Numerical Methods

V. Leclère (ENPC)

February 21st, 2023

Contents

The set-up

- $G = (V, E)$ is a directed graph
- x_e for $e \in E$ represent the flux (number of people per hour) taking edge e
- $\ell_e : \mathbb{R} \to \mathbb{R}^+$ the cost incurred by a given user to take edge e
- We consider K origin-destination vertex pair $\{o^k, d^k\}_{k \in [K]}$, such that there exists at least one path from o^k to d^k.
- r_k is the rate of people going from o^k to d^k
- $\mathcal{P}_k$ the set of all simple (i.e. without cycle) path form o^k to d^k
- We denote f_p the flux of people taking path $p \in \mathcal{P}_k$

Some physical relations

People going from o^k to d^k have to choose a path

$$r^k = \sum_{p \in \mathcal{P}^k} f_p .$$

People going through an edge are on a simple path taking this edge

$$x_e = \sum_{p \ni e} f_p .$$

The fluxes are non-negative

$$\forall p \in \mathcal{P}, \quad f_p \geq 0, \qquad \text{and} \qquad , \forall e \in E, \quad x_e \geq 0$$

Some physical relations

People going from o^k to d^k have to choose a path

$$r^k = \sum_{p \in \mathcal{P}^k} f_p.$$

People going through an edge are on a simple path taking this edge

$$x_e = \sum_{p \ni e} f_p.$$

The fluxes are non-negative

$$\forall p \in \mathcal{P}, \quad f_p \geq 0, \qquad \text{and} \qquad , \forall e \in E, \quad x_e \geq 0$$

Some physical relations

People going from o^k to d^k have to choose a path

$$r^k = \sum_{p \in \mathcal{P}^k} f_p.$$

People going through an edge are on a simple path taking this edge

$$x_e = \sum_{p \ni e} f_p.$$

The fluxes are non-negative

$$\forall p \in \mathcal{P}, \quad f_p \geq 0, \qquad \text{and} \qquad , \forall e \in E, \quad x_e \geq 0$$

System optimum problem

The system optimum consists in minimizing the sum of all costs over the admissible flux $x = (x_e)_{e\in E}$

- Given x, the cost of taking edge e for one person is $\ell_e(x_e)$.
- The cost for the system for edge e is thus $x_e\ell_e(x_e)$.
- Thus minimizing the system costs consists in solving

$$
\begin{aligned}
\min_{x,f} \quad & \sum_{e\in E} x_e\ell_e(x_e) && (SO)\\
s.t. \quad & r_k = \sum_{p\in\mathcal{P}_k} f_p && k\in[K]\\
& x_e = \sum_{p\ni e} f_p && e\in E\\
& f_p \geq 0 && p\in\mathcal{P}
\end{aligned}
$$

System optimum problem

The system optimum consists in minimizing the sum of all costs over the admissible flux $x = (x_e)_{e\in E}$

- Given x, the cost of taking edge e for one person is $\ell_e(x_e)$.
- The cost for the system for edge e is thus $x_e\ell_e(x_e)$.
- Thus minimizing the system costs consists in solving

$$
\begin{aligned}
\min_{x,f} \quad & \sum_{e\in E} x_e\ell_e(x_e) && (SO)\\
s.t. \quad & r_k = \sum_{p\in\mathcal{P}_k} f_p && k\in[K]\\
& x_e = \sum_{p\ni e} f_p && e\in E\\
& f_p \geq 0 && p\in\mathcal{P}
\end{aligned}
$$

System optimum problem

The system optimum consists in minimizing the sum of all costs over the admissible flux $x = (x_e)_{e \in E}$

- Given x, the cost of taking edge e for one person is $\ell_e(x_e)$.
- The cost for the system for edge e is thus $x_e \ell_e(x_e)$.
- Thus minimizing the system costs consists in solving

$$
\begin{aligned}
\min_{x,f} \quad & \sum_{e \in E} x_e \ell_e(x_e) & & (SO) \\
s.t. \quad & r_k = \sum_{p \in \mathcal{P}_k} f_p & & k \in [K] \\
& x_e = \sum_{p \ni e} f_p & & e \in E \\
& f_p \geq 0 & & p \in \mathcal{P}
\end{aligned}
$$

Path intensity formulation

- We can reformulate the (SO) problem only using path-intensity $f = (f_p)_{p\in\mathcal{P}}$.
- Define $x_e(f) := \sum_{p\ni e} f_p$, and $x = (x_e)_{e\in E}$.
- Define the loss along a path $\ell_p(f) = \sum_{e\in p} \ell_e\big(\underbrace{\sum_{p'\ni e} f_{p'}}_{x_e(f)}\big)$
- The total cost is thus

$$C(f) = \sum_{p\in\mathcal{P}} f_p \ell_p(f) = \sum_{e\in E} x_e \ell_e(x_e(f)) = C(x(f)).$$

