Day 1 - Linear programming

Richard Oberdieck

Ørsted A/S

June 10, 2019

Outline

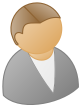
Outline

Linear programming

Anne

Production scheduling

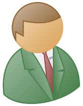
We have a 5-step fermentation process that we are still running manually! Can't we do better?



Martin

Dieting

I want to make sure that I get all my nutrients, but I am strapped for cash. Can't I do better?



Banker

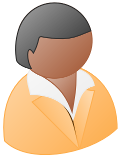
Portfolio optimization

We like money, but we're still picking our stocks based on what we read in the news. Can't we do better?

Have you had any experience with linear programming?

Outline

An example



Anne

Anne

Our warehouses in Copenhagen and Hamburg are consistently running low on stock even though we are producing more than enough! Can't we do better?

The freight cost is 90€ per 100 km per liter, the freight loading cost is 25€ per liter and you are handed this table:

Supply		Demand	
Odense	350	Copenhagen	325
Aarhus	600	Hamburg	275

What to do?

Formulating the actual question

"Can't we do better?" is not a very specific question:

- What is "better"?
- Who is "we"?

So how about something like this:

Actual problem formulation

We want to find out how much we have to ship from Odense/Aarhus to Copenhagen/Hamburg such that we **minimize** cost and **subject to** the market demand, supply capacities and physical constraints.

$$\begin{aligned} z = \quad & \text{minimize} \quad & \text{Shipping costs} \\ & \text{subject to} \quad & \text{Market demand} \\ & & \text{Supply capacities} \end{aligned}$$

Let's open the file `Supply chain example.ipynb` and have a look.

Outline

Now it's your turn!

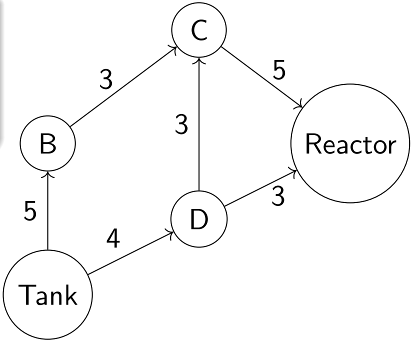
Anne

Now we are having problems with our production! Can you see whether we can squeeze more out of Odense?

The problem is then quickly identified in the raw material flow going into the reactor.

Actual problem formulation

Given the following pipe network in your factory, what is the maximum flow you can get out of it?

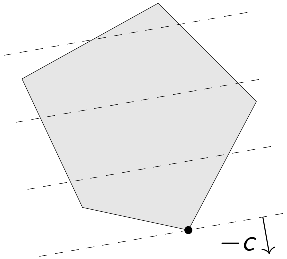


maximize	Flow
subject to	Capacity constraints
	Conservation of mass
	Physical constraints

Outline

General definition of linear programming

$$
\begin{aligned}
z = \quad & \underset{x}{\text{minimize}} && c^T x \\
& \text{subject to} && Ax \leq b \\
& && A_{eq} x = b_{eq} \\
& && x \in \mathbb{R}^n
\end{aligned}
$$



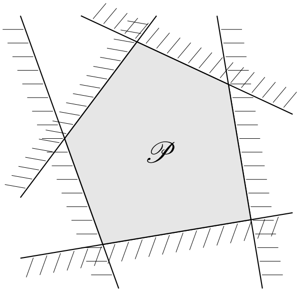
- The solution(s) are **always** at a vertex of the polytope spanned by the constraints.
- Therefore, a solution is characterized by: the optimal objective function value z, the values of the variables x^* and the indices $\mathscr{I}$ of the constraints which are active at the solution:

$$
\begin{aligned}
A_i x^* &= b_i \quad \forall i \in \mathscr{I} \\
A_j x^* &< b_j \quad \forall j \notin \mathscr{I}
\end{aligned}
$$

Analysis of a linear program

- **Variables** are the degrees of freedom of your system.
- The **objective function** shows how good a given choice of values for the variables is.

- The **equality** constraints are always active, and are therefore omitted in the following.
- The **inequality** constraints form a "room" for the variables to live in called a "polytope".
- There is an insane amount of research on polytopes, and we will only scratch the surface of the surface here.



