Vincent Leclère

Recherche Opérationelle et Transport

April 4, 2022

Contents

CHAPTER 1

Introduction

1.1 Urban transportation network analysis

1.1.1 Transportation planning process

Among the many applications of operational reserch, we can quote its key position in the chain of transportation planning.

Transportation projects have several specificities that differentiate them from others infrastructure projects:

- They have a very long lifespan;
- They are part of a network and cannot be thought alone;
- Thus, their effects can extented far from their immediate territories;
- They are rarely financially profitable, but play key roles in the operation of a territory.

For all these reasons, upstream studies are crucial to really predict the impacts of a transportation projects. Those studies combined territorial diagnostic, trafic studies, environnemental studies, socioeconomics studies...

In complex urban systems, a common tool to evaluate changes in the transportation systems is four steps modeling. Those models are built to recreate a current situation of mobility in the territory (a year where a household survey has been made), and are then used to make predictions. They divide the territory in zones, and linked them by one a several transportation networks (typically, transit network and road network). They then follow four stages:

1. Trip generation: according to socioeconomics data, the number of trips emitted and received by each zone are estimated;
2. Trip distribution: we calculate the number of trips between each pairs of zone;
3. Modal split: how many trips are made by car, walking, public transit etc.
4. Traffic assignements, which the subject of this class.

1.2 Game theory

Game theory is a field at the frontier between mathematics and economy that study the interaction between multiple players, each acting for his own gain but whose action affects others as well.

We can define a few different type of games:

- Number of player
  - 2 (most results)
  - $n > 2$ (hard, even with 3)
  - an infinity.
- Objective
  - zero-sum game (the sum of gains is 0, e.g. chess)
  - cooperative : everybody share the same objective (e.g. pandemia)
  - generic (e.g. Prisonner dilemna)
- Repeated or not
- Deterministic or stochastic
- ...

1.2.1 Nash Equilibrium, Pareto Optimum and Social Optimum

We start with a very well known example.

Example 1.1 (Prisonner's Dilemna)**.** *Two guys got caught while dealing chocolate. As he is missing concrete evidence the judge offer them a deal.*

- *If both deny their implication they will get 2 month each.*
- *If one speak, and the other deny their implication, the first will get 1 month while the other will get 5 months.*
- *If both speak, they get 4 month each.*

	collaborate	deny
collaborate	(4,4)	(1,5)
deny	(5,1)	(2,2)

Question : what is the equilibrium ?

We can see that if the other denies, it is better for me to collaborate, and if the other collaborates, it is better for me to collaborate. Hence the equilibrium consists in collaborating with the judge, which leads to the worst global case : a total of 8 *months.*

In game theory we consider multiple agents $a \in \mathcal{A}$, each having a set of possible action $u_a \in \mathcal{U}_a$. Let denote $\mathcal{U} = \times_{a \in \mathcal{A}} \mathcal{U}_a$. Each agent earn a reward $r_a : \mathcal{U} \to \mathbb{R}$ depending on his action, as well as others players actions.

Definition 1.1. *A* Nash equilibrium *is a set of actions such that no player can unilaterally improve its pay-off by changing is action:*

$$\forall a \in \mathcal{A}, \quad \forall u'_a \in \mathcal{U}_a, \qquad r_a(u'_a, u_{-a}) \leqslant r_a(u_a, u_{-a}). \tag{1.1}$$

Definition 1.2. *A* Pareto efficient solution *is a set of action such that no other set of actions can strictly improve at least one player's pay-off without decreasing at least another:*

$$\exists a \in \mathcal{A}, r_a(\tilde{u}) > r_a(u) \quad \Longrightarrow \quad \exists \tilde{a} \in \mathcal{A}, \quad r_{\tilde{a}}(\tilde{u}) < r_{\tilde{a}}(u) \tag{1.2}$$

Definition 1.3. *A* social optimum *is a set of action minimizing the average pay-off.*

Let's make a few interpretative remarks about those notions.

- To be followed a recommandation need to be a Nash Equilibrium. Otherwise one player, assuming that the others follow the recommandation, will have interest not to follow the recommandation.
- A decider (for all player) should always choose a Pareto efficient solution : otherwise he could improve the gain of some player without impeding any others.
- The social optimum is not always well defined as the ponderation is not always obvious, especially when players are of different types (e.g. consumer and companies).

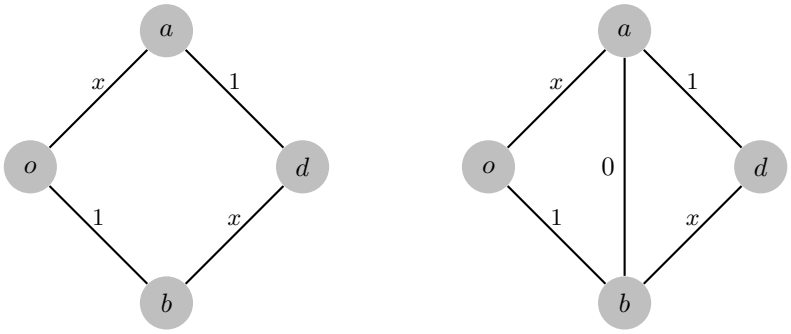
Exercise 1.1.
- *If possible, find social optimum, Nash equilibrium and Pareto efficient solution of Prisonner's Dilemma ?*
- *If possible, find social optimum, Nash equilibrium and Pareto efficient solutions of Zero Sum games ?*

1.2.2 Braess paradox

In road networks, people choose their means of transport (e.g. car versus public transport), their time of departure, their itinerary. Each user chooses in its own interest (mainly the shortest time / lowest cost). The time depends on the congestion, which means on the choice of other users. Hence, we are in a game framework : users interact with conflicting interest. As the number of people is very high we consider that there is an infinite number of people, which means that the choice of a single user does not affect the costs of others.

The generic case will be treated in Chapter 3, now we will just focus on a very simple yet very interesting example, represented in Figure

Example 1.2. *Consider a large group of person want to go from the same origin o to the destination d, at the same time, with the same car. We look at a very simple graph with two roads, each composed of two edges. The time on each edges of the road is given as a function of the number of person taking the given edge.*



(a) Original graph (b) Modified graph

Figure 1.1: Braess's Paradox example

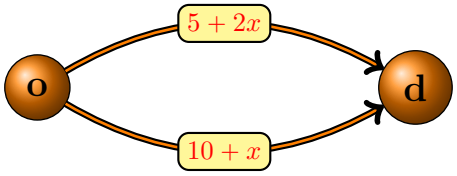


Figure 1.2: Exercise 1.2 graph. The cost of each arc is given as a function of x the number of trip going through the arc.

1.3 Exercises

Solution to starred exercises can be found in Section B.1.

Exercise 1.2. *1. Consider the graph given in figure 1.2, with* 1000 *trips. What is the equilibrium ? What is the social optimum ?*

2. Same question with $c_1(x_1) = 15(1+0.15(\frac{x_1}{1000})^4)$, $c_2(x_2) = 20(1+0.15(\frac{x_2}{3000})^4)$ *?*

Exercise* 1.3. *Consider a game where rewards (to be maximized) are given by the following table where actions of player* 1 *correspond to the lines, actions of player* 2 *to the columns, rewards being given in the order of player.*

	a	b	c
a	(1,2)	(4,3)	(4,4)
b	(2,6)	(5,5)	(2,6)
c	(3,2)	(2,1)	(1,1)

1. Find the Nash equilibrium(s)

2. Find the social optimum(s)

3. Find the Pareto optimum(s)

Exercise* 1.4. *Consider a game where rewards (to be maximized) are given by the following table where actions of player* 1 *correspond to the lines, actions of*

player 2 *to the columns, rewards being given in the order of player. For example, if player* 1 *play* a*, and player* 2 *play* c*, then player* 1 *gains* 0 *and player* 2 *gains* 1*.*

	a	b	c	d
a	(1,-1)	(0,0)	(0,1)	(-1,4)
b	(-1,2)	(2,3)	(3,2)	(-2,3)

1. *Find the Nash equilibrium(s)*
2. *Find the social optimum(s)*
3. *Find the Pareto optimum(s)*

Exercise* 1.5. *Consider a game where rewards (to be maximized) are given by the following table where actions of player* 1 *correspond to the lines, actions of player* 2 *to the columns, rewards being given in the order of player. For example, if player* 1 *play* a*, and player* 2 *play* c*, then player* 1 *gains* 2 *and player* 2 *gains* 3*.*

	a	b	c
a	(7,1)	(0,0)	(2,3)
b	(-1,2)	(2,3)	(3,2)
c	(-1,4)	(1,3)	(1,7)

1. *Find the Nash equilibrium(s)*
2. *Find the social optimum(s)*
3. *Find the Pareto optimum(s)*

Exercise* 1.6. *Consider a game where rewards (to be maximized) are given by the following table where actions of player* 1 *correspond to the lines, actions of player* 2 *to the columns, rewards being given in the order of player.*

	a	b
a	(-5,-5)	(1,-1)
b	(-1,1)	(0,0)

1. *Find the Nash equilibrium(s), social optimum(s) and Pareto optimum(s)*
2. *We now want to consider random strategies. More precisely we consider that player one play a with probability* p_1 *and player* 2 *play a with probability* p_2 *(independently of the action of* 1*). We assume that each wants to maximize its expected reward.*
   (a) *For given* p_1 *and* p_2 *what is the expected reward of player* 1 *?*
   (b) *For a given* p_2 *what are the set of* p_1 *maximizing the expected reward of player* 1 *?*

(c) *Justify that, when looking for a Nash-Equilibrium, only* 3 *value of* p_1 *and* p_2 *should be considered, and give the reward matrix associated.*

(d) *What are the Nash Equilibrium(s) ? Is it better than in the original deterministic version ?*

CHAPTER 2

Shortest path problem

The shortest path problems are very standard optimization problems, incredibly important in all parts of operations research, and especially for transport. Roughly speaking, it consists in finding the fastest (or cheapest) way of going from one point to another. Apart from obvious applications, a large number of optimization problems can be seen as shortest path problems. For example maintenance problems, or (deterministic) unit commitment problems (deciding which unit should produce at what time to meet some demand) can be represented as shortest path problems.

In this chapter we first gives basic definitions over graphs, before detailing three efficient algorithms that solve the shortest path problem.

2.1 Graphs

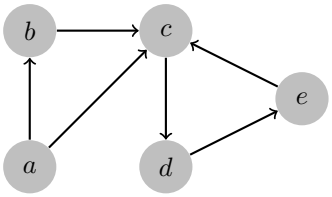
In this section we gives some definitions on graphs.

2.1.1 Directed graphs

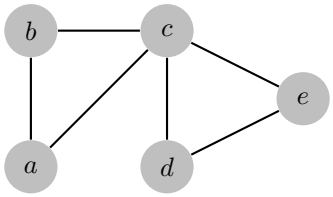
A *graph* is one of the elementary and ubiquitous modelisation tools of Operation Research. A *directed graph* (V, E) is defined by

- A finite set of n *vertices* V
- A finite set of m *edges* each having an origin in V and a destination in V. If there is no confusion possible we identify each edges e of E with the pair origin-destination $(u, v) \in V^2$.

A graph is said to be *undirected* if we do not distinguish between the origin and the destination.



(a) Directed graph



(b) Undirected graph

Consider a directed graph (V, E).

