Lecture 1 - Introduction to Optimization

Optimization and Approximation - ENS M1

Nicolas Bousquet

Lectures:	Tuesday, 10.15-12.15.
Instructor:	Nicolas Bousquet.
Exercise sessions:	Friday, 13.30-15.30
Intructors:	Florent Brehard et Luc Pellissier
Website:	`pagesperso.g-scop.grenoble-inp.fr/~bousquen/OA/`
Email:	`nicolas.bousquet@grenoble-inp.fr`
Last update:	September 18, 2017

1 Introduction to optimization problems.

1.1 Optimization problems.

All along these lectures we will consider optimization problems. An *optimization problem* is a problem of the form

$$\text{maximize } f(x) \text{ subject to } x \in F$$

where

- $F \subseteq \mathbb{R}^n$ is called the *feasible region*.
- $f : F \to \mathbb{R}$ is called the *objective function*.
- x is the vector of *decision variables*.

Optimization problems cover a large scope of mathematics (from statistics to discrete mathematics including financial mathematics...). However all the optimization problems cannot be solved in the same way and rather different approaches are necessary in order to tackle them. In particular, during these lectures we will consider two kinds of problems: linear problems and non-linear problems.

Note that we assume the input of the function f is in $\mathbb{R}$. Nevertheless, we can imagine more general functions f with input values in various other sets or fields. For instance, the set F can be included in a boolean space, or in the set of integers, or it could be the complex space. In all these cases, decision variables can easily be "represented" as real numbers. During these lectures, we will concentrate on the case where F is either $\mathbb{R}$ or $\mathbb{N}$.

One can also note that if we replace the function f by its opposite function $(-f)$, the maximization problem becomes a minimization problem. Indeed maximizing $f(x)$ subject to $x \in F$ obviously corresponds to minimizing $-f(x)$ subject to $x \in F$ in the sense that the set of decision variables x reaching the maximum in the first case is precisely the set of vectors of decision variables reaching the minimum in the second one (and the optimal value of the minimization problem is the opposite of the optimal value of the maximization problem). In mathematical programming, results are usually stated from a maximization point of view, but the above remark ensures that any maximization result can be easily transformed into a minimization result.

When the feasible region is empty, we say that the optimization problem is *unfeasible*. When a vector of decision variables x is in F, we say that x is *feasible*. There is a *feasible solution* if there exists a feasible vector x, *i.e.* F is not empty. A couple $(x, f(x))$ is an *optimal solution* of a maximization problem if $x \in F$ and $f(x)$ reaches the maximum value. By abuse of notations, we will often say that x is an optimal

solution. When we consider maximization problems, we often say that $x \in F$ is a *maximizer* (resp. *minimizer*) if $f(x)$ is maximum (resp. minimum). If x is an optimal solution then $f(x)$ is the *optimal value*.

However the maximum may not be reached. We then say that the optimal value is *unbounded* if for every real number A, there exists a solution $(x, f(x))$ such that $|f(x)| \geq A$. Note that if the optimal value is unbounded, we say that the optimal value is equal to $+\infty$ for maximization problems and $-\infty$ for minimization problems. (we will assume that the maximum is reached when the optimal value is bounded, which is essentially correct in general and totally correct when F is bounded or when we restrict to Linear Programming).

An optimization problem is *unconstrained* if $F = \mathbb{R}^n$. Usually unconstrained optimization problems are quite different from constraint ones but are not necessarily simpler. Sometimes, even determining if the problem is feasible is not so simple (we will see that a trick will be necessary in order to initialize the solutions of the simplex algorithm).

1.2 Optimization

Optimization has two main branches. The first one is *discrete optimization* where the feasible regions are finite (in particular, it will be the case when the decision variables lie on a $\mathbb{N}$ or any discrete sets) and *continuous optimization* where feasible regions are continuous (understand that they are somehow isomorphic to balls in real spaces). Surprisingly, we will see that linear continuous optimization can be much simpler (in terms of computational complexity) than discrete optimization when we want to optimize over linear functions.