Path intensity formulation

- We can reformulate the (SO) problem only using path-intensity $f = (f_p)_{p \in \mathcal{P}}$.
- Define $x_e(f) := \sum_{p \ni e} f_p$, and $x = (x_e)_{e \in E}$.
- Define the loss along a path $\ell_p(f) = \sum_{e \in p} \ell_e\big(\underbrace{\sum_{p' \ni e} f_{p'}}_{x_e(f)}\big)$
- The total cost is thus

$$C(f) = \sum_{p \in \mathcal{P}} f_p \ell_p(f) = \sum_{e \in E} x_e \ell_e(x_e(f)) = C(x(f)).$$

Path intensity formulation

- We can reformulate the (SO) problem only using path-intensity $f = (f_p)_{p \in \mathcal{P}}$.
- Define $x_e(f) := \sum_{p \ni e} f_p$, and $x = (x_e)_{e \in E}$.
- Define the loss along a path $\ell_p(f) = \sum_{e \in p} \ell_e\big(\underbrace{\sum_{p' \ni e} f_{p'}}_{x_e(f)}\big)$
- The total cost is thus

$$C(f) = \sum_{p \in \mathcal{P}} f_p \ell_p(f) = \sum_{e \in E} x_e \ell_e(x_e(f)) = C(x(f)).$$

Path intensity formulation

- We can reformulate the (SO) problem only using path-intensity $f = (f_p)_{p\in\mathcal{P}}$.
- Define $x_e(f) := \sum_{p\ni e} f_p$, and $x = (x_e)_{e\in E}$.
- Define the loss along a path $\ell_p(f) = \sum_{e\in p} \ell_e\Big(\underbrace{\sum_{p'\ni e} f_{p'}}_{x_e(f)}\Big)$
- The total cost is thus

$$C(f) = \sum_{p\in\mathcal{P}} f_p \ell_p(f) = \sum_{e\in E} x_e \ell_e(x_e(f)) = C(x(f)).$$

Path intensity problem

$$
\begin{aligned}
\min_{f} \quad & \sum_{p\in\mathcal{P}} f_p \ell_p(f) && (SO) \\
s.t. \quad & r_k = \sum_{p\in\mathcal{P}_k} f_p && k \in [K] \\
& f_p \geq 0 && p \in \mathcal{P}
\end{aligned}
$$

Contents

Equilibrium definition

John Wardrop defined a traffic equilibrium as follows. "Under equilibrium conditions traffic arranges itself in congested networks such that all used routes between an O-D pair have equal and minimum costs, while all unused routes have greater or equal costs."

Equilibrium definition

John Wardrop defined a traffic equilibrium as follows. "Under equilibrium conditions traffic arranges itself in congested networks such that all used routes between an O-D pair have equal and minimum costs, while all unused routes have greater or equal costs."
A mathematical definition reads as follows.

Definition

A user flow f is a User Equilibrium if

$$\forall k \in [K], \quad \forall (p,p') \in \mathcal{P}_k^2, \qquad f_p > 0 \quad \Longrightarrow \quad \ell_p(f) \leq \ell_{p'}(f).$$

A new cost function

We are going to show that a user-equilibrium f is defined as a vector satisfying the KKT conditions of a certain optimization problem.

Let's define a new edge-loss function by

$$L_e(x_e) := \int_0^{x_e} \ell_e(u)du.$$

The Wardrop potential is defined (for edge intensity) as

$$W(f) = W(x(f)) = \sum_{e \in E} L_e(x_e(f)).$$

A new cost function

We are going to show that a user-equilibrium f is defined as a vector satisfying the KKT conditions of a certain optimization problem.

Let's define a new edge-loss function by

$$L_e(x_e) := \int_0^{x_e} \ell_e(u)du.$$

The Wardrop potential is defined (for edge intensity) as

$$W(f) = W(x(f)) = \sum_{e\in E} L_e(x_e(f)).$$

User optimum problem

Theorem

A flow f *is a user equilibrium if and only if it satisfies the first order KKT conditions of the following optimization problem*

$$\begin{aligned}
\min_{x,f} \quad & W(x) \\
s.t. \quad & r_k = \sum_{p \in \mathcal{P}_k} f_p && k \in [K] \\
& x_e = \sum_{p \ni e} f_p && e \in E \\
& f_p \geq 0 && p \in \mathcal{P}
\end{aligned}$$

Convex case : equivalence

If the loss functions (in edge-intensity) are non-decreasing then the Wardrop potential W is convex.

Theorem

Assume that the loss functions ℓ_e are non-decreasing for all $e \in E$. Then there exists at least one user equilibrium, and a flow f is a user equilibrium if and only if it solves (UE)

Contents

Descent methods

Consider the unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} \quad f(x). \tag{2}$$

A *descent direction algorithm* is an algorithm that constructs a sequence of points $(x^{(k)})_{k\in\mathbb{N}}$, that are recursively defined with:

$$x^{(k+1)} = x^{(k)} + t^{(k)}d^{(k)} \tag{3}$$

where

- $x^{(0)}$ is the initial point,
- $d^{(k)} \in \mathbb{R}^n$ is the descent direction,
- $t^{(k)}$ is the step length.

