Introduction to Algorithms

Chapter 6: Linear programming

Manuel

Fall 2017

Outline

1. Understanding linear programming
2. The simplex method
3. Formulating problems as linear programming

Setup

A new type of problem:

- Given: limited resources or competing constrains
- Goal: maximize/minimize an objective

Problem (Linear Programming Problem)

Given a mathematical model where an *objective* is represented as a linear function and some *constraints* are expressed as equalities or inequalities, maximize the objective while respecting the constraints.

Linear programming defines a method to solve this problem.

Applications

Common applications

- Business: maximize the profit when producing various items sold at different prices
- Politics: what to say such as getting the more votes
- Airlines: following the regulation on what crew can do, maximize the number of flights possible given the number of crew members
- Exploration: decide where to drill for oil in order to maximize profit
- Diet: select food such as having enough nutriments while minimizin the cost

Solving linear programming problems

Three main types of algorithm exist:

- Ellipsoid algorithm: polynomial time complexity but slow in practice
- Interior-point methods: polynomial time complexity and fast in practice for large input
- Simplex algorithm: exponential time complexity but very fast in practice

Formalizing linear programming

Definition (Standard form)

A linear program can be expressed in *standard form*. Given n real numbers $c_1, \cdots, c_n$, m real numbers $b_1, \cdots, b_m$, and mn real numbers a_{ij}, $1 \leq i \leq m$, $1 \leq j \leq n$, find n real numbers $x_1, \cdots, x_n$ that maximize $\underline{\sum_{j=1}^{n}}\, c_j x_j$, while being subject to

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j \leq b_i, & 1 \leq i \leq m; \\ x_j \geq 0, & 1 \leq j \leq n. \end{cases} \tag{6.1}$$

In standard form all the constraints (eq. (6.1)) are defined as inequalities. Although a linear program does not necessarily requires non negative constrains, the standard form does.

Compact standard form

Express the standard form (6.6) in a more compact way:

- Represent the a_{ij} as an $m \times n$ matrix A
- Represent the b_i as an m-vector b
- Represent the c_j as an n-vector c
- Represent the x_j as an n-vector x

In this form a linear program consists in maximizing $c^T x$, knowing that x is subject to $Ax \leq b$ and $x \geq 0$.

From linear programming to standard form

Problem	Solution
Minimizing vs. maximizing	Negate the coefficients of the objective function
Nonnegative constraint on x_j	Write $x_j = x_j' - x_j''$ and add constraints $x_j', x_j'' \geq 0$
Equality constrain $x_j = l$	Write $x_j \leq l$ and $x_j \geq l$
Greater than constrain $x_j \geq l$	Multiply the inequality by -1

From linear programming to standard form

Example.

Express the following linear program into standard form. Minimize $-2x_1 + 3x_2$, subject to

$$\begin{cases} x_1 + x_2 & = 7 \\ x_1 - 2x_2 & \leq 4 \\ x_1 & \geq 0. \end{cases} \tag{6.2}$$

By negating the coefficients from the objective this is equivalent to maximizing $2x_1 - 3x_2$ while keeping the constraints (6.2).

As there is no constraint on x_2 we rewrite it $x_2 = x_2' - x_2''$ and the problem becomes: maximize $2x_1 - 3x_2' + 3x_2''$, subject to

$$\begin{cases} x_1 + x_2' - x_2'' & = 7 \\ x_1 - 2x_2' + 2x_2'' & \leq 4 \\ x_1, x_2', x_2'' & \geq 0. \end{cases}$$

From linear programming to standard form

Finally the first constraint is rewritten as two “less or equal” inequalities such that the initial linear program can be expressed in the following standard form, with variables x_1, x_2 and x_3 in place of x_1, x_2' and x_2'', respectively.

Maximize $2x_1 - 3x_2 + 3x_3$, subject to

$$\begin{cases} x_1 + x_2 - x_3 & \leq 7 \\ -x_1 - x_2 + x_3 & \leq -7 \\ x_1 - 2x_2 + 2x_3 & \leq 4 \\ x_1, x_2, x_3 & \geq 0. \end{cases} \tag{6.3}$$

The standard form allows to define a generation formalization for any linear program. However to solve such problem it is more desirable to deal with equalities rather inequalities.

The slack form

Definition (Slack form)

A linear program is transformed into *slack form* by rewriting all the inequalities, but the nonnegativity constraints $\underline{\text{which is omitted}}$, as equalities and using the variable z to denote the value of the objective function.

