**Optimisation** **Part IB - Easter 2019**

# Lecture 1: Introduction and convexity

Lecturer: Quentin Berthet

**Introduction.** The objective of this course is the study of optimization problems

$$\min_{x \in \mathcal{X}} f(x)\,,$$

where $f$ is a real-valued function in a variable $x$ over a set $\mathcal{X}$. These problems arise in many areas of mathematics and the natural sciences, and are at the center of recent technological advances in information science. We will describe methods to solve some of these problems, exactly or approximately, and study the properties of the solutions.

## 1. Convexity.

1.1. *Definitions.* A convenient property in optimization problems is that of *convexity*, for sets and functions.

Definition 1.1 (Convexity). A subset $C$ of $\mathbf{R}^n$ is *convex* if for all $x, y \in C$, and all $\lambda \in [0,1]$, we have

$$(1-\lambda)x + \lambda y \in C\,.$$

Remark.

- This definition can be extended to more general spaces than $\mathbf{R}^n$, but we will only consider this case here.

- Intuitively, this means that every segment between two elements of $C$ is also contained in $C$.

- An important property of convex sets is that they are stable by intersection. In the second part of this course, we will particularly focus on sets of the form $Ax \leq b$. More examples and properties are in the Examples sheet.

Definition 1.2 (Convex functions). Let $\mathcal{X}$ be a convex subset of $\mathbf{R}^n$. A function $f : \mathcal{X} \to \mathbf{R}$ is *convex* if for all $x, y \in \mathcal{X}$ and all $\lambda \in [0,1]$, we have

$$f\big((1-\lambda)x + \lambda y\big) \leq (1-\lambda)f(x) + \lambda f(y)\,.$$

Remark.

- This definition means that any segment (or "cord") between two points $(x, f(x))$ and $(y, f(y))$ is above the graph of the function.

- In practice, this might not be always easy to verify for a given function, a more "analytic" point of view is given at the end of this lecture.

- More precisely, convex functions are those whose epigraph is convex. For a convex function, level sets are convex, however the converse is not true. These two notions are made formal in Examples sheets.

- Optimization problems involving convex functions are "well-behaved" in some sense, and this is due mainly to the fact that local properties give information about the global behaviour of the function. One manifestation of this phenomenon is the following property.

PROPOSITION 1.1. *For any differentiable function $f$ on $\mathcal{X}$, $f$ is convex if and only if, for all $x, y \in \mathcal{X}$*

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x)\,.$$

PROOF. In one direction, we give the proof in one dimension, i.e. that

$$f(y) \geq f(x) + f'(x)(y - x)$$

For $x < y$, and for all $t \in (0,1)$, we introduce $x_t = x + t(y-x) = (1-t)x + ty$. By convexity of $f$, we have

$$f(x_t) \leq (1-t)f(x) + tf(y)\,.$$

After rearranging,

$$f(x_t) - f(x) \leq t(f(y) - f(x))\,.$$

Dividing on both sides by $x_t - x = t(y-x) > 0$, we have

$$\frac{f(x_t) - f(x)}{x_t - x} \leq \frac{f(y) - f(x)}{y - x}\,.$$

This means that slopes between points of the graph are "increasing" in the destination. Letting $t$ go to 0 on the left hand side yields

$$f'(x) \leq \frac{f(y) - f(x)}{y - x}$$

Multiplying by $y - x > 0$ on both sides gives the desired inequality. In the case of $y < x$, the inequalities are reversed between the two uses of "$y - x > 0$" in this proof. The proof in any dimension is in the Examples sheet.

For the other direction, define again $x_\lambda = (1-\lambda)x + \lambda y$, we have

$$f(x) \geq f(x_\lambda) + \nabla f(x_\lambda)^\top (x - x_\lambda)$$
$$f(y) \geq f(x_\lambda) + \nabla f(x_\lambda)^\top (y - x_\lambda)$$

Adding the first line multiplied by $(1-\lambda)$ and the second multiplied by $\lambda$ yields

$$(1-\lambda)f(x) + \lambda f(y) \geq f((1-\lambda)x + \lambda y)$$

so $f$ is convex. $\square$

Remark. One way to interpret this is that the function is above the best local linear approximation of $f$ for any $x_0 \in \mathcal{X}$,

$$\ell_{x_0}(x) = f(x_0) + \nabla f(x_0)^\top (x - x_0)\,.$$

It gives a lot of information about the values of $f$ at all points in $\mathcal{X}$, given only information at one point. The following shows the only statement we can make for a general function $g$.

Proposition 1.2. *For $g$ that is differentiable on $\mathcal{X} = \mathbf{R}^n$, if $x$ is a local minimum of $g$, then $\nabla g(x) = 0$.*

Remark. The converse is not true in all generality. However, the situation is much easier for convex functions, where local minima are global minima. This is summarized in the following.

Proposition 1.3. *If $f$ is a differentiable convex function on $\mathcal{X} = \mathbf{R}^n$, the following are equivalent*

- *$x$ is a global minimum of $f$.*

- *$\nabla f(x) = 0$.*

Proof. The first direction is direct: if $x$ is a global minimum, it is a local minimum, so $\nabla f(x) = 0$. The second direction is almost as direct, by Proposition 1.1: if $\nabla f(x) = 0$, then for all $y \in \mathcal{X}$, $f(y) \geq f(x)$. $\square$

This shows that unconstrained convex optimization problem are, in theory, very simple. It can be shown that if $f(x) \to \infty$ when $\|x\| \to \infty$, such an $x$ exists. Often in cases considered in practice, there is a unique solution to the equation

$$\nabla f(x) = 0\,,$$

which is the unique global minimum of the convex function $f$. This leaves one question: how does one check easily that a function is convex?

Proposition 1.4. *A twice differentiable function $f$ is convex if and only if it has a semidefinite positive Hessian on the interior of $\mathcal{X}$.*

The proof relies on Taylor's theorem with Lagrange remainder, for any $x, y$ in the interior of $\mathcal{X}$

$$f(y) = f(x) + \nabla f(x)^\top (y - x) + \frac{1}{2}(y - x)\nabla^2 f(\xi)(y - x)\,.$$

**Optimisation** **Part IB - Easter 2019**

# Lecture 2: Gradient descent

Lecturer: Quentin Berthet

1.2. *Algorithms for unconstrained convex optimization.* In the previous lecture, we saw that unconstrained convex optimization problems of the form

$$\min_{x \in \mathbf{R}^n} f(x)\,,$$

when $f$ is a differentiable convex function, are conceptually easy. They are equivalent to solving a first order equation, i.e. to find a stationary point satisfying

$$\nabla f(x) = 0\,.$$

It might sometimes be possible to do so explicitly, but not always. In all cases, there exists a very simple algorithm to tackle these types of problem, known as *gradient descent*

Definition 1.3. The gradient descent algorithm with step $\eta > 0$, for a differentiable function $f$, is an update rule for $x_t \in \mathbf{R}^n$, with $x_0 \in \mathbf{R}^n$ and for all $t \geq 0$

$$x_{t+1} = x_t - \eta \nabla f(x_t)\,.$$

Remark. This algorithm relies on the following intuition: locally around $x$, the line of steepest descent is along $-\nabla f(x)$. For a one dimensional convex function, since $f'(x)$ is increasing in $x$, and equal to 0 at the minimum, the derivative is a good indicator of the direction and distance in which we should step. The sequence of points can be also thought of as the discrete version of the continuous *gradient flow* $\dot{x}(t) = -\nabla f(x(t))$.

Proposition 1.5. *Let $f$ be a differentiable convex function with global minimum $x*$. If $\psi : x \to x - \eta\nabla f(x)$ satisfies $\|\psi(x) - \psi(y)\| \leq \lambda \|x - y\|$ for $\lambda \in [0, 1)$, then*

$$\|x_t - x^*\| \leq \lambda^t \|x_0 - x^*\|\,.$$

Proof. We have

$$\|x_t - x^*\| = \|\psi(x_{t-1}) - \psi(x^*)\| \leq \lambda \|x_{t-1} - x^*\| \leq \lambda^t \|x_0 - x^*\|\,.$$

□

Remark. The main idea is therefore to show that $\psi$ is a contraction. Upon first inspection, we have

$$\|\psi(x) - \psi(y)\|^2 = \|x-y\|^2 + \eta^2\|\nabla f(x) - \nabla f(y)\|^2 - 2\eta(\nabla f(x) - \nabla f(y))^\top(x-y)\,.$$

Since $f$ is convex, writing the first order condition twice, reversing the roles of $x$ and $y$, yields

$$(\nabla f(x) - \nabla f(y))^\top(x-y) \geq 0\,.$$

This is a good indication that the last term can "subtract" from $\|x-y\|^2$, but it seems clear that we should consider assumptions linking the behavior of $f(x)$, $\nabla f(x)$, and quadratic terms.

Definition 1.4. A differentiable convex function is $\beta$-smooth, if for some $\beta \geq 0$, it holds that

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta\|x-y\|\,.$$

Remark. Writing out Taylor's theorem, this property is equivalent to the following upper bound on the function (see Examples sheet)

$$f(y) \leq f(x) + \nabla f(x)^\top(y-x) + \frac{\beta}{2}\|y-x\|^2\,.$$

- Not all convex function are $\beta$-smooth: as an example, $f(x) = x^4$ is not.
- If the function is twice differentiable, this is equivalent to the Hessian having eigenvalues upper bounded by $\beta$.

Definition 1.5. A convex function is $\alpha$-strongly convex, if for some $\alpha \geq 0$, it holds that

$$f(y) \geq f(x) + \nabla f(x)^\top(y-x) + \frac{\alpha}{2}\|y-x\|^2\,.$$

Remark. In comparison with the smoothness assumption, this is a lower bound on the function, with a different second order term. It can be thought of as a quantitative version of strict convexity.

- Not all function are strongly convex: a linear function, or $f(x) = e^{-x}$ are not.
- If the function is twice differentiable, this is equivalent to the Hessian having eigenvalues lower bounded by $\alpha$.
- As a consequence, if $f$ is $\alpha$-strongly convex and $\beta$-smooth, then $\beta \geq \alpha$. In this case, $g(x) = f(x) - \alpha\|x\|^2/2$ is convex and $(\beta-\alpha)$-smooth (see Examples sheet).

Lemma 1.1. *Let $f$ be a convex, $\beta$-smooth function. We have that*

$$f(y) \geq f(x) + \nabla f(x)^\top(y-x) + \frac{1}{2\beta}\|\nabla f(x) - \nabla f(y)\|^2\,.$$

The proof of this lemma is also done in Examples sheets. This result shows that even for smooth convex functions, a stronger lower bound than the linear approximation holds. However, there is a catch: this lower bound depends on the variations of the function itself. Indeed, for a linear form, the linear approximation is tight, and no better lower bound can be obtained.

Remark. As a direct corollary, switching the role of $x$ and $y$, we have that

$$(\nabla f(x) - \nabla f(y))^\top (x-y) \geq \frac{1}{\beta}\|\nabla f(x) - \nabla f(y)\|^2\,.$$

This is closer to what we need in order to show that $\psi$ is a contraction, but has no term in $\|x-y\|^2$. We therefore apply the inequality above to $g(x) = f(x) - \alpha\|x\|^2/2$, which is a convex, $(\beta-\alpha)$-smooth function to obtain

$$(\nabla f(x) - \nabla f(y))^\top (x-y) \geq \frac{\alpha\beta}{\alpha+\beta}\|x-y\|^2 + \frac{1}{\beta+\alpha}\|\nabla f(x) - \nabla f(y)\|^2\,.$$

Theorem 1.1. *Let $f$ be a $\beta$-smooth, $\alpha$-strongly convex function. The iterates of the gradient descent algorithm with $\eta = 2/(\alpha+\beta)$ satisfy*

$$\|x_t - x^*\| \leq \Big(\frac{\kappa-1}{\kappa+1}\Big)^t \|x_0 - x^*\|$$
$$f(x_t) - f(x^*) \leq \frac{\beta}{2}\Big(\frac{\kappa-1}{\kappa+1}\Big)^{2t}\|x_0 - x^*\|^2\,,$$

*where $\kappa = \beta/\alpha > 1$ is the conditioning number of $f$.*

Proof. We have, using notations above

$$\begin{aligned}\|\psi(x) - \psi(y)\|^2 &= \|x-y\|^2 + \eta^2\|\nabla f(x) - \nabla f(y)\|^2 - 2\eta(\nabla f(x) - \nabla f(y))^\top(x-y)\\ &\leq \Big(1 - 2\frac{\eta\alpha\beta}{\alpha+\beta}\Big)\|x-y\|^2 + \Big(\eta^2 - 2\frac{\eta}{\alpha+\beta}\Big)\|\nabla f(x) - \nabla f(y)\|^2\\ &= \Big(\frac{\kappa-1}{\kappa+1}\Big)^2\|x-y\|^2\,.\end{aligned}$$

The first inequality follows from Proposition 1.5. The second follows from $\beta$-smoothness.

$$f(x_t) - f(x^*) \leq \underbrace{\nabla f(x^*)^\top(x_t - x)}_{\nabla f(x^*)=0} + \frac{\beta}{2}\|x_t - x\|^2\,.$$

□

**Optimisation** **Part IB - Easter 2019**

# Lecture 3: Convex optimization

Lecturer: Quentin Berthet

In the previous lecture, we saw that for a differentiable, convex function with $\beta$-smoothness and $\alpha$-strong convexity (they can be thought of as having Hessian upper and lower bounded), we have

$$\|x_t - x^*\| \leq \Big(\frac{\kappa-1}{\kappa+1}\Big)^t \|x_0 - x^*\|$$
$$f(x_t) - f(x^*) \leq \frac{\beta}{2}\Big(\frac{\kappa-1}{\kappa+1}\Big)^{2t} \|x_0 - x^*\|^2\,,$$

where $\kappa = \beta/\alpha \geq 1$ is the conditioning number of the function (and of the Hessian).