- If $(u, v) \in E$, u is a *predecessor* of v, and v is a *successor* of u.
- The *adjacency matrix* of the graph is an application $A : V \times V \to \{0, 1\}$ such that $A(u, v) = 1$ iff $(u, v) \in E$.
- The *incidence matrix* of the graph is an application $N : V \times A \to \{-1, 0, 1\}$ such that $N(u, (u, v)) = -1$, $N(v, (u, v)) = 1$ and N is null elsewhere.
- A *path* is a sequence of edges $\{e_k\}_{k \in [\![1,n]\!]}$, such that the destination of one edge is the origin of the next. The origin of the first edge is the *origin* of the path, and the destination of the last edge is the *destination* of the path.
- A (directed) graph is *connected* if for all $u, v \in V$, there is a u-v-path.
- A *cycle* is a path where the destination vertex is the origin. A graph is termed *acyclic*, if there is no cycle in it. Directed Acyclic Graphs (DAG) have interesting properties has will be seen in § 2.3.1.

2.1.2 Weighted graphs

In order to speak of "shortest" path, we need to define the length (or weight) of a path.

- A weighted (directed) graph is a (directed) graph (V, E) with a weight function $\ell : E \to \mathbb{R}$.
- The weight of a $s-t-$path p is sum of the weights of the edges contained in the path :
$$\ell(p) := \sum_{e \in p} \ell(e).$$
- An *absorbing cycle* is a cycle of strictly negative weight.

The *shortest path problem* from $o \in V$ to $d \in V$ consist of finding a path of minimal weight with origin o and destination d.

Exercise 2.1. *Let G be a connected directed graph containing an absorbing cycle. What is the value of the shortest path problem for any pair of vertexes ? Does a shortest path exists ?*

2.2 Dijkstra algorithm

2.2.1 Generic label algorithm

An optimality condition

The methods we are going to present are based on a label function over the vertices. This function should be understood as *an estimate cost of the shortest path cost* between the origin and the current vertex.

Theorem 2.1. *Suppose that there exists a function* $\lambda : V \mapsto \mathbb{R} \cup \{+\infty\}$*, such that*

$$\forall (i,j) \in E, \qquad \lambda_j \leqslant \lambda_i + \ell(i,j). \tag{2.1}$$

Then, if p *is an* s*-*t*-path, we have* $\ell(p) \leqslant \lambda(t) - \lambda(s)$[1]

In particular, if p *is such that*

$$\forall (i,j) \in p, \qquad \lambda_j = \lambda_i + \ell(i,j),$$

then p *is a shortest path.*

The condition 2.1 is a triangular inequality condition : the cost to reach j is at least the cost to reach i plus the cost to go from i to j. Consequently the first result consists simply in summing the inequality along edges in p, from which we directly obtain the caracterisation of a shortest path.

An abstract algorithm

The generic label algorithm 1 consists in keeping a list of candidates vertices $U \subset V$ to visit, and updating a label function $\lambda : V \mapsto \mathbb{R} \cup \{+\infty\}$, that satisfies the condition 2.1.

```
Data: Graph, weight, origin
U := {o} ;
λ(o) := 0 ;
∀v ≠ o,   λ(v) = +∞ ;

while U ≠ ∅ do
    choose u ∈ U ;
    for v successor of u do
        if λ(v) > λ(u) + ℓ(u, v) then
            λ(v) := λ(u) + ℓ(u, v);
            U := U ∪ {v};
    U := U\{u} ;
```

Algorithm 1: Generic label algorithm to find shortest path to any vertexes of G from o

We have the following properties of the label algorithm.

1. If $\lambda(u) < \infty$ then $\lambda(u)$ is the cost of a o-u-path.
2. If $u \notin U$ then
   - either $\lambda(i) = \infty$ (never visited)
   - or
     $$\text{for all successor } v \text{ of } u, \qquad \lambda(v) \leqslant \lambda(u) + \ell(u,v).$$
3. If the algorithm ends $\lambda(u)$ is the smallest cost to go from o to u.
4. Algorithm end iff there is no path starting at o and containing an absorbing circuit.

The proof is left as exercise.

[1] with the convention $\infty - \infty = \infty$.

2.2.2 Dijkstra's algorithm

We assume in this section that all costs are non-negative.

Algorithm 1 is abstract in the sense that we do not specify how to choose, at the beginning of any iteration, the vertex $u \in U$ to treat. Dijkstra's algorithm 2 is a special case of Algorithm 1 where we select a vertex of minimum label.

$U := \{o\}$;
$\lambda(o) := 0$;
$\forall v \neq o, \quad \lambda(v) = +\infty$;

while $U \neq \varnothing$ **do**
  choose $u \in \arg\min_{u' \in U} \lambda(u')$;
  for v *successor of* u **do**
    if $\lambda(v) > \lambda(u) + \ell(u, v)$ **then**
      $\lambda(v) := \lambda(u) + \ell(u, v)$;
      $U := U \cup \{v\}$;
  $U := U \backslash \{u\}$;

Algorithm 2: Dijkstra algorithm

An instance of Dijsktra is represented in Figure 2.2. Other can be found in exercises.

The main interest of this method of selection is that each node will be selected at most once in Dijkstra's algorithm.

Theorem 2.2. *Let $G = (V, E)$ be a directed graph, $o \in V$ and a cost function $\ell : E \to \mathbb{R}_+$.*

When applying Dijkstra's algorithm, each node is visited at most once. Once a node v has been visited its label is constant accross iterations and equal to the cost of shortest o-v-path.

In particular, a shortest path from o to any vertex v can be found in $O(n^2)$, where $n = |V|$.

Proof. By induction we show that after a node $v \in V$ has been selected, the label $\lambda^k(v)$ is constant in k and equal to the shortest o-v-path cost.

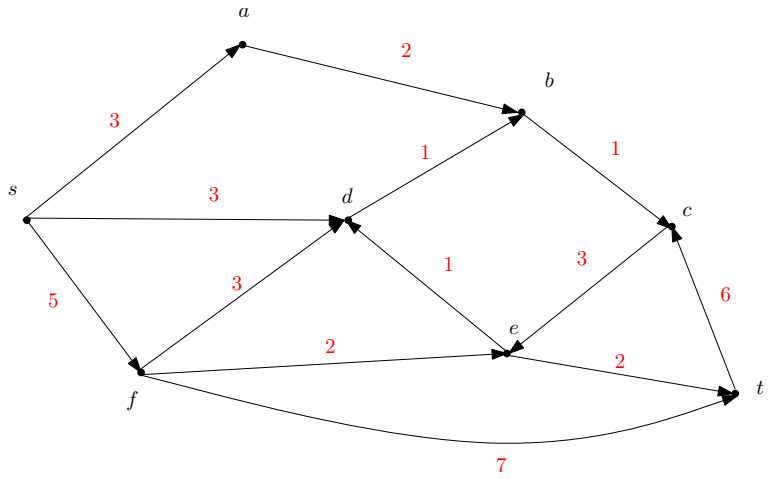
The costs being non-negative it's obvious for o. Let v be the current selected node, and assume that the properties hold true for all previously selected node. Thus $\lambda^k(v)$ is minimal among all non-visited node, consequently, by triangular inequality, any optimal o-v-path can only go through already selected nodes, and by construction $\lambda^k(v)$ is the cost of a shortest path.

Finally, there are at most n nodes to treat, and treating them consists in comparing the costs of at most n arcs, which gives the $O(n^2)$ complexity. □

Note that with specific implementation (e.g. in binary tree of nodes) we can obtain a complexity in $O(n + m \log(\log(m)))$.

2.3 Dynamic Programming

Dynamic Programming (DP) is also a label-like algorithm. Unfortunately it does not exactly fall in the framework of Algorithm 1. As we will see, in DP,



(a) Graphs with non-negative costs

k	s	a	b	c	d	e	f	t
0	(**0**)	∞	∞	∞	∞	∞	∞	∞
1	0	(**3**)	∞	∞	(3)	∞	(5)	∞
2	0	3	(5)	∞	(**3**)	∞	(5)	∞
3	0	3	(**4**)	∞	3	∞	(5)	∞
4	0	3	4	(**5**)	3	∞	(5)	∞
5	0	3	4	5	3	(8)	(**5**)	∞
6	0	3	4	5	3	(**7**)	5	(12)
7	0	3	4	5	3	7	5	(**9**)
8	0	3	4	5	3	7	5	9

(b) Dijkstra's iterations

Figure 2.2: Each line of the table correspond to an iteration of Dijkstra's algorithm, giving for each vertex the current label λ. At iteration k, the number in parenthesis determine the vertexes in U, and the number in bold determined which vertex is selected.

instead of continously decreasing the label, we directly affect the cost of the shortest o-v-path to the selected node v. In order to do that we need to select the nodes in a specific *topological order*.

2.3.1 Topological Order

Definition 2.1. *A topological ordering of a graph is an ordering (injective function from V to $\mathbb{N}$) of the vertices such that the tail of every edge occurs earlier in the ordering than the head of the edge.*

Finding a topological ordering is necessary in a number of applications. For example, in order to compile a software there is a number n of actions the compiler has to perform, and a number m of precedence constraints (action i as to be done before action j). This can be represented as a graph, each node representing an action, and each edge a precedence constraint (the tail of the edge has to be done before doing the head). A topological order over this graph

is a sequence of task that satisfy the precedence constraints.

The following theorem characterize the graphs that admits a topological ordering : Directed Acyclic Graphs (DAG).

Theorem 2.3. *A directed graph is acyclic if and only if there exist a topological ordering. A topological ordering can be found in* $O(|V| + |A|)$.

We give the sketch of the proof.

Proof.

- If G has a topological ordering then it is acyclic. By contradiction : consider a cycle, find the vertex with lowest ordering, it's predecessor can't have a lower ordering.
- If G is a DAG, then it has a root node (with no incoming edges). By contradiction : if there is no root node, you can take the predecessor at most n times before constructing a cycle.
- If G is a DAG then G has a topological ordering. By induction: take a root node, add it at the beginning of the ordering of the subgraph.
- Done in $O(|V| + |E|)$ (maintain $count(v)$: number of incoming edges, S: set of remaining nodes with no incoming edges).

□

2.3.2 Forward Dynamic Programming algorithm

Dynamic Programming principle

The Dynamic Programming principle is simultaneously extremely simple and very useful : a part of an optimal path is still optimal. More precisely, consider a directed weigthed graph $G = (V, E)$, and a starting point $o \in V$. Let $\lambda(v)$ be the shortest o-v-path cost. Then, the Dynamic Programming Principle simply read

$$\lambda(v) = \min_{(u,v)\in E} (\lambda(u) + \ell(u, v)) \tag{2.2}$$

Which can be understood as: there exist a predecessor u of v such that the shortest path between o and v is given by the shortest path between o and u adding the edge (u, v).

A shortest path algorithm

Assume that the graph is *connected and without cycle.*

Data: Graph, cost function
$\lambda(s) := 0$;
$\forall v \neq s, \quad \lambda(v) = +\infty$;

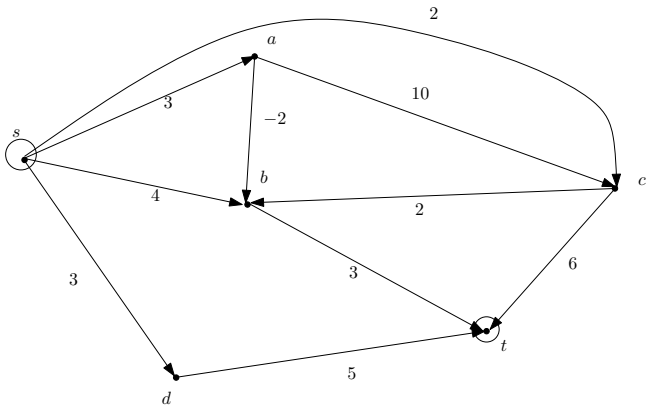
while $\exists v \in V, \quad \lambda(v) = \infty$ **do**
  choose a vertex v such that all predecessors u have a finite label ;
  $\lambda(v) := \min\{\lambda(u) + \ell(u, v) \quad | \quad (u, v) \in E\}$;

Algorithm 3: Bellman Forward algorithm

The while loop can be replaced by a for loop over the nodes in a topological order.

