbf{y})$, where $\mathbf{D}$ is a positive diagonal matrix, and
- the entropy function $\phi(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n}\{(x_i + \epsilon) \ln\left(\frac{x_i+\epsilon}{y_i+\epsilon}\right) - (x_i - y_i)\}$, where $\epsilon > 0$ is a small constant.

In the regularized Frank-Wolfe approach, the subproblems (83) in the original algorithm are replaced by

$$\left.\begin{array}{ll} \min & \nabla_i f(\mathbf{x}^k)^T \mathbf{x}_i + t_k \phi_i(\mathbf{x}_i, \mathbf{x}_i^k) \\ \text{s.t.} & \mathbf{x}_i \in \mathcal{X}_i \end{array}\right\} \ \forall i \in \mathcal{N}, \tag{85}$$

where $t_k > 0$ is some positive constant and
$\phi(\mathbf{x}, \mathbf{x}^k) = \sum_{i=1}^n \phi_i(\mathbf{x}_i, \mathbf{x}_i^k)$.
The next iterate $\mathbf{x}^{k+1}$ is computed in the same manner as in the partial linearization algorithm.

And **FW-SUB**^{k} is replaced by the regularized subproblem

$$\begin{array}{lll} \textbf{(RFW-SUB}^k\textbf{)} & \min & \nabla f(\mathbf{x}^k)^T \mathbf{x} + t_k \phi(\mathbf{x}, \mathbf{x}^k) \\ & \text{s.t.} & \mathbf{x} \in \mathcal{X}, \end{array}$$

Convergence of the Regularized Frank-Wolfe Algorithm

Lemma 1

If $\mathbf{x}^k$ *solves* **RFW-SUB**^{k}, *then* $\mathbf{x}^k$ *is also a solution to* **P1**. *The converse is also true.*

Lemma 2

Let $\bar{\mathbf{x}}^k$ *be the unique optimal solution in* **RFW-SUB**^{k} *and assume that* $\bar{\mathbf{x}}^k \neq \mathbf{x}^k$. *Then* $\mathbf{d}^k = \bar{\mathbf{x}}^k - \mathbf{x}^k$ *is a feasible direction of descent.*

Theorem 17

The regularized Frank-Wolfe algorithm either terminates in a finite number of iterations or it generates an infinite sequence $\{\mathbf{x}^k\}$ *such that any accumulation point is an optimal solution to* **P1**.

Cyclic Linearization

The cyclic linearization aims at to take advance of the new informations generated during the loop of the the subproblems (83) in the sequential implementation of the Frank-Wolfe algorithm. In this approach the loop (83) in the original algorithm is replaced by the following scheme:

$$\left.\begin{array}{rcl} \bar{\mathbf{x}}_i^k & \in & \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} \nabla_i f(\mathbf{x}_{i-}^{k+1}, \mathbf{x}_i^k, \mathbf{x}_{i+}^k)^T \mathbf{x}_i \\ \alpha_i^k & \in & \arg\min_{\alpha \in [0,1]} f(\mathbf{x}_{i-}^{k+1}, \mathbf{x}_i^k + \alpha(\bar{\mathbf{x}}_i^k - \mathbf{x}_i^k), \mathbf{x}_{i+}^k) \\ \mathbf{x}_i^{k+1} & = & \mathbf{x}_i^k + \alpha_i^k(\bar{\mathbf{x}}_i^k - \mathbf{x}_i^k) \end{array}\right\} \forall i \in \mathcal{N} \quad (86)$$

The PARTAN or parallel tangents approach

- The PARTAN or parallel tangents approach has been suggested in particular in order to provide an acceleration scheme for the Frank-Wolfe algorithm that overcomes to certain extend the zig-zagging behavior.
- Thus, given two consecutive Frank-Wolfe iterates, $\mathbf{x}^k$ and $\mathbf{x}^{k+1}$,
  1. a line search is performed along the descent direction $\bar{\mathbf{d}}^{k+1} = \mathbf{x}^{k+1} - \mathbf{x}^k$ and
  2. the new point $\bar{\mathbf{x}}^{k+1}$ obtained as $f(\bar{\mathbf{x}}^{k+1}) = f(\mathbf{x}^k + \alpha_k \bar{\mathbf{d}}^{k+1}) = \min_{\alpha \in [0, \bar{\alpha}_k]} f(\mathbf{x}^k + \alpha \bar{\mathbf{d}}^{k+1})$ replaces $\mathbf{x}^{k+1}$ in the next Frank-Wolfe iteration.

Simplicial Decomposition and Column Generation

Consider problem **P1**,using the Caratheodory's Theorem 4 it can be restated in the equivalent form:

$$\begin{aligned} \textbf{(MP1)} \quad \min_{\mathbf{y}} \quad & f\Big(\sum_{k=1}^{n} y_k \bar{\mathbf{x}}^k\Big), \\ \text{s.t.} \quad & \sum_{k=1}^{K} y_k = 1, \\ & y_k \geq 0, \ k = 1, \ldots, K, \end{aligned}$$

where $\bar{\mathbf{x}}^k$ are the extreme points of $\mathcal{X}$ and K is the number of its extreme points. If $\mathbf{x}^*$ is an optimal solution to **MP1**, the optimal solution to **P1** is calculated as $\mathbf{x}^* = \sum_{k=1}^{K} y_k^* \bar{\mathbf{x}}^k$. Recall that since $\mathcal{X} \subset \mathbb{R}^n$, by Caratheodory's theorem, at most $n+1$ extreme points are needed to represent the optimum point $\mathbf{x}^*$.

The problem **CPP**, can be restated as follows:

$$\textbf{(MP2)} \quad \min_{\mathbf{y}} \quad f\Big(\sum_{j=1}^{K_1} y_{1j}\bar{\mathbf{x}}_1^j, \ldots, \sum_{j=1}^{K_m} y_{mj}\bar{\mathbf{x}}_m^j\Big),$$

$$\text{s.t.} \quad \sum_{j=1}^{K_i} y_{ij} = 1, \ i = 1, \ldots, m,$$

$$y_{ij} \geq 0, \ j = 1, \ldots, K_i, \ i = 1, \ldots, m,$$

where $\bar{\mathbf{x}}_i^j$ are the K_i in total extreme points of the ith polytope $\mathcal{X}_i$ in the Cartesian product $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_m$. Clearly, **MP2** includes a convexity constraint for each polytope in the product and is therefore itself a minimization problem over a Cartesian product of simplices.

Hybridization of Simplicial Decomposition and Frank-Wolfe Regularization

Consider problem (**MP2**) in the following form

$$
\begin{aligned}
\textbf{(MP)} \qquad & \min_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{y}), \\
\text{s.t.} \quad & \\
& \mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m], \\
& \mathbf{y}_i \in \mathcal{Y}_i = \left\{ \mathbf{y}_i \Big| \sum_{j=1}^{K_i} y_{ij} = 1,\ y_{ij} \geq 0 \right\},\ i = 1, \ldots, m, \quad \sum_{i=1}^{m} K_i = n,
\end{aligned}
$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and non-separable and $\mathcal{Y} = \prod_{i=1}^{m} \mathcal{Y}_i$ is the Cartesian product of the m simplices. These conditions are also sufficient to guarantee the existence of a solution to (**MP**).