1.3 Prototype of optimization problem.

A general *prototype* for an optimization problem is

$$\text{maximize}_{\ x \in \mathbb{R}^n} f(x) \text{ subject to } \begin{cases} f_i(x) \leq b_i \text{ for every } i \leq m. \\ x_j \geq 0 \text{ for every } j \leq n. \end{cases}$$

The function f is the *objective function* while the functions f_i's (and the inequalities $x_j \geq 0$) are the *constraints* of the problem. The constraints $x_j \geq 0$ are the *non-negativity constraints*. For every i, the function f_i is a function from $\mathbb{R}^n$ to $\mathbb{R}$. Given this set of constraints, we can define the feasible region as:

$$F = \{x \in \mathbb{R}^n \text{ such that } \forall j \leq n, x_j \geq 0 \text{ and } \forall i \leq m, f_i(x) \leq b_i\}$$

All along these notes, all the vectors x will be column vectors and we will denote by x^T the transpose row vector of x. [1]

The non-negativity constraints naturally arise in almost all the real-world problems. Indeed, the decision variable x_j usually represents the number of produced items or the proportion of a given ingredient, and thus obviously cannot be negative.

2 Linear Programming and Integer Linear Programming.

2.1 Introduction to LP

Assume that we have an optimization problem of the above shape. If the function f and all the functions $f_1, \ldots, f_m$ are linear, the optimization problem is a *Linear Programming* problem (or LP for short). In other words a LP consists in *optimizing a linear function under some linear constraints.*

If at least one of these functions is not linear, it is a *non-linear program* problem. When the constraint f_i is linear, the constraint f_i can be rephrased as follows: $\sum_{i=1}^{n} a_{i,j} \cdot x_i \leq b_i$ where the $a_{i,j}$ are real numbers. So the constraint f_i can be seen as the product of a row vector of size n (the vector $(a_{i,1}, \ldots, a_{i,n})$) with the vector x. Note that non-negativity constraints are also linear constraints and thus can similarly be transformed.

[1] For a pair of vector x, b, by $x \leq b$ we mean that each coefficient of x is less than or equal to the corresponding coefficient of b.

Let us denote by A the matrix whose row i is the vector $(a_{i,1}, \ldots, a_{i,n})$. Let b^T be the vector $(b_1, \ldots, b_m)$. The vector b is called the *vector of constraints*. We can similarly create a vector c, called the *objective vector*, of $\mathbb{R}^n$ whose coefficient i is the coefficient of x_i in the objective function. Note that n is not necessarily equal to m. So finally, the LP can be rephrased in the following way:

$$\begin{aligned} &\text{maximize } _{x\in\mathbb{R}^n} c^T x \\ &\text{subject to} \\ &Ax \leq b \text{ and} \\ &x_j \geq 0 \text{ for every } j \end{aligned}$$

In the following, b will always denote the vector of constraints and c will denote the objective vector.

Integer Linear Programming (ILP) is the important special case where the functions f_i and f are linear and the vector of decision variables x is additionally constrained to lie in $\mathbb{Z}^n$. In this case, the domain of our prototype model changes from $\mathbb{R}^n$ to $\mathbb{Z}^n$:

$$\text{maximize } _{x\in\mathbb{Z}^n} c^T x \text{ subject to } x \in \mathbb{R}^n \text{ such that } Ax \leq b \text{ and } x_j \geq 0 \text{ for every } j$$

Note that obviously, the maximum of a linear program and the maximum of the "corresponding" integer linear program are not necessarily the same. Consider for instance the following simple linear program

$$\text{maximize } _{(x_1,x_2,x_3)\in\mathbb{R}^3}(x_1 + x_2 + x_3) \text{ subject to } \begin{cases} x_1 + x_2 \leq 1 \\ x_1 + x_3 \leq 1 \\ x_2 + x_3 \leq 1 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

By setting $x_1 = x_2 = x_3 = \frac{1}{2}$, we obtain a vector x which is feasible. And the value of this solution is $\frac{3}{2}$. So the optimal value of this LP is at least $\frac{3}{2}$. On the contrary, if we assume x_1, x_2, x_3 to be non-negative integers, then at most one of them can be positive and the constraints ensure that its value is at most one. So the optimal value of the ILP cannot necessarily reach the value of its corresponding LP. We will see further in these lectures that it is even worse.

If some of the x_i 's are integers and others are real numbers, it is a *mixed integer programming* (MIP) problem and the domain defines which variables are real, and which are integer. If we allow integer, non-integer variables, and nonlinear constraint functions and objective function, then we have the class of mixed integer nonlinear program (MINLP) which has only recently become an active area attracting leading researchers.

2.2 Modelization of a production problem

All the notions introduced in this section will be illustrated on the following simple example of "Dovetail". This example will be used several times all along these lectures.