Descent methods

Consider the unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} \quad f(x). \tag{2}$$

A *descent direction algorithm* is an algorithm that constructs a sequence of points $(x^{(k)})_{k\in\mathbb{N}}$, that are recursively defined with:

$$x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)} \tag{3}$$

where

- $x^{(0)}$ is the initial point,
- $d^{(k)} \in \mathbb{R}^n$ is the descent direction,
- $t^{(k)}$ is the step length.

Descent methods

Consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} \quad f(x). \tag{2}$$

A *descent direction algorithm* is an algorithm that constructs a sequence of points $(x^{(k)})_{k \in \mathbb{N}}$, that are recursively defined with:

$$x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)} \tag{3}$$

where

- $x^{(0)}$ is the initial point,
- $d^{(k)} \in \mathbb{R}^n$ is the descent direction,
- $t^{(k)}$ is the step length.

Video explanation

https://www.youtube.com/watch?v=n-Y0SDSOfUI

Descent direction

For a differentiable objective function f, $d^{(k)}$ will be a descent direction iff $\nabla f(x^{(k)}) \cdot d^{(k)} \leq 0$, which can be seen from a first order development:

$$f\left(x^{(k)} + t^{(k)} d^{(k)}\right) = f(x^{(k)}) + t\langle \nabla f(x^{(k)}) \, , d^{(k)} \rangle + o(t).$$

The most classical descent direction is $d^{(k)} = -\nabla f(x^{(k)})$, which correspond to the gradient algorithm.

Descent direction

For a differentiable objective function f, $d^{(k)}$ will be a descent direction iff $\nabla f(x^{(k)}) \cdot d^{(k)} \leq 0$, which can be seen from a first order development:

$$f\left(x^{(k)} + t^{(k)} d^{(k)}\right) = f(x^{(k)}) + t\langle \nabla f(x^{(k)}) \, , d^{(k)} \rangle + o(t).$$

The most classical descent direction is $d^{(k)} = -\nabla f(x^{(k)})$, which correspond to the gradient algorithm.

Step-size choice

The step-size $t^{(k)}$ can be:

- fixed $t^{(k)} = t^{(0)}$, for all iteration,
- optimal $t^{(k)} \in \arg\min_{t \geq 0} f\left(x^{(k)} + td^{(k)}\right)$,
- a "good" step, following some rules (e.g Armijo's rules).

Finding the optimal step size is a special case of unidimensional optimization (or linear search).

Step-size choice

The step-size $t^{(k)}$ can be:

- fixed $t^{(k)} = t^{(0)}$, for all iteration,
- optimal $t^{(k)} \in \arg\min_{t \geq 0} f\left(x^{(k)} + td^{(k)}\right)$,
- a "good" step, following some rules (e.g Armijo's rules).

Finding the optimal step size is a special case of unidimensional optimization (or linear search).

Contents

Unidimensional optimization

We assume that the objective function $J : \mathbb{R} \to \mathbb{R}$ is strictly convex.

We are going to consider two types of methods:

- interval reduction algorithms: constructing $[a^{(l)}, b^{(l)}]$ containing the optimal point;
- successive approximation algorithms: approximating J and taking the minimum of the approximation.

Bisection method

We assume that J is differentiable over $[a, b]$. Note that, for $c \in [a, b]$, $t^* < c$ iff $J'(c) > 0$. From this simple remark, we construct the bisection method.

```
while b^(l) − a^(l) > ε do
    c^(l) = (b^(l) + a^(l)) / 2 ;
    if J'(c^(l)) > 0 then
        a^(l+1) = a^(l) ; b^(l+1) = c^(l) ;
    else if J'(c^(l)) < 0 then
        a^(l+1) = c^(l) ; b^(l+1) = b^(l) ;
    else
        return interval [a^(l), b^(l)]
    l = l + 1
```

while $b^{(l)} - a^{(l)} > \varepsilon$ **do**

- $c^{(l)} = \dfrac{b^{(l)} + a^{(l)}}{2}$;
- **if** $J'(c^{(l)}) > 0$ **then**
  - $a^{(l+1)} = a^{(l)}$; $b^{(l+1)} = c^{(l)}$;
- **else if** $J'(c^{(l)}) < 0$ **then**
  - $a^{(l+1)} = c^{(l)}$; $b^{(l+1)} = b^{(l)}$;
- **else**
  - return interval $[a^{(l)}, b^{(l)}]$
- $l = l + 1$

Note that $L_l = b^{(l)} - a^{(l)} = \dfrac{L_0}{2^l}$.

Golden section

I

Consider $a < t_1 < t_2 < b$, we are looking for $t^* = \arg\min_{t \in [a,b]} J(t)$

Note that

- if $J(t_1) < J(t_2)$, then $t^* \in [a, t_2]$;
- if $J(t_1) > J(t_2)$, then $t^* \in [t_1, b]$;
- if $J(t_1) = J(t_2)$, then $t^* \in [t_1, t_2]$.