For any given feasible solution $\mathbf{y}^k = [\mathbf{y}_1^k, \ldots, \mathbf{y}_m^k]$, let $\mathbf{c}_i = \nabla_i f(\mathbf{y}^k),\ i = 1, \ldots, m$, denote the gradient of f with respect to $\mathbf{y}_i$ evaluated at $\mathbf{y}^k$. Then $\mathbf{P}$ is approximated at $\mathbf{y}^k$ by m linear subproblems:

$$\textbf{(FW)} \qquad \min \sum_{j=1}^{K_i} c_{ij} y_{ij}$$

$$\text{s.t.}$$

$$\sum_{j=1}^{n_i} y_{ij} = 1$$

$$y_{ij} \geq 0,\ j = 1, \ldots, n_i.$$

Let $\widehat{\mathbf{y}}_i$ be the solution to each **FW**, and let $\widehat{\mathbf{y}} = [\widehat{\mathbf{y}}_1, \ldots, \widehat{\mathbf{y}}_m]$. A search direction $\mathbf{d}^k = \widehat{\mathbf{y}} - \mathbf{y}^k$ is defined and a new feasible solution $\mathbf{y}^{k+1} = \mathbf{y}^k + \alpha_k \mathbf{d}^k$ is generated by finding $\alpha_k \in [0, 1]$ that minimizes $f(\mathbf{y}^k + \alpha \mathbf{d}^k)$.

(RFWP)

$$\min \sum_{j=1}^{K_i} c_{ij} y_{ij} + \phi_i(\mathbf{y}_i, \mathbf{y}_i^k)$$

s.t.

$$\sum_{j=1}^{K_i} y_{ij} = 1$$

$$y_{ij} \geq 0, \ j = 1, \ldots, n_i.$$

Corollary 1

If $\mathbf{y}^k = [\mathbf{y}_1^k, \ldots, \mathbf{y}_m^k]$ *solves the corresponding* m *subproblems* **RFWP**, *then* $\mathbf{y}^k$ *is also a solution to* **P**. *The converse is also true.*

Corollary 2

For feasible $\mathbf{y}^k = [\mathbf{y}_1^k, \ldots, \mathbf{y}_m^k]$, *let* $\widehat{\mathbf{y}}^k = [\widehat{\mathbf{y}}_1^k, \ldots, \widehat{\mathbf{y}}_m^k]$ *solve the corresponding* m *subproblems* **RFWP**, *and assume that* $\mathbf{y}^k \neq \widehat{\mathbf{y}}^k$. *Then* $\mathbf{d}^k = [\mathbf{d}_1^k, \ldots, \mathbf{d}_m^k] = \widehat{\mathbf{y}}^k - \mathbf{y}^k$ *is a feasible direction of descent of* f *at* $\mathbf{y}^k$

Corollary 3

For given feasible $\mathbf{y}^k$ *and any* $i = 1, \ldots, m$, **RFWP** *is feasible and its optimal solution is unique.*

Lemma 3

Let μ_i be the least eigenvalue of $\mathbf{D}_i$. Then $\nabla_i f(\mathbf{y}^k)^T \mathbf{d}_i^k \leq -\mu_i \|\mathbf{d}_i^k\|$.

Lemma 4

The maximum feasible step length is 1.

1. **algorithm** `regularized_fw`
2. Initialize $\mathbf{y}^0 \in \mathcal{Y}$, $\text{LBD}^{-1} \longleftarrow -\infty$, $k \longleftarrow 0$
3. **do**
4. $\quad \text{UBD}^k \longleftarrow f(\mathbf{y}^k)$
5. $\quad$ Solve all **FW** and compute LBD^k
6. $\quad$ Let $\widehat{\mathbf{y}}^k = [\widehat{\mathbf{y}}^k_1, \ldots, \widehat{\mathbf{y}}^k_m]$ be the unique solutions to all **RFWP**
7. $\quad \mathbf{d}^k \longleftarrow \widehat{\mathbf{y}} - \mathbf{y}^k$
8. $\quad$ **if** $(\mathbf{d}^k = 0)$ **exit**
9. $\quad \text{LBD}^k \longleftarrow \max\{\text{LBD}^{k-1}, \text{LBD}^k\}$
10. $\quad$ **if**($(\text{UBD}^k - \text{LBD}^k)/\text{LBD}^k \leq \epsilon$) **exit**
11. $\quad \alpha_k \longleftarrow \arg \min_{\alpha \in [0,1]} f(\mathbf{y}^k + \alpha \mathbf{d}^k)$
12. $\quad \mathbf{y}^{k+1} \longleftarrow \mathbf{y}^k + \alpha_k \mathbf{d}^k$
13. $\quad k \longleftarrow k + 1$
14. **end do**
15. **end** `regularized_fw`

Theorem 18

Under the additional assumption of Lipschitz continuity of the gradient, the `regularized_fw` *algorithm with Armijo steps converges globally to an optimal solution of* **MP**.

Polynomially bounded dual algorithm for the solution of the subproblems

We assume here that at iteration k, the regularization functions $\phi_i(\mathbf{y}_i, \mathbf{y}_i^k) = \frac{1}{2}(\mathbf{y}_i - \mathbf{y}_i^k)^T \mathbf{D}_i(\mathbf{y}_i - \mathbf{y}_i^k$ with diagonal $\mathbf{D}_i$, e.g., $\mathbf{D}_i = \text{diag}(\nabla_i^2 f(\mathbf{y}^k))$, perturbed if necessary to positiveness, are chosen for each $i \in \{1, \ldots, m\}$. Dropping constant terms and indexing, each subproblem is thus of the form:

$$\textbf{(QP)} \qquad \min \sum_{j=1}^n \frac{1}{2} d_j y_j^2 - l_j y_j$$

$$\text{s.t.}$$

$$\sum_{j=1}^n y_j = 1$$

$$0 \leq y_j \leq 1, \; j = 1, \ldots, n.$$

Let $\lambda \in \mathbb{R}$ be the Lagrangian multiplier associated with the equality constraint and consider the Lagrangian inner subproblem:

$$\textbf{[QP}(\lambda)\textbf{]} \qquad \min \sum_{j=1}^{n} \tfrac{1}{2} d_j y_j^2 + (\lambda - l)_j y_j$$

$$\text{s.t.}$$

$$0 \leq y_j \leq 1, \ j = 1, \ldots, n.$$

Since $d_j > 0, \forall j$, $\textbf{QP}(\lambda)$ has a unique finite solution $\mathbf{y}(\lambda)$ defined by

$$y_j(\lambda) = \begin{cases} 0, & \text{if } (l_j - \lambda)/d_j \leq 0 \\ 1, & \text{if } (l_j - \lambda)/d_j \geq 1 \\ (l_j - \lambda)/d_j, & \text{otherwise,} \end{cases}$$

Thus, the Lagrangian dual to **QP** is

$$\textbf{[QPLD]} \qquad \max \ -\lambda + \sum_{j=1}^{n} \tfrac{1}{2} d_j y_j(\lambda)^2 + (\lambda - l)_j y_j(\lambda).$$

The λ^* is computed using the bisection algorithm of the nonlinear programming.