Theorem 2.4. *Let $D = (V, E)$ be a directed graph without cycle, and $w : A \to \mathbb{R}$ a cost function. The shortest path from o to any vertex $v \in V$ can be computed in $O(n + m)$.*

Note that we do not require the costs to be positive for the Bellman-Forward algorithm. In particular we can also compute the *longest path*.



(a) Acircuitic graph

s	a	c	b	d	t
0	∞	∞	∞	∞	∞
0	$0+3$	∞	∞	∞	∞
0	3	$\min\{0+2, 10+3\}$	∞	∞	∞
0	3	2	$\min\{0+4, 3-2, 2+2\}$	∞	∞
0	3	2	1	$0+3$	∞
0	3	2	1	3	4

(b) Bellman's iterations

Figure 2.3: Each line of the table corresponds to an iteration of Bellman's algorithm, giving for each vertex the current label λ.

2.4 $A^\star$ algorithm

Dijkstra's algorithm is used in industry for a wide range of problems. However, most of the times it is not implemented as presented in section 2.2. There are a number of ways to improve the algorithm (like starting from both the starting point and the end, or precomputing some itineraries, and so on). We present here one of the most common way to improve Dijkstra's algorithm, to turn it into the so-called $A^\star$ algorithm

2.4.1 The algorithm

The base idea consists in realizing that Dijkstra's algorithm will compute the shortest path from the starting point in any direction, and not particularly in

the destination's direction. Let's say that you want the shortest path from Paris to Marseille, you will probably obtain the shortest path to Lille, Bruxelles or Strasbourg before obtaining Marseille's. Thus we are going to add an "heuristic" that orient the algorithm toward its destination.

More precisely, To reach destination d from origin o in a weighted directed graph we keep a label function $\lambda(n)$, with

$$\lambda(n) = g(n) + h(n) \tag{2.3}$$

where $g(n)$ is the cost of a o-n-path, and $h(n)$ is an (user-given) heuristic of the cost of a n-d-path. Then the algorithm reads as in Algorithm 4. Notice that it is a specific implementation of the generic label algorithm 1, and the only difference with Djikstra's algorithm is that instead of selecting the node closest to the origin, we select the node v with minimal λ where λ is an estimation of the total path cost, not just minimal o-v-distance.

$U := \{s\}$; $\lambda(s) := h(s)$; $\forall v \neq s, \quad \lambda(v) = g(v) = +\infty$;
while $U \neq \varnothing$ **do**
  choose $u \in \arg\min_{u' \in U} \lambda(u')$;
  for v *successor of* u **do**
    if $g(v) > g(u) + \ell(u, v)$ **then**
      $g(v) := g(u) + \ell(u, v)\}$;
      $\lambda(v) := g(v) + h(v)$;
      $U := U \cup \{v\}$;
  $U := U \backslash \{u\}$;

Algorithm 4: $A^\star$ algorithm

2.4.2 Heuristics and properties

Definition 2.2 (admissible heuristic)**.** *A heuristic is* admissible *if it underestimate the actual cost to get to the destination, i.e. if for all vertex $v \in V$, $h(v)$ is lower or equal to the cost of a shortest path from v to d.*

Example : in the case of a graph in $\mathbb{R}^2$ with a cost proportional to the euclidean distance, an admissible heuristic is the euclidean distance between v and t (the "direct flight" distance). $A^\star$ with an admissible heuristic yields the shortest path.

Definition 2.3 (consistent heuristic)**.** *The heuristic h is* consistent *if it is admissible and satisfies a triangle equality:*

$$\forall (u, v) \in E, \quad h(u) \leqslant \ell(u, v) + h(v). \tag{2.4}$$

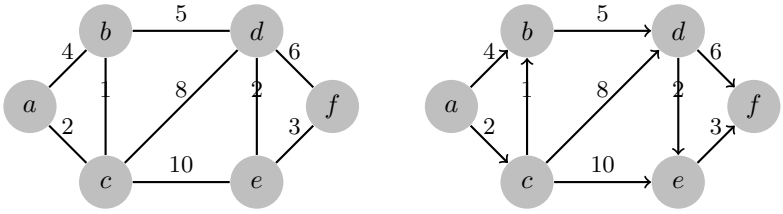
If h is consistent, $A^\star$ can be implemented more efficiently. Roughly speaking, no node needs to be processed more than once, and $A^\star$ is equivalent to running Dijkstra's algorithm with the reduced cost $\tilde{\ell}(u, v) = \ell(u, v) + h(v) - h(u)$.

Exercise 2.2.
1. *Show that $h \equiv 0$ is consistent, and in this case $A^\star$ reduce to Dijkstra.*
2. *Show that if h is exact ($h(v)$ is the cost of the shortest v-d-path), then h is consistent and $A^\star$ only visit the shortest o-d-path.*

2.5 Exercises

Answer to starred exercises can be found in Section B.2

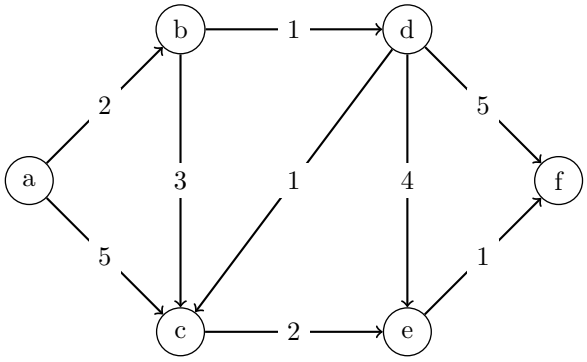
Exercise* 2.3. *Consider the following non-oriented weighted graph.*



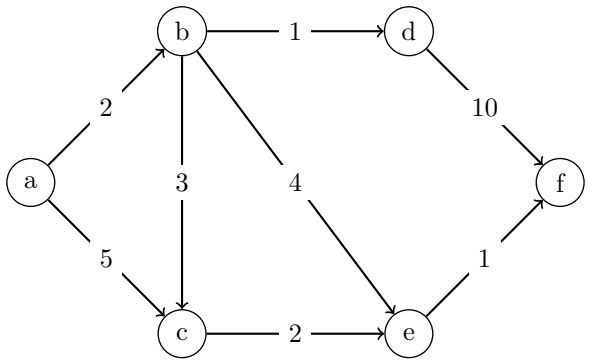
(a) non-oriented graph (b) oriented graph

1. *Why can Dijkstra's algorithm be used to find a shortest path on this graph (Figure 1) ?*
2. *Use Dijkstra's algorithm to find the shortest path between node a and node f. The results can be presented in a table of the labels where each column corresponds to a node of the graph, and each line to an iteration of the Dijkstra algorithm. Give the shortest path and its cost.*
3. *Can we find a topological order for this graph (Figure 2.4a) ? for the next one (Figure 2.4b) ?*
4. *Find the shortest a-f-path for the graph in Figure 2.*

Exercise* 2.4. *Consider the following weighted graph.*



1. *Use Dijkstra's algorithm to find the cost of the shortest path between node a and node f. The results can be presented in a table of the labels where each column corresponds to a node of the graph, and each line to an iteration of the Dijkstra algorithm.*
2. *Find a topological ordering for the graph. Use the topological ordering to compute the cost of the shortest path from a to every nodes by Dynamic Programming.*



3. *Give the shortest path from a to f.*

Exercise* 2.5. *Consider the following weighted graph.*

1. *Use Dijkstra's algorithm to find the cost of the shortest path between node a and node f. The results can be presented in a table of the labels where each column corresponds to a node of the graph, and each line to an iteration of the Dijkstra algorithm. Note the order in which the nodes are treated.*

2. *We have the following heuristic h giving an estimate of the distance between a given node and f.*

a	b	c	d	e	f
20	4	7	6	0	0

Apply the A algorithm using this heuristic. Note the order of nodes treated. Comment.*

CHAPTER 3

Wardrop Equilibrium

3.1 Definitions and social-optimum

3.1.1 Notations

Consider a (finite) directed graph $G = (V, E)$. We consider K origin-destination vertex pairs $\left\{o^k, d^k\right\}_{k\in[\![1,K]\!]}$, such that there exists at least one path from o^k to d^k. Let denotes:

- r^k the intensity of the flow of user entering in o^k and exiting in d^k;
- $\mathcal{P}_k$ the set of all simple (i.e. without cycle) path form o^k to d^k, and by $\mathcal{P} = \bigcup_{k=1}^{K} \mathcal{P}_k$;
- f_p the flux of user taking path $p \in \mathcal{P}$;
- $f = \left\{f_p\right\}_{p\in\mathcal{P}}$ the vector of path-flux;
- $x_e = \sum_{p\ni e} f_p$ the flux of user taking the edge $e \in E$;
- $x = \left\{x_e\right\}_{e\in E}$ the vector of edge-flux;
- $\ell_e : \mathbb{R} \to \mathbb{R}^+$ the cost incurred by a given user to take edge e, if the edge-intensity is x_e;
- $L_e(x_e) := \int_0^{x_e} \ell_e(u)du$.

3.1.2 First formulation

Minimizing the total cost of the system is an optimization problem that reads

$$\min_{x,f} \quad \sum_{e\in E} x_e \ell_e(x_e) \tag{3.1a}$$

$$s.t. \quad r_k = \sum_{p\in\mathcal{P}_k} f_p \qquad k \in [\![1, K]\!] \tag{3.1b}$$

$$x_e = \sum_{p\ni e} f_p \qquad e \in E \tag{3.1c}$$

$$f_p \geqslant 0 \qquad p \in \mathcal{P} \tag{3.1d}$$

Where constraint (3.1b) ensure that the flux going from o^k to d^k is spread among the different possible paths. Constraint (3.1c) is the definition of x_e as the sum of the users taking the different path containing edge e.

3.1.3 Reformulations

We can write the total cost, or system cost, in different ways using

$$x_e(f) := \sum_{p \ni e} f_p \qquad \forall e \in E, \tag{3.2}$$

and $x(f) = \{x_e \mid e \in E\}$

The total loss is given in function of arc-intensity, by

$$C(x) = \sum_{e \in E} x_e \ell_e(x_e),$$

where x_e is the number of people using edge e.