The company 'Dovetail' produces two kinds of matches: long ones and short ones. The company makes a profit of 300\$ for every $100,000$ boxes of long matches, and 200\$ for every $100,000$ boxes of short matches. The company has one machine that can produce both long and short matches, with a total of at most 9 ($\times 100,000$) boxes per year. For the production of matches the company needs wood and boxes: 3 cubic meters of wood are needed for one box of long matches, and 1 cubic meter of wood is needed for one box of short matches. The company has 18 cubic meters of wood available for the next year. Moreover, 'Dovetail' has 7 ($\times 100,000$) boxes for long matches, and 6 ($\times 100,000$) for short matches available at its production site. The company wants to maximize its profit in the next year. It is assumed that 'Dovetail' can sell any amount it produces. To write this production problem in mathematical terms, we first introduce the decision variables x_1 and x_2 :

- x_1 is the number ($\times 100,000$) of boxes of long matches produced during the next year.
- x_2 is the number ($\times 100,000$) of boxes of short matches produced during the next year.

The company makes a profit of $\$300$ each time they produce $100,000$ boxes of long matches. So their profit when they produce $x_1 \cdot 100,000$ boxes of long matches is $\$300 \cdot x_1$. Similarly, the company makes a profit of $\$200$ each time they produce $100,000$ boxes of long matches. So their profit when they produce $x_2 \cdot 100,000$ boxes of long matches is $\$300 \cdot x_2$. Since Dovetail wants to maximize the profit during the upcoming year, the objective function is the following:

$$\max_{(x_1,x_2)\in\mathbb{R}^2} 300 \cdot x_1 + 200 \cdot x_2.$$

Note that maximizing a function f precisely corresponds to maximizing the function λf for every positive λ. So, for simplicity, we consider the objective function

$$d = \max_{(x_1,x_2)\in\mathbb{R}^2} 3 \cdot x_1 + 2 \cdot x_2.$$

And we will keep in mind that if the optimal value is d then the maximum profits of Dovetail is $\$100d$.

Let us now consider the constraints. The statement mentioned several constraints: the number of boxes of long and short matches, the quantity of wood, the productivity of the machine. First, the company has 18 cubic meters of wood available for the next year. The production of $100,000$ long matches boxes needs 3 cubic meters of wood and the production of $100,000$ boxes of short matches needs 1 cubic meter of wood. Since only 18 cubic meters of wood are available, we have the constraint:

$$3 \cdot x_1 + x_2 \leq 18. \tag{1}$$

Moreover, since the machine can produce at most 9 ($\times 100,000$) boxes of matches a year, we have the following constraint (due to the machine) which is:

$$x_1 + x_2 \leq 9. \tag{2}$$

Since there are only 7 ($\times 100,000$) boxes of long matches available on the production site, we cannot produce more 7 ($\times 100,000$) long matches boxes. So:

$$x_1 \leq 7. \tag{3}$$

Similarly we have

$$x_2 \leq 6. \tag{4}$$

Finally, the amount of produced boxes is non negative so both x_1 and x_2 are non-negative.

$$x_1, x_2 \geq 0$$

Note that the function we want to maximize and all the constraint functions are linear. Moreover we consider that, since the amount of boxes we produce is huge, a real solution will provide a good approximation of an optimal integer solution. Let us now describe how we can solve this linear program.

Linear vs Integer Linear Programming for production. When we solve a LP, we usually solve it using a solver where variables are real values. There are two reasons for that: the first one is that the algorithm are simpler and more efficient. The second is that, in most of the cases, we are optimizing on variables whose values will be thousands or millions. Thus the rounding approximation will not be a problem for production. In other words, we do not lose the quality of the production by rounding our solution.

In many other cases, we will see that the gap between the optimal solution using real variables and the optimal solution using integer variables will be very different. This gap between optimal real solution and integral solution is called the integrality gap.

2.3 Modelization of Discrete Mathematics problems

All along the course, we will use the notion of graph (or network). A *graph* $G = (V, E)$ is a pair where V is a set of "points" (not in a geometric but a combinatorial way) called *vertices*, and E is a set of *edges* which is a subset of pairs of vertices. Graphs are standard tools to represent many discrete objects that will be introduced all along the course. Many optimization problems have been studied on graphs.

Vertex Cover. One of them is the VERTEX COVER problem. A *vertex cover* is a subset of vertices such that each edge contains at least one of its enpoints in the vertex cover. The vertex cover problem consists in finding a vertex cover of minimum size.