1. Starting from an initial interval $[a_0, b_0]$ with $g(a_0)g(b_0) < 0$.
2. The midpoint of $\frac{(a_0+b_0)}{2}$ bisecting the $[a_0, b_0]$ is computed and tested.
3. At each iteration, the interval is reduced by half.
4. For a desired reduction of the initial interval of length ε, the maximum number of iterations required is $\log_2 \frac{b_0-a_0}{\varepsilon}$.
5. The final is obtained by linear interpolation on the final interval

$$\lambda^* = a_k + \frac{(b_k - a_k)(1 - g(a_k)}{g(b_k) - g(a_k)}. \tag{87}$$

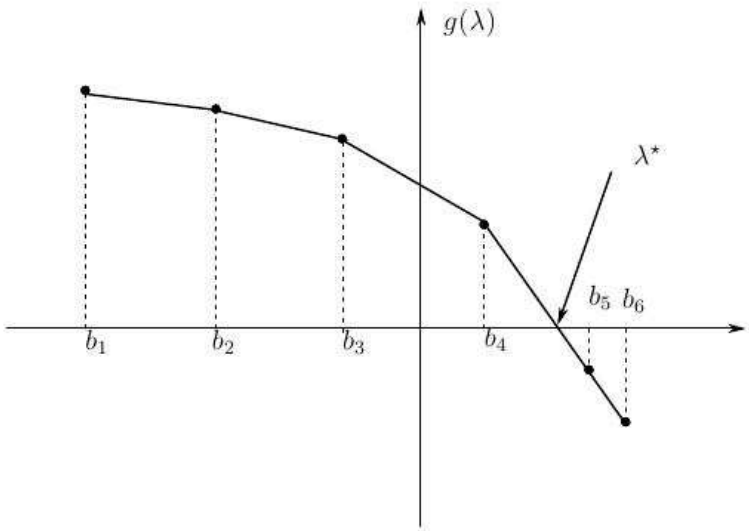


Figure 4: The derivative of the objective function in **QPLD**: Breakpoints and the optimal multiplier.

The application of the simplicial decomposition method to **STP** results in column generation sub-problems of the form:

$$\textbf{(CGS)} \quad \min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} \bar{c}_{ij}^{(k)} x_{ij} \tag{88}$$

$$\text{s.t} \quad \sum_{j=1}^{n} x_{ij} \leq q_i, \ \forall i \tag{89}$$

$$x_{ij} \geq 0 \ \forall i, \ \forall j, \tag{90}$$

where

$$\bar{c}_{ij}^{(k)} = \frac{\partial f(x^{(k)}, r^{(k)})}{\partial x_{ij}} + \frac{\partial f(x^{(k)}, r^{(k)})}{\partial r_j}, \forall i, \ \forall j \tag{91}$$

with

$$r_j^{(k)} = \sum_{i=1}^{m} x_{ij}^{(k)}, \ \forall j, \tag{92}$$

and $f(x, r)$ and denotes the objective function of the initial

Table 3: Comparison of the runtime of the algorithms

Problem size	Execution time (secs)			
n	Helgason	Brucker	Pardalos	Bisection $\varepsilon = 0.00001$)
100	5.99E-3	3.000E-3	7.998E-3	1.000E-3
500	3.99E-2	4.099E-2	2.599E-2	2.000E-3
1000	9.29E-2	9.399E-2	6.499E-2	3.999E-3
2000	0.112	9.69E-2	7.099E-2	3.999E-3
5000	0.257	0.263	0.178	1.199E-2
7000	0.390	0.398	0.253	1.500E-2
10000	0.619	0.628	0.312	2.200E-2

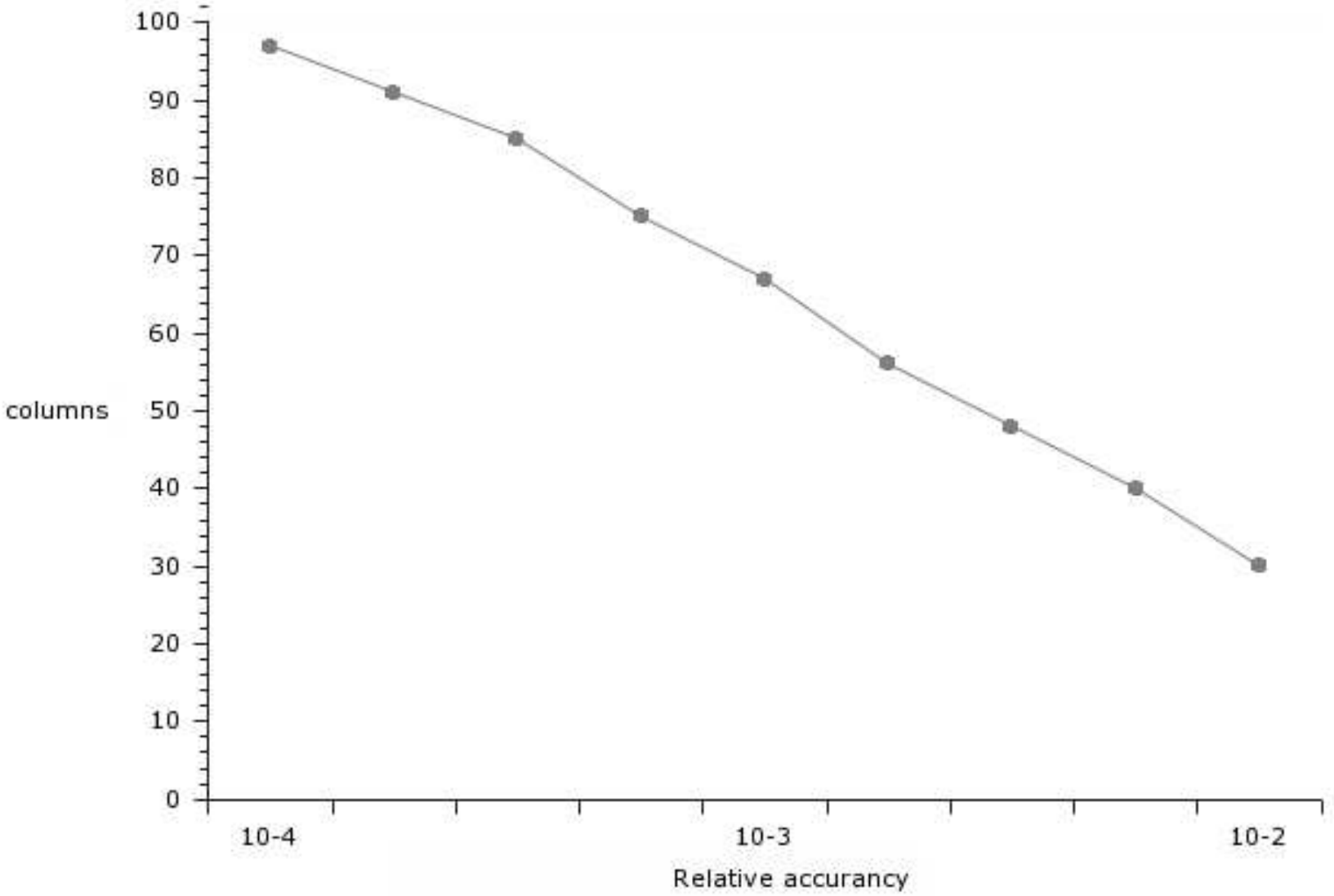


Figure 5: Size of restricted master problem in columns compared to relative accuracy