Hence, at each iteration the interval $[a^{(l)}, b^{(l)}]$ is updated into $[a^{(l)}, t_2^{(l)}]$ or $[t_1^{(l)}, b^{(l)}]$.

Golden section I

Consider $a < t_1 < t_2 < b$, we are looking for $t^* = \arg\min_{t\in[a,b]} J(t)$

Note that

- if $J(t_1) < J(t_2)$, then $t^* \in [a, t_2]$;
- if $J(t_1) > J(t_2)$, then $t^* \in [t_1, b]$;
- if $J(t_1) = J(t_2)$, then $t^* \in [t_1, t_2]$.

Hence, at each iteration the interval $[a^{(l)}, b^{(l)}]$ is updated into $[a^{(l)}, t_2^{(l)}]$ or $[t_1^{(l)}, b^{(l)}]$.

Golden section II

We now want to know how to choose $t_1^{(l)}$ and $t_2^{(l)}$. To minimize the worst-case complexity we want equity between both possibilities, hence $b^{(l)} - t_1^{(l)} = t_2^{(l)} - a^{(l)}$. Now assume that $J(t_1^{(l)}) < J(t_2^{(l)})$. Hence, $a^{(l+1)} = a^{(l)}$, and $b^{(l+1)} = t_2$. We would like to reuse the computation of $J(t_1^{(l)})$ by defining $t_1^{(k+1)} = t_2^{(l)}$.
In order to satisfy this constraint we need to have

$$\begin{cases} L_2 + L_1 = L \\ \dfrac{L_2}{L} = \dfrac{L_1}{L_2} =: R \end{cases} \tag{4}$$

where $L = b^{(l)} - a^{(l)}$, $L_1 = t_1^{(l)} - a^{(l)}$ and $L_2 = t_2^{(l)} - a^{(l)}$.
This implies

$$1 + R = \frac{1}{R} \tag{5}$$

Golden section II

We now want to know how to choose $t_1^{(l)}$ and $t_2^{(l)}$. To minimize the worst-case complexity we want equity between both possibilities, hence $b^{(l)} - t_1^{(l)} = t_2^{(l)} - a^{(l)}$. Now assume that $J(t_1^{(l)}) < J(t_2^{(l)})$. Hence, $a^{(l+1)} = a^{(l)}$, and $b^{(l+1)} = t_2$. We would like to reuse the computation of $J(t_1^{(l)})$ by defining $t_1^{(k+1)} = t_2^{(l)}$. In order to satisfy this constraint we need to have

$$\begin{cases} L_2 + L_1 = L \\ \dfrac{L_2}{L} = \dfrac{L_1}{L_2} =: R \end{cases} \tag{4}$$

where $L = b^{(l)} - a^{(l)}$, $L_1 = t_1^{(l)} - a^{(l)}$ and $L_2 = t_2^{(l)} - a^{(l)}$.

This implies

$$1 + R = \frac{1}{R} \tag{5}$$

Golden section II

We now want to know how to choose $t_1^{(l)}$ and $t_2^{(l)}$. To minimize the worst-case complexity we want equity between both possibilities, hence $b^{(l)} - t_1^{(l)} = t_2^{(l)} - a^{(l)}$. Now assume that $J(t_1^{(l)}) < J(t_2^{(l)})$. Hence, $a^{(l+1)} = a^{(l)}$, and $b^{(l+1)} = t_2$. We would like to reuse the computation of $J(t_1^{(l)})$ by defining $t_1^{(k+1)} = t_2^{(l)}$. In order to satisfy this constraint we need to have

$$\begin{cases} L_2 + L_1 = L \\ \dfrac{L_2}{L} = \dfrac{L_1}{L_2} =: R \end{cases} \tag{4}$$

where $L = b^{(l)} - a^{(l)}$, $L_1 = t_1^{(l)} - a^{(l)}$ and $L_2 = t_2^{(l)} - a^{(l)}$. This implies

$$1 + R = \frac{1}{R} \tag{5}$$

Golden section III

$$R = \frac{\sqrt{5}-1}{2}. \tag{6}$$

Finally, in order to satisfy equity and reusability it is enough to set

$$t_1^{(l)} = a^{(l)} + (1-R)(b^{(l)} - a^{(l)})$$
$$t_1^{(l)} = a^{(l)} + R(b^{(l)} - a^{(l)})$$

The same happens for the $J(t_1^{(l)}) > J(t_2^{(l)})$ case.