As many optimization problems on graphs, this problem can be modelized as an ILP in the following way.

$$\min \sum_{v \in V} x_v$$

subject to

$$x_u + x_v \geq 1 \text{ for all } e = (u, v) \in E$$

$$x_u \in \{0, 1\}$$

Note that the VERTEX COVER problem is an NP-hard problem. Thus optimizing over an ILP is NP-hard.

3 Approximation Algorithms

3.1 General definitions

Often, there is no fast efficient algorithm to solve an optimization problem Π. We have a choice to make between efficiency and running time. An *approximation algorithm* is an algorithm giving a solution that "approximates" the best possible solution of Π. All along this course we will only consider approximation algorithm running in polynomial time.

An algorithm is an α-approximation algorithm if the solution output by the algorithm is, in the worst case, of size $\alpha \cdot OPT$ where OPT is the size of the optimal solution. α is the *approximation factor* (or ratio) of the algorithm. If α is a constant, the algorithm is a *constant approximation algorithm*.

3.2 An example: 2-approximation for Vertex Cover

Here we give a very simple approximation algorithm to illustrate this notion.

Procedure 1 Approximation algorithm for Vertex Cover

$X = \emptyset$
$H = G$
while there is an edge in H **do**
  Let $e = (u, v)$ be an edge of H.
  $H := H \setminus \{u, v\}$ — Delete u, v and the edges incident to them.
  $X := X \cup \{u, v\}$
end while
Return X

Lemma 1. *Algorithm 1 returns a Vertex Cover.*

Proof. Let us prove that for every edge, at least one of the endpoints is selected in X. Either the edge is selected at some point of the algorithm and the conclusion immediately holds. Or the edge $f = (a, b)$ is not selected by the algorithm. Since f exists at the beginning of the algorithm in H (since $H = G$) and does not exist at the end (since H is edge-less), there is a step i such that f is in H_i, the graph at the beginning of step i and not in H_{i+1}, the graph at the end of step i. Let $g = (x, y)$ be the edge selected at step i. Since the only edges deleted between H_i and H_{i+1} are edges incident to x or y, the edge f must contain either x or y. Since both vertices are added in X, the conclusion holds. □

Lemma 2. *The set X returned by Algorithm 1 has size at most $2OPT$.*

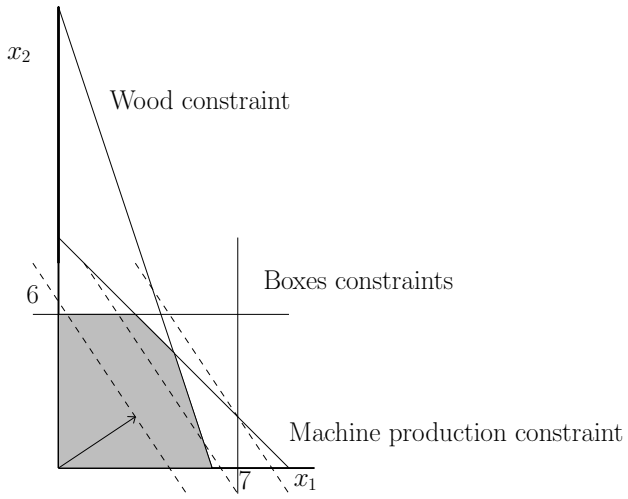

Figure 1: Geometrical representation of the Dovetail LP. In dotted, lines with the same value for the objective function. The gray part is the feasible region.

Proof. Let $e_1, e_2, \cdots, e_\ell$ be the set of edges selected during this procedure. Note that they are vertex disjoint since the endpoints of the edges selected during the algorithm are deleted during the procedure. So any vertex cover must select at least one endpoints of all of these edges. In particular, its size is at least ℓ, which completes the proof. □

Note that the proof of Lemma 2 essentially ensures that the size of a vertex cover is at most twice the size of a maximum matching... It will important within a few weeks !

4 Geometric representations

Before explaining how we can find a solution using a graphical argument, let us first explain how a linear program can be represented on a geometrical space. Since there are two decision variables x_1 and x_2, our feasible region is a subset of $\mathbb{R}^2$. We will represent the LP in a 2-dimensional space (the plane) and we will try to determine the feasible region. All along this part, you are refered to Figure 1 for an illustration.

Since both x_1 and x_2 are non-negative, it means that three quarter of the plane is not in the feasible region: the feasible region is included in the "north-east" quarter of the plane: indeed if $x_1 < 0$ or $x_2 < 0$, then the point $x = (x_1, x_2)$ is not feasible. More precisely, when we consider the constraint $x_1 \leq 0$, then a half plane is not feasible. Actually this fact is much more general: every linear constraint (which is an inequality) eliminate one half of the space from the set of feasible solutions. Moreover the constraints $x_1 \leq 7$ and $x_2 \leq 6$ ensure that the set of feasible solutions is in the axis-parallel rectangle with corners $(0, 0)$ and $(7, 6)$.