Remark.

- Under less restrictive assumptions, other rates of convergence can be shown for a convex function $f$.

- This method can also be used to minimize a function with noise on the gradient. This is known as stochastic optimization, and is particularly important in statistics and machine learning.

- The assumptions (smoothness, strong convexity) and quantities (conditioning) appeared as inspired by the proof, but they actually describe accurately the performance of this algorithm.

Example 1.1. In $\mathbf{R}^2$, for some $x^* \in \mathbf{R}^2$ let $f$ be the convex quadratic function defined by

$$f(x) = f(x^*) + \frac{1}{2}(x-x^*)^\top Q(x-x^*)\,, \quad \text{with} \quad Q = \begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix}\,.$$

By definitions, for $\beta \geq \alpha > 0$, $Q$ is positive definite and $f$ is $\beta$-smooth, $\alpha$-strongly convex, with a unique global minimum at $x^*$. The iterates of gradient descent satisfy

$$x_{t+1} - x^* = x_t - \eta\nabla f(x_t) - x^* = x_t - \eta Q(x_t - x^*) - x^* = (I - \eta Q)(x_t - x^*)\,.$$

Decomposing in each coordinate, this yields (recalling that $\eta = 2/(\alpha+\beta)$)

$$\begin{pmatrix} x_{t+1}^{(1)} - x_*^{(1)} \\ x_{t+1}^{(2)} - x_*^{(2)} \end{pmatrix} = \begin{pmatrix} 1-\eta\alpha & 0 \\ 0 & 1-\eta\beta \end{pmatrix} \begin{pmatrix} x_t^{(1)} - x_*^{(1)} \\ x_t^{(2)} - x_*^{(2)} \end{pmatrix} = \begin{pmatrix} \frac{\kappa-1}{\kappa+1} & 0 \\ 0 & -\frac{\kappa-1}{\kappa+1} \end{pmatrix} \begin{pmatrix} x_t^{(1)} - x_*^{(1)} \\ x_t^{(2)} - x_*^{(2)} \end{pmatrix}\,.$$

One can explicitly draw the behaviour of $x_t$ to understand the behaviour of the algorithm. More precisely, we have exactly

$$\|x_t - x^*\| = \Big(\frac{\kappa - 1}{\kappa + 1}\Big)^t \|x_0 - x^*\| \,.$$

Remark. The example above illustrates well the impact of conditioning on the performance of gradient descent. This can be understood in the following way: after $t$ iterations, given $f(x_t)$ and $\nabla f(x_t)$, the update $x_{t+1} = x_t - \eta \nabla f(x_t)$ is the minimizer of the local quadratic approximation

$$q_{x_t}(x) = f(x_t) + \nabla f(x_t)^\top (x - x_t) + \frac{\eta^{-1}}{2}\|x - x_t\|^2 \,.$$

The better this approximation is, i.e. the more the second-order term is close to a quadratic with the same eigenvalue (isotropic), the closer $x_{t+1}$ will be to the true minimum of $f$. In particular, if $f$ is a quadratic function with $\alpha = \beta$, we have in one step $x_1 = x^*$. Note that $\eta$ corresponds in the approximation above to the inverse of the eigenvalue of the quadratic. This justifies further why in the algorithm we chose $\eta^{-1} = (\alpha + \beta)/2$, the mean of the two extreme eigenvalues of the Hessian.

If the Hessian has very different eigenvalues, it is possible to use this information to minimize another approximation of this function of the form

$$q_H(x) = f(x_t) + \nabla f(x_t)^\top (x - x_t) + \frac{1}{2}(x - x_t)^\top \nabla^2 f(x_t)(x - x_t) \,.$$

Definition 1.6. Newton's method for a twice differentiable function $f$ is an update rule for $x_t \in \mathbf{R}^n$, with $x_0 \in \mathbf{R}^n$ and all $t \geq 0$

$$x_{t+1} = x_t - [\nabla^2 f(x_t)]^{-1} \nabla f(x_t) \,.$$

Remark.

- The first intuition for this algorithm is given above: the function is approximated by a second-order approximation. There are no conditioning issues anymore: if the function $f$ is indeed any quadratic (no matter what the eigenvalue of the Hessian are), then $x_1 = x^*$.

- For general twice differentiable functions, it can converge much faster than gradient descent. Under more assumptions on the function and on the starting point, it is possible to show that
$$\|x_{t+1} - x^*\| \leq C\|x_t - x^*\|^2 \,.$$
however there are some caveats. The function needs to be twice differentiable, we need to know or compute the Hessian, we also need to invert it.

- In one dimension, this simply corresponds to

$$x_{t+1} = x_t - \frac{f'(x_t)}{f''(x_t)}\,.$$

Taking $g(x) = f'(x)$, minimizing $f$ convex is equivalent to finding a root of nondecreasing $g$ and we have

$$x_{t+1} = x_t - \frac{g(x_t)}{g'(x_t)}\,.$$

This is also known as Newton's method, and has a nice geometric interpretation.

1.3. *Barrier function method for constrained optimization.* Newton's method can be particularly useful to solve constrained optimization problems of the form

$$\begin{array}{rl} \min & f(x) \\ s.t. & h_i(x) \leq 0 \quad 1 \leq i \leq m\,, \end{array}$$

for convex, differentiable functions $f$ and $h_i$. Indeed, taking

$$I_-(u) = 0 \text{ if } u \leq 0\ ,\ \infty \text{ otherwise}$$

this problem is equivalent to

$$\min_{x \in \mathbf{R}^n} f(x) + \sum_{i=1}^{m} I_-(h_i(x))\,.$$

We can approximate $I_-(u)$ by $I_{-,t}(u) = -\frac{1}{t}\log(-u)$, for any $t > 0$, and consider the approximate optimization problem

$$\min_{x \in \mathbf{R}^n} f(x) + \sum_{i=1}^{m} -(1/t)\log(-h_i(x))\,.$$

Taking $\phi(x) = \sum_{i=1}^{m} -(1/t)\log(-h_i(x))$ (known as the barrier function), this is equivalent to solving

$$\min_{x \in \mathbf{R}^n} tf(x) + \phi(x)\,.$$

The objective is convex, since $I_{-,t}$ is convex and increasing. For any $t \geq 0$, it is possible to approximately solve this problem towards its solution $x^*(t)$. As expected, when $t$ is taken very large this approximately solves the original problem.

**Optimisation** **Part IB - Easter 2019**

# Lecture 4: Lagrangian duality

Lecturer: Quentin Berthet

**2. Lagrange duality.** Many optimization problems over $\mathcal{X}$ with additional constraints $h(x) = b$, where $h : \mathcal{X} \to \mathbf{R}^m$ and $b \in \mathbf{R}^m$.

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & h(x) = b\,, \\ & x \in \mathcal{X}\,. \end{aligned} \tag{1}$$

This captures many problems, main idea is that it is "simple" to optimize certain types of functions over $\mathcal{X}$, and to see how this can be used to optimize over the more complicated set $\mathcal{X}(b) = \{x \in \mathcal{X} \,:\, h(x) = b\}$. To do so, we introduce the following notion.

Definition 2.1. The *Lagrangian* associated with the optimization problem (1) is defined as

$$L(x, \lambda) = f(x) - \lambda^\top (h(x) - b)\,.$$

2.1. *Lagrange sufficiency.* If optimizing over $\mathcal{X}$ is a simple problem, we can optimize $L(\cdot, \lambda)$ over $\mathcal{X}$. This is related to the original optimization problem, as shown in the following.

Theorem 2.1 (Lagrange sufficiency theorem). *Let $\lambda \in \mathbf{R}^m$, and $x^*(\lambda) \in \mathcal{X}$ such that*

$$L(x^*(\lambda), \lambda) = \inf_{x \in \mathcal{X}} L(x, \lambda)\,, \quad \text{and} \quad h(x^*(\lambda)) = b\,.$$

*Then, $x^*(\lambda)$ is an optimal solution of problem (1) .*

Proof. We have the following

$$\begin{aligned} \inf_{x \in \mathcal{X}(b)} f(x) &= \inf_{x \in \mathcal{X}(b)} \left\{ f(x) - \lambda^\top (h(x) - b) \right\} \\ &\geq \inf_{x \in \mathcal{X}} \left\{ f(x) - \lambda^\top (h(x) - b) \right\} \\ &= \inf_{x \in \mathcal{X}} L(x, \lambda) \\ &= L(x^*(\lambda), \lambda) \\ &= f(x^*(\lambda))\,. \end{aligned}$$

□

For such a $\lambda$, the function $L(\cdot, \lambda)$ is bounded from below, which might not always be the case. To formally discuss this result and a strategy that uses it, we introduce the following notation

Definition 2.2. We denote by $\mathcal{Y}$ the set of dual feasible vectors, defined as

$$\mathcal{Y} = \{\lambda \in \mathbf{R}^m \,:\, \inf_x L(x,\lambda) > -\infty\}$$

Remark. We can therefore follow this strategy to solve these types of optimization problems:

- *i)* For any $\lambda \in \mathbf{R}^m$, find out if $\lambda \in \mathcal{Y}$.
- *ii)* For any $\lambda \in \mathcal{Y}$, find a solution $x^*(\lambda)$ to minimizing $L(\cdot, \lambda)$ over $\mathcal{X}$.
- *iii)* Find $\lambda$ such that $h(x^*(\lambda)) = b$.

We have not given any guarantee that these steps are easy, or even possible. However, by Theorem 2.1, we know that if the steps above succeed, we have solved the optimization problem. It is not a necessary, but a sufficient condition for optimality.

Example 2.1. We consider the following problem, with $m = 2$, over $\mathcal{X} = \mathbf{R}^3$.

$$\begin{aligned} \min \quad & x_1 - x_2 - 2x_3 \\ s.t. \quad & x_1 + x_2 + x_3 = 5\,, \\ & x_1^2 + x_2^2 = 4\,. \end{aligned}$$

We write out the Lagrangian associated with this problem

$$\begin{aligned} L(x,\lambda) &= x_1 - x_2 - 2x_3 - \lambda_1(x_1 + x_2 + x_3 - 5) - \lambda_2(x_1^2 + x_2^2 - 4) \\ &= (-\lambda_2 x_1^2 + (1-\lambda_1)x_1) + (-\lambda_2 x_2^2 + (-1-\lambda_1)x_2) + (-2-\lambda_1)x_3 + 5\lambda_1 + 4\lambda_2\,. \end{aligned}$$

Unless $\lambda_1 = -2$, letting $x_3$ go to $-\infty$ or $+\infty$, $L(x,\lambda)$ goes to $-\infty$. Similarly, unless $\lambda_2 < 0$, $L(x,\lambda)$ is not lower bounded. Under these conditions, we have that

$$\nabla_x^2 L(x,\lambda) = \begin{pmatrix} -2\lambda_2 & 0 & 0 \\ 0 & -2\lambda_2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \succeq 0\,,$$

so $L(\cdot,\lambda)$ is a convex function of $x$. To find its global minimum, we solve $\nabla_x L(x,\lambda) = 0$

$$\begin{aligned} \frac{\partial L}{\partial x_1} &= 1 - \lambda_1 - 2\lambda_2 x_1 = 3 - 2\lambda_2 x_1 \\ \frac{\partial L}{\partial x_2} &= -1 - \lambda_1 - 2\lambda_2 x_2 = 1 - 2\lambda_2 x_2 \\ \frac{\partial L}{\partial x_3} &= -2 - \lambda_1 = 0\,. \end{aligned}$$

We obtain that any $x^*(\lambda)$ with $x_1^*(\lambda) = 3/(2\lambda_2)$ and $x_2^*(\lambda) = 1/(2\lambda_2)$ is solution. Solving so that $(x_1^*(\lambda))^2 + (x_2^*(\lambda))^2 = 4$ gives $\lambda_2 = -\sqrt{5/8}$ and by Theorem 2.1, $x = (-3\sqrt{2/5}, -\sqrt{2/5}, 5 + 4\sqrt{2/5})$ is optimal.

