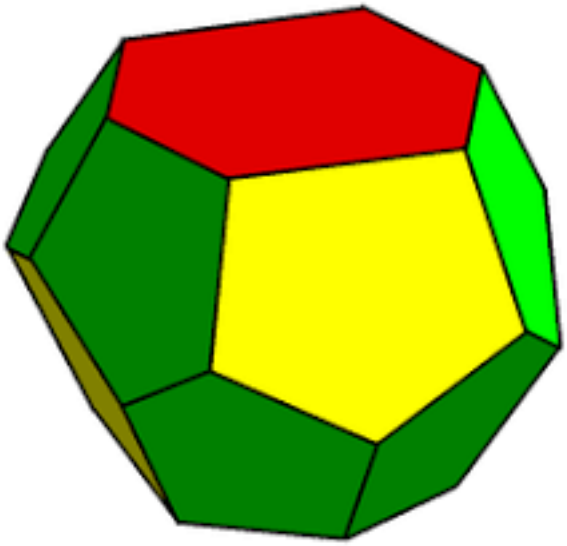
Linear Programming: An Introduction

Alex Rankine



Contents

You may be familiar with the idea of optimization as it was presented in your introductory calculus course. Given some function, we want to find the point(s) at which it is maximized or minimized; we may refer to these points as **optimal**.[1] Given a function such as $f(x) = -x^2$, we know that we find its global maximum by finding the point where the derivative vanishes.

But what if we were to impose some **constraints** on the optimal solution? For instance, what if I wanted to maximize $f(x) = -x^2$, but I also wanted the solution x to satisfy the inequality $x \geq 2$? What if I also require solution to satisfy $x \leq 10$? Our old solution $x = 0$ no longer works because it fails to satisfy these constraints!

We quickly see that these elementary methods for optimization are not enough, because they have no way of accounting for these restrictions. Problems concerned with maximizing some linear function with respect to a set of linear constraints on the solution are referred to **linear programming** problems and are the treatment of this short introduction.

1 Examples of linear programs

Linear programming problems are very much practical problems. In fact, optimization problems that are unconstrained are rather rare in practice. For instance, if we're a chocolate manufacturer and we're trying to finding the optimal quantity x^* of chocolate bars to sell, we must at least have the constraint that $x^* \geq 0$ since we certainly can't sell a negative quantity!

Before providing a formalization of the form of linear programming problems, we'll take a look at some examples to build intuition.

1.1 Alex's Apples

Alex is an apple grower. He grows red apples, green apples, and yellow apples. He sells the red apples for c_r dollars, green ones for c_g dollars, and yellow ones for c_y dollars. Let x_r, x_g, and x_y reflect the quantities of the respective apples that Alex sells. Then Alex's total revenues are described by the function

$$f(x_r, x_g, x_y) = c_r x_r + c_g x_g + c_y x_y$$

which he would like to maximize.

[1] You may also be familiar with the distinction between global and local maxima/minima. In the linear programming setting, any local optimum is also a global optimum, but this does not always hold, particularly when dealing with nonlinear or nonconvex functions/constraints.

However, Alex can only grow so many apples because he has only a_r seeds for red apples, a_g seeds for green ones, and a_y seeds for yellow ones. Of course, Alex cannot sell a negative quantity of apples either. The constraints on Alex's apple production are thus

$$x_r \leq a_r$$
$$x_g \leq a_g$$
$$x_y \leq a_y$$
$$x_r, x_g, x_y \geq 0$$

This is an example of a problem that is **feasible** and bounded; that is, the set of solutions to the problem is nonempty and the optimal revenue $f(x_r^*, x_g^*, x_y^*)$ is finite.

1.2 Xela's Xylophones

Xela sells xylophones. She sells plastic xylophones and rosewood xylophones for c_p and c_w dollars respectively. Using the similar notation as the prior problem, Xela's revenues are described by the function

$$f(x_p, x_w) = c_p x_p + c_w x_w$$

For every xylophone she produces, Xela must use a_b units of base materials (think nails, screws, etc). Additionally, plastic xylophones require a_p units of plastic, and rosewood xylophones require a_w units of rosewood. The constraints are thus

$$x_p + x_w \leq a_b$$
$$x_p \leq a_p$$
$$x_w \leq a_w$$
$$x_w, x_p \geq 0$$

But in addition, because rent is due, Xela wants to sell at least μ rosewood xylophones, where $\mu > a_b$. Then, with this additional constraint

$$x_p + x_w \geq \mu \geq a_b + 1$$

the problem becomes **infeasible** as there is no solution that can satsify this constraint and the first one.