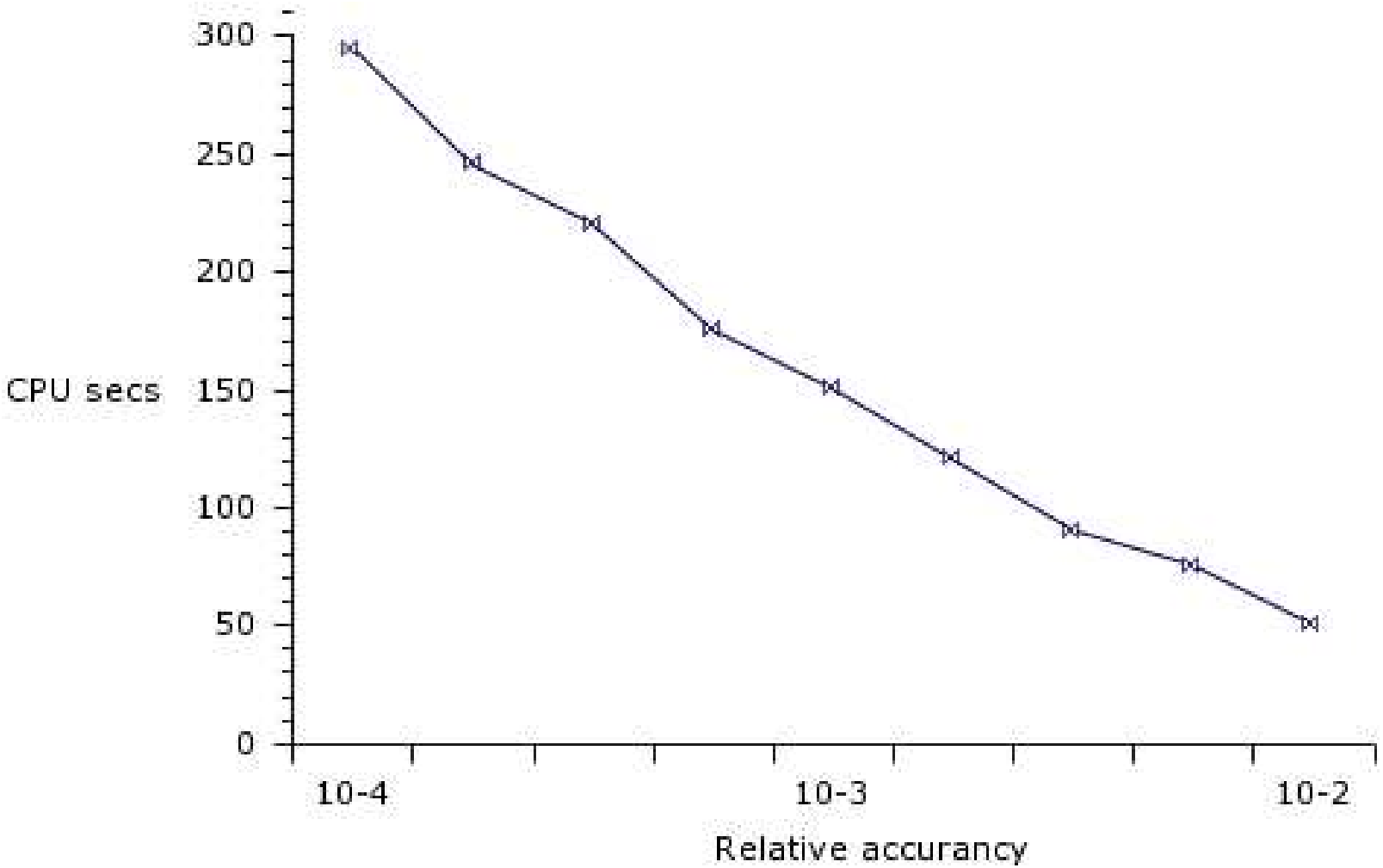
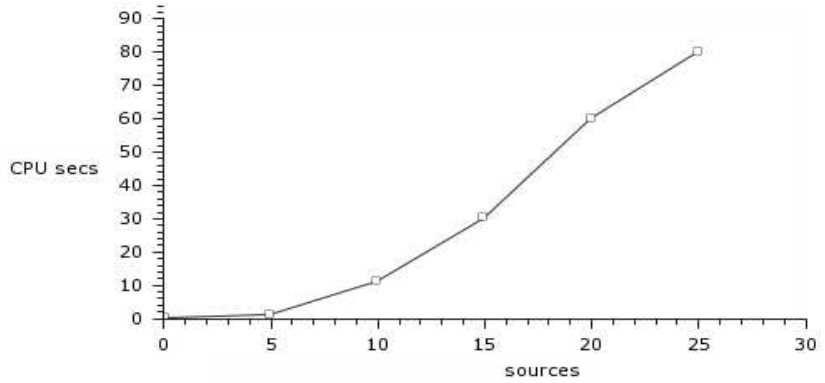


Figure 6: Computational time compared to the required relative accuracy

Problems created by the principle of Cooper and LeBlanc



Problems created by the principle of LeBlanc et al.

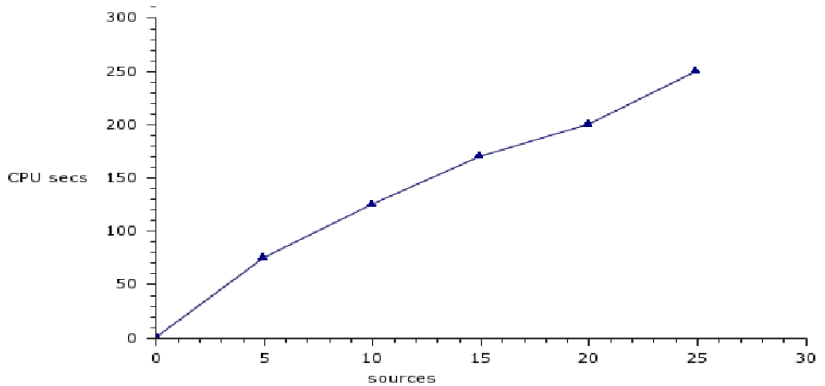


Figure 7: Comparison of the computational results and the problem's size with accuracy 0.01 %

Tailoring the Algorithm to the Traffic Assignment

Solving **FTAP** requires repeated solutions of restricted master problems in the form **MP** and more specifically of the following form when using disaggregate representation :

$$\textbf{(RMP)} \qquad \min \sum_{a \in \mathcal{A}} f_a\Big(\sum_{k \in \mathcal{K}} r_k \sum_{p \in \Pi_k} \delta_{kap} \lambda_{pk}\Big)$$

s.t.

$$\sum_{p \in \Pi_k} \lambda_{pk} = 1, \ \forall k \in \mathcal{K}$$

$$\lambda_{pk} \geq 0, \ \forall p \in \mathcal{P}_k, \ \forall k \in \mathcal{K}$$

where λ_{pk} is the fraction of the total flow origin-destination (O-D) demand on path $p_k \in \mathcal{P}_k$.