Golden section algorithm

$a^{(0)} = a, \quad b^{(0)} = b;$
$t_1^{(0)} = a + (1-R)b, \quad t_2^{(0)} = a + Rb;$
$J_1 = J(t_1^{(0)}), \quad J_2 = J(t_2^{(0)}), \ \ell = 0;$
while $b^{(\ell)} - a^{(\ell)} > \varepsilon$ **do**
  if $J_1 < J_2$ **then**
    $a^{(\ell+1)} = a^{(\ell)}$; $b^{(\ell+1)} = t_2^{(\ell)}$;
    $t_1^{(\ell+1)} = a^{(\ell+1)} + (1-R)b^{(\ell+1)}$; $t_2^{(\ell+1)} = t_1^{(\ell)}$;
    $J_2 = J_1;$
    $J_1 = J(t_1^{(\ell+1)});$
  else
    $a^{(\ell+1)} = t_1^{(\ell)}$; $b^{(\ell+1)} = b^{(\ell)}$;
    $t_1^{(\ell+1)} = t_2^{(\ell)}$; $t_2^{(\ell+1)} = a^{(\ell+1)} + Rb^{(\ell+1)}$;
    $J_1 = J_2;$
    $J_2 = J(t_2^{(\ell+1)});$
  $\ell = \ell + 1$

Note that $L_\ell = R^\ell L_0$.

Video explanation

Golden section

https://www.youtube.com/watch?v=6NYp3td3cjU

Curve fitting : Newton method

If J is twice-differentiable (with non-null second-order derivative) is to determine $t^{(k+1)}$ as the minimum of the second order Taylor's of J at $t^{(k)}$:

$$
\begin{aligned}
t^{(l+1)} - t^{(l)} &= \arg\min_t J(t^{(l)}) + J'(t^{(l)})t + \frac{t^2}{2} J''(t^{(l)}) \\
&= \left[J''(t^{(l)})\right]^{-1} J'(t^{(l)})
\end{aligned}
$$

This is the well-known, and very efficient, Newton method.

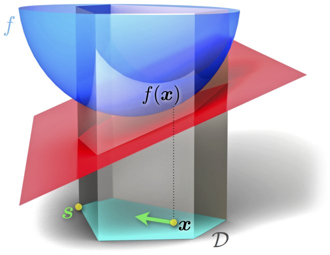
Conditional gradient algorithm

We address an optimization problem with convex objective function f and compact polyhedral constraint set X, i.e.

$$\min_{x \in X \subset \mathbb{R}^n} \qquad f(x)$$

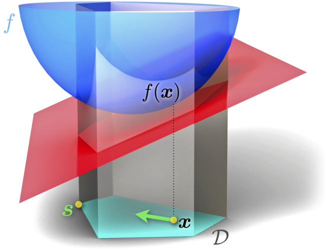
where

$$X = \left\{ x \in \mathbb{R}^n \quad | \quad Ax \leq b, \quad \tilde{A}x = \tilde{b} \right\}$$



Conditional gradient algorithm

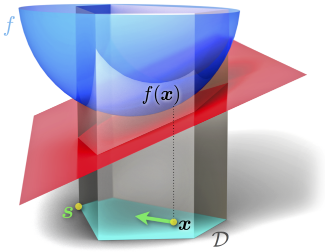
It is a descent algorithm, where we first look for an admissible descent direction $d^{(k)}$, and then look for the optimal step.



Conditional gradient algorithm

It is a descent algorithm, where we first look for an admissible descent direction $d^{(k)}$, and then look for the optimal step.
As f is convex, we know that for any point $x^{(k)}$,

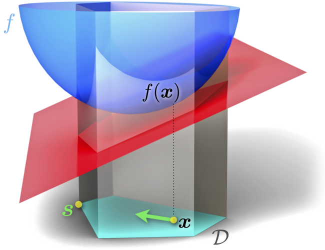
$$f(y) \geq f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)})$$



Conditional gradient algorithm

It is a descent algorithm, where we first look for an admissible descent direction $d^{(k)}$, and then look for the optimal step.
As f is convex, we know that for any point $x^{(k)}$,

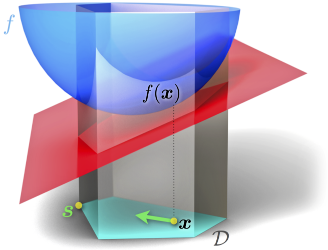
$$f(y) \geq f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)})$$



The conditional gradient method consists in choosing the descent direction that minimizes the linearization of f over X.

Conditional gradient algorithm

The conditional gradient method consists in choosing the descent direction that minimizes the linearization of f over X. More precisely, at step k we solve



$$y^{(k)} \in \arg\min_{y \in X} \quad f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)}).$$

Remarks on conditional gradient

$$y^{(k)} \in \arg\min_{y \in X} \quad f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)}).$$

- This problem is linear, hence easy to solve.
- By the convexity inequality, the value of the linearized Problem is a lower bound to the true problem.
- As $y^{(k)} \in X$, $d^{(k)} = y^{(k)} - x^{(k)}$ is a *feasable direction*, in the sense that for all $t \in [0,1]$, $x^{(k)} + td^{(k)} \in X$.
- If $y^{(k)}$ is obtained through the simplex method it is an extreme point of X, which means that, for $t > 1$, $x^{(k)} + td^{(k)} \notin X$.
- If $y^{(k)} = x^{(k)}$ then we have found an optimal solution.
- We also have $y^{(k)} \in \arg\min_{x \in X} \nabla f(x^{(k)}) \cdot y$, the lower-bound being obtained easily.