Proposition 2.1. *The inequality-constrained optimization problem*

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & h(x) \leq b \,, \\ & x \in \mathcal{X} \,. \end{aligned}$$

*can be reduced to the equivalent problem*

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & h(x) + z = b \,, \\ & x \in \mathcal{X} \,,\ z \geq 0 \,. \end{aligned}$$

In this case, the optimization problem has variable $(x, z)$ in $\mathcal{X} \times \mathbf{R}^m_{\geq 0}$, with equality constraint. The associated Lagrangian is written out as

$$L(x, z, \lambda) = f(x) - \lambda^\top (h(x) + z - b) = f(x) - \sum_{i=1}^m \lambda_i h_i(x) - \sum_{i=1}^m \lambda_i z_i + \sum_{i=1}^m \lambda_i b_i \,.$$

Theorem 2.2. *For all $\lambda \in \mathcal{Y}$, we have $\lambda_i \leq 0$ and $\lambda_i z_i^*(\lambda) = 0$ for all $i \in [m]$, i.e.*

$$\begin{aligned} \lambda_i \neq 0 \quad & \textit{implies} \quad z_i^*(\lambda) = 0 \\ z_i^*(\lambda) \neq 0 \quad & \textit{implies} \quad \lambda_i = 0 \,. \end{aligned}$$

*In particular, for any $\lambda^*$ such that $h(x^*(\lambda^*)) + z^*(\lambda^*) = b$, we have*

$$\begin{aligned} \lambda_i^* < 0 \quad & \textit{implies} \quad h_i(x_i^*(\lambda)) = b_i \\ h_i(x_i^*(\lambda)) < b_i \quad & \textit{implies} \quad \lambda_i = 0 \,. \end{aligned}$$

Proof. From the form taken by the Lagrangian, if $\lambda_i > 0$, letting $z_i$ go to $+\infty$ lets $L(x, z, \lambda)$ go to $-\infty$, so $\lambda_i \leq 0$ for any $\lambda \in \mathcal{Y}$. Further, if $\lambda_i < 0$ and $z_i > 0$, then $z$ cannot be optimal: the value of the Lagrangian can be reduced by taking smaller $z_i$. $\square$

**Optimisation** **Part IB - Easter 2019**

# Lecture 5: Dual problem

Lecturer: Quentin Berthet

2.2. *Dual problems.* Another important notion related to the Lagrangian is that of the dual problem

Definition 2.3 (Dual function). For optimization problem (1), the *dual function* is defined for any $\lambda \in \mathcal{Y}$ as

$$g(\lambda) = \inf_{x \in \mathcal{X}} L(x, \lambda)\,.$$

Since this function is defined as an infimum on a larger space than $\mathcal{X}(b)$, on which $L(x, \lambda)$ and $f(x)$ coincide, it provides a lower bound for $f$ on $\mathcal{X}(b)$. This property is known as weak duality

Theorem 2.3 (Weak duality). *For any $x \in \mathcal{X}(b)$ and $y \in \mathcal{Y}$, we have $g(\lambda) \leq f(x)$, and in particular*

$$\sup_{\lambda \in \mathcal{Y}} g(\lambda) \leq \inf_{x \in \mathcal{X}(b)} f(x)\,.$$

Proof. We have, for any $x \in \mathcal{X}(b)$ and $\lambda \in \mathcal{Y}$

$$\begin{aligned} f(x) &= f(x) - \lambda^\top (h(x) - b) \\ &= L(x, \lambda) \\ &\geq \inf_{x' \in \mathcal{X}} L(x', \lambda) \\ &= g(\lambda)\,. \end{aligned}$$

Taking the supremum and infimum on both sides yields the desired result. □

It is therefore interesting to consider the maximum of this lower bound

Definition 2.4. Given optimization problem (1), the *dual problem* is defined as

$$\begin{aligned} \max \quad & g(\lambda) \\ s.t. \quad & y \in \mathcal{Y}\,. \end{aligned}$$

Remark. Further than simply giving a lower bound for this problem, under certain conditions, the result of Theorem 2.3 holds with equality, which is known as *strong duality.* In these cases, it can be simpler to solve this problem than problem (1) directly. Whenever some $x \in \mathcal{X}(b)$ and some $\lambda \in \mathcal{Y}$ have the same value for these two functions, we know that we are at optimality.

As an example, in the problem considered in the previous lecture, we have for any $\lambda \in \{\lambda \in \mathbf{R}^2 \,:\, \lambda_1 = -2, \lambda_2 < 0\}$ that

$$g(\lambda) = \inf_{x \in \mathcal{X}} L(x, \lambda) = L(x^*(\lambda), \lambda) = \frac{10}{\lambda_2} + 4\lambda_2 - 10\,.$$

This is maximized at $\lambda_2 = -\sqrt{5/8}$, and in this case there is strong duality.

In order to understand when strong duality holds, it is helpful to have the following interpretation of duality.

Definition 2.5. The *value function* $\varphi$ of optimization problem (1) is defined for $c \in \mathbf{R}^m$ as

$$\varphi(c) = \inf\{f(x) \,:\, h(x) = c\,,\ x \in \mathcal{X}\}\,.$$

Definition 2.6. We say that the function $\psi$ has a *supporting hyperplane* at $b \in \mathbf{R}^m$ if there exists $\lambda \in \mathbf{R}^m$ such that

$$\psi(c) \geq \psi(b) + \lambda^\top (c - b)\,.$$

Remark. The supporting hyperplane is a linear form that matches the function at $b \in \mathbf{R}^m$ and is below it everywhere. In particular, we have seen that differentiable convex functions have a supporting hyperplane at all points.

To grasp the link with dual problems, one can rely on the following fact, given without proof: for any $\lambda \in \mathbf{R}^m$, if the function defined by $\alpha(c) = \beta + \lambda^\top(c - b)$ is a supporting hyperplane at any point in $\mathbf{R}^m$, the value $\alpha(b) = \beta$ is equal to $g(\lambda)$. It is a lower bound for $\varphi(b)$, as known from weak duality. A special case is shown in the following result: if such a function is a supporting hyperplane at $b$, and in this case $\beta = \varphi(b)$, there is strong duality.

Theorem 2.4. *The value function $\varphi$ has a supporting hyperplane at $b$ if and only if there is strong duality for problem (1).*

Proof. There is strong duality if and only if there exists a $\lambda \in \mathbf{R}^m$ such that

$$\varphi(b) = \inf_{x \in \mathcal{X}} L(x, \lambda) = \inf_{x \in \mathcal{X}} \{f(x) - \lambda^\top (h(x) - b)\}\,.$$

There is a supporting hyperplane at $b$ if and only if there exists $\lambda \in \mathbf{R}^m$ such that

$$\varphi(b) = \inf_{c \in \mathbf{R}^m} \{\varphi(c) - \lambda^\top (c - b)\}$$

To show equivalence of these two properties, we see that for every $\lambda \in \mathcal{Y}$, we have

$$\begin{aligned}\inf_{x \in \mathcal{X}} \{f(x) - \lambda^\top (h(x) - b)\} &= \inf_{c \in \mathbf{R}^m} \inf_{x \in \mathcal{X}(c)} \{f(x) - \lambda^\top (h(x) - b)\} \\ &= \inf_{c \in \mathbf{R}^m} \inf_{x \in \mathcal{X}(c)} \{f(x) - \lambda^\top (c - b)\} \\ &= \inf_{c \in \mathbf{R}^m} \{\varphi(c) - \lambda^\top (c - b)\}\,.\end{aligned}$$

□

Remark. Of course, not every function has a supporting hyperplane at any given point. It is possible to generalize our remark on differentiable convex functions in the following theorem, given without proof.

Theorem 2.5. *A function is convex if and only if it has a supporting hyperplane at every point.*

This implies that if the value function $\varphi$ from $\mathbf{R}^m$ to $\mathbf{R}$ is convex, then we have strong duality, and the strategy from the previous lecture will work, provided that we can easily optimize $L(\cdot, \lambda)$ over $\mathcal{X}$. We now give very general conditions under which such conditions are met. We recall that for an optimization problem with variable $x \in \mathcal{X}$ constraint $h(x) \leq b$, it is equivalent to an optimization problem with variable $(x, z) \in \mathcal{X} \times \mathbf{R}^m_{\geq 0}$ and equality constraint $h(x) + z = b$, and the analysis above goes through. These problems are particularly important because of the following result.

Proposition 2.2. *The value function $\varphi$ to the inequality constrained problem defined by*

$$\begin{aligned} \varphi(b) = \min \quad & f(x) \\ s.t. \quad & h(x) \leq b\,, \\ & x \in \mathcal{X}\,, \end{aligned}$$

*is convex over the set where $\varphi$ is finite, if $f$,$h$, and $\mathcal{X}$ are convex.*

This is proved in Examples sheets, and is a callback to the message of Section 1: convex problems are "well-behaved", from the point of view of algorithms and of duality. Note that if $h$ is convex, then $\mathcal{X}(b) = \{x \in \mathcal{X} \,:\, h(x) \leq b\}$ is a convex set. Further, in the case of equality constraints $g(x) = c$, they can be interpreted as two constraints $g(x) \leq b$ and $-g(x) \leq -b$, and if both $g$ and $-g$ are convex these conditions are met again. In this case, $g$ is a linear function.

One way to interpret this result is to see this optimization problem as a convex cost minimization (or, up to a minus sign, concave utility maximization), under resource constraints $h_i(x) \leq b_i$: using the decision variable $x$ will consume $h_i(x)$ of the $i$-the resource, of which there is a budget $b_i$. If all sets and functions are convex, we are in a setting with diminishing return: as $b_i$ increases, any additional unit added allows for less improvement in the value of $\varphi(b)$. In mathematical terms, $\varphi$ is convex.

**Optimisation** **Part IB - Easter 2019**

# Lecture 6: Linear programming

Lecturer: Quentin Berthet

We saw in the previous lecture that one could understand minimization problems with inequality constraints as cost minimization under resource budgets. Convexity of $f$, $h$, and $\mathcal{X}$ implies convexity of the value function $\varphi(b)$, that can be interpreted as a diminishing returns. Another property obtained from strong duality also related to economics is an interpretation of dual variables, by showing that $\lambda_i^*$ is the "shadow price" for the $i$-th resource.

THEOREM 2.6. *If there is strong duality for optimization problem, with dual solution $\lambda^*$, and $\varphi$ is a differentiable function, then we have*

$$\frac{\partial \varphi}{\partial b_i}(b) = \lambda_i^*$$

PROOF. If there is strong duality, then there is a supporting hyperplane with slope $\lambda^*$, i.e.

$$\varphi(c) \geq \varphi(b) + {\lambda^*}^\top (c - b)\,.$$

If the function $\varphi$ is differentiable, the only vector that can have this property is its gradient, so $\nabla \varphi(b) = \lambda^*$. □

This result illustrates further complementary slackness: if the $i$-th resource is not fully used, then $\partial\varphi/\partial b_i(b) = 0$ : added value of any additional amount of resource $i$ is 0.

**3. Linear programming.** In the next part of this course, we mainly focus on the case of optimization problems with linear objectives and constraints. They are traditionally presented in the general form, with $c \in \mathbf{R}^n$, $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^m$ in the form

$$\begin{aligned} \max \quad & c^\top x \\ s.t. \quad & Ax \leq b\,, \\ & x \geq 0\,. \end{aligned}$$

Note that they are *convex minimization* problems, formally as maximizing $c^\top x$ is equivalent to minimizing $-c^\top x$. Linear optimization is the only case where minimization and maximization are both convex minimization problems, since a linear function is both concave and convex. The constraint $x \geq 0$ can always be used, as any unconstrained variable can be represented by $x_+ \geq 0$ and $x_- \geq 0$ with $x = x_+ - x_-$. Further, by introduction of a slack variable, all linear programs can be written in the following manner, called *standard form.*

DEFINITION 3.1. A linear program is in *standard form* if it is presented as

$$\begin{aligned} \max \quad & c^\top x \\ s.t. \quad & Ax = b\,, \\ & x \geq 0\,. \end{aligned}$$

PROOF. To show that this is equivalent to the general form, we can take $\tilde{x} = (x, z) \geq 0$, $\tilde{A} = (A \,|\, I_m)$, and $\tilde{A}\tilde{x} = b$ is equivalent to $Ax + z = b$, or $Ax \leq b$ since $z \geq 0$. □

REMARK. We note that if inequalities are in the other direction, changing one row from $a_i$ to $-a_i$. We recall that the feasible set $\mathcal{X}(b) = \{x \in \mathbf{R}^n \,:\, Ax = b\,,\, x \geq 0\}$ is convex, as proved in Examples sheet.