1.3 On notation

The example linear programs we studied described the **objective function**[2] and constraints in terms of a system of equations and inequalities. As you may have learned, any such system can be equivalently expressed in terms of matrices. For instance, with the "Alex's Apples" program, we may let

$$\mathbf{c} = \begin{bmatrix} c_r \\ c_g \\ c_y \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} x_r \\ x_g \\ x_y \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} a_r \\ a_g \\ a_y \end{bmatrix}$$

and

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Then the problem may be equivalently expressed as maximizing the function

$$\mathbf{c}^T\mathbf{x}$$

with respect to the constraint

$$A\mathbf{x} \leq \mathbf{b}$$

$$\mathbf{x} \geq 0$$

It would be a good exercise to convince yourself that this is an equivalent problem. The key idea is that we can express a linear programming problem in a variety of ways, whether that be in terms of systems of equations or matrices. So we may choose which ever representation is most convenient. When discussing algorithms for solving linear programming problems, many prefer to use the matrix form since it's not only more compact but also more convenient for implementing these algorithms in code.

[2] That is, the function we're trying to maximize. If we're specifically trying to minimize a function, we coud also refer to it as a **cost function**.

2 Defining linear programs

Before going any further, we formally define the general linear programming problem.

Definition (Linear program).

A linear program features a linear[a] **cost function** $\mathbf{c}^T\mathbf{x}$, where $\mathbf{c}$ is some n-dimensional **cost vector** and $\mathbf{x}$ is a n-dimensional vector of decision variables. It also features a set S of m linear equality and inequality constraints, where the ith constraint is of either of the forms

$$\mathbf{a}_i\mathbf{x} \leq b_i$$
$$\mathbf{a}_i\mathbf{x} = b_i$$
$$\mathbf{a}_i\mathbf{x} \geq b_i$$

for some n-dimensional vector $\mathbf{a}_i$ and scalar b_i. We consider the problem of minimizing $\mathbf{c}^T\mathbf{x}$ subject to every constraint in S. A solution $\mathbf{x}$ satisfying all the constraints in S is called **feasible**. The set of all feasible solutions is called the **feasible set**. An optimal feasible solution $\mathbf{x}^*$ is one that which minimizes the objective function such that $\mathbf{c}^T\mathbf{x}^* \leq \mathbf{c}^T\mathbf{y}$ for all feasible solutions $\mathbf{y}$. The value $\mathbf{c}^T\mathbf{x}^*$ is called the **optimal cost**.

[a] a linear function is synomonous with a linear transformation here; informally, it is a function mapping its inputs to some linear combination of those inputs.

For now, we're assuming that cost function, constraints, and vectors are all real-valued, but a linear program could also be integer-valued, for example. Note that any problem of maximizing a linear function $\mathbf{c}^T\mathbf{x}$ can be expressed as minimizing $-\mathbf{c}^T\mathbf{x}$, so this definition accounts for either case. It is also not too important whether or not the constraints are strict (why?)

A linear program may not have an optimal feasible solution. However, it need not be the case that the program has no feasible solutions (although this is a possibility). Instead, it could be the case that the feasible set is unbounded and that for any supposedly optimal solution $\mathbf{x}^*$ and optimal cost $\mathbf{c}^T\mathbf{x}^*$, there always exists some other solution $\mathbf{y}^*$ such that $\mathbf{c}^T\mathbf{y}^* < \mathbf{c}^T\mathbf{x}^*$. The optimal cost in this scenario is this scenario is $-\infty$.

2.1 Standard form linear programming problems

In the literature, linear programming problems are usually presented in the **standard form**. The basic idea of it is that most of the constraints is expressed as an equality, with the additional constraint that every solution $\mathbf{x}$ has nonnegative components.

Definition (Standard form linear program).