Remarks on conditional gradient

$$y^{(k)} \in \arg\min_{y \in X} \quad f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)}).$$

- This problem is linear, hence easy to solve.
- By the convexity inequality, the value of the linearized Problem is a lower bound to the true problem.
- As $y^{(k)} \in X$, $d^{(k)} = y^{(k)} - x^{(k)}$ is a *feasable direction*, in the sense that for all $t \in [0,1]$, $x^{(k)} + td^{(k)} \in X$.
- If $y^{(k)}$ is obtained through the simplex method it is an extreme point of X, which means that, for $t > 1$, $x^{(k)} + td^{(k)} \notin X$.
- If $y^{(k)} = x^{(k)}$ then we have found an optimal solution.
- We also have $y^{(k)} \in \arg\min_{x \in X} \nabla f(x^{(k)}) \cdot y$, the lower-bound being obtained easily.

Remarks on conditional gradient

$$y^{(k)} \in \arg\min_{y \in X} \quad f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)}).$$

- This problem is linear, hence easy to solve.
- By the convexity inequality, the value of the linearized Problem is a lower bound to the true problem.
- As $y^{(k)} \in X$, $d^{(k)} = y^{(k)} - x^{(k)}$ is a *feasable direction*, in the sense that for all $t \in [0, 1]$, $x^{(k)} + td^{(k)} \in X$.
- If $y^{(k)}$ is obtained through the simplex method it is an extreme point of X, which means that, for $t > 1$, $x^{(k)} + td^{(k)} \notin X$.
- If $y^{(k)} = x^{(k)}$ then we have found an optimal solution.
- We also have $y^{(k)} \in \arg\min_{x \in X} \nabla f(x^{(k)}) \cdot y$, the lower-bound being obtained easily.

Remarks on conditional gradient

$$y^{(k)} \in \arg\min_{y \in X} \quad f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)}).$$

- This problem is linear, hence easy to solve.
- By the convexity inequality, the value of the linearized Problem is a lower bound to the true problem.
- As $y^{(k)} \in X$, $d^{(k)} = y^{(k)} - x^{(k)}$ is a *feasable direction*, in the sense that for all $t \in [0, 1]$, $x^{(k)} + td^{(k)} \in X$.
- If $y^{(k)}$ is obtained through the simplex method it is an extreme point of X, which means that, for $t > 1$, $x^{(k)} + td^{(k)} \notin X$.
- If $y^{(k)} = x^{(k)}$ then we have found an optimal solution.
- We also have $y^{(k)} \in \arg\min_{x \in X} \nabla f(x^{(k)}) \cdot y$, the lower-bound being obtained easily.

Remarks on conditional gradient

$$y^{(k)} \in \arg\min_{y\in X} \quad f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)}).$$

- This problem is linear, hence easy to solve.
- By the convexity inequality, the value of the linearized Problem is a lower bound to the true problem.
- As $y^{(k)} \in X$, $d^{(k)} = y^{(k)} - x^{(k)}$ is a *feasable direction*, in the sense that for all $t \in [0, 1]$, $x^{(k)} + td^{(k)} \in X$.
- If $y^{(k)}$ is obtained through the simplex method it is an extreme point of X, which means that, for $t > 1$, $x^{(k)} + td^{(k)} \notin X$.
- If $y^{(k)} = x^{(k)}$ then we have found an optimal solution.
- We also have $y^{(k)} \in \arg\min_{x\in X} \nabla f(x^{(k)}) \cdot y$, the lower-bound being obtained easily.

Remarks on conditional gradient

$$y^{(k)} \in \arg\min_{y \in X} \quad f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)}).$$

- This problem is linear, hence easy to solve.
- By the convexity inequality, the value of the linearized Problem is a lower bound to the true problem.
- As $y^{(k)} \in X$, $d^{(k)} = y^{(k)} - x^{(k)}$ is a *feasable direction*, in the sense that for all $t \in [0, 1]$, $x^{(k)} + td^{(k)} \in X$.
- If $y^{(k)}$ is obtained through the simplex method it is an extreme point of X, which means that, for $t > 1$, $x^{(k)} + td^{(k)} \notin X$.
- If $y^{(k)} = x^{(k)}$ then we have found an optimal solution.
- We also have $y^{(k)} \in \arg\min_{x \in X} \nabla f(x^{(k)}) \cdot y$, the lower-bound being obtained easily.