The two other constraints are slightly more involved (but don't worry, not a lot, the constraints are still linear). We have $3 \cdot x_1 + x_2 \leq 18$. The equality is reached when $3x_1 + x_2 = 18 \Leftrightarrow x_2 = 18 - 3x_1$. So the set of tight points for this constraint can be represented using a line passing through the point $(0, 18)$ with a slope of -3. The inequality is satisfied for every x_2 which is above this line, which, again, defines a half-plane. Similarly, the constraint $x_1 + x_2 \leq 9$ can be represented using the line $x_2 = 9 - x_1$ passing through $(0, 9)$ with a slope of -1.

Finally the set of feasible solution is the intersection of all these half-spaces (see Figure 1). Note that no point of the feasible region is tight for the constraint $x_1 \leq 7$. The constraint is called *redundant* and any redundant constraint can be deleted from the set of constraints (at least at this point, we will nevertheless see later that it can be useful to keep this constraint if we want to study the behavior of our system when we modify the constraints).

More details on the geometrical representation of LP and reminders (and new stuff) concerning linear algebra and polyhedron will follow in the upcoming lectures.

5 Solving a LP via geometric methods

5.1 Via "moving" a line.

Imagine now that you have a feasible region F and that you want to determine the point $x \in F$ which maximizes the first coordinate. Then you will just draw a vertical line and move this line horizontally in order to find the point maximizing the first coordinate.

For more complicated maximization problems, the method is the same. Since we want to maximize $3 \cdot x_1 + 2 \cdot x_2$, we want to find an element in the feasible region which is maximum in the direction $(3, 2)$. The points of the plane which have the same value in that direction form a line (in larger dimensions, it will be a hyperplane of $\mathbb{R}^n$) perpendicular to the vector $(3, 2)$, *i.e.* a line of the form $3x - 2y = c$ where c is a constant (several such lines are represented as dotted lines in Figure 1).

So we just have to "move" the line $3x - 2y = c$ (using variations of c to considering lines parallel to the vectorial line $3x - 2y = 0$) in order to find the best solution. By doing it on our example we can show that the optimal solution is reached for the point

$$\begin{pmatrix} \frac{9}{2} \\ \frac{9}{2} \end{pmatrix}.$$

Then we just have to compute the objective function on this point in order to obtain the maximum value. So the optimal value is 22.5 and then the maximum revenue for the company is $\$2,250$. This profit is achieved if Dovetail produces $450,000$ boxes of long matches and $450,000$ boxes of short matches. Note that the optimal point is a vertex (the formal definition of vertex will come soon but vertex essentially means "corner" of the feasible region) of the feasible region which is a crucial fact in linear programming.

5.2 Via corner-walking.

Although it was obvious from the figure what the optimal solution is for the Dovetail model, computers cannot "mentally move" lines (and even you will have some trouble to do it if the number of decision variables increases: it is quite complicated to mentally or physically represent spaces of dimension 4 or more). So in order to construct an optimal solution, we will have to do something else. We will see later in this course that in fact, if a linear optimization problem has an optimal solution, then it has an optimal solution at a vertex. So it suffices to consider only vertices of the feasible region and walk from any corner to any other corner trying to improve the best solution. This suggests the following method :

- Step 1: Start at some initial vertex v.
- Step 2: If we can walk along a constraint v to a neighbor v', while increasing the objective value, then set $v := v'$, and repeat Step 2. If not, then we are done.

Again, if we use this corner-walking solution, we find the same optimal solution. The most famous algorithm which solves LP, called the *Simplex Algorithm* is based on this method.

5.3 Shadows prices.

A constraint C is *tight* at a point x if x is in the hyperplane defined by the constraint C. Solving a linear program gives much more information than the optimal value. Indeed, after the computation of the best solution, we can also make a post-analysis in order to determine how our profit can increase. In particular, we can evaluate the possible gain if we improve our equipment. This vast area is known as sensitivity analysis. We will study in details this topic later in these lectures.

Acknowledgments: These lectures are inspired from the lectures notes of Bruce Shepherd and Yori Zwols. The Dovetail example (and a few other example that will appear in the lectures) are taken from the book *"Applied Mathematical Programming"* of Bradley, Hax, Magnanti.