A standard form linear program is of the form

$$\text{minimize } \mathbf{c}^T\mathbf{x}$$
$$\text{subject to } \begin{cases} A\mathbf{x} = \mathbf{b} \\ \mathbf{x} \geq 0 \end{cases}$$

where A is some $m \times n$ matrix and $\mathbf{b}$ a m-dimensional vector representing the constraints.

It can be shown that any linear programming problem can be reduced into the standard form. We do not provide a formal proof of this, but the intuition goes as follows: for any constraint that is linear inequality like $\mathbf{a}_i\mathbf{x} \leq b_i$, we can add or subtract a **slack variable** s_j such that the constraint becomes an equation like $\mathbf{a}_i\mathbf{x} + s_j = b_i$, also introducing the positivity constraint $s_j \geq 0$. Note that s_j is added to the set of decision variables. If the sign of some decision variable x_j is unconstrained[3], we can replace it with the expression $x'_j - x''_j$, where x'_j and x''_j are two new decision variables subject to the positivity constraints $x'_j, x''_j \geq 0$. This follows the fact that any real number can be expressed as the difference of two nonnegative real numbers.

Theorem (Standard form reduction). Any linear program P can be reduced to a linear program P' in standard form via the elimination of inequalities using slack variables and the elimination of unconstrained variables. Any feasible solution x to P with cost vector $\mathbf{c}$ can be used to derive a feasible solution x' to P' with the same cost $\mathbf{c}^T\mathbf{x}$.

The standard form is primarily adopted for the sake of constructing algorithms that solve linear programs, since expressing the constraints as equations is better suited for implementing said algorithms.

[3] Some literature refers to such variables as **free**. Variables whose signs are restricted are unsurprisingly said to be **restricted**.

2.2 Geometric intuition

The algebraic presentation of the linear programming problem can be rather dense, which is why studying the geometric interpretation of these problems can be helpful for building intuition. Our study of the geometry of linear programs will be biased towards the case where the feasible set is two-dimensional, but the intuition applies for feasible sets of higher dimensions without any major hiccups.

Let us consider the linear program

$$\begin{aligned} &\text{minimize } \mathbf{c}^T\mathbf{x} \\ &\text{subject to } A\mathbf{x} \leq \mathbf{b} \end{aligned}$$

where

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} 5 \\ 5 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{c} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$$\mathbf{x} \in \mathbb{R}^2$$

Equivalently, we're interested in minimizing the function

$$f(x, y) = 2x + 3y$$

with the constraint $0 \leq x$, $x \leq 5$, $0 \leq y$, and $y \leq 5$. If we plot these inequalities on a graph and shade the regions, we get Figure 1.

The intersection of all the inequalities' respective regions is called the **feasible region**. The lines $y = 5$, $y = 0$, $x = 5$, $x = 0$ bound the feasible region. The linear programming can now be thought of asking: *which point* (x, y) *in the feasible region will minimize the expression* $2x + 3y$*?*

It turns that this optimal solution in this case is $(0, 0)$, which is a "corner" of our feasible region. This hints at a more general result, which for any linear program that which is feasible and has an optimal solution, then there exists