Polytopes

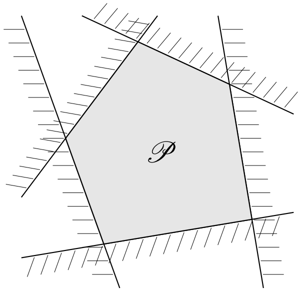
Definition

The set $\mathscr{P}$ is called a *n*-dimensional polytope if it satisfies:

$$\mathscr{P} := \left\{ x \in \mathbb{R}^n \,|\, a_i^T x \leq b_i, i = 1, ..., m \right\},$$

where m is finite.

- A subset of a polytope is called a face of $\mathscr{P}$.
- The faces of polytopes of dimension $n-1$, 1 and 0 are referred to as facets, edges and vertices, respectively.
- A polytope is called bounded if and only if there exists a finite $x_{\min} \in \mathbb{R}^n$ and $x_{\max} \in \mathbb{R}^n$ such $x_{\min} \leq x \leq x_{\max}$ for all $x \in \mathscr{P}$.



Outline

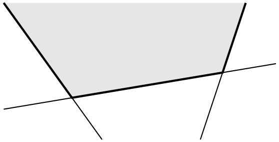
The art of linear programming modelling - introduction

So far we have seen fairly straightforward linear constraints. But what if I told you that this is a linear programming problem?

$$
\begin{aligned}
z = \quad & \underset{x_j}{\text{minimize}} && \frac{\left(\sum_{j \in J} c_j x_j + \alpha\right)}{\left(\sum_{j \in J} d_j x_j + \beta\right)} \\
& \text{subject to} && \sum_{j \in J} a_{ij} x_j \leq b_i, \quad \forall i \in I \\
& && x_j \geq 0, \quad \forall j \in J
\end{aligned}
$$

All of the tricks though rely on one of the two:

- Introduce **new variables** defined in a clever way.
- Use the **epigraph** of the constraints to create curves.



Absolute values

$$\begin{aligned} \underset{x_j}{\text{minimize}} \quad & \sum_{j \in J} c_j \left| x_j \right| \\ \text{subject to} \quad & \sum_{j \in J} a_{ij} x_j \leq b_i, \quad \forall i \in I \end{aligned}$$

To use linear programming on this problem, we need to avoid using the absolute value terms $|x_j|$, as shown below:

$$\begin{aligned} x_j &= x_j^+ - x_j^- \\ |x_j| &= x_j^+ + x_j^- \\ x_j^+, x_j^- &\geq 0 \end{aligned}$$

Using these, the problem can be reformulated as:

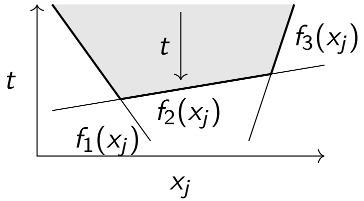
$$\begin{aligned} \underset{x_j}{\text{minimize}} \quad & \sum_{j \in J} c_j \left(x_j^+ + x_j^- \right) \\ \text{subject to} \quad & \sum_{j \in J} a_{ij} \left(x_j^+ - x_j^- \right) \leq b_i \\ & x_j^+, x_j^- \geq 0 \end{aligned}$$

Minimum of a maximum - minimax

$$\begin{aligned} \underset{x_j}{\text{minimize}} \quad & \max_{k \in K} f_k(x_j) \\ \text{subject to} \quad & \sum_{j \in J} a_{ij} x_j \leq b_i, \quad \forall i \in I \end{aligned}$$

where $f_k(x_j) = \sum_{j \in J} c_{kj} x_j$. To resolve this, we introduce a new variable $t = \max_{k \in K} \sum_{j \in J} c_{kj} x_j$, and describe it using the epigraph:

$$\begin{aligned} \underset{x_j, t}{\text{minimize}} \quad & t \\ \text{subject to} \quad & \sum_{j \in J} a_{ij} x_j \leq b_i, \quad \forall i \in I \\ & \sum_{j \in J} c_{kj} x_j \leq t, \quad \forall k \in K \end{aligned}$$



Fractional objective of linear objective functions

$$\begin{aligned} z = \quad & \underset{x_j}{\text{minimize}} && \frac{\left(\sum_{j\in J} c_j x_j + \alpha\right)}{\left(\sum_{j\in J} d_j x_j + \beta\right)} \\ & \text{subject to} && \sum_{j\in J} a_{ij} x_j \le b_i, \quad \forall i \in I \\ & && x_j \ge 0, \quad \forall j \in J \end{aligned}$$

1. Rewrite the objective in terms of $t = \frac{1}{\left(\sum_{j\in J} d_j x_j + \beta\right)}$:

$$\sum_{j\in J} c_j x_j t + \alpha t.$$

   Add $t > 0$ as a constraint.