To write the system cost in function of the path-intensity f we need first to define the cost along a path $\ell_p(f)$, for a given flow f given by

$$\ell_p(f) = \sum_{e \in p} \ell_e\big(\underbrace{\sum_{p' \ni e} f_{p'}}_{x_e(f)} \big).$$

Thus the system cost in function of the flow f is given by

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

3.2 Wardrop Equilibrium

3.2.1 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 3.1. *A user flow f is a User Equilibrium if*

$$\forall k \in [\![1, K]\!], \quad \forall (p, p') \in \mathcal{P}_k^2, \qquad f_p > 0 \quad \Longrightarrow \quad \ell_p(f) \leqslant \ell_{p'}(f).$$

3.2.2 Equilibrium as optimality conditions

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 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

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

Theorem 3.1. *A flow f is a user equilibrium if and only if it satisfies the first order conditions of the following optimization problem*

$$\min_{x,f} \quad W(x) \tag{3.3a}$$

$$s.t. \quad r_k = \sum_{p \in \mathcal{P}_k} f_p \qquad k \in [\![1, K]\!] \tag{3.3b}$$

$$x_e = \sum_{p \ni e} f_p \qquad e \in E \tag{3.3c}$$

$$f_p \geqslant 0 \qquad p \in \mathcal{P} \tag{3.3d}$$

Proof. Let write Problem (3.3) only in path-intensity variables.

$$\min_{f} \quad \sum_{e \in E} L_e\Big(\sum_{p \ni e} f_p\Big) \tag{3.4a}$$

$$s.t. \quad r_k = \sum_{p \in \mathcal{P}_k} f_p \qquad k \in [\![1, K]\!] \tag{3.4b}$$

$$f_p \geqslant 0 \qquad p \in \mathcal{P} \tag{3.4c}$$

with Lagrangian

$$\mathcal{L}(f, \lambda, \mu) := W(f) + \sum_{k=1}^{K} \lambda_k \Big(r_k - \sum_{p \in \mathcal{P}_k} f_p\Big) + \sum_{p \in \mathcal{P}} \mu_p f_p.$$

Now note that we have

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

and the constraints of Problem (3.4) are qualified. Consequently its first-order KKT conditions reads

$$\begin{cases}
\frac{\partial \mathcal{L}(f,\lambda,\mu)}{\partial f_p} = \ell_p(f) - \lambda_k + \mu_p = 0 & \forall p \in \mathcal{P}_k, \forall k \in [\![1, K]\!] \\
\frac{\partial \mathcal{L}(f,\lambda,\mu)}{\partial \lambda_k} = r_k - \sum_{p \in \mathcal{P}_k} f_p = 0 & \forall k \in [\![1, K]\!] \\
\mu_p = 0 \text{ or } f_p = 0 & \forall p \in \mathcal{P} \\
\mu_p \leqslant 0, f_p \geqslant 0 & \forall p \in \mathcal{P}
\end{cases}$$

From which we deduce that, f satisfies the KKT conditions iff for all origin-destination pair $k \in [\![1, K]\!]$, and all path $p \in \mathcal{P}_k$ we have

$$\begin{cases}
\ell_p(f) = \lambda_k & \text{if } f_p > 0 \\
\ell_p(f) \geqslant \lambda_k & \text{if } f_p = 0
\end{cases} \tag{3.5}$$

In other words, if the path $p \in \mathcal{P}_k$ is used, then its cost is λ_k, and all other path $p' \in \mathcal{P}_i$ have a greater or equal cost, which is the definition of a User Equilibrium. □

Note that if the edge-loss ℓ_e is constant, then $W = C$, and equilibrium and system optimum are the same.

3.2.3 Convex case

If the loss functions (in edge-intensity) are non-decreasing then the Wardrop potential W is convex. Hence we have a stronger version of Theorem (3.1):

Theorem 3.2. *Assume that the loss function ℓ_e is 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 Problem* (3.3)*.*

Proof. For $e \in E$, if ℓ_e is non-decreasing, then L_e is convex, hence $\sum_{e\in E} L_e(\cdot)$ is convex, and by composition with a linear mapping $x \mapsto W(x)$ is convex as well. Thus Problem (3.4) is convex, with qualified constraint.

Consequently f is an equilibrium iff it satisfies KKT conditions of Problem (3.4), iff it solves Problem (3.4), iff it solves Problem (3.3) (x being deduced from the constraints). □

We end with another characterization of the equilibrium in the convex case. This characterization states that the flow minimizes the total cost with travel cost frozen at the equilibrium flow.

First, we need to define the system cost of a flow f' for a given flow f, as

$$C^f(f') := \sum_{e\in E} x_e(f')\ell_e\big(x_e(f)\big).$$

Theorem 3.3. *Assume that the cost functions ℓ_e are continuous and non-decreasing. Then, f^{UE} is a user equilibrium iff*

$$\forall f \in F^{ad}, \qquad C^{f^{UE}}(f^{UE}) \leqslant C^{f^{UE}}(f),$$

where F^{ad} is the set of admissible flows.

Proof. We are in the convex case, hence f^{UE} is an optimal solution of Problem (3.4) iff

$$\nabla W(f^{UE}) \cdot (f - f^{UE}) \geqslant 0, \qquad \forall f \in F^{ad}$$

which is equivalent to

$$\sum_{p\in\mathcal{P}} \underbrace{\frac{\partial W}{\partial f_p}(f^{UE})}_{\ell_p(f^{UE})} f_p \quad \geqslant \quad \sum_{p\in\mathcal{P}} \underbrace{\frac{\partial W}{\partial f_p}(f^{UE})}_{\ell_p(f^{UE})} f_p^{UE}, \qquad \forall f \in F^{ad}$$

which can be written

$$C^{f^{UE}}(f^{UE}) \leqslant C^{f^{UE}}(f), \qquad \forall f \in F^{ad}.$$

□

3.2.4 Price of anarchy

Definition 3.2. *Consider increasing loss functions ℓ_e. Let f^{UE} be a user equilibrium, and f^{SO} be a system optimum (i.e. a solution of Problem (3.1)). Then the price of anarchy of our network is given by*

$$PoA := \frac{C(f^{UE})}{C(f^{SO})} \geqslant 1.$$

Theorem 3.4. *Let ℓ_e be the affine function $x_e \mapsto b_e x_e + c_e$, with $b_e, c_e \geqslant 0$. Then the price of anarchy is lower than $4/3$, and the bound is tight.*

Proof. Let f be a feasible flow, and f^{UE} be the user equilibrium. For ease of notation we fix $x^{UE} = x(f^{UE})$, and $x = x(f)$. By Theorem 3.3 we have

$$\begin{aligned}
C(f^{UE}) &\leqslant C^{f^{UE}}(f) \\
&= \sum_{e\in E} \big(b_e x_e^{UE} + c_e\big)x_e \\
&\leqslant \sum_{e\in E} \Big[\big(b_e x_e + c_e\big)x_e + \frac{1}{4} b_e \big(x_e^{UE}\big)^2\Big] \qquad \text{as } (x_e - x_e^{UE}/2)^2 \geqslant 0 \\
&\leqslant C(f) + \frac{1}{4}\sum_{e\in E} \big(b_e x_e^{UE} + c_e\big)x_e^{UE} \qquad \text{as } c_e x_e^{UE} \geqslant 0 \\
&= C(f) + \frac{1}{4} C^{f^{UE}}(f^{UE})
\end{aligned}$$

Hence we have $3/4 C(f^{UE}) \leqslant C(f)$. Minimizing over admissible flow f ends the proof.

The Braess paradox example show that the bound is tight. □

3.3 Exercises

Answer to starred exercises can be found in Section B.3

Exercise 3.1.

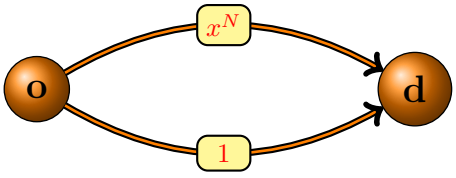


Figure 3.1: Pigou example

On a graph with two nodes: one origin, one destination, a total flow of 1, a fixed cost of 1 on one edge, and a cost of x^N on the other, where $N \in \mathbb{N}$ and x is the intensity of the flow using this edge (see Figure 3.1).

1. *Compute the system optimum for a given N.*

2. *Compute the user equilibrium for a given N.*

3. *Compute the price of anarchy on this network when $N \to \infty$.*

Exercise* 3.2. *Consider a (finite) directed, strongly connected, graph $G = (V, E)$. We consider K origin-destination vertex pair $\left\{o^k, d^k\right\}_{k\in[\![1,K]\!]}$, such that there exists at least one path from o^k to d^k.*

We want to find bounds on the price of anarchy, assuming that, for each arc e, $\ell_e : \mathbb{R}^+ \to \mathbb{R}^+$ is non-decreasing, and that we have

$$x\ell_e(x) \leqslant \gamma L_e(x), \qquad \forall x \in \mathbb{R}^+$$

1. *Recall which optimization problems solves the social optimum x^{SO} and the user equilibrium x^{UE}. We will denote $W(x)$ the objectif function of the user equilibrium problem and $C(x)$ the objective function of the social optimum problem.*

2. *Let x be a feasable vector of arc-intensity. Show that $W(x) \leqslant C(x) \leqslant \gamma W(x)$.*

3. *Show that the price of anarchy $C(x^{UE})/C(x^{SO})$ is lower than γ.*

4. *If the cost per arc ℓ_e are polynomial of order at most p with non-negative coefficient, find a bound on the price of anarchy. Is this bound sharp ?*

Exercise* 3.3. *Consider a (finite) directed, strongly connected, graph $G = (V, E)$. We consider K origin-destination vertex pair $\left\{o^k, d^k\right\}_{k\in[\![1,K]\!]}$. We denote by (G, ℓ, r) the congestion game with rate r_k from o^k to d^k. We want to compare the cost of the user equilibrium of (G, ℓ, r), denoted $f^{UE,r}$, with the cost of the social optimum $f^{SO,2r}$ of $(G, \ell, 2r)$, that is the same game with twice the inflows. Accordingly we denote $x^{UE,r} = x(f^{UE,r})$, and $x^{SO,2r} = x(f^{SO,2r})$. Finally, edge-loss ℓ_e are assumed to be non-negative and non-decreasing.*

We construct new loss functions $\bar{\ell}_e(x)$ given by

$$\bar{\ell}_e(x) = \begin{cases} \ell_e(x^{UE,r}) & \text{if } x \leqslant x^{UE,r} \\ \ell_e(x) & \text{else} \end{cases}$$

Accordingly we denote $\bar{\ell}_p(f) = \sum_{e\in p} \bar{\ell}_e(x_e(f))$ and

$$C(x) = \sum_{e\in E} x_e\ell_e(x_e) \qquad \text{and} \qquad \bar{C}(x) = \sum_{e\in E} x_e\bar{\ell}_e(x_e).$$

1. *Justify that for all $k \in [\![1, K]\!]$, there exists $\lambda_k \in \mathbb{R}_+$ such that for all path $p \in \mathcal{P}_k$,*
$$f_p^{UE,r} > 0 \;\Rightarrow\; \ell_p(f^{UE,r}) = \lambda_k.$$

2. *Show that, for any $x \in \mathbb{R}_+^{|E|}$, $C(x) \leqslant \bar{C}(x)$, and that $C(x^{UE,r}) = \bar{C}(x^{UE,r})$.*

3. *Show that, for any $x \in \mathbb{R}_+^{|E|}$, $x_e(\bar{\ell}_e(x_e) - \ell_e(x_e)) \leqslant x_e^{UE,r}\ell_e(x_e^{UE,r})$.*

4. *Deduce that, $\bar{C}(x^{SO,2r}) - C(x^{SO,2r}) \leqslant C(x^{UE,r})$.*

5. *On the other hand, show that, for every path $p \in \mathcal{P}_k$, $\bar{\ell}_p(f^{SO,2r}) \geqslant \lambda_k$.*

6. *Write C and $\bar{C}$ as function of f instead of x (we keep the same notation).*

7. *Deduce that,* $\bar{C}(f^{SO,2r}) \geqslant 2C(f^{UE,r})$.