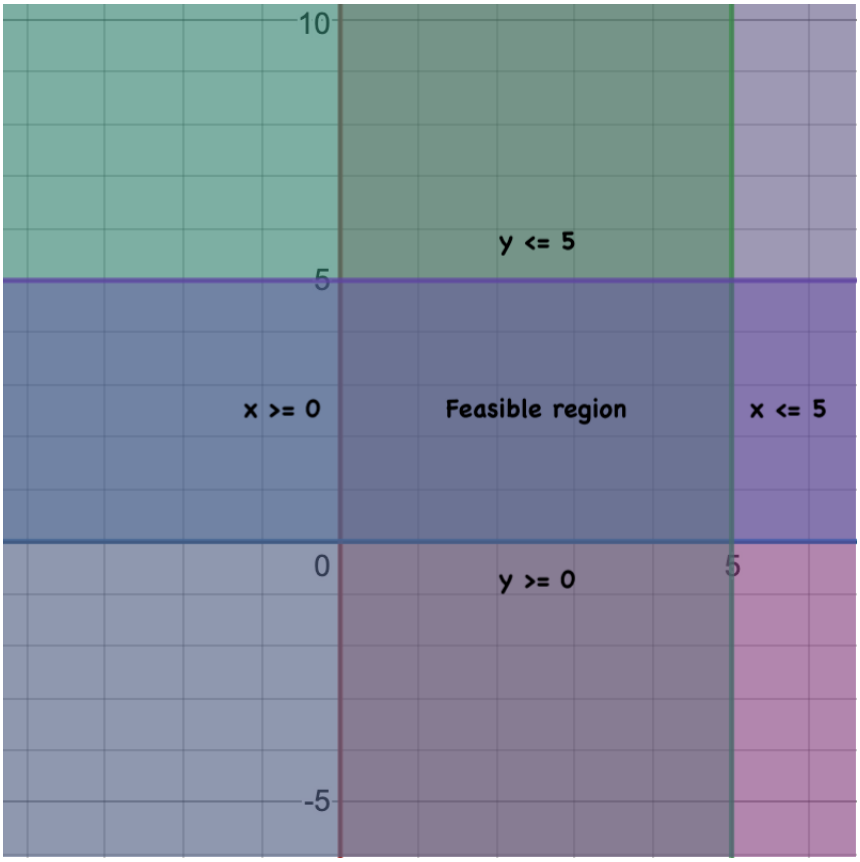


Fig. 1: The constraints $0 \leq x_1$, $x_1 \leq 5$, $0 \leq x_2$, and $x_2 \leq 5$ as plotted on a graph. The intersection of all the colored regions is the set of feasible solutions.

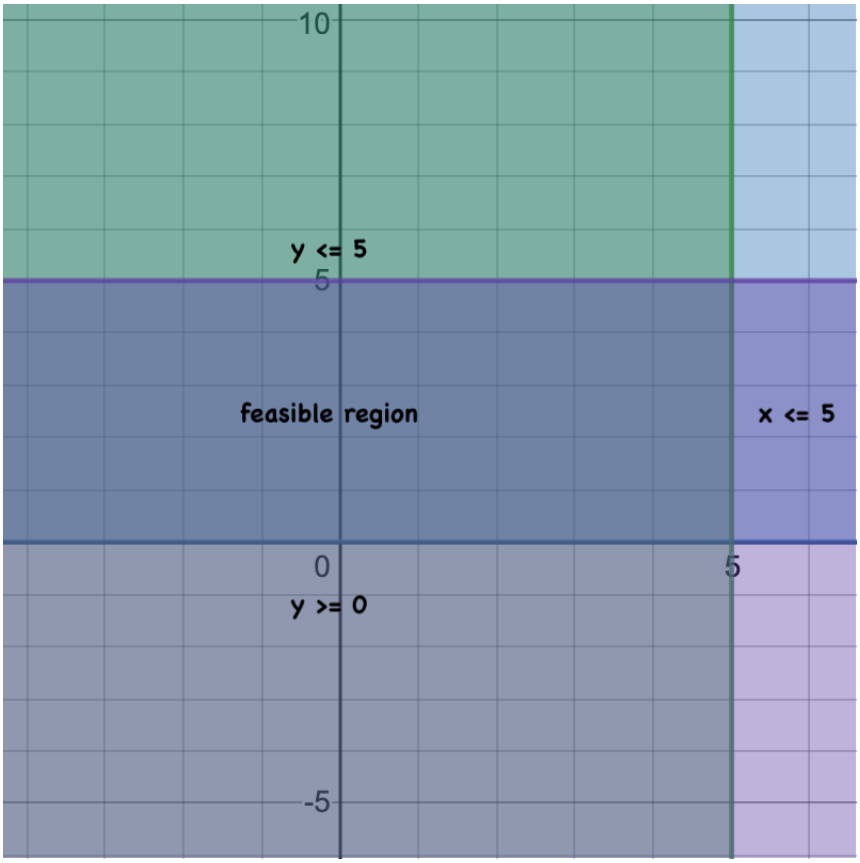


Fig. 2: The constraints $x \leq 5$, $0 \leq y$, and $y \leq 5$ as plotted on a graph.

an optimal solution that is a "corner" (alternatively, an extreme point) of the feasible region. This follows a theorem stating that every (basic) feasible solution to a linear program is a corner point of the corresponding feasible region[4].