EXAMPLE 3.1. As an example, let

$$\begin{aligned} \max \quad & x_1 + x_2 \\ s.t. \quad & x_1 + 2x_2 \leq 6\,, \\ s.t. \quad & x_1 - x_2 \leq 3\,, \\ & x_1, x_2 \geq 0\,. \end{aligned}$$

It is possible to draw the feasible set, and even to see the solution, in an example with few variables. The constraint can be represented as

$$\begin{pmatrix} 1 & 2 & 1 & 0 \\ 1 & -3 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} 6 \\ 3 \end{pmatrix}.$$

3.1. *Solutions of linear programs.* Even though this is a convex optimization problem, and formal guarantees on algorithms to solve it are obtained by using techniques described in the first part of this course (barrier function, Newton's method), we can also use the fact that it is also a convex maximization problem, to make formal the intuition that these problems are maximized "in the corners" of the optimization domain. Informally, for any convex function $f$, and $x = \lambda y + (1-\lambda)z$, we have

$$f(x) = f(\lambda y + (1-\lambda)z) \leq \lambda f(y) + (1-\lambda)f(z) \leq \max\{f(y), f(z)\}\,.$$

It is therefore intuitive that to maximize a convex function on a convex set, one should only worry about the extremities of segments or more formally, extreme points, as defined here.

DEFINITION 3.2. Let $C$ be a convex set. We say that $x \in C$ is an *extreme point* of $C$ if for all $y, z$ in $C$ such that $x = (1-\lambda)y + \lambda z$ with $\lambda \in (0,1)$, then $x = y = z$.

Informally, $x$ does not belong to any non-trivial segment of $C$. Note that this is stronger than requiring $x$ not to be in the interior of $C$, as points on the boundary of a convex set are not always extreme points. We often show that a point $x$ is extreme if for all nonzero $h \in \mathbf{R}^n$, for no $\varepsilon > 0$ do we have $x + \varepsilon h$ and $x - \varepsilon h$ both in $C$.

Extreme points of the domain $\{x \in \mathbf{R}^n \ : \ Ax = b, x \geq 0\}$ have a very special structure, as seen in the following

Definition 3.3. A solution to $Ax = b$, with $x \in \mathbf{R}^n$ and $A \in \mathbf{R}^{m \times n}$ is called a *basic solution* if it has at most $m$ non-zero entries. This set of non-zero entries is denoted by $B$. It is called a *basis* and a variable $x_i$ is called *basic* if $i \in B$ and *non-basic* if $i \notin B$. A basic solution that satisfies $x \geq 0$ is called a *basic feasible solution* (BFS).

In this course, we make the following assumptions:

(i) The $m$ rows of $A$ are linearly independent vectors of $\mathbf{R}^n$.

(ii) Every set of $m$ rows is linearly independent.

(iii) Every basic solution is *non-degenerate*, i.e. has exactly $m$ non-zero entries.

Under these assumptions, the idea is that for every basis $B$ of $m$ variables, there is exactly one solution to $A_B x = b$, since $A_B$ is a square invertible matrix, and therefore only one basic solution with basis $B$. The first two are

Theorem 3.1. *The extreme points of $\mathcal{X}(b) = \{x \geq 0 \ : \ Ax = b\}$ are the basic feasible solutions of $Ax = b$.*

Proof. Let $x$ be a basic feasible solution and $y, z \in \mathcal{X}(b)$ satisfy $x = \delta y + (1-\delta) z$ with $\delta \in (0, 1)$. For the non-basic entries $x_i = 0$, so $y_i = z_i = 0$, since they are nonnegative. As a consequence, $y$ and $z$ are basic with basis $B$, as $x$ is. Since $A_B$ is invertible, $x = y = z$.

To show the reverse, we prove that a non-basic solution is not an extreme point. Let $i_1, \ldots, i_r$ be the non-zero entries of a non-basic solution $x$, with $r > m$. The $r$ columns $a_{i_1}, \ldots, a_{i_r}$ are linearly dependent in $\mathbf{R}^m$, so there exists $w \in \mathbf{R}^n$ with the same non-zero entries such that

$$a_{i_1} w_{i_1} + \ldots + a_{i_r} w_{i_r} = 0$$

As a consequence, $Aw = 0$ so for any $\varepsilon > 0$, $A(x \pm \varepsilon w) = b$, and for $\varepsilon > 0$ small enough $x \pm \varepsilon w \geq 0$ in both cases. As a consequence, $x$ is not an extreme point. $\square$

**Optimisation** **Part IB - Easter 2019**

# Lecture 7: Solutions of linear programs

Lecturer: Quentin Berthet

In the previous lecture, we saw that in order to solve linear optimization problems in the standard form

$$\begin{aligned} \max \quad & c^\top x \\ s.t. \quad & Ax = b\,, \\ & x \geq 0\,. \end{aligned}$$

It seems important to consider the extreme points of the set $\mathcal{X}(b) = \{x \geq 0 \,:\, Ax = b\}$, which correspond to basic feasible solutions. We show this formally in the following theorem

Theorem 3.2. *A linear program that is feasible and bounded is maximized at one of the basic feasible solutions.*

Proof. Let $x$ be a maximum of $c^\top x$ on $\mathcal{X}(b)$. If $x$ has $m$ non-zero entries, then it is a basic feasible solution. If not, it has $r > m$ non-zero entries by assumptions. We will then show that there is a maximizer, with at least one more zero. By induction, this yields a feasible maximizer with at most $m$ non-zero entries.

In this case, since $r > m$ (this is how the induction stops at $m$), by Theorem 3.1, $x$ is not an extreme point of $\mathcal{X}(b)$, so there exist $y \neq z \in \mathcal{X}(b)$ and $\lambda \in (0,1)$ such that $x = \lambda y + (1-\lambda)z$. As a consequence, we have that $c^\top x = \lambda c^\top y + (1-\lambda)c^\top z$, so $c^\top x = c^\top y = c^\top z$, so $x, y, z$ are all optimal. Further, similarly to the proof of Theorem 3.1, $x_i = 0$ implies $y_i = 0$ and $z_i = 0$, so $y, z$ share at least the zeros of $x$. If one of them has one more zero, we are done.

Otherwise, we can consider $x' = \lambda' y + (1-\lambda')z = z + \lambda'(y-z)$, for any $\lambda' \in \mathbf{R}$. By the properties above, $x'$ satisfies $Ax = b$ and does not have more nonzero entries. Further, if $\lambda'$ is close enough to $\lambda$, it is still feasible. All of its entries are linear functions of $\lambda'$, with at least one of them having a non-zero slope, since $y \neq z$. As a consequence, it is possible to take $\lambda'$ small enough so that $x'$ is still feasible, with one more zero entry. $\square$

Remark. This gives a formal confirmation to the idea that only extreme points are important for linear programs. Further, there is an explicit, algebraic way to describe extreme points of this feasible set. One can in theory compute, for all choices of basis $B$ of size $m$, the basic solution $x_B$, check if $x_B \geq 0$, and compute $c^\top x_B$. If the program is feasible and bounded, one of these is optimal, and we can find it by examining all the possibilities.

While that may be viable when $m, n$ are small numbers as in our examples, this is in general not a practical approach, as there are $\binom{n}{m}$ possible choices of basis: if $m$ and $n$ are large integers of the same order, this is exponential in $n$! Even if we are lucky and find the optimal basis in of the first tries, we also have no way of knowing, as we have no optimality criteria. In order to find one, we can turn to duality: this is a problem with convex constraints and a convex objective (even when written in the minimization form), so strong duality holds

3.2. *Duality in linear programming.*

Definition 3.4. For a linear program in the general form, the Lagrangian is

$$L(x, z, \lambda) = c^\top x - \lambda^\top (Ax + z - b)\,.$$

Remark. The problem is equivalent to minimizing $-c^\top x$ such that $-Ax - z = -b$, giving a Lagrangian equal to $-c^\top x - \lambda^\top(-Ax - z + b)$, following our definitions so far. This is the same Lagrangian, up to a minus sign. Traditionally, convex optimization is concerned with minimization of convex functions, and linear programming with maximization of concave functions, so to go from one to the other sometimes requires to change the sign of notations, as in the following.

The dual function is defined, for $\lambda \in \mathbf{R}^m$, by

$$g(\lambda) = \max_{x \geq 0, z \geq 0} L(x, z, \lambda)$$

and $\mathcal{Y} = \{\lambda \in \mathbf{R}^m \,:\, \max_{x \geq 0, z \geq 0} L(x, z, \lambda) < \infty\}$. The dual problem is defined as

$$\begin{array}{ll} \min & g(\lambda) \\ s.t. & \lambda \in \mathcal{Y}\,. \end{array}$$

Computation of the LP dual. We have, rearranging terms

$$L(x, z, \lambda) = (c - A^\top \lambda)^\top x - \lambda^\top z + b^\top \lambda\,.$$

When maximizing over $z \geq 0$, $\lambda \in \mathcal{Y} \Rightarrow \lambda \geq 0$, otherwise, letting an entry of $z$ go to $\infty$, the problem is unbounded. Similarly, when maximizing over $x \geq 0$, $\lambda \in \mathcal{Y} \Rightarrow c - A^\top \lambda \leq 0$, otherwise, letting an entry of $x$ go to $\infty$, the problem is unbounded.

For any $\lambda \geq 0$ such that $c - A^\top \lambda \leq 0$, we therefore have

$$\max_{x \geq 0, z \geq 0} L(x, z, \lambda) = b^\top \lambda = g(\lambda)\,,$$

taking $z_i \neq 0$ and $x_i \neq 0$ only when the corresponding entries of $\lambda$ and $c - A^\top \lambda$ are not zero. As a consequence, by double inclusion, $\mathcal{Y} = \{\lambda \geq 0 \,:\, c - A^\top \lambda \leq 0\}$ and the dual problem is

$$\begin{array}{ll} \max & b^\top \lambda \\ s.t. & A^\top \lambda \geq c\,, \\ & \lambda \geq 0\,. \end{array}$$

One finds that the dual problem in the standard form is the same, without the constraint $\lambda \geq 0$. More importantly, the dual of the dual problem is the original problem, called the *primal problem.* This leads to the following result, a consequence of strong duality □

THEOREM 3.3. *For an LP in the general form, if $x^*$ and $\lambda^*$ satisfy*

- $Ax^* \leq b$ *and* $x^* \geq 0$ *(primal feasibility)*
- $A^\top \lambda^* \leq c$ *and* $\lambda^* \geq 0$ *(dual feasibility)*
- $\lambda^*(b - Ax^*) = x^{*\top}(c - A^\top \lambda^*) = 0$ *(complementary slackness)*

*then $x^*$ is optimal for the primal problem and $\lambda^*$ is optimal for the dual, there is strong duality and $c^\top x^* = b^\top \lambda^*$ is the value of the problem.*

REMARK. When the problem is in the standard form, this is replaced by

- $Ax^* = b$ and $x^* \geq 0$ (primal feasibility)
- $A^\top \lambda^* \leq c$ (dual feasibility)
- $x^{*\top}(c - A^\top \lambda^*) = 0$ (complementary slackness)

In this case, we have $c^\top x^* = c^\top x^* - \lambda^{*\top}(Ax^* - b) = (c - A^\top \lambda^*)^\top x^* + b^\top \lambda = b^\top \lambda^*$.

Further, for any basic feasible solution in the standard form, we can find a corresponding dual solution using complementary slackness. If the solution is optimal, this dual is feasible. There is a corespondance between feasibility of the dual and optimality of the primal (and vice-versa). Indeed, for any basis $B$ and complement $N$, we have $x = x_B + X_N$, with $x_N = 0$ for a BFS. Feasibility implies $Ax = A_B x_B + A_N x_N = b$, so $x_B = A_B^{-1} b$, and the BFS is uniquely determined.

In this case, we can look for a dual solution $\lambda$ that satisfies the optimality properties

$$0 = (c - A^\top \lambda)^\top x = (c_B - A_B^\top \lambda)^\top x_B + (c_N - A_N^\top \lambda)^\top x_N = (c_B - A_B^\top \lambda)^\top x_B \,.$$

Since $x_B \geq 0$ and we are looking for $\lambda$ such that at least $c_B - A_B^\top \lambda \leq 0$, this imposes $\lambda = (A_B^\top)^{-1} c_B$.

Just like the basic feasible solution is entirely determined by $B$, with $x_N = 0$ and $x_B = A_B^{-1} b$, so is the dual variable satisfying complementary slackness. It satisfies $c_B - A_B^\top \lambda = 0$, and if it satisfies $c - A^\top \lambda \leq 0$, $\lambda$ is dual feasible, and the BFS is optimal.