Frank Wolfe algorithm

Data: objective function f, constraints, initial point $x^{(0)}$, precision ε
Result: ε-optimal solution $x^{(k)}$, upper-bound $f(x^{(k)})$, lower-bound $\underline{f}$

$\underline{f} = -\infty$;
$k = 0$;

while $f(x^{(k)}) - \underline{f} > \varepsilon$ **do**

- solve the LP $\min_{y \in X} f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)})$;
- let $y^{(k)}$ be an optimal solution, and $\underline{f}$ the optimal value ;
- set $d^{(k)} = y^{(k)} - x^{(k)}$;
- solve $t^{(k)} \in \arg\min_{t \in [0,1]} f\left(x^{(k)} + td^{(k)}\right)$;
- update $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$;
- $k = k + 1$;

Contents

All-or nothing

A very simple heuristic consists in:

1. Set $k = 0$.
2. Assume initial cost per edge $\ell_e^{(k)} = \ell_e(x_e^{ref})$.
3. For each origin-destination pair (o_i, d_i) find the shortest path associated with $\ell^{(k)}$.
4. Associate the full flow r_i to this path, which form a flow of user $f^{(k)}$.
5. Deducing the travel cost per edge is $\ell_e^{(k+1)} = \ell_e(f^{(k)})$.
6. Go to step 3.

This method is simple and requires only computing the shortest path in a fixed cost graph.
However, it is not converging as it can cycle.

All-or nothing

A very simple heuristic consists in:

1. Set $k = 0$.
2. Assume initial cost per edge $\ell_e^{(k)} = \ell_e(x_e^{ref})$.
3. For each origin-destination pair (o_i, d_i) find the shortest path associated with $\ell^{(k)}$.
4. Associate the full flow r_i to this path, which form a flow of user $f^{(k)}$.
5. Deducing the travel cost per edge is $\ell_e^{(k+1)} = \ell_e(f^{(k)})$.
6. Go to step 3.

This method is simple and requires only computing the shortest path in a fixed cost graph.

However, it is not converging as it can cycle.

All-or nothing

A very simple heuristic consists in:

1. Set $k = 0$.
2. Assume initial cost per edge $\ell_e^{(k)} = \ell_e(x_e^{ref})$.
3. For each origin-destination pair (o_i, d_i) find the shortest path associated with $\ell^{(k)}$.
4. Associate the full flow r_i to this path, which form a flow of user $f^{(k)}$.
5. Deducing the travel cost per edge is $\ell_e^{(k+1)} = \ell_e(f^{(k)})$.
6. Go to step 3.

This method is simple and requires only computing the shortest path in a fixed cost graph.
However, it is not converging as it can cycle.

Smoothed all-or-nothing

The all-or-nothing method can be understood as follow: each day every user chooses the shortest path according to the traffic on the previous day. We can smooth the approach by saying that only a fraction ρ of users is going to update its path from one day to the next.

Hence the smoothed all-or-nothing approach reads

1. Set $k = 0$.
2. Assume initial cost per arc $\ell_e^{(k)} = \ell_e(x_e^{ref})$.
3. For each pair origin destination (o_i, d_i) find the shortest path associated with $\ell^{(k)}$.
4. Associate the full flow r_i to this path, which form a flow of user $\tilde{f}^{(k)}$.
5. Compute the new flow $f^{(k)} = (1-\rho)f^{(k-1)} + \rho\tilde{f}^{(k)}$.
6. Deducing the travel cost per arc as $\ell_e^{(k+1)} = \ell_e(f^{(k)})$.
7. Go to step 3.

Smoothed all-or-nothing

The all-or-nothing method can be understood as follow: each day every user chooses the shortest path according to the traffic on the previous day. We can smooth the approach by saying that only a fraction ρ of users is going to update its path from one day to the next.

Hence the smoothed all-or-nothing approach reads

1. Set $k = 0$.
2. Assume initial cost per arc $\ell_e^{(k)} = \ell_e(x_e^{ref})$.
3. For each pair origin destination (o_i, d_i) find the shortest path associated with $\ell^{(k)}$.
4. Associate the full flow r_i to this path, which form a flow of user $\tilde{f}^{(k)}$.
5. Compute the new flow $f^{(k)} = (1-\rho)f^{(k-1)} + \rho\tilde{f}^{(k)}$.
6. Deducing the travel cost per arc as $\ell_e^{(k+1)} = \ell_e(f^{(k)})$.
7. Go to step 3.

Contents

UE problem

Recall that, if the arc-cost functions are non-decreasing finding a user-equilibrium is equivalent to solving

$$\min_{f \geq 0} \quad W(x(f))$$
$$s.t. \quad r_k = \sum_{p \in \mathcal{P}_k} f_p \qquad k \in [K]$$

where

$$W(f) = W(x(f)) = \sum_{e \in E} L_e(x_e(f)),$$

with

$$L_e(x_e) := \int_0^{x_e} \ell_e(u)du,$$

and

$$x_e(f) = \sum_{p \ni e} f_p.$$

Frank-Wolfe for UE

Let's compute the linearization of the objective function. Consider an admissible flow $f^{(\kappa)}$ and a path $p \in \mathcal{P}_i$. We have

$$\begin{aligned}
\frac{\partial W \circ x}{\partial f_p}(f^{(\kappa)}) &= \frac{\partial}{\partial f_p}\Big(\sum_{e \in E} L_e(\sum_{p' \ni e} f^{(\kappa)}_{p'})\Big) \\
&= \sum_{e \in p} \frac{\partial}{\partial x_e} L_e(x_e(f^{(\kappa)})) \\
&= \sum_{e \in p} \ell_e(x_e(f^{(\kappa)}) = \ell_p(f^{(\kappa)}).
\end{aligned}$$