Now imagine we did not have the constraint $x >= 0$, as in Figure 2. This illustrates the case where the feasible region is not bounded and in particular the case where there exists no optimal solution. To see this, notice that $(0, 0)$ can't be an optimal solution, since the cost for the solution $(-1, 0)$ is lower $(-2 < 0)$. But $(-1, 0)$ cannot be optimal either, since the cost for $(-2, 0)$ is even lower. This chain of reasoning leads us to see that there exists no optimal solution and that the optimal cost is $-\infty$.

[4] When discussing the geometry of linear programs, these feasible regions are better known as **polyhedra**, an example of which accompanies the title page

3 Finding optimal solutions to linear programs

So far, much of our discussion has centered on shaping and motivating the linear programming problem. We now turn towards discussing how to find optimal solutions to linear programs.

When there do not exist convenient closed forms for finding optimal solutions to any optimization problem, we typically turn towards iterative algorithms that help us move from one feasible solution towards another feasible solution that has *lower* cost. This is the essence of the **simplex** method: starting with some initial feasible solution to a linear program, we move along the boundary of the feasible set towards a cost-reducing direction. After doing this enough times, we will find an optimal solution.

3.1 Basic solutions, basis matrices, and feasible directions

Detailing the simplex method requires us to establish some ideas about solutions to linear programs. That is because the simplex method requires the initial feasible solution to have certain properties that are useful towards finding feasible solutions with a lower cost.

Definition (Basic solution). Consider some solution $\mathbf{x} \in \mathbb{R}^n$ to a linear program P. $\mathbf{x}$ is said to be a **basic solution** if and only if $\mathbf{x}$ satisfies all the equality constraints of P and out of all the constraints satisfied by $\mathbf{x}$, exactly n of those constraints are linearly independent[a]; that is, their corresponding vectors a_i are linearly independent.

A basic solution that also happens to satisfies all the constraints of P is known as a **basic feasible solution**.

[a] If it satisfies more than n constraints, then the solution is said to be degenerate. We will not consider degeneracy here, since degenerate solutions complicate the optimization problem. However, it's worth knowing that the simplex algorithm can deal with degeneracy when augmented with certain rules.

For a geometric interpretation, the aforementioned extreme points of the feasible region constitute the basic feasible solutions. With this notion of basic solutions, we can also define basic variables and the basis matrix with the following theorem.

Lemma (Basicity). Consider the standard form linear program P with constraints $A\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geq 0$, where A is an $m \times n$ matrix[a]. Let $\mathbf{x}^* \in \mathbb{R}^n$ be some solution for P. Then $\mathbf{x}^*$ is a basic solution if and only if there exists a set of indices I such that for $i \in I$ the columns A_i are linearly independent and for all $j \in [n]$ where $j \notin I$ we have that $x_j^* = 0$.

Definition (Basic variables and basis matrix). The decision variables x^i for $i \in I$ are known as **basic variables**, every other decision variable x_j is **nonbasic**. The $m \times m$ matrix

$$B = \left[A_1 \dots A_{\max(I)}\right]$$

is a **basis matrix**.

[a] It must be the case $m \leq n$; otherwise, the program fails to have a solution, i.e. it is **infeasible**.

Assuming that we know what the basic variables are, this lemma enables us to find basic solutions with the system $A\mathbf{x} = \mathbf{b}$ by setting $x_i = 0$ for $i \notin I$ and solving for the variables $i \in I$. Where $\mathbf{x}_B$ indicates the vector containing the basic components of $\mathbf{x}$, this amounts to calculating $\mathbf{x}_B = B^{-1}\mathbf{b}$.[5]

The crux of the simplex algorithm is moving from one basic feasible solution $\mathbf{x}$ to another whilst remaining in the feasible set; geometrically, this can be interpreted choosing a vector $\mathbf{d}$ representing direction to move in, provided that that the resulting solution $\mathbf{x} + \theta\mathbf{d}$ for some scalar θ is still in the feasible region. This vector $\mathbf{d}$ is in some respect core of the simplex algorithm, so it warrants its own definition.

Definition (Feasible direction). Let $\mathbf{x}$ be any feasible solution. A vector $\mathbf{d} \in \mathbb{R}^n$ is said to be a feasible direction for $\mathbf{x}$ if there exists positive $\theta \in \mathbb{R}$ such that the vector $\mathbf{x} + \theta\mathbf{d}$ is feasible.