**Optimisation** **Part IB - Easter 2019**

# Lecture 8: The simplex method

Lecturer: Quentin Berthet

3.3. *The simplex method.* We saw in the previous lecture that solutions of linear programs were always at the extreme points, corresponding to basic feasible solutions. Further, duality gives us an optimality criteria: for a basic feasible solution with basis $B$, $x_N = 0$ and $x_B = A_B^{-1}b$, the dual variable $\lambda = (A_B^\top)^{-1}c_B$ satisfies complementary slackness and $c_B - A_B^\top \lambda = 0$. If it satisfies $c - A^\top \lambda \leq 0$, it is dual feasible and the BFS $x$ is optimal.

This formulation can be used to design an optimization method: for every $x'$ such that $Ax' = b$, we have for any basis $B$ $Ax' = A_B x'_B + A_N x'_N = b$, so $x'_B = A_B^{-1}(b - A_N x'_N)$. Fixing all but $m$ entries of $x'$ leaves only one choice for the $m$ remaining. The value of the objective at $x'$ is given by

$$\begin{aligned} f(x') = c^\top x' &= c_B^\top x'_B + c_N^\top x'_N \\ &= c_B^\top A_B^{-1}(b - A_N x'_N) + c_N^\top x'_N \\ &= c_B^\top A_B^{-1} b + (c_N^\top - c_B^\top A_B^{-1} A_N) x'_N \,. \end{aligned}$$

Taking $x_B = A_B^{-1}b$ and $x_N = 0$, the basic solution with basis $B$, the first term is the objective value at $x$: $c^\top x = c_B^\top x_B = c_B^\top A_B^{-1} b$. The second term is equal to $(c_N - A_N^\top (A_B^\top)^{-1} c_B)^\top x'_N = (c_N - A_N^\top \lambda)^\top x'_N$, where $\lambda$ corresponds to the basic solution $x$. This confirms the analysis with duality: if $A_B^{-1}b$ is feasible and $c_N - A_N^\top \lambda \leq 0$, then the optimal choice is $x'_N = 0$, and $x$ is optimal for this problem.

More importantly, if this optimality criteria is not met and there is some $j \in N$ such that $(c_N - A_N^\top (A_B^\top)^{-1} c_B)_j > 0$, this suggests to start from $x' = x$ and to increase $x'_j$, thus increasing the objective. For any positive value of $x'_j$, letting the other non-basic coefficients at 0, the basic coefficients are determined by $x'_B = A_B^{-1}(b - A_N x'_N) = A_B^{-1}(b - a^{(j)} x'_i) = A_B^{-1} b - x'_j A_B^{-1} a^{(j)}$, where $a^{(j)}$ is the $j$-th row of $A$. The basic entries are linear functions of $x'_j$: if they are all increasing in $x'_j$, and the BFS $A_B^{-1}b$ is feasible, then $x'$ remains feasible, for any value of $x'_j$, and the problem is unbounded. If at least one of the entries is decreasing in $x'_j$, it will eventually reach 0, and we have a new basis $B'$, with this variable removed, and the $j$-th variable added. Under the assumptions on $A$, there are not two variables that reach 0 at the same time: this would yield a basic solution with less than $m$ non-zero entries, which is not possible.

Together, this procedure can be summarized as follows:

Definition 3.5. Simplex algorithm:

1. Start in a basic feasible solution $x$ with basis $B$, compute corresponding dual variables, check if optimality conditions are met.
2. If not, identify $j$ such that $(c_N - A_N^\top (A_B^\top)^{-1} c_B)_j > 0$ and increase $x_j$, keeping the other variables such that $Ax = b$, until a new variable reaches 0.
3. We now have a new basis $B'$, repeat this procedure until optimality conditions are satisfied.

Remark. As noted in the previous lecture and at the beginning of this one, a dual variable satisfying complementary slackness corresponding to the BFS can always be obtained by solving $c_B - A_B^\top \lambda = 0$. If $c_N - A_N^\top \lambda \leq 0$, then the BFS is optimal.

If in the second phase, no variable reaches 0, and $x_j$ can be increased indefinitely, the problem is unbounded.

Following this algorithm, we go from basis to basis, always increasing the objective. Because there are finitely many possible basis, we are bound to find an optimal solution.

Information to run this procedure by hand, without often inverting matrices, can be stored in the following form.

Definition 3.6. The *simplex tableau* is an $(n+1) \times (m+1)$ array that summarizes information about the problem, and the state of the algorithm at basis $B$

| $B$: $m$ entries | $N$: $n - m$ entries | 1 entry |
|---|---|---|
| $A_B^{-1} A_B = I$ | $A_B^{-1} A_N$ | $A_B^{-1} b$ |
| $c_B^T - c_B^T A_B^{-1} A_B = 0$ | $c_N^T - c_B^T A_B^{-1} A_N$ | $-c_B^T A_B^{-1} b$ |

Remark. The first $m$ rows, consist of $A$ and $b$, multiplied by $A_B^{-1}$. In particular, in the last column we observe the $m$ nonzero (basic) entries of $x_B = A_B^{-1} b$. In the last row, the first $m$ columns are $c^\top - \lambda^\top A$, for $\lambda = (A_B^\top)^{-1} c_B$. In the last corner, we have $-f(x) = -c_B^\top x_B$, for $x$ being the BFS with basis $B$.

The dual optimality condition can be checked by observing the first $m$ elements of the last row. If they are all nonpositive, the basis $B$ is optimal.

The second step of the simplex algorithm can be implemented, by performing the following operations, using the notation

| | |
|---|---|
| $a_{ij}$ | $a_{i0}$ |
| $a_{0j}$ | $a_{00}$ |

1. First, at any round, we check if $a_{0j} \leq 0$ for every $j > 0$. If this holds, the basis $B$ is optimal.

2. If not, we choose $j$ such that $a_{0j} > 0$ and we take $i \in \{i' : a_{i'j} > 0\}$ that minimizes $a_{i0}/a_{ij}$. This is equivalent to finding the first basic variable whose entry will reach 0 when increasing $x_j$ and maintaining feasibility, among all of those that will decrease.

$*$ Issue 1: if $a_{ij} \leq 0$ for all $i$, all entries will increase and as seen above, the problem is unbounded.

$*$ Issue 2: if multiple rows minimize $a_{i0}/a_{ij}$, this means that a solution is degenerate, which is not covered in this course.

3. Update the tableau: We multiply row $i$ by $1/a_{ij}$. For each row $k \neq i$, we add a multiple $-(a_{kj}/a_{ij})$ of row $i$ to row $k$. With this last step, we are changing to the tableau for the following basis $b'$ in the simplex algorithm, without having to invert $A_{B'}$

Example 3.2. We consider the linear program in the general form

$$\begin{aligned} \max \quad & x_1 + x_2 \\ s.t. \quad & x_1 + 2x_2 \leq 6\,, \\ & x_1 - x_2 \leq 3\,, \\ & x_1, x_2 \geq 0\,. \end{aligned}$$

We convert it into standard form

$$\begin{aligned} \max \quad & x_1 + x_2 \\ s.t. \quad & x_1 + 2x_2 + z_1 = 6\,, \\ & x_1 - x_2 + z_2 = 3\,, \\ & x_1, x_2, z_1, z_2 \geq 0\,. \end{aligned}$$

This particular linear program can conveniently be started in the basic feasible solution with basis $B = \{3, 4\}$ on the variables $z_1, z_2$, and basic feasible solution $x = (0, 0, 6, 3)^\top$. This is particularly convenient, as $A_B = I_2$ for this basis.

The corresponding tableau is therefore directly written as

| $x_1$ | $x_2$ | $z_1$ | $z_2$ | |
|---|---|---|---|---|
| 1 | 2 | 1 | 0 | 6 |
| 1 | $-1$ | 0 | 1 | 3 |
| 1 | 1 | 0 | 0 | 0 |

1. Since $a_{0j} > 0$ for $j = 1, 2$, variables $x_1$ and $x_2$ can be increased. We take $j = 1$, for $x_1$.

2. If $a_{ij} \leq 0$ for all $i$, the problem is unbounded, which is not the case here. We have $\{i' : a_{i'j} > 0\} = \{1, 2\}$, both basic variables will decrease when increasing $x_1$ (as an aside, this makes sense looking at the problem: the basic variables are slack, and for both inequalities, the slack decreases when $x_1$ increases). We take $i$ that minimizes $a_{i0}/a_{ij}$, this is $i = 2$, corresponding to $i = 2$, for variable $z_2$, which will reach zero first while maintaining feasibility.

3. We multiply row $i = 2$ by $1/a_{ij}$ (here equal to 1), and add a multiple by $-a_{kj}/a_ij$ of row $i$ to row $k = 1, 3$, obtaining the new tableau with basis $B' = \{3, 1\}$ and BFS $(3, 0, 3, 0)^\top$.

| $x_1$ | $x_2$ | $z_1$ | $z_2$ | |
|---|---|---|---|---|
| 0 | 3 | 1 | $-1$ | 3 |
| 1 | $-1$ | 0 | 1 | 3 |
| 0 | 2 | 0 | $-1$ | $-3$ |

We stopped here during the lecture, but we can now notice that for $j = 2$, $a_{0j} > 0$, and in this column, the only corresponding row is $i = 1$, with $a_{12} = 3$. Performing the same operations, we obtain the new tableau, with basis $B' = \{2, 1\}$ and BFS $(4, 1, 0, 0)^\top$. The maximum is at $x_1 = 4$, $x_2 = 1$, with objective 5.

| $x_1$ | $x_2$ | $z_1$ | $z_2$ | |
|---|---|---|---|---|
| 0 | 1 | $1/3$ | $-1/3$ | 1 |
| 1 | 0 | $1/3$ | $2/3$ | 4 |
| 0 | 0 | $-2/3$ | $-1/3$ | $-5$ |

**Optimisation** **Part IB - Easter 2019**

# Lecture 9: Flow and transportation problems

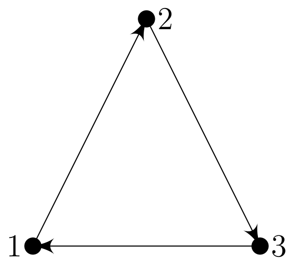
Lecturer: Quentin Berthet

**4. Applications of linear programming.** Linear programming has many applications, and its origins are actually in solving optimization problems related to flows on graphs.

Definition 4.1. A directed *graph* $G = (V, E)$ consists in a set of vertices $V$ (e.g. $\{1, \ldots, n\}$) and a set of ordered edges $E \subseteq V \times V$.

When $E$ is symmetric (i.e. $(v, v') \in E$ iff $(v', v) \in E$), $G$ is a *directed graph.*

Example 4.1. We consider the following exampe of a graph with three vertices



$$V = \{1, 2, 3\}$$
$$E = \{\{1, 2\}, \{2, 3\}, \{3, 1\}\}$$
$$\{1, 3\} \notin E$$

4.1. *Minimum cost flow problem.* For a graph $G = (V, E)$ with $|V| = n$, we can consider a *flow* on the network, denoted by $x_{ij}$, the quantity flowing from $i$ to $j$. This can represent water, goods, people, etc. depending on the application. The constraints on this flow are inspired by physical or real-life constraints, and driven by

- A vector $b \in \mathbf{R}^n$: the amount of flow $b_i$ that enters or leaves the graph at vertex $i$. If $b_i > 0$, we say that there is a *source* in $i$ and if $b_i > 0$, that there is *a sink.*
- A matrix $C \in \mathbf{R}^{n\times n}$: the *cost* $c_{ij}$ associated to an edge $(i, j)$. Note that it is sufficient to define this matrix on $E$.
- Matrices $\underline{M}$ and $\overline{M}$: lower and upper bounds, sometimes called capacities, on the flow $x_{ij}$ between $i$ and $j$.

Definition 4.2. The *minimum cost flow problem* is the linear program

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} c_{ij} x_{ij} \\ s.t. \quad & b_i + \sum_{j:(j,i)\in E} x_{ji} = \sum_{j:(i,j)\in E} x_{ij} \,\text{, for all } i \in V \\ & \underline{m}_{ij} \le x_{ij} \le \overline{m}_{ij}\,. \end{aligned}$$

Remark. The first line of constraints states that for every vertex $i \in V$, the amount of flow entering the vertex (LHS of the equation) must be equal to the amount leaving (RHS of the equation): no flow is stored in the vertex. This is a physical constraint, making sure that the variable $x$ denotes indeed a real flow. The inequalities represent capacity constraints for each edge. The equality constraints can be written in the form $Ax = b$, with each edge corresponding to a column, and for the $k$-th column: $a_{ik} = 1$ for vertex $i$ where the $k$-th edge starts, $-1$ for the vertex where it ends, and 0 otherwise.