8. *Finally, show that,* $C(f^{UE,r}) \leqslant C(f^{SO,2r})$. *Give an interpretation of this result.*

Exercise* 3.4. *The price of anarchy of a class* $\mathcal{C}$ *of cost functions is the highest price of anarchy obtained by choosing any (finite) graph, with any rate (i.e. input/output flow) and any cost function in the class.*

We will assume that $\mathcal{C}$ *contains the constant functions. Moreover we assume that* $\mathcal{C}$ *contains only non-decreasing functions.*

1. *What is the price of anarchy if* $\mathcal{C}$ *is the class of non-decreasing affine functions?*

2. *What is the price of anarchy if* $\mathcal{C}$ *is the class of non-decreasing polynomial functions?*

A Pigou network is a network with two nodes: one origin o*, one destination* d*, and two arcs linking* o *to* d*, and a flow rate between* o *and* d *of* $r > 0$*. The cost function of the first arc is* $c \in \mathcal{C}$*, and the cost function of the second arc is constant equal to* $c(r)$.

3. *What is the user equilibrium, social optimum and price of anarchy of a Pigou network with given rate* $r > 0$*? (The result is not a closed formula but contains a max or a min).*

4. *Deduce a lower bound of the price of anarchy of the class* $\mathcal{C}$.

5. *Show that this lower bound is exact for the class of affine non-decreasing functions.*

Exercise* 3.5. *Consider a (finite) directed, strongly connected, graph* $G = (V, E)$*. We consider* K *origin-destination vertex pair* $\left\{o^k, d^k\right\}_{k\in[\![1,K]\!]}$*. We denote by* (G, ℓ, r) *the congestion game with inflow vector* r.

- r^k *is the intensity of the flow of users entering in* o^k *and exiting in* d^k*;*
- $\mathcal{P}_k$ *is the set of all simple (i.e. without cycle) paths from* o^k *to* d^k*, and by* $\mathcal{P} = \bigcup_{k=1}^{K} \mathcal{P}_k$ *;*
- f_p *the number of users taking path* $p \in \mathcal{P}$ *per hour (intensity);*
- $f = \left\{f_p\right\}_{p\in\mathcal{P}}$ *the vector of path intensity;*
- $x_e = \sum_{p \ni e} f_p$ *the flux of user taking the edge* $e \in E$*;*
- $x = \left\{x_e\right\}_{e\in E}$ *the vector of edge intensity;*
- $x(f)$ *is the vector of edge-intensity induced by the path intensity* f*;*
- $\ell_e : \mathbb{R} \to \mathbb{R}^+$ *the cost incurred by a given user to take edge* e*, if the edge-intensity is* x_e*;*
- $L_e(x_e) := \int_0^{x_e} \ell_e(u)du$.

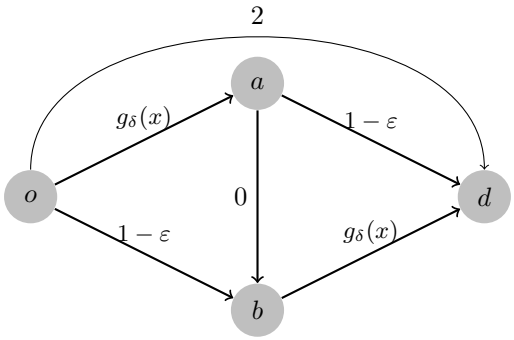


Figure 3.2: A graph example

We say that an admissible flow f^ε, for $\varepsilon \in [0,1[$ is a **ε-Nash Equilibrium** *if*

$$\forall k \in [\![1,K]\!], \quad \forall p_1,p_2 \in \mathcal{P}_k, \quad f^\varepsilon_{p_1} > 0 \quad \Longrightarrow \ell_{p_1}(f^\varepsilon) \leqslant (1+\varepsilon)\ell_{p_2}(f^\varepsilon).$$

We want to compare the cost of a given ε-equilibrium of (G,ℓ,r), denoted $f^{\varepsilon,r}$, with the cost of the social optimum $f^{SO,2r}$ of $(G,\ell,2r)$, that is the same game with twice the inflows. Accordingly we denote $x^{\varepsilon,r} = x(f^{\varepsilon,r})$, and $x^{SO,2r} = x(f^{SO,2r})$. Finally, edge-loss ℓ_e are assumed to be non-negative and non-decreasing.

Both parts are largely independent.

Part I : an example

We consider, for $\varepsilon \in [0,1[$, the congestion game (G,ℓ,r) given in Figure 3.2 with the unique origin destination pair $o-d$. Here, g_δ is a continuous non-decreasing function with value 0 on $]-\infty, 1-\delta]$ and value $1+\varepsilon$ on $[1,+\infty[$.

1. *Show that a flow $f^{\varepsilon,1}$ getting 1 through $o \to a \to b \to d$, and 0 on other paths, is a ε-Nash Equilibrium of $(G,\ell,1)$.*
2. *Construct an admissible flow of $(G,\ell,2)$ of cost $4\delta + 2(1-\varepsilon)(1-\delta)$.*
3. *Show that the social optimum of $(G,\ell,2)$ can be found by solving an uni-dimensional optimization problem, and propose an adapted optimization algorithm.*

Part II : bounding the cost of ε-Nash Equilibrium

We construct new loss functions $\bar{\ell}_e(x)$ given by

$$\bar{\ell}_e(x) = \begin{cases} \ell_e(x^{\varepsilon,r}) & \text{if } x \leqslant x^{\varepsilon,r} \\ \ell_e(x) & \text{else} \end{cases}$$

Accordingly we denote $\bar{\ell}_p(f) = \sum_{e\in p} \bar{\ell}_e(x_e(f))$ and

$$C(x) = \sum_{e\in E} x_e \ell_e(x_e) \qquad \text{and} \qquad \bar{C}(x) = \sum_{e\in E} x_e \bar{\ell}_e(x_e).$$

For $k \in [\![1, K]\!]$, denote $\lambda_k(x)$ the minimum cost of an o_k-d_k-path with costs given by edge-intensity vector x.

1. *Give an interpretation of an ε-Nash Equilibrium. What happens if $\varepsilon = 0$?*
2. *Show that $C(x^{\varepsilon,r}) \leqslant (1+\varepsilon)\sum_{k=1}^{K} r_k \lambda_k(x^{\varepsilon,r})$.*
3. *Show that, for any $x \in \mathbb{R}_+^{|E|}$, $x_e(\bar{\ell}_e(x_e) - \ell_e(x_e)) \leqslant x_e^{\varepsilon,r}\ell_e(x_e^{\varepsilon,r})$.*
4. *Deduce that, $\bar{C}(x^{SO,2r}) - C(x^{SO,2r}) \leqslant C(x^{\varepsilon,r})$.*
5. *Show that, for all $p \in \mathcal{P}_k$, $\bar{\ell}_p(f^{SO,2r}) \geqslant \lambda_k(x^{\varepsilon,r})$.*
6. *Show that*
$$\sum_{p\in\mathcal{P}} \bar{\ell}_p(f^{SO,2r}) f_p^{SO,2r} \geqslant \frac{2}{1+\varepsilon} C(x^{\varepsilon,r})$$
7. *Find a constant K_ε such that $C(x^{\varepsilon,r}) \leqslant K_\varepsilon C(x^{SO,2r})$.*
8. *Using part I show that this bound is tight.*

CHAPTER 4

Numerical Methods

4.1 Some optimization algorithms

Consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} \quad f(x). \tag{4.1}$$

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

$$x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)} \tag{4.2}$$

where

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

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

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

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

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

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

In the remain of this section we will first give algorithm for finding optimal step size, and then present a new algorithm for constrained optimization.

4.1.1 Unidimensional optimization

Here we assume that the objective function $J : \mathbb{R} \to \mathbb{R}$ is strictly convex[1] and we propose algorithms that find the minimum over $[a, b]$. We start with two interval reduction algorithms, that constructs intervals $[a^{(l)}, b^{(l)}]$ containing the optimal solution t^*. We note $L_l = b^{(l)} - a^{(l)}$ the length of the interval, and the speed of the algorithm is given by the speed at which L_l goes toward 0.

[1] extension to simple convexity, or even lesser conditions is straightforward.

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.

Data: objective function J, interval $[a, b]$, error ε
Result: interval $[a^{(l)}, b^{(l)}]$ containing t^* such that $b^{(l)} - a^{(l)} < \varepsilon$
$a^{(0)} = a$; $b^{(0)} = b$;

while $b^{(l)} - a^{(l)} > \varepsilon$ **do**
  $c^{(l)} = \frac{b^{(l)}+a^{(l)}}{2}$;
  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 $[a^{(l)}, b^{(l)}]$
  $l = l + 1$

Algorithm 5: Bisection algorithm

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

Golden section method

The bisection method is simple and fast but requires computing the gradient at each iteration. The golden section method does not require any gradient computation.

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 $[t_1^{(l)}, b^{(l)}]$. 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 possibility, 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

$$\begin{cases} L_2 + L_1 = L \\ \frac{L_2}{L} = \frac{L_1}{L_2} =: R \end{cases} \tag{4.3}$$

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

$$1 + R = \frac{1}{R} \tag{4.4}$$

and thus

$$R = \frac{\sqrt{5}-1}{2}. \tag{4.5}$$

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. Thus the golden section algorithm reads

Data: objective function J, interval $[a, b]$, error ε
Result: interval $[a^{(l)}, b^{(l)}]$ containing t^* such that $b^{(l)} - a^{(l)} < \varepsilon$
$a^{(0)} = a, \quad b^{(0)} = b;$
$t_1^{(0)} = a + (1-R)b, \quad t_2^{(0)} = a + Rb;$
$J_1 = J(t_1^{(0)}), \quad J_2 = J(t_2^{(0)});$

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

Algorithm 6: Golden section algorithm

Note that $L_l = R^l L_0$.

Curve fitting methods

Another approach consists in fitting a polynomial to J and finding its minimum. There are different ways to do that.

One of them consists in keeping three points $t_1 < t_2 < t_3$ such that $J(t_1) > J(t_2) < J(t_3)$, then fit a polynomial $P^{(k)}$ to these points, find the minimum of $P^{(k)}$ and update one of the three points.

Another one, 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.

4.1.2 Conditional Gradient Algorithm

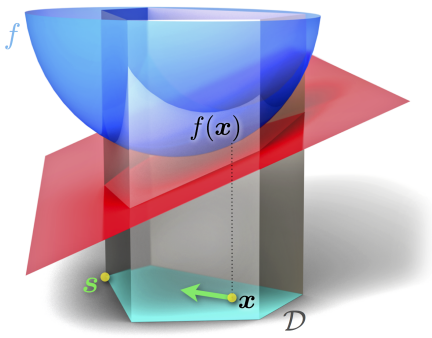


Figure 4.1: Conditional gradient algorithm[2]

The conditional gradient algorithm, also known as Frank-Wolfe algorithm, or convex combination method, address an optimization problem with convex objective function f and compact polyhedral constraint set X, i.e.

$$\min_{x \in X \subset \mathbb{R}^n} \quad f(x) \tag{4.6}$$

where

$$X = \left\{x \in \mathbb{R}^n \quad | \quad Ax \leqslant b, \quad \tilde{A}x = \tilde{b}\right\} \tag{4.7}$$

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 $x^{(k)}$,

$$f(y) \geqslant f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)}). \tag{4.8}$$

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

$$y^{(k)} \in \arg\min_{y \in X} \quad f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (y - x^{(k)}). \tag{4.9}$$

Note that:

- Problem (4.9) is linear, hence easy to solve.

[2] illustration by S.Stutz and M.Jaggi

- By the convexity inequality (4.8), the value of Problem (4.9) is a lower bound to our problem (4.6).
- As $y^{(k)} \in X$, $d^{(k)} = y^{(k)} - x^{(k)}$ is a *feasable direction*, in the sense that we can move infinitesimally from $x^{(k)}$ in the direction $d^{(k)}$.
- More precisely, for all $t \in [0, 1]$, $x^{(k)} + td^{(k)} \in X$.
- Furthermore, if $y^{(k)}$ is obtained through the simplex method it is an extreme point of X, which means that, for $t > 1$, $x^{(k)} + td^{(k)} \notin X$.
- Finally, if $y^{(k)} = x^{(k)}$ then they are equal to an optimal solution of Problem (4.6).
- We also have $y^{(k)} \in \arg\min_{x \in X} \nabla f(x^{(k)}) \cdot y$, the lower-bound being obtained easily.