Let j be the index of some nonbasic variable x_j for the constraints $A\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geq 0$. Then the jth basic direction is defined as the n-dimensional feasible direction $\mathbf{d}$ where the ith component $d_i = -(BA_j)_i$ for basic variables, $d_j = 1$, and 0 otherwise (for all other nonbasic variables).

Why the interest in basic directions? Well, as stated in the prior section, an optimal solutions to a linear program under usual conditions will end up

[5] B is $m \times m$, and since the columns are linearly independent, its rank is m and so the corresponding linear transformation is an isomorphism, which implies that B is invertible.

being one of the extreme points of the feasible region. The motivation for (feasible) basic directions is that they're always parallel to the edges of the feasible region.[6] So, if $\mathbf{x}$ is some extreme point on the feasible region (i.e. it's on one of the corners), if we can move in some feasible basic direction towards another extreme point, and if the move would be cost-reducing, then we should take the move since doing so will take us closer to the optimal solution.

Given any feasible basic direction $\mathbf{d}$ at a basic feasible solution $\mathbf{x}$, we're interested in finding how out much the cost reduces as we move towards that direction $\mathbf{d}$ from $\mathbf{x}$. This gives rise to one last definition quantifying the "rate" of change in the cost of a variable as we move in some basic direction:

Definition (Reduced cost). Given a cost vector $\mathbf{c}$ and basis matrix B for some feasible solution x_j, the **reduced cost** $\mathbf{c}'_j$ of the decision variable of x_j is

$$\mathbf{c}'_j = \mathbf{c}_j - \mathbf{c}_B^T B^{-1} A_j = \mathbf{c}_j - \mathbf{c}_B^T \mathbf{d}_j$$

where $\mathbf{d}_j$ denotes the jth basic direction. The reduced cost for basic variables is 0.

If the reduced cost for the jth variable is positive, then moving in the jth basic direction from some basic feasible solution will increase the cost. If it's negative, then moving in that direction will decrease the cost, which approaches what we want.

Synthesizing everything we've discussed so far, if we wanted to find the optimal solution for some linear program, then we should start with some basic feasible solution. We can then calculate all the reduced costs $\mathbf{c}'$ and for some nonbasic variable $\mathbf{x}_j$ such that $\mathbf{c}'_j < 0$, we move in its respective (basic) direction $\mathbf{d}$ towards an adjacent basic feasible solution. The basic feasible solution $\mathbf{x} + \theta\mathbf{d}$ that we move to will have a lower cost. Figure 3 offers a visual summary of this process.

Ignoring some details such as the choice of j or θ, the simplex algorithm just iterates this process on each basic feasible solution obtained until the reduced cost for every variable is nonnegative, in which case the solution will be optimal[7].

[6] Basic directions are always feasible when the solution $\mathbf{x}$ is not degenerate.

[7] This conclusion only holds because the aforementioned phenomenon where in linear programs local minima of the cost function are also global minima. Indeed, if you've worked with a nonlinear optimization technique like gradient descent, you'll be familiar with how these techniques can sometimes get "stuck" in local minima. So be thankful for linearity!

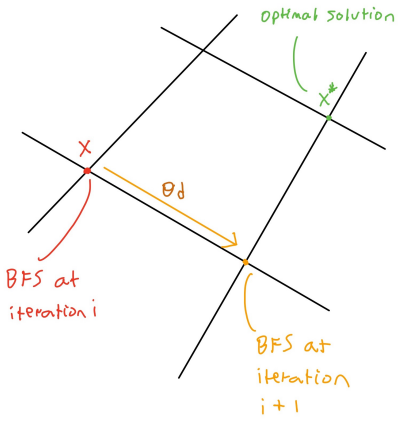


Fig. 3: Starting at an initial basic feasible solution (bfs) of $\mathbf{x}$, if we identify $\mathbf{d}$ as a basic feasible direction corresponding to some nonbasic variable x_j, then there exists some scalar θ such that $\mathbf{x}+\theta\mathbf{d}$ is an adjacent basic feasible solution. If the reduced cost for x_j is negative, this solution $\mathbf{x}+\theta\mathbf{d}$ will bear a lower cost. By applying this process again starting with $\mathbf{x}+\theta\mathbf{d}$, we will eventually reach the optimal solution $\mathbf{x}^*$.