If the **RMP** solution $[\lambda_{pk}]$ does not solve **FTAP**, where $h_{pk} = r_k \lambda_{pk}, p \in \mathcal{P}_k, k \in \mathcal{K}$, then the set $\mathcal{P}_k$ is augmented by solving the following column generating shortest path subproblems which are separated over k and are obtained by linearizing **AFTAP** at the current flow $x_a = \sum_{k \in \mathcal{K}} \sum_{p \in \Pi_k} \delta_{kap} h_{pk}$ (where $c_a = s_a(x_a)$):

$$[\mathbf{SPP}_k] \qquad \min \ \widehat{f}_k(\mathbf{x}) = \sum_{a \in \mathcal{A}} c_a x_a^k$$

s.t.

$$\sum_{a \in \mathcal{S}(i)} x_a^k - \sum_{a \in \mathcal{T}(i)} x_a^k = \left\{ \begin{array}{rl} r_k & \text{if } o(k) = i \\ -r_k & \text{if } d(k) = i \\ 0 & \text{otherwise} \end{array} \right\} \forall i \in \mathcal{N}$$

$$x_a^k \geq 0, \ \forall a \in \mathcal{A}$$

Applying the `regularized_fw` to the **RMP**, we can compute the **QP** objective terms as $d_{pk} = r_k^2 \sum_{a \in \mathcal{A}} \delta_{kap} s'_a(x_a),\ p \in \mathcal{P}_k, k \in \mathcal{K}$, and $l_{pk} = d_{pk}\lambda_{pk} - r_k \sum_{a \in \mathcal{A}} \delta_{kap} s_a(x_a),\ p \in \mathcal{P}_k, k \in \mathcal{K}$.

The hybrid algorithm for the traffic assignment problem I

algorithm DSD-RFW

Heuristic. Generate the first path for each O-D pair at zero arc flow, and assign the full O-D demand to the path, i.e., perform an all-or-nothing assignment.

1. $x_a \longleftarrow 0$, $c_a \longleftarrow s_a(x_a)$, $\mathcal{P}_k \longleftarrow \emptyset$.
2. $\widehat{p}_k \Longleftarrow$ Solve **SPP**$_k$, $\lambda_{\widehat{p}_k} \longleftarrow 1$, $\mathcal{P}_k \longleftarrow \mathcal{P}_k \cup \widehat{p}_k$.
3. $x_a \longleftarrow \sum_{k\in\mathcal{K}} r_k \sum_{p\in\mathcal{P}_k} \delta_{kap}\lambda_{pk}$.
4. UBD $\longleftarrow f(\mathbf{v})$, LBD $\longleftarrow -\infty$.

Main solver. Generate paths based on the current arc delays. Augment the set of generated paths. Solve the master problem over this restricted set.

Column generation. Shortest path based on current arc delays. Augment the set of the generated paths if not previously included. Compute a lower bound on the objective.

5. $c_a \longleftarrow s_a(x_a)$.
6. $\{\widehat{f}_k(\mathbf{x}),\ \widehat{p}_k\} \Longleftarrow$ Solve **SPP**$_k$.
7. If $\widehat{p}_k \notin \mathcal{P}_k$ then $\mathcal{P}_k \longleftarrow \mathcal{P}_k \cup \widehat{p}_k$, $\lambda_{\widehat{p}_k} \longleftarrow 0$.
8. LBD$\longleftarrow \max\{\text{LBD}, \text{UBD} + \sum_{k\in\mathcal{K}} \widehat{f}_k(\mathbf{x}) - \sum_{a\in\mathcal{A}} c_a x_a\}$.

Convergence test. Terminate the algorithm if the relative objective error is lower than some a priori set constant.

The hybrid algorithm for the traffic assignment problem II

9. If $|(\text{UBD-LBD})/\text{LBD}| \leq \epsilon_1$ then terminate.

Restricted master. Solve the restricted master defined for the set of generated paths. In each iteration, a **QP** is solved for each O-D pair.

10. $\text{MLBD} \longleftarrow -\infty$.
11. $d_{pk} \longleftarrow r_k^2 \sum_{a \in \mathcal{A}} \delta_{kap} s'_a(x_a)$, $l_{pk} \longleftarrow d_{pk}\lambda_{pk} - r_k \sum_{a \in \mathcal{A}} \delta_{kap} s_a(x_a)$.
12. $\text{MLBD} \longleftarrow \max[\text{MLBD}, \text{UBD} + \sum_{k \in \mathcal{K}} r_k \min_{p_k \in \mathcal{P}_k} \{\sum_{a \in \mathcal{A}} \delta_{kap} s_a(x_a)\} - \sum_{a \in \mathcal{A}} c_a x_a]$.
13. If $|(\text{UBD-MLBD})/\text{MLBD}| \leq \epsilon_2$ then the master is sufficiently solved. Goto 5.
14. $\lambda_{pk}^{\text{old}} \longleftarrow \lambda_{pk}$, $x_a^{\text{old}} \longleftarrow x_a$.
15. $\lambda_{pk} \Longleftarrow$ Solve **QP**.
16. $x_a \longleftarrow \sum_{k \in \mathcal{K}} r_k \sum_{p \in \mathcal{P}_k} \delta_{kap} \lambda_{pk}$.
17. $\lambda_{pk}^{\text{dir}} \longleftarrow \lambda_{pk} - \lambda_{pk}^{\text{old}}$, $x_a^{\text{dir}} \longleftarrow x_a - x_a^{\text{old}}$.
18. step$\Longleftarrow$Apply Armijo line search.

18′. (or step$\longleftarrow \min\{1, -\sum_{a \in \mathcal{A}} s_a(x_a^{\text{old}}) x_a^{\text{dir}} / \sum_{a \in \mathcal{A}} s'_a(x_a^{\text{old}})(x_a^{\text{dir}})^2\}$)

19. $\lambda_{pk} \longleftarrow \lambda_{pk}^{\text{old}} + \text{step} * \lambda_{pk}^{\text{dir}}$, $x_a \longleftarrow x_a^{\text{old}} + \text{step} * x_a^{\text{dir}}$.
20. $\text{UBD} \longleftarrow f(\mathbf{x})$.
21. Goto 11.

na doume ti tha bgalei tora

Computational Results I

- The algorithm is implemented in C++
- Nine tests on three real world networks requesting three levels of relative objective error:
  - (i) the network of the city of Barcelona, Spain, which has 1020 nodes, 2522 arcs and 7922 origin-destination (O-D) pairs,
  - (ii) the network of Linköping, Sweden, which has 335 nodes, 882 arcs and 12372 O-D pairs, and
  - (iii) the network of Winnipeg, Canada, which has 1052 nodes, 2836 arcs and 4344 O-D pairs.
- (`DSD-RFW`) is compared to the Fortran 77 code (`DSD`) by Larson and Patriksson (1992)
- The essential difference between the two codes is the restricted master solver, where `DSD` uses a reduced gradient technique, while our `DSD-RFW` utilizes the `regularized_fw` algorithm.

Computational Results II

- We adjusted the parameter γ (with $\beta = 2$ constant) in the Armijo line search procedure to obtain the fastest possible running times for both codes.
- The best γ was always between 0.2 and 0.3.
- We also tested various C++ and Fortran compilation parameters since this has great impact on the computing speed, resulting in the following switches when using Sun (Oracle) C++ and Fortran compilers:
  - `-Bstatic -dalign -fsimple -O5 -xlibmopt -xlibmil`.

Computational Results III

Table 4: Statistics on CPU time (in seconds), number of main iterations (#MI), and number of line searches (#LS) for algorithms `DSD` and `DSD-RFW`. Relative objective error achieved: 0.5%.