Let $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$ be an inequality constraint. Define a *slack variable* s and rewrite the previous constrain as

$$\begin{cases} s & = b_i - \sum_{j=1}^{n} a_{ij}\overset{x_j}{xj}, \\ s & \geq 0. \end{cases}$$

In the new set of equalities *non-basic variables* are one the right and appear in the objective function, while *basic variables* are on the left.

For consistency reasons, when slack variables are added their name usually starts at x_{n+1} instead of s.

From standard to slack form

Example.
We carry on with example 6.9, now transforming the standard form into slack form.
From constraints (6.3) three inequalities should be rewritten as equalities. Therefore three slack variables x_4, x_5, and x_6 are added. Moreover the terms "maximize" and "subject to", as well as the nonnegativity constraints can be omitted. Hence the initial linear problem can be expressed in slack form as

$$\begin{cases} z = \quad 2x_1 - 3x_2 + x_3 \\ x_4 = 7 - x_1 - x_2 + x_3 \\ x_5 = -7 + x_1 + x_2 - x_3 \\ x_6 = 4 - x_1 + 2x_2 - 2x_3. \end{cases}$$

Concise slack form

In a more compact definition a linear program can be expressed in it slack form as

$$\begin{cases} z = v + \sum_{j \in N} c_j x_j, \\ x_i = b_i - \sum_{j \in N} a_{ij} x_j, \quad \text{for } i \in B, \end{cases}$$

where z, x_i, b_i, a_{ij}, and c_j are as previously defined, while N defines the set of the non-basic variables, B represents the indices for the set of the basic variables, and v is an optional constant term in the objective function.

Therefore, in term of algorithm a slack form can be represented by a structure composed of the fields N, B, A, b, c, v.

Outline

1. Understanding linear programming
2. The simplex method
3. Formulating problems as linear programming

Underlying idea

Given a linear program in slack form the goal is to solve it while maximizing the value z. In fact the system of equations provided by the constraints is very similar to a regular system of equations, which can usually be solved by *Gaussian elimination*.

However note that a slack system usually features more variables than equations. Therefore there might be an infinite number of solutions. A solution such that $x_j \geq 0$ for all j is called a *feasible solution*.

A *basic solution* is found when setting all the non-basic variables to 0 on the right hand while computing the value of the basic variables on the left hand side. Assuming this basic solution is feasible the goal is to reformulate the linear program such as to maximize z.

Underlying idea

The value z is maximized by selecting a non-basic variable x_k and increasing it as much as possible without violating any of the constraints. Then x_k is transfered on the left side to become a basic variable and the previous basic variable x_l is moved to the right. All other constraints, as well as the objective function are updated.

Example.
Given the following linear problem in slack form determine x_1, x_2, and x_3 such as to maximize z.

$$\begin{cases} z = & 3x_1 + x_2 + 2x_3 \\ x_4 = 30 - & x_1 - x_2 - 3x_3 \\ x_5 = 24 - & 2x_1 - 2x_2 - 5x_3 \\ x_6 = 36 - & 4x_1 - x_2 - 2x_3. \end{cases}$$

Underlying idea

The basic solution is $(x_1, \cdots, x_6) = (0, 0, 0, 30, 24, 36)$. Then the "fastest" way to increase is to use x_1. Looking at the constraints we observe that augmenting x_1 decreases x_4, x_5, and x_6. However because of the nonnegativity constraint none of them can become negative. Therefore x_1 should not be increased by more than 9 and

$$x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}.$$

As x_1 is now a basic variable all the constraints and the objective are rewritten

$$\underline{\left\{\begin{aligned} z &= 27 + \frac{x_2}{4} + \frac{x_3}{2} - \textcircled{\frac{3x_6}{4}} \quad \text{Used} \\ x_1 &= 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4} \\ x_4 &= 21 - \frac{3x_2}{4} - \frac{5x_3}{2} + \frac{x_6}{4} \\ x_5 &= 6 - \frac{3x_2}{2} - \underline{\underline{4x_3}} + \frac{x_6}{2}. \end{aligned}\right.}$$

Underlying idea

The previous operation consisting in maximizing the objective while transforming a non-basic variable into a basic one is called a *pivot*. (Moreover x_1 is referred to as the *entering variable* and x_6 as the *leaving variable*.) Both used

We now repeat the same process, choosing x_3 which allows the objective to increase faster than x_2. The third constrain being the tightest one we rewrite it $x_3 = 3/2 - 3x_2/8 - x_5/4 + x_6/8$. The linear program becomes

$$\begin{cases} z = \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \\ x_1 = \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \\ x_3 = \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \\ x_4 = \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16}. \end{cases}$$