For feasibility of the problem (i.e. to have a flow $x$ that is feasible), we need $\sum_{i \in V} b_i = 0$, as can be seen from summing all the equality constraints. It can be understood as a global physical constraint: what enters the network must come out. This can be transformed into a problem where $b_i = 0$ for all $i \in V$, known as a *circulation problem*, by adding new vertex and edges: An "outside" vertex connected to all other vertices with uncapacited edges, with flow $b_i$ along the edge between the outside and vertex $i$.

Proposition 4.1. *The Lagrangian for this problem, defined over the feasible set* $\mathcal{X} = \{x \,|\, \underline{m}_{ij} \leq x_{ij} \leq \overline{m}_{ij}\}$, *with* $h(x) = b$ *as* $\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = b_i$ *can be written directly for the circulation problem (i.e. where* $b_i = 0$*)*

$$\begin{aligned} L(x,\lambda) &= \sum_{(i,j)\in E} c_{ij} x_{ij} - \sum_{i \in V} \lambda_i \Big( \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} \Big) \\ &= \sum_{(i,j)\in E} (c_{ij} - \lambda_i + \lambda_j) x_{ij} \,. \end{aligned}$$

Recalling the Lagrange sufficiency theorem (if $x$ minimizes $L(x,\lambda)$ over $\mathcal{X}$ for some $\lambda$ and $h(x) = b$, then $x$ is optimal), we have the following

Theorem 4.1. *Let $x$ be a feasible flow (i.e. $x \in \mathcal{X}$ and $h(x) = b$), and $\lambda \in \mathbf{R}^n$ s.t.*

$$\begin{aligned} &\textit{For all } i,j \textit{ such that} && c_{ij} - \lambda_i + \lambda_j > 0\,, && \textit{we have} && x_{ij} = \underline{m}_{ij} \\ &\textit{For all } i,j \textit{ such that} && c_{ij} - \lambda_i + \lambda_j < 0\,, && \textit{we have} && x_{ij} = \overline{m}_{ij} \\ &\textit{For all } i,j \textit{ such that} && \underline{m}_{ij} \leq x_{ij} \leq \overline{m}_{ij}\,, && \textit{we have} && c_{ij} - \lambda_i + \lambda_j = 0\,, \end{aligned}$$

*then $x$ is a minimal flow.*

Proof. The conditions directly imply that $x$ minimizes $L(x,\lambda)$ over $\mathcal{X}$. □

4.2. *Transportation problem.* They are a special case of minimum cost flow problems, with disjoint sets of suppliers $S = \{1,\dots,n\}$ and of consumers $C = \{1,\dots,m\}$.

Definition 4.3. Over a bipartite graph between $S$ and $C$ (meaning with edges only going from $S$ to $C$, i.e. $E \subset S \times C$), given a list of *supplies* $s_1,\dots,s_n$ and *demands* $d_1,\dots,d_n$ such that $\sum_{i=1}^n s_i = \sum_{j=1}^m d_j$, the *transportation problem* (or optimal transport problem) is defined as

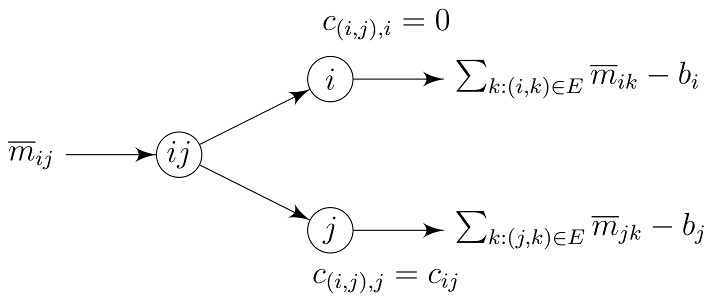
$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \\ s.t. \quad & \sum_{i=1}^{m} x_{ij} = s_i \,, \text{for all } j \in S \\ & \sum_{j=1}^{n} x_{ij} = d_i \,, \text{for all } i \in C \\ & x \geq 0\,. \end{aligned}$$

Even though this is a special case, it actually covers up to modification all cost minimization problems

THEOREM 4.2. *Every minimum cost flow problem with finite capacities or nonnegative costs has an equivalent transportation problem.*

PROOF. We can first assume that $\underline{m}_{ij} = 0$ for all $(i,j) \in E$. If $\underline{m}_{ij} \neq 0$ for some specific $i$ and $j$, it can be replaced by 0 without changing the problem, by also changing $\overline{m}_{ij}$ to $\underline{m}_{ij}$, $b_i$ to $b_i - \underline{m}_{ij}$ and $b_j$ to $b_j + \underline{m}_{ij}$. With these new constraints, for every feasible $x_{ij}$ in the old problem, $x_{ij} + \underline{m}_{ij}$ is feasible in the new problem. This can be interpreted as measuring the flow in another unit, with zero placed in another value (think Celsius and Kelvin). With no further loss of generality, all edges have a finite capacity. If not, we can always replace them with an arbitrarily large number, say greater than $\sum_{i \in V} |b_i|$. As long as all costs are nonnegative, this does not affect the minimization problem.

Under these unrestrictive assumptions, we can create the following transport problem, between suppliers associated to an edge $(i,j)$ and consumers to vertices. They are connected in the following manner.


$c_{(i,j),i} = 0$
$i$
$\sum_{k:(i,k)\in E} \overline{m}_{ik} - b_i$
$\overline{m}_{ij}$
$ij$
$j$
$\sum_{k:(j,k)\in E} \overline{m}_{jk} - b_j$
$c_{(i,j),j} = c_{ij}$

Any feasible flow in the original problem can be decomposed into $\overline{m}_{ij} - x_{ij}$ flowing in the top edge (between $(i,j)$ and $i$) and $x_{ij}$ in the bottom edge (between $(i,j)$ and $j$). One can check directly that feasibility of $x$ for the original problem implies feasibility of this new flow for the transport problem. The value of the objective is also the same for both problems. □

**Optimisation** **Part IB - Easter 2019**

# Lecture 10: Transport algorithm

Lecturer: Quentin Berthet

We recall that the transport problem between $n$ suppliers and $m$ customers, introduced in the previous lecture, is defined as

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \\
s.t. \quad & \sum_{j=1}^{m} x_{ij} = s_i \,, \text{for all } i \in S \\
& \sum_{j=1}^{n} x_{ij} = d_i \,, \text{for all } i \in C \\
& x \geq 0 \,.
\end{aligned}$$

Proposition 4.2. *If for some feasible $x$ we have dual variables $\lambda \in \mathbf{R}^n$ (for suppliers) and $\mu \in \mathbf{R}^m$ (for customers) such that*

$$c_{ij} \geq \lambda_i + \mu_j \quad \textit{for all } i, j$$

*and*

$$(c_{ij} - (\lambda_i + \mu_j))x_{ij} = 0 \quad \textit{for all } i, j \,,$$

*then $x$ is optimal.*

Proof. We write the Lagrangian of this problem

$$\begin{aligned}
L(x, \lambda, \mu) &= \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} + \sum_{i=1}^{n} \lambda_i \Big(s_i - \sum_{j=1}^{m} x_{ij}\Big) + \sum_{j=1}^{m} \mu_j \Big(d_j - \sum_{i=1}^{n} x_{ij}\Big) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} (c_{ij} - (\lambda_i + \mu_j))x_{ij} + \sum_{i=1}^{n} \lambda_i s_i + \sum_{j=1}^{m} \mu_j d_j \,.
\end{aligned}$$

By Theorem 3.3 on optimality conditions for linear programs, there is primal and dual feasibility, as well as complementary slackness, and the claim holds. $\square$

For transportation problems, the simplex algorithm can be efficiently run by using a transportation tableau containing at any iteration the flow variables $x_{ij}$, the dual variables $\lambda_i$ and $\mu_j$, as well as the costs $c_{ij}$ and demands and supplies $d_i$ and $s_j$, constant throughout the algorithm iteration.

$$
\begin{array}{c|cc|cc|cc|cc|c}
 & \mu_1 & & \mu_2 & & \cdots & & \mu_m & & \\
\hline
\lambda_1 & x_{11} & \boxed{c_{11}} & x_{12} & \boxed{c_{12}} & \cdots & \boxed{\cdots} & x_{1m} & \boxed{c_{1m}} & s_1 \\
\hline
\vdots & \cdots & \boxed{\cdots} & \cdots & \boxed{\cdots} & \cdots & \boxed{\cdots} & \cdots & \boxed{\cdots} & \vdots \\
\hline
\lambda_n & x_{n1} & \boxed{c_{n1}} & x_{n2} & \boxed{c_{n2}} & \cdots & \boxed{\cdots} & x_{nm} & \boxed{c_{nm}} & s_n \\
\hline
 & d_1 & & d_2 & & \cdots & & d_m & &
\end{array}
$$

It can also be written with sums of dual variables $\lambda_i + \mu_j$, particularly for variables set to 0 in order to track the condition $c_{ij} \geq \lambda_i + \mu_j$, as in this 4-by-3 tableau.

$$
\begin{array}{c|cc|cc|cc|cc|c}
 & \mu_1 & & \mu_2 & & \mu_3 & & \mu_4 & & \\
\hline
 & \lambda_1+\mu_1 & & \lambda_1+\mu_2 & & \lambda_1+\mu_3 & & \lambda_1+\mu_4 & & \\
\lambda_1 & x_{11} & \boxed{c_{11}} & x_{12} & \boxed{c_{12}} & x_{13} & \boxed{c_{13}} & x_{14} & \boxed{c_{14}} & s_1 \\
\hline
 & \lambda_2+\mu_1 & & \lambda_2+\mu_2 & & \lambda_2+\mu_3 & & \lambda_2+\mu_4 & & \\
\lambda_2 & x_{21} & \boxed{c_{21}} & x_{22} & \boxed{c_{22}} & x_{23} & \boxed{c_{23}} & x_{24} & \boxed{c_{24}} & s_2 \\
\hline
 & \lambda_3+\mu_1 & & \lambda_3+\mu_2 & & \lambda_3+\mu_3 & & \lambda_3+\mu_4 & & \\
\lambda_3 & x_{31} & \boxed{c_{31}} & x_{32} & \boxed{c_{32}} & x_{33} & \boxed{c_{33}} & x_{34} & \boxed{c_{34}} & s_3 \\
\hline
 & d_1 & & d_2 & & d_3 & & d_4 & &
\end{array}
$$

4.3. *The transport algorithm.* The first part of the algorithm is to construct an initial basic feasible solution. This is achieved by following these steps:

- Start with any edge variable (say $x_{11}$), and increase it until either supply $s_i$ or demand $d_j$ (here, $s_1$ or $d_1$ for the choice $x_1$1) is satisfied.

- If supply is satisfied, move to the next supplier and repeat the first step.

- If demand is satisfied, move to the next customer and repeat the first step.

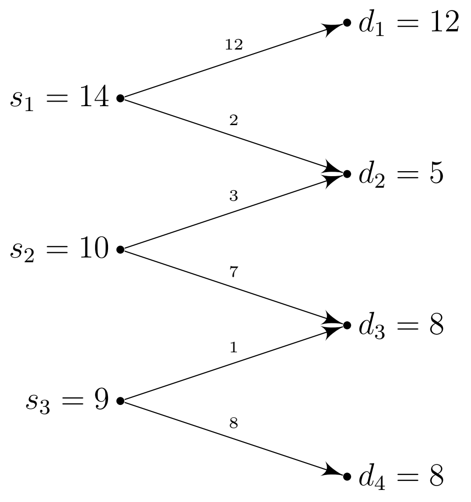
At the end, supply and demand should be satisfied simultaneously. If they are satisfied earlier, the problem is degenerate. We illustrate this process through the following example. with supplies $\{14, 10, 9\}$ and demands $\{12, 5, 8, 8\}$ (both summing to 33: the problem is feasible).

- We start with the edge between the first supplier (with $s_1 = 14$) and the first customer (with $d_1 = 12$). We augment the flow until one of these is met, so $x_{11} = 12$.

- As demand is satisfied first, we move to the next customer, with $d_2 = 5$. We augment the flow until $x_{12} = 2$, and the overall supply of the first supplier is exhausted, i.e. $x_{11} + x_{12} = 12 + 2 = 14 = s_1$.

- We repeat these steps until the end.

We obtain the following feasible flow, where the values $x_{ij}$ are written in small font above the corresponding arrows.



The initial corresponding tableau is

|  | 5 | 3 | 0 | 2 |  |
|---|---|---|---|---|---|
| 0 | 12 $\boxed{5}$ | 2 $\boxed{3}$ | 0<br>$\boxed{4}$ | 2<br>$\boxed{6}$ | 14 |
| 4 | 9<br>$\boxed{2}$ | 3 $\boxed{7}$ | 7 $\boxed{4}$ | 6<br>$\boxed{1}$ | 10 |
| 2 | 7<br>$\boxed{5}$ | 5<br>$\boxed{6}$ | 1 $\boxed{2}$ | 8 $\boxed{4}$ | 9 |
|  | 12 | 5 | 8 | 8 |  |

The flows are in the left part of each cell, written only when they are not 0. The basic dual variables are determined by setting $x_{ij} = \lambda_i + \mu_j$ when $x_{ij} > 0$, and $\lambda_i + \mu_j$ is only written in the top part of the cell when $x_{ij} = 0$, as otherwise the information is redundant.