Network	Code	CPU	#MI	#LS
Barcelona	`DSD`	61.3	5	100
	`DSD-RFW`	31.3	10	28
Linköping	`DSD`	54.2	4	20
	`DSD-RFW`	48.3	6	12
Winnipeg	`DSD`	53.8	6	114
	`DSD-RFW`	30.4	12	35

Computational Results IV

Table 5: Statistics on CPU time (in seconds), number of main iterations (#MI), and number of line searches (#LS) for algorithms `DSD` and `DSD-RFW`. Relative objective error achieved: 0.1%.

Network	Code	CPU	#MI	#LS
Barcelona	`DSD`	92.6	6	149
	`DSD-RFW`	58.8	10	59
Linköping	`DSD`	53.8	7	94
	`DSD-RFW`	31.7	6	36
Winnipeg	`DSD`	135.5	7	281
	`DSD-RFW`	59.6	11	81

Computational Results V

Table 6: Statistics on CPU time (in seconds), number of main iterations (#MI), and number of line searches (#LS) for algorithms `DSD` and `DSD-RFW`. Relative objective error achieved: 0.05%.

Network	Code	CPU	#MI	#LS
Barcelona	`DSD`	114.3	7	181
	`DSD-RFW`	81.3	12	83
Linköping	`DSD`	106.3	7	185
	`DSD-RFW`	65.9	8	77
Winnipeg	`DSD`	187.7	8	382
	`DSD-RFW`	114.3	15	159

Distributed System Implementation I

In this subsection we describe a synchronous single program multiple data (SPMD) implementation of the hybridized simplicial decomposition and regularized Frank-Wolfe algorithm as adapted by Damberg and Migdalas (1997) There are four main reasons for this choice:

i) the convergence properties are exactly the same as for the sequential algorithm,

ii) it is relatively easy to implement since the compute nodes (processors) are doing (virtually) the same thing,

iii) the **FTAP** problem structure suits this parallelization model nicely, and

iv) the approach is easily adapted to modern shared-memory multicore systems under the OpenMP parallel computing model.

Data structures and distribution I

By using the **TAP**$_h$ model, we can readily see that there are two main data structures:

1. **Network**.
   - The network is defined by the number of nodes, links (with corresponding delay functions) and the underlying graph.
   - The graph is stored as a sparse adjacency list of size $|\mathcal{N}| + |\mathcal{A}|$ and the link data (flow, three delay function terms, and two for temporary calculations) are stored in vectors of size $|\mathcal{A}|$.
   - All compute nodes hold the entire network data so (given that the compute nodes know the present link flow) the link delay can be computed i parallel and the shortest path problems can be solved in parallel without any communication.
2. **OD-pairs**.
   - An OD-pair is defined by its origin, destination and the demand of flow that is to carried between the two.

Data structures and distribution II

- All processors hold this (constant) information (of size $O(|\mathcal{K}|)$ to reduce the amount of data to be communicated if load balancing is used.
- Furthermore, in the disaggregated simplicial decomposition case, there are also routes associated with the OD-pairs.
- For each OD-pair there is a structure holding a linked list of the routes (a dynamically allocated vector of network link indices) which has been generated, i.e., the set $\mathcal{P}_k$ (or $\mathcal{L}_k$) from the subproblem phase $\mathbf{SPP}_k$. The same structure also holds the present route flow.
- The total size of these structures can be estimated with $O(|\mathcal{K}||\mathcal{A}|)$, since, in general, there are only a handful of routes generated per OD-pair in an user equilibrium solution. This is the (possibly huge set of) data which is to be divided among the compute nodes.

Data structures and distribution III

- By distributing the OD-pairs over the processors we obtain a data distribution which allows for the communication-less solution of the route generating shortest path problems $\mathbf{SPP}_k$ in parallel.
- Furthermore, within the master problem **RMP** we can solve the quadratic knapsack problems (see §1) in parallel without communication.

Communication (message passing) I

- Only one type (essentially) of communication is used in the proposed algorithm.
- Denote it `reduce-add_and_multicast` for further reference.
- It works as follows:
  1. all nodes compute their share of the data in question.
  2. A 'reduction with add' operation is then performed to gather and compute (summation) the total result,
  3. which in its turn is sent (multicasted) back to all compute nodes.
- Optimal algorithms and their time complexity for these operations can be found in the literature.

Load balancing I

- It performs local load balancing between pairs of processors by comparing their compute time and transferring OD-pairs, i.e., the structure holding the route data (see Section 1, item 2) in order to attempt to equalize the compute time (see Figure 8).
- Denote this procedure `balance-load` for further reference.

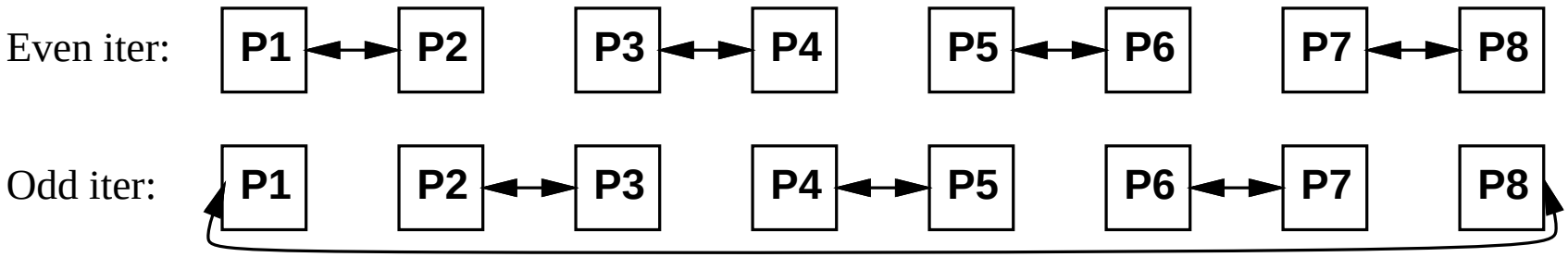


Figure 8: Local load balancing scheme.

Load balancing II

- Computational tests has proven this to be efficient, especially if the algorithm is used for solving several snapshots of the network with varying OD-pair demand for each snapshot.
- One snapshot is in general solved with only a handful major iterations, so there is little to gain in this case.

The distributed disaggregate simplicial decomposition algorithm I

- Let $\mathcal{C}$ denote the set of compute nodes and
- let $\mathcal{K}^c$ be the set of OD-pairs that compute node $c \in \mathcal{C}$ deals with. Clearly $\cup_{c \in \mathcal{C}} \mathcal{K}^c = \mathcal{K}, \mathcal{K}^i \cap \mathcal{K}^j = \emptyset, \forall i, j \in \mathcal{C}, i \neq j$.
- Let $\mathbf{x}^c$ denote the vector of link flows $[x_a^c]$, where $x_a^c = \sum_{k \in \mathcal{K}^c} \sum_{p_k \in \mathcal{P}_k} h_{pk} \delta_{kap}$, i.e., the part of the link flow that the generated routes held by compute node c carry.
- Let $\widehat{f}^c(\mathbf{x}) = \sum_{k \in \mathcal{K}^c} \widehat{f}_k(\mathbf{x})$ be the total shortest path cost for problems $\mathbf{SPP}_k$ which are solved by processor c.
- Finally, define UBD as the upper bound on the objective and by LBD the lower bound.
- Note that all processors run the same program, but each have different data sets (which correspond to $\mathcal{K}^c$), i.e., it is a SPMD algorithm.