Underlying idea

The objective can now only be increased by adjusting x_2, and the tightest constraint being the second we get

$$\begin{cases} z = 28 - \dfrac{x_3}{6} - \dfrac{x_5}{6} - \dfrac{2x_6}{6} \\ x_1 = 8 + \dfrac{x_3}{6} + \dfrac{x_5}{6} - \dfrac{x_6}{3} \\ x_2 = 4 - \dfrac{8x_3}{3} - \dfrac{2x_5}{3} + \dfrac{x_6}{3} \\ x_4 = 18 - \dfrac{x_3}{2} + \dfrac{x_5}{2}. \end{cases}$$

All the coefficients in the objective function now being negative the solution $(8, 4, 0, 18, 0, 0)$ with objective value 28 is optimal. Therefore, for the original linear problem x_1, x_2, and x_3 are 8, 4 and 0, respectively. On their side the slack variables measure how much difference remains in each equality.

Pivoting

Algorithm.

Input : a linear problem in slack form (N, B, A, b, c, v), l and e the indices of the leaving and entering variables, respectively

Output : an equivalent linear problem in slack form

```
Function Pivot(N, B, A, b, c, v, l, e):
    $b'_l \leftarrow b_l / a_{le}$;
    foreach $j \in N \setminus \{e\}$ do $a'_{ej} \leftarrow a_{ij} / a_{le}$ ;
    $a'_{el} \leftarrow 1/a_{le}$;
    foreach $i \in B \setminus \{l\}$ do
        $b'_i \leftarrow b_i - a_{ie} b'_e$;
        foreach $j \in N \setminus \{e\}$ do $a'_{ij} \leftarrow a_{ij} - a_{ie} a'_{ej}$;
        $a'_{il} \leftarrow -a_{ie} a'_{el}$
    end foreach
    $v \leftarrow v + c_e b'_e$;
    foreach $j \in N \setminus \{e\}$ do $c'_j \leftarrow c_j - c_e a'_{ej}$ ;
    $c'_l \leftarrow -c_e a'_{el}$;
    $N \leftarrow N \setminus \{e\} \cup \{l\}$; $B \leftarrow B \setminus \{l\} \cup \{e\}$;
    return $N, B, A', b', c', v$          /* $A'$ $m \times n$ matrix of the $a'_{ij}$ */
end
```

The simplex method

When looking at the general process illustrated in the previous example (6.16) a pivot operation only represents one step of a more complex process. In particular all the calls to the `pivot` function are coordinated by the main procedure, the simplex algorithm.

Also note that the problem solved in the example is given in the slack form. Therefore it is also necessary to initialize the algorithm by transforming a problem from standard form into slack form.

We now provide the pseudocode corresponding to the simplex algorithm, and then look at its initialization.

Simplex algorithm

Algorithm. (*Simplex*)

Input : a linear problem in standard form (A, b, c)
Output : an optimal solution to the linear problem

```
(N, B, A, b, c, v) ← Init(A,b,c);
while ∃ j ∈ N with c_j > 0 do
    select e such that c_e > 0;
    foreach index i ∈ B do
        if a_ie > 0 then d_i ← b_i/a_ie else d_i ← ∞;
    end foreach
    select l such that d_l is minimal;
    if d_l ≠ ∞ then (N, B, A, b, c, v) ← Pivot(N, B, A, b, c, v, l, e);
    else return unbounded;
end while
for i ← 1 to n do
    if i ∈ B then x_i ← b_i else x_i ← 0;
end for
return x_1, ··· , x_n
```

Initialization

The simplex algorithm (6.22) starts by calling an initialization function. Its role is not only to translate the linear program from standard form into slack form, but also to ensure its feasibility. At times it might be feasible although the basic solution is not, thus requiring more work.

Lemma

Let L be a linear program in standard form. Let x_0 be an additional variable and L_0 be the following linear program with $n+1$ variables. Maximize $-x_0$, subject to

$$\begin{cases} \sum_{j=1}^{n} a_{ij}x_j - x_0 \leq b_i, & 1 \leq i \leq m; \\ x_j \geq 0, & 0 \leq j \leq n. \end{cases}$$

Then the linear program L is feasible if and only if the optimal objective value for L_0 is 0.