Finally, the conditional gradient algorithm goes as follow.

Data: objective function f, constraints, initial point $x^{(0)}$, precision ε
Result: ε-optimal solution $x^{(k)}$, upperbound $f(x^{(k)})$, lowerbound $\underline{f}$
$\underline{f} = -\infty$;
$k = 0$;

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 $t^{(k)} \in \underset{t \in [0,1]}{\arg\min} f\left(x^{(k)} + td^{(k)}\right)$;
  update $x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)}$;
  $k = k + 1$;

4.2 Algorithm for computing User Equilibrium

4.2.1 Heuristics algorithms

All-or-nothing approach

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 to compute the shortest path in a fixed cost graph. However it is not converging as it can cycle.

Smoothed all-or-nothing approach

The all-or-nothing method can be understood as follow: each day every user choose the shortest path according to the traffice on the previous day. We can smooth the approach by saying that only a fraction ρ of user 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.

This method is better behaved.

Incremental loading

For more stability we can adapt the previous methods by modifying the flow only for one origin-destination pair at a time.

4.2.2 Frank-Wolfe algorithm applied to the User-Equilibrium problem

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

$$\begin{aligned}
\min_f \quad & W(x(f)) && && (4.10a)\\
s.t. \quad & r_k = \sum_{p\in\mathcal{P}_k} f_p && k \in [\![1, K]\!] && (4.10b)\\
& f_p \geqslant 0 && p \in \mathcal{P} && (4.10c)
\end{aligned}$$

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

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

Note that the constraints in problem (4.10) are linear with convex objective function. 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

$$\min_{\{y_p\}_{p\in\mathcal{P}}} \quad \sum_{p\in\mathcal{P}} y_p\ell_p(f^{(\kappa)}) \tag{4.11a}$$

$$s.t \quad r_k = \sum_{p\in\mathcal{P}_k} y_p \qquad k \in [\![1,K]\!] \tag{4.11b}$$

$$y_p \geqslant 0 \qquad p \in \mathcal{P} \tag{4.11c}$$

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.

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.

In the end, the Frank-Wolfe algorithm applied to the user-equilibrium prob-

lem is a smoothed all-or-nothing approach, which reads:

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 $t^{(\kappa)} \in \underset{t\in[0,1]}{\arg\min}\, W\left(x^{(\kappa)} + td^{(\kappa)}\right)$;
  update $f^{(k+1)} = f^{(\kappa)} + t^{(\kappa)} d^{(\kappa)}$;
  $\kappa = \kappa + 1$;

4.3 Exercises

Answer to starred exercises can be found in Section B.4

Exercise* 4.1. *We consider the problem*

$$\min_{x\in[-1,1]^2} \quad f(x) := (x_1 - 2)^2 + (x_2 + 1)^2.$$

We want to solve this problem through Frank-Wolfe algorithm (a.k.a Conditional Gradient algorithm).

1. *Will the Frank-Wolfe algorithm converge ?*
2. *Compute the gradient of J.*
3. *Assume that $x^{(0)} = (0, 0)$. Write and solve the linear problem that is part of the first iteration of the Frank-Wolfe algorithm.*
4. *Write and solve the linear search problem that is part of the first iteration of the Frank-Wolfe algorithm. Find the new point $x^{(1)}$.*
5. *Write the linear problem part of the second iteration of the Frank-Wolfe algorithm.*

Exercise* 4.2. *Consider the function $f(x_1, x_2) = 4x_1^4 - 2x_1 + x_2^2 - x_2 + 2$, and the set*

$$X = \left\{x \in \mathbb{R}_+^2 \quad | \quad 2x_2 + x_1 \leqslant 2\right\}$$

and $x^0 = (0, 0)$. A scheme of X representing the iteration and search direction of the algorithm might be helpful.

1. *Justify that X is polyhedral and find its extreme points.*
2. *Compute ∇f*

3. *Justify that this problem can be solved by Frank-Wolfe (aka conditional gradient) algorithm.*

4. *Find the descent direction* d^0 *of the Frank-Wolfe algorithm starting from* x_0.
   (hint : use the extreme points of X*).*

5. *Find the optimal step* t^0 *of the first step of Frank-Wolfe algorithm. What is the new point* x^1 *?*

6. *What is the upper and lower bound obtained along this first iteration ?*

7. *Find the descent direction* d^1 *of the second step of Frank-Wolfe algorithm.*

8. *Write the unidimensional optimisation problem that would determine the next optimal step* t^1 *(do not solve it).*

9. *Compute the lower bound associated to the second step of the algorithm.*

APPENDIX A

Recall on optimization

A.1 Convexity

We recall a few simple results on convexity.

A.1.1 Generic results

A set $C \subset \mathbb{R}^n$ is *convex* iff

$$\forall x, y \in C, \quad \forall t \in [0,1], \qquad tx + (1-t)y \in C.$$

Intersection of convex sets is convex. A closed convex set C is equal to the intersection of all half-spaces containing it.

The *epigraph* of a function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is the set of point that lies above the graph of f, i.e.

$$\mathrm{epi}(f) := \big\{(x,t) \in \mathbb{R}^n \times \mathbb{R} \quad | \quad t \geqslant f(x)\big\}.$$

The *domain* of a function f is the set of points where f does not take value $+\infty$.

$$\mathrm{dom}(f) := \big\{x \in \mathbb{R}^n \quad | \quad f(x) < +\infty\big\}$$

The function f is said to be *convex* iff its epigraph is convex, in other words iff

$$\forall x, y \in \mathrm{dom}(f), \quad \forall t \in [0,1], \qquad f(tx + (1-t)y) \leqslant tf(x) + (1-t)f(y).$$

The function f is said to be *strictly convex* iff

$$\forall x, y \in \mathrm{dom}(f), \quad \forall t \in (0,1), \qquad f(tx + (1-t)y) < tf(x) + (1-t)f(y).$$

A.1.2 Differentiable functions

We now assume that f is differentiable on its domain.

Recall that if f is defined on $\mathbb{R}$, then f is convex iff f' is non-decreasing. Generically f is convex iff

$$\forall x, y \in \mathrm{dom}(f), \quad \langle \nabla f(y) - \nabla f(x), y - x \rangle \geqslant 0.$$

Assume further that f is twice differentiable on its domain. Recall that if f is defined on $\mathbb{R}$, then f is convex iff f'' is non-negative. Generically, f is convex iff its hessian $\nabla^2 f(x)$ is semi-definite positive for every $x \in \mathrm{dom}(f)$.

Finally, a convex function remains above all its tangeant. More precisely we have the following Proposition.

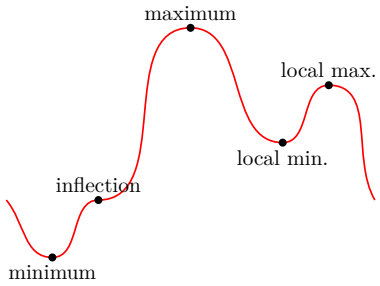


Figure A.1: Example of critical points

Proposition A.1. *Let f be a convex differentiable function, then for all $x \in \mathrm{dom}(f)$, we have*

$$\forall y \in \mathbb{R}^n, \qquad f(y) \geqslant \langle \nabla f(x), y - x \rangle. \tag{A.1}$$

A.1.3 Some operation preserving convexity

Let E be a vector space. Assume that I is a set (not necessarily finite), and for all $i \in I$, $C_i \subset E$ is convex, $f_i : C \mapsto \mathbb{R} \cup \{+\infty\}$.

Then we have that

- $\cap_{i \in I} C_i$ is convex,
- $\sup_{i \in I} f_i(\cdot)$ is convex,
- $\sum_{i=1}^n \alpha_i f_i(\cdot)$, with $\alpha \geqslant 0$ is convex,
- $f \circ A$ is convex (where A is affine).

A.1.4 Why convexity is usefull ?

Convexity largely simplify any optimization problem. Indeed, if f is convex then any local minimizer is global. If f is strictly convex, then there is at most one minimizer of f.

A.2 Optimality conditions

A.2.1 A story of first order conditions

Let start with the unconstrained real case. Consider a differentiable function $f : \mathbb{R} \to \mathbb{R}$, and $x^\sharp$ a minimizer of f. Then, we know that $f'(x^\sharp) = 0$. The *first order condition* is thus to look for all point $x \in \mathbb{R}$ satisfying

$$f'(x) = 0. \tag{A.2}$$

Multiple points satisfies (A.2), see Figure A.1. Hence, satisfying a first order condition is not enough to be a minimizer. For two reasons : first we do not differentiate between minimum and maximum, second the condition is only local. Nonetheless the condition is usefull as it drastically reduce the set of points that can be minimum (from $\mathbb{R}$ to, generally, a few points).

The same thing happens for a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, except that the first order condition now reads

$$\nabla f(x) = 0. \tag{A.3}$$

Also, in addition to being local minimizer / maximizer and inflexion points, an x satisfying the first order condition can be a local minimizer along a variable and a local maximizer along another.

The remain of this section extend this analysis to the constrained case.

A.2.2 Convex and differentiable optimization problem

Consider the following optimization problem.

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) && (P) && \text{(A.4a)}\\ \text{s.t.} \quad & g_i(x) = 0 && \forall i \in [n_E] && \text{(A.4b)}\\ & h_j(x) \leqslant 0 && \forall j \in [n_I] && \text{(A.4c)} \end{aligned}$$

Denote X the set of admissible solution

$$X := \big\{x \in \mathbb{R}^n \mid \forall i \in [n_E], \quad g_i(x) = 0, \quad \forall j \in [n_I], \quad h_j(x) \leqslant 0\big\}.$$

We say that (P) is a *convex optimization problem* if f and X are convex. We say that (P) is a *convex differentiable optimization problem* if f, and h_j (for $j \in [n_I]$) are convex differentiable and g_i (for $i \in [n_E]$) are affine, in which case X is convex.

A.2.3 Optimality conditions

Optimality condition in the convex case

In the convex differentiable case we also have the following necessary and sufficient condition of optimality.

Theorem A.1. *If (P) is a convex differentiable optimization problem, then $x^\sharp \in X$ is an optimal solution iff*

$$\forall y \in X, \quad \nabla f(x) \cdot (y - x) \geqslant 0.$$

KKT conditions

A convex optimization problem (P) satisfies the *Slater* condition if there exists $x_0 \in \mathbb{R}^n$ such that

$$\forall i \in [n_E], \quad g_i(x_0) = 0, \quad \forall j \in [n_I], \quad h_j(x_0) < 0.$$

If the Slater condition is satisfied, then the constraints are qualified at any $x \in X$.