PROPOSITION 4.3 (Properties of basic feasible solutions). *The set of edges with positive flow created with this process form a graph that is connected (it goes through all the vertices) and has no cycle (we do not go back to a previously visited node). It therefore forms a* spanning tree $T$ *(i.e. a connected graph with no cycles).*

- *We have $x_{ij} = 0$ whenever $(i, j) \notin T$, by definition.*

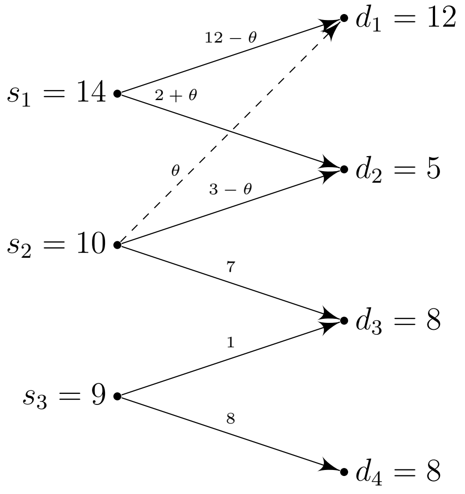
- *By complementary slackness, $\lambda_i + \mu_j = c_{ij}$ for $(i, j) \in T$.*

- *Setting $\lambda_1 = 0$, we have $n+m-1$ linear equations for $n+m-1$ variables (number of edges in the spanning tree on $n+m$ vertices), the process gives a unique solution.*

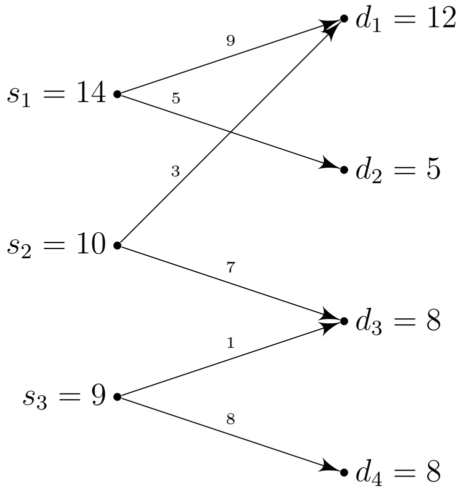
As in every implementation of the simplex, once we have a basic feasible solution, we can pivot to another one with the following rules

- If $c_{ij} \geq \lambda_i + \mu_j$ for all $(i,j) \notin T$, then the flow is optimal.
- Otherwise, $c_{ij} < \lambda_i + \mu_j$ for some $(i,j) \notin T$. This edge and those in $T$ form a unique cycle.
- In the case of non-degeneracy, we have $x_{i'j'} > 0$ on the edges of $T$ so we can increase $x_{ij}$ while keeping the flow feasible, decreasing the value of the Lagrangian until $x_{i'j'} = 0$ for some $(i',j') \in T$.
- Update the dual variables and repeat.

We apply these rules in the example considered above: dual feasibility is violated for $i = 2, j = 1$: $9 = \lambda_2 + \mu_1 > c_{21} = 2$. We can therefore increase $x_{21}$ by $\theta > 0$, adjusting on the cycle along the four top edges for feasibility, and diminish the total cost.



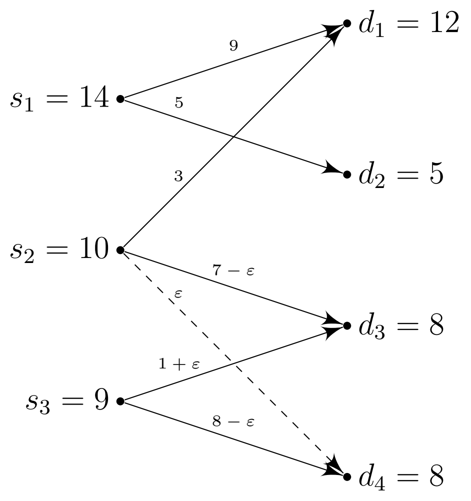
This is done until $x_{22} = 0$, for $\theta = 3$, and we obtain the following flow



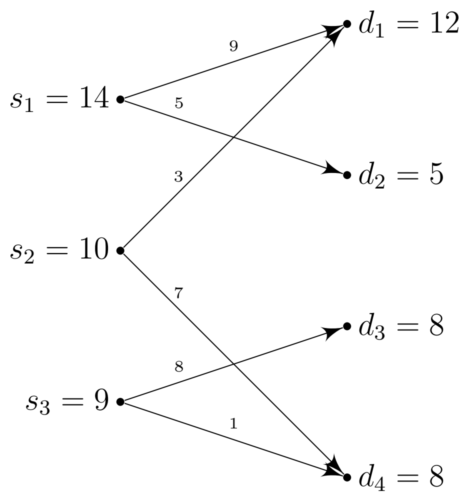
The associated tableau, with updated dual variables is

$$\begin{array}{c|c|c|c|c|c}
 & 5 & 3 & 7 & 9 & \\ \hline
0 & 9\ \boxed{5} & 5\ \boxed{3} & \overset{7}{\quad\boxed{4}} & \overset{9}{\quad\boxed{6}} & 14 \\ \hline
-3 & 3\ \boxed{2} & \overset{0}{\quad\boxed{7}} & 7\ \boxed{4} & \overset{6}{\quad\boxed{1}} & 10 \\ \hline
-5 & \overset{0}{\quad\boxed{5}} & \overset{-2}{\quad\boxed{6}} & 1\ \boxed{2} & 8\ \boxed{4} & 9 \\ \hline
 & 12 & 5 & 8 & 8 &
\end{array}$$

The optimality conditions are again violated for $c_{24} = 1 < 6 = \lambda_2 + \mu_4$, and we can increase $x_{24}$ by $\varepsilon$ while keeping the flow feasible along the newly formed cycle, obtaining the following flow



We increase until $x_{24} = \varepsilon = 7$, when $x_{23} = 0$, with the following solution



The corresponding tableau with updated variables is given by

|  | 5 | 3 | 2 | 4 |  |
|---|---|---|---|---|---|
| 0 | 9 $\boxed{5}$ | 5 $\boxed{3}$ | $\overset{2}{\boxed{4}}$ | $\overset{4}{\boxed{6}}$ | 14 |
| $-3$ | 3 $\boxed{2}$ | $\overset{0}{\boxed{7}}$ | $\overset{-1}{\boxed{4}}$ | 7 $\boxed{1}$ | 10 |
| 0 | $\overset{5}{\boxed{5}}$ | $\overset{3}{\boxed{6}}$ | 8 $\boxed{2}$ | 1 $\boxed{4}$ | 9 |
|  | 12 | 5 | 8 | 8 |  |

We now have $\lambda_i + \mu_j \leq c_{ij}$ for all $i, j$, so the solution is optimal.

**Optimisation** **Part IB - Easter 2019**

# Lecture 11: Maximum flow and minimum cut

Lecturer: Quentin Berthet

4.4. *The maximum flow problem.* We consider in this lecture a particular kind of flow problem, with a single source and sink, and where the objective is to have the maximal amount of flow between the source and the sink, under capacity constraints.

Definition 4.4. The *maximum flow problem* is the linear program

$$\begin{aligned}
\max \quad & \delta \\
s.t. \quad & \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = 0\,, \text{for all } 1 < i < n \\
& \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \delta\,, \text{for } i = 1 \\
& \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = -\delta\,, \text{for } i = n \\
& 0 \le x_{ij} \le C_{ij}\,.
\end{aligned}$$

Remark.

- The constraints describe a total flow of $\delta$, flowing from 1 to $n$. The first constraint line describes the flow constraint for all nodes except the source and the sink: what comes in must come out. The second constraint describes that a total flow of $\delta$ flows out of 1, and the third one that a total flow of $\delta$ flows into $n$. The final constraint is that on any directed edge $(i,j)$, the nonnegative flow $x_{ij}$ is bounded by $C_{ij}$.

- The problem is a special case of the minimum cost problem, by taking a cost equal to 0 for all $(i,j) \in E$, and add an edge $(n,1)$ with cost $-1$ and infinite capacity on this edge. The total cost is minimized by maximizing the flow along $(n,1)$, i.e. the whole network.

4.4.1. *Relationship between cuts and flows.* There is a very simple way to give upper bounds for maximum flow problems, through the understanding of “bottlenecks”, or cuts.

Definition 4.5. A *cut* of a graph $G$ is the partition of $V$ in two sets $(S, V \setminus S)$.

The *capacity* of a cut is given by

$$C(S) = \sum_{(i,j)\in S\times(V\setminus S)} C_{ij}\,.$$

Intuitively, for any partition, the amount of flow that can go from $S$ to $V \setminus S$ cannot be greater than this capacity. This can be made formal in the following result.

PROPOSITION 4.4. *Let $x$ be a feasible flow with objective $\delta$. For any cut $(S, V\setminus S)$ such that $1 \in S$ and $n \in V \setminus S$, we have $\delta \le C(S)$.*

PROOF. Let $X, Y \subseteq V$, and $f_x(X,Y) = \sum_{(i,j)\in E\cap(X\times Y)} x_{ij}$. We denote by $E\cap(X\times Y)$ the edges that start in $X$ and end in $Y$. Note that $X$ and $Y$ need not be disjoint. For any $S$ such that $1 \in S$ and $n \in V \setminus S$, we have by summing the flow constraints in $S$

$$\begin{aligned}
\delta &= \sum_{i\in S}\Big(\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji}\Big)\\
&= f_x(S,V) - f_x(V,S)\\
&= f_x(S,S) + f_x(S,V\setminus S) - f_x(V\setminus S,S) - f_x(S,S)\\
&\le f_x(S,V\setminus S) \le C(S)\,.
\end{aligned}$$

□

As a consequence, it seems that finding the partition separating 1 and $n$ with the smallest possible cut can give a very good upper bound on the maximum flow. Again, this intuition is confirmed by the following result

THEOREM 4.3 (Max-flow Min-cut). *Let $\delta^*$ be the optimal solution of the maximum flow problem. Then, we have*

$$\delta^* = \min\{C(S) \,:\, S\subseteq V\,, 1\in S\,, n\in V\setminus S\}\,.$$

PROOF. The idea is to construct a cut with capacity $C(S)$. A *path* $v_0, \dots, v_k$ is a sequence of vertices where every vertex in the sequence is connected to the following one in one direction or another. It is called an *augmenting path* if $x_{v_{i-1}v_i} < C_{v_{i-1}v_i}$ or $x_{v_iv_{i-1}} > 0$ for every $i = 1 \dots k$. If there exists an augmenting path from $i$ to $n$, the flow is not optimal: indeed, the variables can be slightly modified to add an arbitrarily small amount of flow from $i$ to $n$, as none of the constraints along this path are saturated.

Assume that $x$ is optimal and that $S = \{1\}\cup\{i \in V \,:\, \exists\,\text{augmenting path from 1 to } i\}$. By the point proved above, since $x$ is optimal, $n \in V\setminus S$. As a consequence, we have that

$$\delta = f_x(S,V\setminus S) - f_x(V\setminus S,S) = f_x(S,V\setminus S) = C(S)\,.$$

- The first equality derives from previous computations.

- The second equality holds because $x_{ij} = 0$ for every $(i, j) \in E \cap (V \setminus S \times S)$.

- The third equality holds because $x_{ij} = C_{ij}$ for every $(i, j) \in E \cap (S \times V \setminus S)$.

Indeed, if the second and third equality do not hold, $j$ can be added to the augmenting path, and does not belong to $S$. □

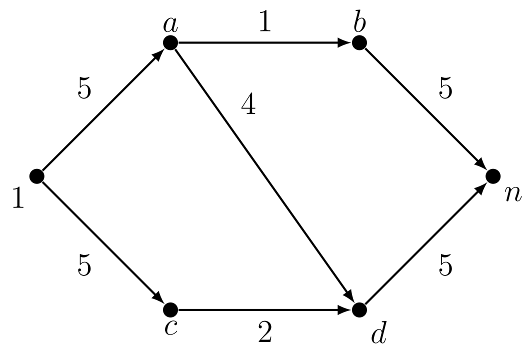
4.4.2. *The Ford-Fulkerson algorithm.* There is an easy-to-implement algorithm to find an optimal solution to the maximum flow problem, based on the notion of augmenting path introduced above.

Definition 4.6 (Ford-Fulkerson algorithm).