Hence, the linearized problem around $f^{(k)}$ reads

$$\begin{aligned}
\min_{\{y_p\}_{p \in \mathcal{P}}} \quad & \sum_{p \in \mathcal{P}} y_p \ell_p(f^{(\kappa)}) \\
s.t \quad & r_k = \sum_{p \in \mathcal{P}_k} y_p \qquad k \in [K]
\end{aligned}$$

Frank-Wolfe for UE

Let's compute the linearization of the objective function. Consider an admissible flow $f^{(\kappa)}$ and a path $p \in \mathcal{P}_i$. We have

$$\begin{aligned}
\frac{\partial W \circ x}{\partial f_p}(f^{(\kappa)}) &= \frac{\partial}{\partial f_p}\Big(\sum_{e \in E} L_e(\sum_{p' \ni e} f_{p'}^{(\kappa)})\Big) \\
&= \sum_{e \in p} \frac{\partial}{\partial x_e} L_e(x_e(f^{(\kappa)})) \\
&= \sum_{e \in p} \ell_e(x_e(f^{(\kappa)}) = \ell_p(f^{(\kappa)}).
\end{aligned}$$

Hence, the linearized problem around $f^{(k)}$ reads

$$\begin{aligned}
\min_{\{y_p\}_{p \in \mathcal{P}}} \quad & \sum_{p \in \mathcal{P}} y_p \ell_p(f^{(\kappa)}) \\
s.t \quad & r_k = \sum_{p \in \mathcal{P}_k} y_p & k \in [K]
\end{aligned}$$

Frank-Wolfe for UE II

$$\begin{aligned} \min_{\{y_p\}_{p\in\mathcal{P}}} \quad & \sum_{p\in\mathcal{P}} y_p \ell_p(f^{(\kappa)}) & \\ s.t \quad & r_k = \sum_{p\in\mathcal{P}_k} y_p & k \in [K] \\ & y_p \geq 0 & p \in \mathcal{P} \end{aligned}$$

Note that this problem is an all-or-nothing iteration and can be solved (o, d)-pair by (o, d)-pair by solving a shortest path problem. As the cost $t_a^k := \ell_e(f^{(\kappa)})$ is non-negative we can use Djikstra's algorithm to solve this problem.

Frank-Wolfe for UE II

$$\min_{\{y_p\}_{p\in\mathcal{P}}} \quad \sum_{p\in\mathcal{P}} y_p \ell_p(f^{(\kappa)})$$

$$s.t \quad r_k = \sum_{p\in\mathcal{P}_k} y_p \qquad k \in [K]$$

$$y_p \geq 0 \qquad p \in \mathcal{P}$$

Note that this problem is an all-or-nothing iteration and can be solved (o, d)-pair by (o, d)-pair by solving a shortest path problem. As the cost $t_a^k := \ell_e(f^{(\kappa)})$ is non-negative we can use Djikstra's algorithm to solve this problem.

Frank-Wolfe for UE III

Having found $y^{(\kappa)}$, we now have to solve

$$\min_{t\in[0,1]} J(t) := W\Big((1-t)f^{(\kappa)} + ty^{(\kappa)}\Big).$$

As J is convex, the bisection method seems adapted. We have

$$\begin{aligned} J'(t) &= \nabla W\Big((1-t)f^{(\kappa)} + ty^{(\kappa)}\Big) \cdot (y^{(\kappa)} - f^{(\kappa)}) \\ &= \sum_{p\in\mathcal{P}} (y_p^{(\kappa)} - f_p^{(\kappa)})\ell_p\big((1-t)f^{(\kappa)} + ty^{(\kappa)}\big) \end{aligned}$$

hence the bisection method is readily implementable.

Frank Wolfe is a smoothed all-or-nothing

Data: cost function ℓ, constraints, initial flow $f^{(0)}$
Result: equilibrium flow $f^{(\kappa)}$
$\underline{W} = -\infty$;
$\kappa = 0$;
compute starting travel time $c_e^{(0)} = \ell_e(x(f^{(\kappa)}))$;

while $W(x^{(\kappa)}) - \underline{W} > \varepsilon$ **do**
  foreach *pair origin-destination* (o_i, d_i) **do**
    find a shortest path p_i from o_i to d_i for the loss $c^{(\kappa)}$;
  deduce an auxiliary flow $y^{(\kappa)}$ by setting r_i to p_i ;
  set descent direction $d^{(\kappa)} = y^{(\kappa)} - f^{(\kappa)}$;
  find optimal step $t^{(\kappa)} \in \arg\min_{t\in[0,1]} W\left(x^{(\kappa)} + td^{(\kappa)}\right)$;
  update $f^{(k+1)} = f^{(\kappa)} + t^{(\kappa)}d^{(\kappa)}$;
  $\kappa = \kappa + 1$;