Initialization

Example.
Maximize $2x_1 - x_2$, subject to

$$\begin{cases} 2x_1 - x_2 \leq 2 \\ x_1 - 5x_2 \leq -4 \\ x_1, x_2 \geq 0. \end{cases}$$

If we follow the basic strategy to translate the linear program into slack form then this yields the basic solution $x_1 = x_2 = 0$ which violates the second constraint. Therefore we now formulate an *auxiliary linear program* for which we can find a slack form with a feasible basic solution. Applying lemma 6.23 we get the problem:
Maximize $-x_0$, subject to

$$\begin{cases} 2x_1 - x_2 - x_0 \leq 2 \\ x_1 - 5x_2 - x_0 \leq -4 \\ x_1, x_2, x_0 \geq 0. \end{cases}$$

Initialization

By lemma 6.23 the existence of a feasible solution to the original linear program depends on the optimal value of the objective being 0 or not, for the new linear program. The latter can be rewritten in slack form as

$$\begin{cases} z = & & & & - & x_0 \\ x_3 = & 2 & - 2x_1 & + x_2 & + & x_0 \\ x_4 = & -4 & - x_1 & + 5x_2 & + & x_0. \end{cases}$$

This auxiliary linear problem still has no basic solution since it would set x_4 to -4. However we can reshape it by running the `Pivot` function. The entering variable chosen is x_0 and the leaving one is the basic variable corresponding to the constraint featuring the smallest b_i, i.e. x_4 in our case.

Initialization

This yields the following slack form

$$\begin{cases} z = -4 - x_1 + 5x_2 - x_4 \\ x_0 = 4 + x_1 - 5x_2 + x_4 \\ x_3 = 6 - x_1 - 4x_2 + x_4 . \end{cases}$$

Although the associated basic solution $(4, 0, 0, 6, 0)$ is now feasible, it remains to determine an optimal solution for L_0. In particular it is sufficient to apply a single call to `Pivot` with entering and leaving variables x_2 and x_0, respectively. This yields

$$\begin{cases} z = - x_0 \\ x_2 = \frac{4}{5} - \frac{x_0}{5} + \frac{x_1}{5} + \frac{x_4}{5} \\ x_3 = \frac{14}{5} + \frac{4x_0}{5} - \frac{9x_1}{5} + \frac{x_4}{5} . \end{cases}$$

Initialization

As the objective value for the auxiliary linear program is 0 the initial problem is feasible by lemma 6.23. This also means that x_0 can be removed from the constraints since it is equal to 0.

The last stage consists in restoring the initial objective function ensuring it only contains non-basic variables. We get

$$2x_1 - x_2 = 2x_1 - \left(\frac{4}{5} + \frac{x_1}{5} + \frac{x_4}{5}\right).$$

This yields the following slack form which has a feasible solution

$$\begin{cases} z = -\frac{4}{5} + \frac{9x_1}{5} - \frac{x_4}{5} \\ x_2 = \frac{4}{5} + \frac{x_1}{5} + \frac{x_4}{5} \\ x_3 = \frac{14}{5} - \frac{9x_1}{5} + \frac{x_4}{5}. \end{cases}$$

Initialization

Algorithm.

Input : a linear problem in standard form (A, b, c)
Output : an initial feasible slack form or infeasible

```
Function Init(A, b, c):
    select $k$ such that $b_k$ is minimal among all the $b_i$;
    if $b_k \geq 0$ then return $\{1, \ldots, n\}, \{n+1, \cdots, n+m\}, A, b, c, 0$;
    foreach constraint do add $x_0$ on the right side;
    set the objective function to $-x_0$;                    /* define $L_0$ */
    $(N, B, A, b, c, v) \leftarrow L_0$ in slack form;
    $l \leftarrow n + k$;
    $(N, B, A, b, c, v) \leftarrow$ Pivot($N,B,A,b,c,v,l,0$);
    Run algorithm 6.22 (lines 2-10) until an optimal solution is found for $L_0$;
    if the optimal solution to $L_0$ sets $x_0$ to 0 then
        if $x_0$ is basic then perform one pivot to make it non-basic;
        remove $x_0$ from the constraints, restore the original objective function,
          and replace any basic variable from the objective function by the right
          hand side of its associated constraint;
        return the modified final slack form;
    else
        return infeasible;
    end if
end
```

Fundamental theorem

Theorem (Fundamental theorem of Linear Programming)

For a linear program L in standard form, one of the following holds.

- L has an optimal solution with a finite objective value;
- L is infeasible;
- L is unbounded;

In any case, the Simplex algorithm returns the appropriate state.