1. Start with a feasible flow, (often $x = 0$).
2. Find an augmenting path from 1 to $n$
   - If there is none, STOP. By the proof of Theorem 4.3, the flow is optimal.
   - If there is one, increase (push) the flow while all constraints are satisfied.
3. Go to step 2.

In practice, augmenting paths can be found and tested "by eye", as can be seen in the following example.

Example 4.2. Consider the following graph, with capacities denoted above the directed edges.



Starting with a flow $x = 0$ everywhere, we apply the algorithm along the following augmenting paths from 1 to $n$

- Along path $1, a, b, n$: push 1 unit of flow (i.e. until edge $a$ to $b$ is at capacity).
- Along path $1, a, d, n$: push 4 unit of flow (i.e. until edge $a$ to $d$ is at capacity).
- Along path $1, c, d, n$: push 1 unit of flow (i.e. until edge $d$ to $n$ is at capacity).

There is afterwards no augmenting path, the flow is optimal. This can be confirmed by considering the cut of this graph into $\{1, a, c, d\}$ and $\{b, n\}$: the capacity of this cut is $5 + 1 = 6$, the sum of the capacities of the edges crossing the cut ($a$ to $b$ and $d$ to $n$), it is equal to the amount of flow in the proposed solution.

Remark.

- As described in the example, the max-flow min-cut theorem allows to confirm the optimality of a cut, by finding a cut whose capacity matches the proposed flow.

- Another interesting property that can be obtained from this theorem is the integrality or rationality of optimal solutions: If the capacities are integer and we start from an integral solution (e.g. $x = 0$), the algorithm maintains at all times the integrality of the solution. For a bounded program, it is therefore guaranteed to terminate. When it terminates, it gives an optimal flow with integer entries. The same is true for rational capacities.

## 4.5. *The bipartite matching problem.*

Definition 4.7. A *matching* of a graph $(V, E)$ is a subset of edges with no shared vertices. A matching $M$ is a *perfect matching* if it covers every vertex, i.e. if $|M| = |V|/2$.

Definition 4.8. A graph is $k$-regular if every vertex has degree $k$.

An interesting property about regular graphs can be proved using what we know about flow problems, even though it is seemingly disconnected from this topic.

Theorem 4.4. *Every $k$-regular bipartite graph has a perfect matching.*

Proof. We create another graph, whose vertices are composed of: the original vertices, partitioned in two sets of $L$ and $R$ (for left and right), a source $s$, and a sink $t$. We note by $n$ the number of original vertices $|V|$, and note that since $G$ is $k$-regular and bipartite, $|L| = |R| = n/2$.

The source $s$ is connected to all vertices of $L$ by edges of capacity 1. The sink $t$ is connected to all vertices of $R$ by edges of capacity 1. On the original edges between $L$ and $R$, the capacity is set to $+\infty$.

We consider the maximum flow problem on this graph, from $s$ to $t$. A flow with 1 in every new edge and $1/k$ in every old edge is feasible: it satisfies the flow equations at each vertex, and satisfies the capacity constraints. The total value of this flow is $n/2 = |V|/2 = |L| = |R|$.

Furthermore, cutting between the source and the rest of the newly constructed graph, the capacity of this cut is also $n/2 = |L|$. As a consequence, the optimal value of a cut is $n/2$.

If we start the Ford-Fulkerson at $x = 0$, by the remark above, the algorithm terminates at an optimal integer solution with value $n/2$. As a consequence, every vertex in $L$ has an incoming flow of 1, and an integral outgoing flow of 1, to only one vertex in $R$. Similarly, every vertex in $R$ has an outgoing flow of 1, and an integral incoming flow of 1, from only one vertex in $L$. The edges on which there is a flow therefore form a perfect matching. □

More generally, it is possible to characterize all the bipartite graphs that have perfect matchings. It can also be proved using the max-flow min-cut theorem. It is sometimes referred to as Hall's marriage theorem.

THEOREM 4.5 (Hall's theorem). *A bipartite graph with $V = L \cup R$ and $|L| = |R|$ has a perfect matching if and only if $|N(X)| \geq |X|$ for every subset $X \subseteq L$, where*

$$N(X) = \{j \in R \,:\, i \in X\,,\, (i,j) \in E\}\,.$$

**Optimisation** **Part IB - Easter 2019**

# Lecture 12: Noncooperative games

Lecturer: Quentin Berthet

**5. Game theory.** Another application of linear programming is the analysis of games: situations where several players make actions in order to maximize some reward. In this course, we will consider cases where there are two players and each one has a finite set of actions.

5.1. *Notions and definitions.*

Definition 5.1. In a two-player game, with $m$ actions for player 1 and $n$ actions for player 2, we define the *payoff matrices* as $P, Q \in \mathbf{R}^{m\times n}$ such that, when player 1 plays action $i$ and player 2 action $j$

- $P_{ij}$ = payoff of player 1.
- $Q_{ij}$ = payoff of player 2.

These matrices contain all the information necessary to the analysis of a game. We consider that players can have a strategy where they chose randomly one of the actions, with some probabilities.

Definition 5.2. The set of possible *strategies* are sets $\mathcal{X}$ and $\mathcal{Y}$ defined by

$$\mathcal{X} = \Big\{x \in \mathbf{R}^m_{\geq 0} \ : \ \sum_{i=1}^m x_i = 1\Big\}$$
$$\mathcal{Y} = \Big\{y \in \mathbf{R}^n_{\geq 0} \ : \ \sum_{i=1}^n y_i = 1\Big\}\,.$$

A *pure strategy* is one that choses one of the actions with probability one. It is a vector of the canonical basis, with all entries equal to 0 except the $i$-th entry, equal to 1.

Both players chose their actions, potentially at random, without cooperating or communicating. The choices are independent, and they have no knowledge of what the other player will do.

PROPOSITION 5.1. *For strategies $x$ and $y$, the profile $(x,y) \in \mathcal{X} \times \mathcal{Y}$ has expected payoff $x^\top Py$ for the first, or* row *player. It is sometimes written $p(x,y)$.*

PROOF.

$$\begin{aligned} x^\top Py &= \sum_{i=1}^m x_i(Py)_i = \sum_{i=1}^m x_i \sum_{j=1}^n P_{ij}y_j = \sum_{i,j} P_{ij}x_iy_j \\ &= \sum_{i,j} P_{ij} \times \mathbf{P}(\text{player 1 plays } i, \text{ player 2 plays } j) = \mathbf{E}[\text{payoff}] \end{aligned}$$

□

EXAMPLE 5.1. We often represent games with both payoff matrices in the same table with, at entry $i,j$ the information $(P_{ij}, Q_{i,j})$. As an example, in the prisoner's dilemna game, the two players (suspects interrogated by the authorities) can chose to be silent ($S$) or to talk ($T$). This is written out in the following table.

| | S | T |
|---|---|---|
| S | (2,2) | (0,3) |
| T | (3,0) | (1,1) |

There are three possible outcomes:

- Both suspects are silent, they stay a few weeks in detention (payoff = 2 for both players).
- Both suspects talk, and both get five years of detention (payoff = 1 for both of them, worst than in the first case).
- One talks and gets no detention (payoff = 3) and the other one stays silent and gets ten years because of his former accomplice's testimony (payoff = 0).

We note that in this game, whatever the other player does, it is always bertter for a player to talk.

DEFINITION 5.3. For two strategies $x, x'$ of the row player, $x$ is said to *strictly dominate* $x'$ if, for every $y \in \mathcal{Y}$

$$p(x,y) > p(x',y) .$$

DEFINITION 5.4. If $x, y$ both strictly dominate all other strategies, $(x,y)$ is called a *dominant strategy equilibrium.*

EXAMPLE 5.2. In the game of *Chicken*, two players are going in opposite directions on a one-way street. They can either *chicken out* ($C$) and yield, letting the other one go. They can also decide to *dare* ($D$) and go ahead without stopping. The outcomes are given as follows.

| | C | D |
|---|---|---|
| C | (2,2) | (1,3) |
| D | (3,1) | (0,0) |

There are three possible outcomes:

- If both players chicken, they both lose some time, but not more than the other driver, and they avoid an accident (payoff = 2). It is the second-best outcome for each of them.
- If both players, they both die (payoff = 0). It is the worst outcome for both of them.
- If one dares (payoff = 3) and the other chickens (payoff = 1), the first player has the best outcome, and the other player loses more time.

We note that in this game, there is no dominant strategy. It is however possible to design a safe strategy: to chose a strategy for which the worst payoff (over all possible actions of the opposite player) is as high as possible. If the first player dares the worst possible payoff is 0 (if the other one dares as well), while if the first player chickens, the worst payoff is 1 (if the other player dares). It is therefore safer to chicken out.

Definition 5.5. We call *maximin strategy* the solution to the problem

$$\min_{j\in\{1,\dots,n\}} \sum_{i=1}^{m} x_i P_{ij}\,.$$

It is a pessimistic approach consisting in maximizing (in $x$) the minimum value of $p(x, a_j)$ (over $a_j$). It can be written as the linear program

$$\begin{aligned} \max \quad & v \\ s.t. \quad & \sum_{i=1}^{m} x_i P_{ij} \geq v\,, \text{ for all } 1 \leq j \leq n \\ & \sum_{i=1}^{m} x_i = 1 \\ & x \geq 0\,. \end{aligned}$$

As noted above, in the game of chicken, the solution to this problem is a pure action. Note that it is not in general the best choice for every action of the other player, in hindsight.

Definition 5.6. A strategy $x \in \mathcal{X}$ is the *best response* to $y \in \mathcal{Y}$ if for all $x' \in \mathcal{X}$.

$$p(x, y) \geq p(x', y)\,.$$

Observe the difference with the notion of domination in Definition 5.3: the quantifiers are important.

Definition 5.7. If a profile $(x,y) \in \mathcal{X} \times \mathcal{Y}$ satisfies

- $x$ is a best response to $y$
- $y$ is a best response to $x$,

the profile is called a *Nash equilibrium.*

Theorem 5.1 (Nash, 1951). *Every bimatrix game has an equilibrium. In the game of chicken, it is $(C,D)$ and $(D,C)$.*

5.2. *Zero-sum games.*

Definition 5.8. A game where $Q = -P$ (i.e. $P_{ij} = -Q_{ij}$) is called a *zero-sum game.*

Theorem 5.2 (von Neuman, 1928). *Let $P \in \mathbf{R}^{m\times n}$ in a zero-sum game, $\mathcal{X}$ and $\mathcal{Y}$ be a set of strategies. It holds that*

$$\max_{x\in\mathcal{X}} \min_{y\in\mathcal{Y}} p(x,y) = \min_{y\in\mathcal{Y}} \max_{x\in\mathcal{X}} p(x,y)\,.$$

Proof. We consider the LP formulation of the left-hand side problem in Definition 5.5 and write its Lagrangian, with slack variable $z$ for the $n$ first inequalities with associated dual variable $y$, and dual variable $w$ for the simplex constraint.

$$\begin{aligned} L(v,x,z,w,y) &= v + \sum_{j=1}^{n} y_j\Big(\sum_{i=1}^{m} x_i P_{ij} - z_j - v\Big) - w\Big(\sum_{i=1}^{m} x_i - 1\Big) \\ &= \Big(1 - \sum_{j=1}^{n} y_j\Big)v + \sum_{i=1}^{m} x_i\Big(\sum_{j=1}^{n} P_{ij}y_j - w\Big) - \sum_{j=1^n} y_j z_j + w\,. \end{aligned}$$

For fixed dual variables $w, y$, this program has a finite maximum over $v \in \mathbf{R}$ and $x \geq 0$ if and only if

$$\sum_{j=1}^{n} y_j = 1\,, \quad \sum_{j=1}^{n} P_{ij}y_j \leq w \quad \text{for all } 1 \leq i \leq m, \quad y \geq 0\,.$$

The dual problem is therefore

$$\begin{aligned} \min \quad & w \\ s.t. \quad & \sum_{j=1}^{n} P_{ij}y_j \leq w\,, \text{ for all } 1 \leq i \leq m \\ & \sum_{j=1}^{n} y_j = 1 \\ & y \geq 0\,. \end{aligned}$$

This dual problem is the right-hand side problem. By strong duality, both sides are therefore equal. □

Every zero-sum game has a Nash equilibrium, characterized by the result above.

Theorem 5.3. *In a zero-sum game, a pair of strategies $(x, y)$ is a Nash equilibrium if and only if*

$$\begin{aligned}\min_{y' \in \mathcal{Y}} p(x, y') &= \max_{x' \in \mathcal{X}} \min_{y' \in \mathcal{Y}} p(x', y') \\ \max_{x' \in \mathcal{X}} p(x', y) &= \min_{y' \in \mathcal{Y}} \max_{x' \in \mathcal{X}} p(x', y') \, .\end{aligned}$$