Algorithm SPMD-DDSD I

Initialization. *Data input and initial distribution of OD-pairs. The $|\mathcal{K}|$ OD-pairs are evenly distributed over the $|\mathcal{C}|$ compute nodes.*

Heuristic. *Generate the first route for each OD-pair at zero link flow, and assign the full OD demand to the route — an all-or-nothing assignment.*

1. $\mathbf{x} \longleftarrow 0, \quad c_a \longleftarrow s_a(\mathbf{x}), \quad \mathcal{P}_k \longleftarrow \emptyset.$
2. $\widehat{p}_k \Longleftarrow$ Solve $\mathbf{SPP}_k(c_a, \ldots)$,
   $h_{\widehat{p}k} \longleftarrow r_k, \quad \mathcal{P}_k \longleftarrow \mathcal{P}_k \cup \widehat{p}_k.$
3. $\mathbf{x} \Longleftarrow$ `reduce-add_and_multicast`$(\mathbf{x}^c)$
4. UBD $\longleftarrow f(\mathbf{x}), \quad$ LBD $\longleftarrow -\infty$

Algorithm SPMD-DDSD II

Main solver. *Generate routes based on the current link delays. Augment the set of generated routes. Solve the master problem over the restricted set of generated routes.*

Subproblem solver. *Shortest path based on current link delays. Augment the set of generated routes if not previously included. Compute a lower bound on the objective.*

5. $c_a \longleftarrow s_a(\mathbf{x})$.
6. $\{\widehat{f}_k(\mathbf{x}), \quad \widehat{p}_k\} \Longleftarrow$ Solve $\mathbf{SPP}_k(c_a, \ldots)$
   If $\widehat{p}_k \notin \mathcal{P}_k$ then $\mathcal{P}_k \longleftarrow \mathcal{P}_k \cup \widehat{p}_k, \quad h_{\widehat{p}k} \longleftarrow 0$.
7. $\widehat{f}(\mathbf{x}) \Longleftarrow$ `reduce-add_and_multicast`$(\widehat{f}^c(\mathbf{x}))$
8. $\mathrm{LBD} \longleftarrow \max\{\mathrm{LBD}, \mathrm{UBD} + \widehat{f}(\mathbf{x}) - \sum_{a \in \mathcal{A}} c_a x_a\}$

Convergence test. *Terminate algorithm if the relative objective error is below some a priori set constant.*

Algorithm SPMD-DDSD III

9. If $(\text{UBD} - \text{LBD})/\text{LBD} \leq \varepsilon$ then Terminate!

Restricted master solver. *Solve the equilibrium problem over the restricted set of generated routes. Each iteration the objective is approximated with a separable (over the OD-pairs) quadratic function; see §1.*

10. $d_{pk} \longleftarrow \sum_{a \in \mathcal{A}} \delta_{kap} s'_a(x_a), \quad l_{pk} = \sum_{a \in \mathcal{A}} \delta_{kap} s_a(x_a).$
11. $h^{\text{old}}_{pk} \longleftarrow h_{pk}, \quad x^{\text{old}}_a \longleftarrow x_a$
12. $h_{pk} \Longleftarrow$ Solve $\mathbf{QPP}_k(h^{\text{old}}_{pk}, \ldots)$
13. $\mathbf{x} \Longleftarrow \texttt{reduce-add\_and\_multicast}(\mathbf{x}^c)$
14. $h^{\text{dir}}_{pk} \longleftarrow h_{pk} - h^{\text{old}}_{pk}, \quad x^{\text{dir}}_a \longleftarrow x_a - x^{\text{old}}_a$
15. $\text{step} \longleftarrow \min\{1, -\sum_{a \in \mathcal{A}} s_a(x^{\text{old}}_a) x^{\text{dir}}_a / \sum_{a \in \mathcal{A}} s'_a(x^{\text{old}}_a)(x^{\text{dir}}_a)^2\}$

Algorithm SPMD-DDSD IV

16. $h_{pk} \longleftarrow h_{pk}^{\text{old}} + \text{step} * h_{pk}^{\text{dir}}$,
    $x_a \longleftarrow x_a^{\text{old}} + \text{step} * x_a^{\text{dir}}$
17. UBD $\longleftarrow f(\mathbf{x})$
18. Terminate master after (a priori set) number of iterations. Return new equilibrium flow.
19. Goto 10.

Load equalization. *Re-distribute OD-pair data to obtain equal running times for the compute nodes.*

20. `balance-load`
21. Goto 5.

Implementation and Computational Results I

Table 7: Summary of network data

Network	# nodes	# links	# OD-pairs	OD-pair to link ratio
Barcelona	1020	2522	7922	3.1
Linköping	335	882	12372	14.0
Winnipeg	1052	2836	4344	1.5

Implementation and Computational Results II

Figure 9: GIS map of the flow on the Linköping network.

Table 8: Wall clock time (not including I/O to disk) in seconds for one snapshot the networks. Requested relative objective error: 0.1 %.

Network	Platform	Number of processors 1	2	4	8	16
Barcelona	MIMD Computer	44.0	27.3	19.5	13.1	12.9
	Multi-computer cluster	30.5	14.6	11.0	7.6	—
Linköping	MIMD Computer	49.8	35.2	22.0	12.0	9.0
	Multi-computer cluster	36.6	17.5	12.3	6.99	—
Winnipeg	MIMD Computer	60.1	33.4	22.7	18.8	18.5
	Multi-computer cluster	39.9	20.1	15.2	12.6	—