3.2 The simplex algorithm

We finally present a simplified version of the simplex algorithm, but given our discussion above you'll find most of its steps are intuitive.

Simplex method

Let $\mathbf{x}^0 \in \mathbb{R}^n$ be any initial basic feasible solution for some standard form linear program P with cost vector $\mathbf{c}$ and the constraints $A\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geq 0$. Let I is the index set of the basic variables and where $\bar{I}$ is the index set for the nonbasic variables. If P is feasible and has no degenerate solutions and if there exists an optimal solution for P, then an optimal solution may be found with the following algorithm.

Algorithm 1: Simplex

1. If it's the first iteration, assign $\mathbf{x} \leftarrow \mathbf{x}^0$.
2. For $i \in \bar{I}$, compute the reduced costs $\mathbf{c}'_i$. If they're all nonnegative, terminate with $\mathbf{x}$ as an optimal solution. Otherwise, choose any j such that $\mathbf{c}'_j < 0$.
3. Let $\mathbf{d} = B^{-1}A_j$ (the feasible direction for the jth variable)[a], and let k be the basic index minimizing $\theta^* = \frac{\mathbf{x}_k}{\mathbf{d}_k}$.
4. Assign $x_j \leftarrow \theta$ and $\mathbf{x} \leftarrow \mathbf{x} + \theta^*\mathbf{d}$.[b] Let the variable x_j enter the basis, and let x_k exit it. Go to step 2.

[a] If we did not make the assumption that P had an optimal solution, then we might've run into the problem that no component of $\mathbf{d}$ was positive, in which case the optimal cost is $-\infty$

[b] The choice of θ^* was such that the new solution will be basic where x_j is now basic and where x_k is no longer basic; more precisely, we are choosing a bfs adjacent to the current one.

Because it's always moving in a cost-reducing direction, this algorithm will always converge to a solution within a finite number of iterations (on a programs with a bounded feasible set).

For the sake of simplicity, we've deliberately ignored several questions. In particular:

1. The algorithm doesn't specify *how* to algorithmically choose an initial basic feasible solution. That is because doing so require us to solve a separate and auxiliary linear program, which also be accomplished using the simplex method but not treated here.
2. We can ignore the caveat of not having degenerate solutions by choosing the smallest applicable index j in step 2, but the proof of this is beyond the scope of this introduction.
3. We assumed that the program P had a solution to avoid dealing with the case where the optimal cost is $-\infty$. We could've actually detected this by checking if none of the components of the chosen basic direction were positive, in which case we can actually conclude that feasible region is unbounded and terminate (but the proof of this is also beyond our scope).
4. We also didn't justify the choice for θ^* for similar reasons as above.

3.3 Why simplex?

In Section 2.2, we hinted that for programs that are feasible, the optimal solution, if it exists, will be a basic feasible solution. It turns out that there are finitely many basic solutions to a linear program, because there are only $\binom{n}{m}$ ways to m columns from a $m \times n$ matrix A to possibly form a basis matrix, and each basis matrix uniquely corresponds to a basic solution.

One could imagine that to find the optimal solution, we could just check every basic solution and identify the one producing the lowest cost as optimal. This certainly works, but for large enough linear programs the simplex method is on average more efficient in terms of the number of solutions it checks before finding the optimal one. The exact time complexity of the simplex method varies by the implementation[8], but for a typical linear program the simplex method runs in time polynomial to m and n (the dimensions of the matrix A). For certain unusual programs, simplex indeed visits all the extreme points of the feasible region, and so its worst-case time complexity is exponential.

For more computationally-minded readers, the supplementary materials provide a Python program implementing the simplex method in nearly one-to-one correspondence with the steps of algorithm provided above.

4 Further readings

(Bertsimas and Tsitsiklis, 1997) offers a comprehensive but dense introduction to linear optimization and in particular devotes an entire chapter towards deriving the simplex method and efficient implementations of it. There are also many great resources scattered across the internet that cover LP and the simplex method with motivating intuition and examples, in particular this, this, and this.

[8] Implementing the simplex method using the same steps verbatim as the algorithm above turns out to be quite inefficient; there exists implementations using **tableaux** representations of linear programs that offers significant performance improvements