Let recall the Karush-Kuhn-Tucker first order conditions

Theorem A.2 (KKT)**.** *Let $x^\sharp$ be an optimal solution to a differentiable optimization problem (P). If the constraints are qualified at $x^\sharp$ then there exists optimal multipliers $\lambda^\sharp \in \mathbb{R}^{n_E}$ and $\mu^\sharp \in \mathbb{R}^{n_I}$ satisfying*

$$\begin{cases} \nabla f(x^\sharp) + \sum_{i=1}^{n} \lambda_i^\sharp \nabla g_i(x^\sharp) + \sum_{j=1}^{n_I} \mu_i^\sharp \nabla h_j(x^\sharp) = 0 & \textit{first order condition} \\ g(x^\sharp) = 0 & \textit{primal admissibility} \\ h(x^\sharp) \leqslant 0 & \\ \mu \geqslant 0 & \textit{dual admissibility} \\ \mu_i h_i(x^\sharp) = 0, \quad \forall i \in [\![1, n_I]\!] & \textit{complementarity} \end{cases}$$

The three last conditions are sometimes compactly written

$$0 \geqslant g(x^\sharp) \perp \mu \geqslant 0.$$

The more complex constraint by far is the last one, also called complementarity condition. It state that, for any $j \in [n_I]$, μ_j or $g_j(x^\sharp)$ is null. In other words if the multiplier μ_j is non-null, then the constraint j is saturated (i.e. the inequality is an equality). Reciprocately if constraint g is not saturated (i.e. inequality is strict) then $\mu_j = 0$.

A.2.4 Lagrangian

A good way of writing down the KKT conditions is to rewrite Problem (A.4) using the Lagrangian.

Before introducing the Lagrangian we start by reformulating Problem (A.4). For any set $X \subset \mathbb{R}^n$, define the indicator function

$$\mathbb{I}_X(x) = \begin{cases} 0 & \text{if } x \in X \\ +\infty & \text{otherwise} \end{cases}$$

Note that, for any function $f : \mathbb{R}^n \to \mathbb{R}$ and any set $X \subset \mathbb{R}^n$, minimizing f over X is equivalent to minimizing $f + \mathbb{I}_X$. Indeed, if $x \in X$, then $f(x) = f + \mathbb{I}_X(x)$; and if $x \notin X$, $f + \mathbb{I}_X(x) = +\infty$ so x cannot be the minimum. Hence Problem (A.4) can be written

$$\min_{x \in \mathbb{R}^n} \quad f(x) + \sum_{i=1}^{n_E} \mathbb{I}_{\{0\}}(g_i(x)) + \sum_{i=1}^{n_I} \mathbb{I}_{\mathbb{R}^-}(h_i(x)) \tag{A.5}$$

Obviously this is just a notational trick. To go further we use the following remark:

$$\sup_{\lambda \in \mathbb{R}} \lambda x = \delta_{\{0\}}(x).$$

Indeed, if $x > 0$, letting λ goes toward $+\infty$ yield $+\infty$, if $x < 0$, letting λ goes toward $-\infty$ yield $+\infty$, while if $x = 0$, $\lambda x = 0$ for any λ. Similarily,

$$\sup_{\mu \in \mathbb{R}^+} \mu x = \delta_{\mathbb{R}^-}(x).$$

Indeed, if x is non-positive, then the product is always non-positive, and the best you can do is 0. Otherwise you can go to $+\infty$. Hence, we can write Problem (A.5) as

$$\min_{x \in \mathbb{R}^n} \quad f(x) + \sum_{i=1}^{n_E} \sup_{\lambda_i \in \mathbb{R}} \lambda_i g_i(x) + \sum_{j=1}^{n_I} \sup_{\mu_j \in \mathbb{R}^+} h_j(x) \tag{A.6}$$

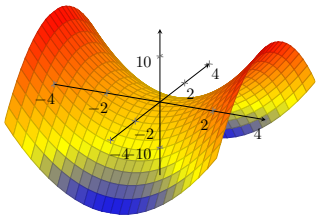


Figure A.2: $(0, 0)$ is a saddle point of this function : minimum along the x axis, maximum along the y axis

Now, define the Lagrangian associated to Problem (A.4) as

$$\mathcal{L}(x, \lambda, \mu) := f(x) + \sum_{i=1}^{n_E} \lambda_i g_i(x) + \sum_{j=1}^{n_I} \mu_i h_j(x) \tag{A.7}$$

then Problem (A.4) is equivalent (without any assumption) to

$$\min_{x \in \mathbb{R}^n} \sup_{\lambda \in \mathbb{R}^{n_E}, \mu \in \mathbb{R}_-^{n_I}} \mathcal{L}(x, \lambda, \mu). \tag{A.8}$$

Which means that, if all goes well (here convexity and qualification are required), an optimal solution $x^\sharp$ is part of a saddle point (see Figure A.2 of the Lagrangian $\mathcal{L}$: minimum in the space of x, maximum in λ, μ. Anyway, being minimum or maximum the first order condition is the same. The KKT condition can be recovered by simply differentiating the Lagrangian: $\partial\mathcal{L}/\partial x = 0$ yields the first order condition. Add primal and dual admissibility conditions and complementarity.

APPENDIX B

Partial solution to some exercises

Disclaimer : only elements of the solution are given. More details are expected from students.

B.1 Answers to Chapter 1 exercises

Solution (to Exercise 1.3). *1. NE :* (a, c) *and* (c, a)

2. SO : (b, b)

3. Pareto : (b, a), (b, b), (b, c).

Solution (to Exercise 1.4). *1. (0.75pt) NE :* (b, b) *and* (a, d)

2. (0.5pt) SO : (b, b) *and* (b, c)

3. (0.75pt) Pareto : (b, b), (b, c), (a, d).

Solution (to Exercise 1.5). *1. NE :* (b, b)

2. SO : (a, a), (c, c)

3. Pareto : (a, a), (a, c), (b, b), (b, c) (c, c),

Solution (to Exercise 1.6). *1. (1.5pt) NE : (a,b), (b,a) ; OS and Pareto : (a,b), (b,a), (b,b)*

2. (a) (1pt) The reward obtained by 1 *is* $-5p_1p_2 + p_1(1-p_2) - p_2(1-p_1) = -5p_1p_2 + p_1 - p_2 = p_1(1-5p_2) - p_2$

(b) (1.5pt) For $p_2 > 1/5$*, the optimal* p_1 *is* 0*. For* $p_2 < 1/5$ *the optimal* p_1 *is* 1*. For* $p_2 = 1/5$*, every* $p_1 \in [0, 1]$ *is optimal.*

(c) (2pt) By symmetry we have the same result for p_2*, hence we have*

	0	0.2	1
0	(0, 0)	(-0.2, 0.2)	(-1, 1)
0.2	(0.2, -0.2)	(-0.2, -0.2)	(-1.8, -0.2)
1	(1, -1)	(-0.2, -1.8)	(-5, -5)

(d) (1pt) The Nash Equilibrium is (0.2,0.2), with a social value of -0.4 *which is worse than* 0*. However it is symmetric.*

B.2 Answers to Chapter 2 exercises

Solution (to Exercise 2.3). *1. We can consider that each arc is two opposite directed arcs with positive costs. Hence, Djikstra algorithm applies.*

2. We have (2 pt)

a	b	c	d	e	f
(0)	∞	∞	∞	∞	∞
0	(4)	(2)	∞	∞	∞
0	(3)	2	(10)	(12)	∞
0	3	2	(8)	(12)	∞
0	3	2	8	(10)	(14)
0	3	2	8	10	(13)
0	3	2	8	10	13

Thus the shortest path is $a-c-b-d-e-f$ for a cost of 13.

3. No for Figure 1 graph is non-oriented. Yes for figure 2 : a-c-b-d-e-f.

4. Graph 2 is a subset of graph 1 dedoubled, and the shortest path in graph 1 is admissible in graph 2, hence it is still the shortest path.

Solution (to Exercise 2.4). *1. We have*

a	b	c	d	e	f
(0)	∞	∞	∞	∞	∞
0	(2)	(5)	∞	∞	∞
0	2	(5)	(3)	∞	∞
0	2	(4)	3	(7)	(8)
0	2	4	3	(6)	(8)
0	2	4	3	6	(7)
0	2	4	3	6	7

Hence the shortest path from a to f as cost 7.

2. topological ordering : a-b-d-c-e-f. By dynamic programming we have:

(a) $\lambda(b) = 2$

(b) $\lambda(d) = 2 + 1 = 3$

(c) $\lambda(c) = \min\{3+2, 3+1\} = 4$

(d) $\lambda(e) = \min\{4+2, 3+4\} = 6$

(e) $\lambda(f) = \min\{6+1, 3+5\} = 7$

3. Shortest path : a-b-d-c-e-f.

Solution (to Exercise 2.5). *1. We have*

Hence the shortest path from a to f as cost 7. *The nodes are treated in the following order: a-b-d-c-e-f.*

2. We compute the label λ in the following table Which gives the distance and shortest path in only 4 iterations (a-b-e-f) instead of 6.

a	b	c	d	e	f
(0)	∞	∞	∞	∞	∞
0	(2)	(5)	∞	∞	∞
0	2	(5)	(3)	(6)	∞
0	2	(5)	3	(6)	(13)
0	2	5	3	(6)	(13)
0	2	5	3	6	(7)
0	2	5	3	6	7

a	b	c	d	e	f
(20)	∞	∞	∞	∞	∞
20	(6)	(12)	∞	∞	∞
20	6	(12)	(9)	(6)	∞
20	6	(12)	(9)	6	(7)
20	6	(12)	(9)	6	7

B.3 Answers to Chapter 3 exercises

Solution (to Exercise 3.2). *1. The user equilibrium and social optimum problem are of the following form*

$$\min_{x,f} \quad J(x) \tag{B.1}$$

$$s.t. \quad r_k = \sum_{p \in \mathcal{P}_k} f_p \qquad k \in [\![1, K]\!] \tag{B.2}$$

$$x_a = \sum_{p \ni e} f_p \qquad e \in E \tag{B.3}$$

$$f_p \geqslant 0 \qquad p \in \mathcal{P} \tag{B.4}$$

where $J(x) = W(x) = \sum_{e \in E} L_e(x_e)$ for the user equilibrium, and $J(x) = C(x) = \sum_{e \in E} x_a \ell_e(x_e)$ for the social optimum.

2. As ℓ_a is non-decreasing and $x_e \geqslant 0$ we have $x_e \ell_e(x_e) \geqslant \int_0^{x_e} \ell_e(u)du = L_e(x_e)$. Furthermore, by assumption we have $x\ell_e(x) \leqslant \gamma L_e(x)$. Summing over $e \in E$ gives the result.

3. We have

$$C(x^{UE}) \leqslant \gamma W(x^{UE}) \leqslant \gamma W(x^{SO}) \leqslant \gamma C(x^{SO}).$$

4. If ℓ_a is a polynomial function of order at most p with non-negative coefficient, then we have $x\ell_e(x) \leqslant (p+1)L_e(x)$. Hence, the price of anarchy is lower than $p+1$. For $p = 1$ we have affine function in which case the price of anarchy is at most $4/3 < 2$, so the bound is not sharp.

Solution (to Exercise 3.3). *1. $f^{UE,r}$ is a Wardrop equilibrium, thus by definition the cost of all used path is the same.*

2. As ℓ_e are non decreasing, $\bar{\ell}_e \geqslant \ell_e$, multiplying by $x_e \geqslant 0$ and summing gives the result. Equality is obvious.