A few more remarks

A few words on the Simplex method:

- Exponential time complexity
- Very efficient in practice
- Many variations exist
- For all the variations there exists a family of linear programs which takes exponential time to solve
- The existence of a variation running in polynomial or sub-exponential time remains an open problem

Outline

1 Understanding linear programming

2 The simplex method

3 Formulating problems as linear programming

Shortest path

Given two vertices s and t in a weighted graph $G = \langle V, E \rangle$, the shortest weighted path problem (3.12) consists in computing the shortest distance d_t between s and t.

To phrase this problem as a Linear Program it is necessary to determine a set of variables and constraints that define when a shortest path is discovered.

This is in fact what the Bellman-Ford algorithm (3.17) does, as when it terminates the distance from s to any vertex v is given by $d_v \leq d_u + w(u, v)$, where u is a vertex adjacent to v and $w(u, v)$ is the weight of the edge (u, v) (lemma 3.16). Noting that $d_s = 0$ this translates into the following linear program:
Maximize d_t, subject to

$$\begin{cases} d_v \leq d_u + w(u, v) & \text{for each edge } (u, v) \in E \\ d_s = 0 \end{cases} \tag{6.4}$$

Shortest path

The reason behind the maximization of the objective function is again provided by lemma 3.16. The solution to the shortest path problem sets the minimum distance for each vertex v to

$$d_v = \min_{(u,v)\in E} \left\{ d_u + w(u,v) \right\}.$$

Therefore d_v is the largest value that is less or equal to all the values in $\left\{ d_u + w(u,v) \right\}$. In other words we are interested in the minimum of the maximum. We want to maximize d_v for all the vertices v on the shortest path from s to t, as it will maximize d_t.

The Linear Program (6.4) can be solved using the simplex algorithm but this results in a slower method than using a specialized algorithm such as the Bellman-Ford (3.17).

Duality

Definition

Given a Linear Program where the objective is to be maximized, its *dual* has the same optimal value and is described as follows, using the notations from definition 6.6. Find m real numbers $y_1, \cdots y_m$ that minimize $\sum_{i=1}^{m} b_i y_i$, while being subject to

$$\begin{cases} \sum_{i=1}^{m} a_{ij} y_i \geq c_j, & 1 \leq j \leq n; \\ y_i \geq 0, & 1 \leq i \leq m, \end{cases}$$

The original Linear Program is called the *primal*.

max flow min cut

Duality

Example.

Determine the dual of the following Linear program.

Maximize $3x_1 + x_2 + 2x_3$, subject to

$$\begin{cases} x_1 + x_2 + 3x_3 \le 30 \\ 2x_1 + 2x_2 + 5x_3 \le 24 \\ 4x_1 + x_2 + 2x_3 \le 36 \\ x_1, x_2, x_3 \ge 0 \end{cases}$$

Duality

Example.
Determine the dual of the following Linear program.
Maximize $3x_1 + x_2 + 2x_3$, subject to

$$\begin{cases} x_1 + x_2 + 3x_3 \leq 30 \\ 2x_1 + 2x_2 + 5x_3 \leq 24 \\ 4x_1 + x_2 + 2x_3 \leq 36 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

Minimize $30y_1 + 24y_2 + 36y_3$, subject to

$$\begin{cases} y_1 + 2y_2 + 4y_3 \geq 3 \\ y_1 + 2y_2 + y_3 \geq 1 \\ 3y_1 + 5y_2 + 2y_3 \geq 2 \\ y_1, y_2, y_3 \geq 0. \end{cases}$$

Duality

Theorem (Linear Programming duality)

Let (A, b, c) be a primal Linear Program and $x = (x_1, \cdots, x_n)$ be a solution returned by the Simplex algorithm. Denoting by c' the vector representing the coefficients in the final slack form, the vector y is defined by $y_i = -c'_{n+i}$ if $n+i$ is in N and zero otherwise. Then x is an optimal solution to the primal Linear Program, y is an optimal solution to the dual Linear Program and

$$\sum_{j=1}^{n} c_j x_j = \sum_{i=1}^{m} b_i y_i.$$

Final remarks

Given a linear program, solve it in:

- Primal form if it has more variables than constraints
- Dual form if it has more constraints than variables

Non-linear optimization function constraints:

- This is not a linear program
- Use a different approach
- If the objective function is quadratic, specific cases can be efficiently solved
- In the general case quadratic programming is $\mathcal{NP}$-complete

Key points

- What is a linear program?
- What are the standard and slack forms?
- Briefly describe the Simplex method
- Explain duality and when to use it in the case of the simplex method

Thank you!