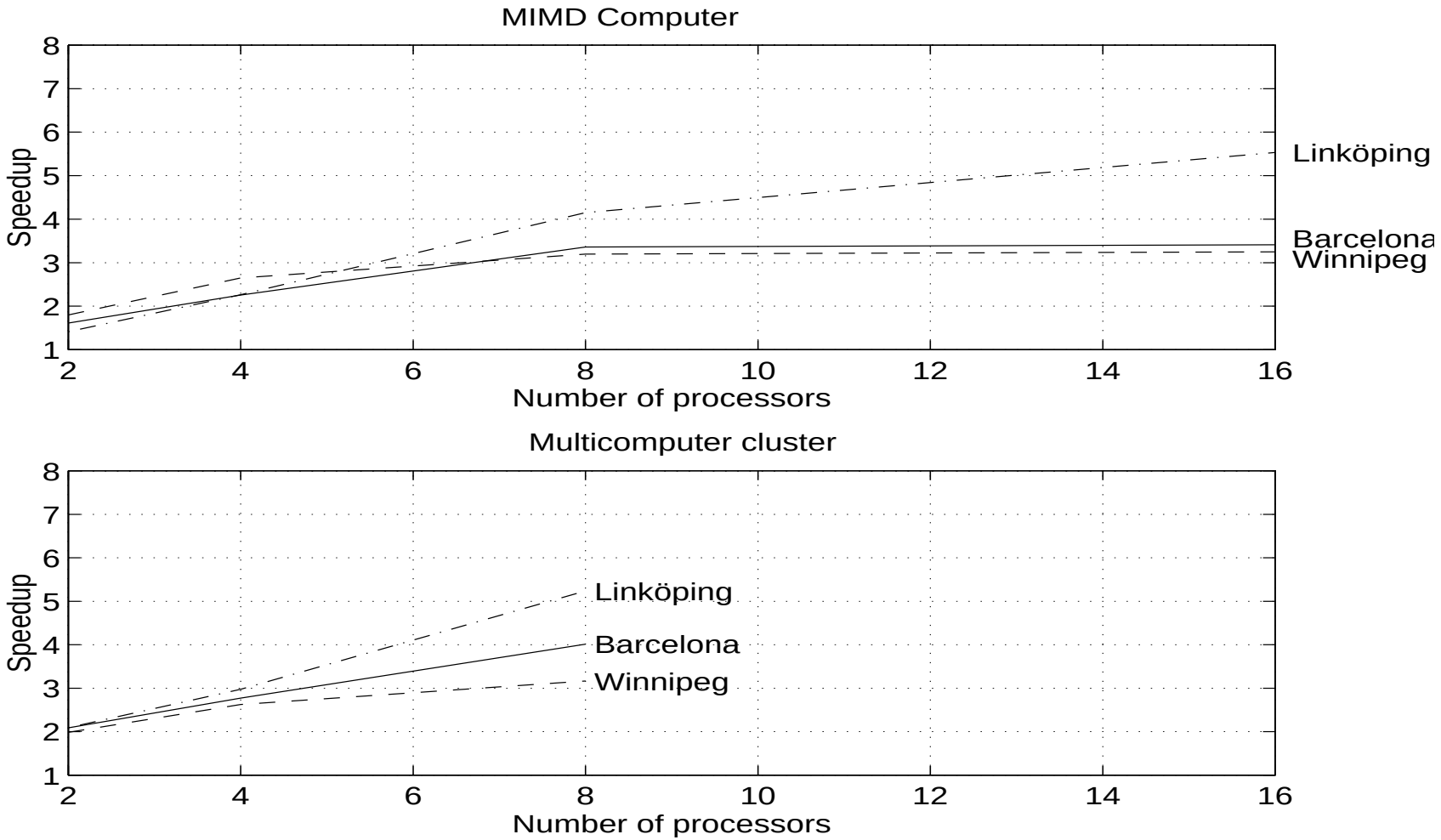

MIMD Computer
Speedup
Number of processors
Linköping
Barcelona
Winnipeg
Multicomputer cluster
Speedup
Number of processors
Linköping
Barcelona
Winnipeg

Figure 10: Speed-up for one snapshot of all three networks (data from Table 8).

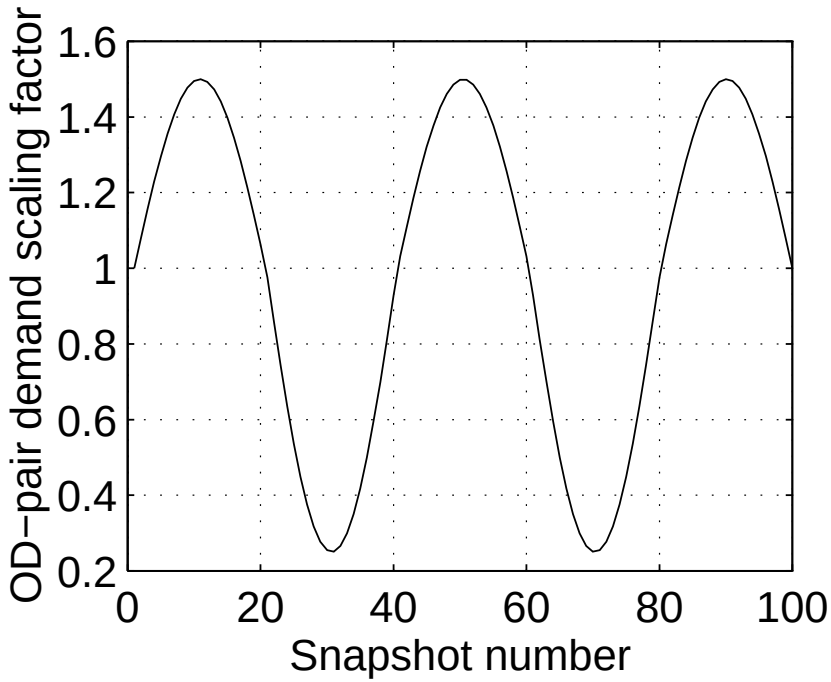


Figure 11: OD-pair demand scaling of original demand for each snapshot.

Table 9: Wall clock time (not including I/O to disk) in seconds for 100 snapshots of the networks. Requested relative objective error for each snapshot: 0.5%.

Network	Platform	Number of processors 1	2	4	8	16
Barcelona	MIMD Computer	2889	1520	822	576	561
	Multi-computer cluster	1970	987	526	356	—
Linköping	MIMD Computer	7426	4014	2196	1329	1045
	Multi-computer cluster	3898	1954	1035	675	—
Winnipeg	MIMD Computer	3294	1734	925	708	689
	Multi-computer cluster	2281	1150	633	450	—

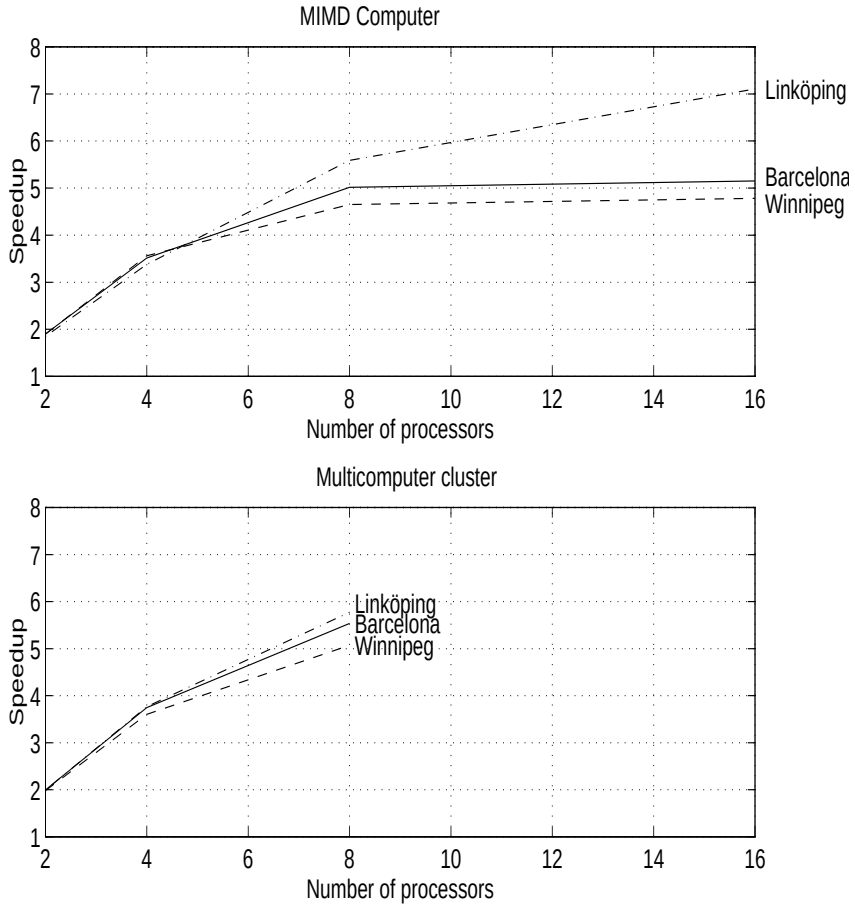


Figure 12: Speed-up for 100 snapshots of all three networks (data from Table 9)