3. *$\bar{\ell}_e(x_e) - \ell_e(x_e)$ is null if $x_e \geqslant x_e^{UE,r}$, and equal to $\ell_e(x_e^{UE,r}) - \ell_e(x_e) \leqslant \ell_e(x_e^{UE,r})$ otherwise. Multiplying by x_e we have the result both for $x_e \geqslant x_e^{UE,r}$ and for $x_e \leqslant x_e^{UE,r}$.*

4. $$\begin{aligned}\bar{C}(x^{SO,2r}) - C(x^{SO,2r}) &= \sum_{e\in E} x_e^{SO,2r}(\bar{\ell}_e(x_e^{SO,2r}) - \ell_e(x_e^{SO,2r})) \\ &\leqslant \sum_{e\in E} x_e^{UE,r}\ell_e(x_e^{UE,r}) \\ &= C(x^{UE,r})\end{aligned}$$

5. *Consider $p \in \mathcal{P}^k$. Then $\ell_p(f^{UE,r}) = \lambda_k$. Furthermore,*
$$\bar{\ell}_p(f^{SO,2r}) = \sum_{e\in p} \bar{\ell}_e(x_e(f^{SO,2r})) \geqslant \sum_{e\in p} \ell_e(x_e^{UE,r}) = \lambda_k$$
where the inequality comes from monotonicity of ℓ_e, and definition of $\bar{\ell}_e$.

6. $$C(x) = \sum_{f\in\mathcal{P}} f_p\ell_p(f) \qquad \text{and} \qquad \bar{C}(x) = \sum_{f\in\mathcal{P}} f_p\bar{\ell}_p(f).$$

7. $$\begin{aligned}\bar{C}(f^{SO,2r}) &= \sum_{k=1}^{K}\sum_{p\in\mathcal{P}_k} f_p^{SO,2r}\bar{\ell}_p(f^{SO,2r}) \\ &\geqslant \sum_{k=1}^{K}\lambda_k \sum_{p\in\mathcal{P}_k} f_p^{SO,2r} \\ &= \sum_{k=1}^{K} 2\lambda_k r^k \\ &= 2C(f^{UE,r})\end{aligned}$$

8. *Combining previous results we have*
$$2C(f^{UE,r}) \leqslant \bar{C}(x^{SO,2r}) \leqslant C(x^{UE,r}) + C(x^{SO,2r}),$$
which give the result, that can be interpreted as "optimizing flux cannot allow more than twice the inflows rates without increasing global cost".

Solution (to Exercise 3.4)**.** *1.* 4/3.

2. *$+\infty$ using $r = 1$ and x^p, $p \to \infty$.*

3. - *User equilibrium is given by $x_1 = r$, $x_2 = 0$, with total cost $rc(r)$.*
   - *Social optimum is given by $\min_{0\leqslant x\leqslant r} xc(x) + (r-x)c(r)$*
   - *Price of anarchy $\max_{0\leqslant x\leqslant r} rc(r)/(xc(x) + (r-x)c(r))$.*

4. *Thus the price of anarchy of $\mathcal{C}$ is greater than*
$$\max_{c\in\mathcal{C}, r>0}\ \max_{0\leqslant x\leqslant r} \frac{rc(r)}{xc(x) + (r-x)c(r)}.$$

5. *Choosing $r = 1$ and $c(x) = x$ yield a lower bound of $4/3$ which is exact.*

Solution (to Exercise 3.5 Part I). *1. (1pt) There are 4 possible paths : $o-d$, $o-a-d$, $o-a-b-d$, $o-b-d$. For $f^{\varepsilon,1}$ their cost is 2, $1+\varepsilon+1-\varepsilon = 2$, $2+2\varepsilon$ and 2. Thus $f^{\varepsilon,1}$ is an ε-Nash Equilibrium.*

2. *(1pt) We put δ on $o-d$, $1-\delta/2$ on $o-a-d$ and $1-\delta/2$ on $o-b-d$.*

3. *(2.5pts) The global cost is*

$$2f_1 + (f_2+f_3)g_\delta(f_2+f_3) + (1-\varepsilon)f_2 + (1-\varepsilon)f_3 + g_\delta(f_3)$$

we can improve the cost of any admissible flow by shifting from path 3 to path 4 (as g_δ is increasing), thus an optimal flow have $f_3 = 0$. By monotonicity, an optimal solution have $f_2 = f_4$, and as $f_1+f_2+f_3+f_4 = 2$ we reduce the problem to

$$\min_{f_4\in[0,1]} 2(2-2f_4) + 2f_4 g_\delta(f_4)$$

which can be further reduced to

$$\min_{f_4\in[1-\delta,1]} 2(2-2f_4) + 2f_4 g_\delta(f_4)$$

Solution (to Exercise 3.5 Part II). *1. (0.5pts) An ε-Nash equilibrium is a flux such that each user can win at most ε by changing trajectory with fixed cost. If $\varepsilon = 0$ we recover the Wardrop equilibrium.*

2. *(1pt) $f^{\varepsilon,r}$ is a ε-Nash equilibrium, thus for every $p \in \mathcal{P}_k$ we have $f_p^{\varepsilon,r}\ell_p(f^{\varepsilon,r}) \leqslant (1+\varepsilon)f_p^{\varepsilon,r}\lambda_k(x(f^{\varepsilon,r}))$, and summing over all $p \in \mathcal{P}$ yields the result.*

3. *(1pt) $\bar{\ell}_e(x_e) - \ell_e(x_e)$ is null if $x_e \geqslant x_e^{\varepsilon,r}$, and equal to $\ell_e(x_e^{\varepsilon,r}) - \ell_e(x_e) \leqslant \ell_e(x_e^{\varepsilon,r})$ otherwise. Multiplying by x_e we have the result both for $x_e \geqslant x_e^{\varepsilon,r}$ and for $x_e \leqslant x_e^{\varepsilon,r}$.*

4. *(1pt)*

$$\begin{aligned}\bar{C}(x^{SO,2r}) - C(x^{SO,2r}) &= \sum_{e\in E} x_e^{SO,2r}(\bar{\ell}_e(x_e^{SO,2r}) - \ell_e(x_e^{SO,2r})) \\ &\leqslant \sum_{e\in E} x_e^{\varepsilon,r}\ell_e(x_e^{\varepsilon,r}) \\ &= C(x^{\varepsilon,r})\end{aligned}$$

5. *(1pt) We have $\bar{\ell}_p(0) \geqslant \lambda_k(x^{\varepsilon,r})$, and as $\bar{\ell}_p$ is non-decreasing we get $\bar{\ell}_p(f^{SO,2r}) \geqslant \lambda_k(x^{\varepsilon,r})$.*

6. *(1.5pts)*

$$\begin{aligned}\sum_{p\in\mathcal{P}} \bar{\ell}_p(f^{SO,2r})f_p^{SO,2r} &\geqslant \sum_k \sum_{p\in\mathcal{P}_k} \lambda_k(x^{\varepsilon,r})f_p^{SO,2r} \quad \textit{by previous question} \\ &= \sum_k \lambda_k(x^{\varepsilon,r})r_k \\ &\geqslant \frac{2}{1+\varepsilon}C(f^{\varepsilon,r}) \quad \textit{by question ??}\end{aligned}$$

7. *(1.5pts) We have*

$$\begin{aligned} C(x^{SO,2r}) &\geqslant \sum_{p\in\mathcal{P}} \bar{\ell}_p(f^{SO,2r}) f_p^{SO,2r} - C(x^{\varepsilon,r}) \\ &\geqslant \frac{2}{1+\varepsilon} C(x^{\varepsilon,r}) - C(x^{\varepsilon,r}) \\ &= \frac{1-\varepsilon}{1+\varepsilon} C(x^{\varepsilon,r}) \end{aligned}$$

8. *(1pt) In the example of part I we have $C(x^{\varepsilon,1}) = 2+2\varepsilon$, and an admissible flow for the double rate with cost $2\delta + (1-\varepsilon)(1-\delta)$. Letting δ goes to zero yields the result.*

9. *(1.5pts) Consider $p \in \mathcal{P}^k$. Then $\ell_p(f^{UE,r}) = c_k$. Furthermore,*

$$\bar{\ell}_p(f^{SO,2r}) = \sum_{e\in E} \bar{\ell}_e(x_e(f^{SO,2r})) \geqslant \sum_{e\in E} \ell_e(x_e^{UE,r}) = c_k$$

where the inequality comes from monotonicity of ℓ_e, and definition of $\bar{\ell}_e$.

10. *(0.5pts)*

$$C(x) = \sum_{f\in\mathcal{P}} f_p \ell_p(f) \qquad \text{and} \qquad \bar{C}(x) = \sum_{f\in\mathcal{P}} f_p \bar{\ell}_p(f).$$

11. *(2pts)*

$$\begin{aligned} \bar{C}(f^{SO,2r}) &= \sum_{k=1}^{K} \sum_{p\in\mathcal{P}_k} f_p^{SO,2r} \bar{\ell}_p(f^{SO,2r}) \\ &\geqslant \sum_{k=1}^{K} c_k \sum_{p\in\mathcal{P}_k} f_p^{SO,2r} \\ &= \sum_{k=1}^{K} 2c_k r^k \\ &= 2C(f^{UE,r}) \end{aligned}$$

12. *(1pts) Combining previous results we have*

$$2C(f^{UE,r}) \leqslant \bar{C}(x^{SO,2r}) \leqslant C(x^{UE,r}) + C(x^{SO,2r}),$$

which give the result, that can be interpreted as "optimizing flux cannot allow more than twice the inflows rates without increasing global cost".

B.4 Answers to Chapter 4 exercises

Solution (to Exercise 4.1)**.** *1. Objective function is (strongly) convex, constraints are polyhedral.*

2. $\nabla J(x) = (2(x_1 - 2), 2(x_2 + 1))^\top$

3. *The linear problem is given by*

$$\min_{y\in[-1,1]^2} -4y_1 + 2y_2$$

with optimal solution $(1,-1)$.

4. *The linear search problem is*

$$\min_{t\in[0,1]} (t-2)^2 + (-t+1)^2.$$

(1pt) The unconstrained problem has an optimal solution of $t = 3/2$*, hence the optimal* t *is* $t = 1$*, and we have (0.5pts)* $x^{(1)} = (1,-1)$.

5. *The new linear problem is given by*

$$\min_{y\in[-1,1]^2} -2y_1.$$

Solution (to Exercise 4.2)**.** 1. $(0,0)$*,* $(0,1)$ *and* $(2,0)$

2. $\nabla f(x) = \begin{pmatrix} 16x_1^3 - 2 \\ 2x_2 - 1 \end{pmatrix}$

3. $\nabla^2 f(x) = \begin{pmatrix} 42x_1^2 & 0 \\ 0 & 2 \end{pmatrix} \geq 0$ *hence* f *is convex, and* X *is polyhedral and bounded.*

4. $\min_{y\in X} -2y_1 - 1y_2$*.* $-2*0-1*1 > -2*2-1*0$*, hence the optimal solution is* $y^0 = (2,0)$*. And the optimal direction is* $d^0 = y^0 - x^0 = (2,0)$.

5. *(1pts)* $\min_{t\in[0,1]} 2^6t^4 - 2^2t$*. By derivating this objective function we obtain that the optimal step is* $t^0 = 1/4$*, and* $x^1 = x^0 + t^0d^0 = (1/2,0)$.

6. *The upper bound is* $f(x^1) = 5/4$*. The lower bound is* $(\nabla(f)(x^0))^T(y^0 - x^0) + f(x^0) = -2$.

7. *To find the direction* d^1 *we need to solve* $\min_{y\in X} \nabla(f)(x^1)^T y$*. The solution being an extreme point of* X *that is neither* $(0,0)$ *nor* $(2,0)$*, we have* $y^1 = (1,0)$*. Thus* $d^1 = (1,-1/2)$.

8. *Finding the optimal step require to solve*

$$\min_{t\in[0,1]} 4(1/2-t)^4 - 2(1/2-t) + t^2 - t$$

9. *The lower bound is given by*

$$(\nabla(f)(x^1))^T(y^1 - x^1) + f(x^1) = 0*(-1/2) - 1*1 + 5/4 = 1/4$$