2. Multiply both sides of the original constrains by t and rewrite the model using $y_j = x_j t$.

Fractional objective of linear objective functions - continued

$$
\begin{aligned}
z = \quad & \underset{y_j, t}{\text{minimize}} && \sum_{j \in J} c_j y_j + \alpha t \\
& \text{subject to} && \sum_{j \in J} a_{ij} y_j \leq b_i t, \quad \forall i \in I \\
& && \sum_{j \in J} d_j y_j + \beta t = 1 \\
& && t > 0 \\
& && y_j \geq 0, \quad \forall j \in J
\end{aligned}
$$

- The values of x_j at the optimal solution can be retrieved from y_j by diving by t.
- This is a quite non-obvious reformulation, but well worth it considering that now we have a LP.
- There are a couple of other tricks related to probability distributions and robust optimization, but they follow the same principles.

And now it's your turn!

Commonly, we fit to the least-mean-square error, i.e. $f(x) = \sum_{j \in J} x_j^2$.

However, now you should write a script that solves for the following:

Least absolute deviations estimation: $f(x) = \sum_{j \in J} |x_j|$

Least maximum deviation estimation: $f(x) = \max_{j \in J} |x_j|$

Download the file `ErrorEstimation.ipynb` from github and get on it!

Outline

An introduction to LP solvers

So far, we have assumed that we can simply solve LP problems by giving them to Xpress. And that's true. But what type of solvers are out there? And how are modern solvers built up?

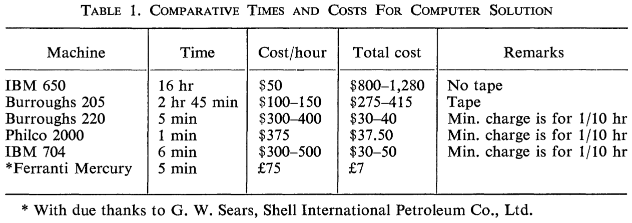


TABLE 1. COMPARATIVE TIMES AND COSTS FOR COMPUTER SOLUTION

Machine	Time	Cost/hour	Total cost	Remarks
IBM 650	16 hr	\$50	\$800–1,280	No tape
Burroughs 205	2 hr 45 min	\$100–150	\$275–415	Tape
Burroughs 220	5 min	\$300–400	\$30–40	Min. charge is for 1/10 hr
Philco 2000	1 min	\$375	\$37.50	Min. charge is for 1/10 hr
IBM 704	6 min	\$300–500	\$30–50	Min. charge is for 1/10 hr
*Ferranti Mercury	5 min	£75	£7	

* With due thanks to G. W. Sears, Shell International Petroleum Co., Ltd.

Figure: Taken from: Koenigsberg (1961) OR, Vol. 12, No. 2, pp. 105-114.

How are solvers typically built up?

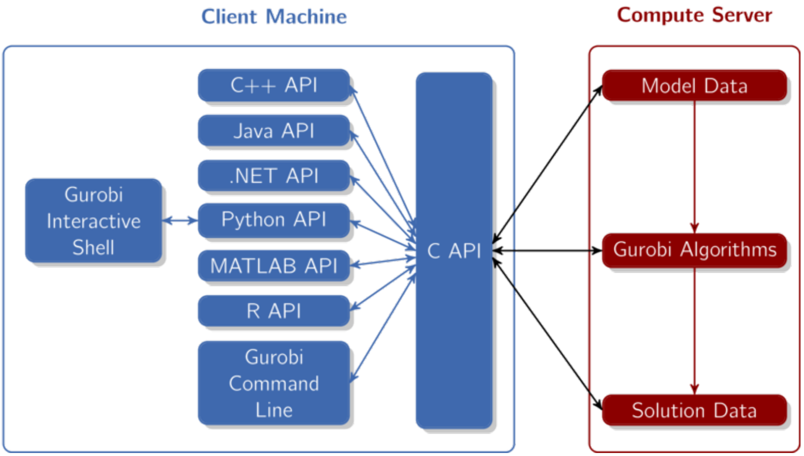


Figure: Taken from:
http://www.gurobi.com/documentation/8.0/remoteservices/compute_server.html

An overview over linear programming solvers

- LP solvers can be classified into **commercial** and **free** software.
- When it comes to LP solvers, the three components that most relevant are (a) performance, (b) modelling capabilities and (c) support.
- Most (if not all) commercial solvers are free to use for academics, however their price tag can be fairly high for industry.

Commercial solvers

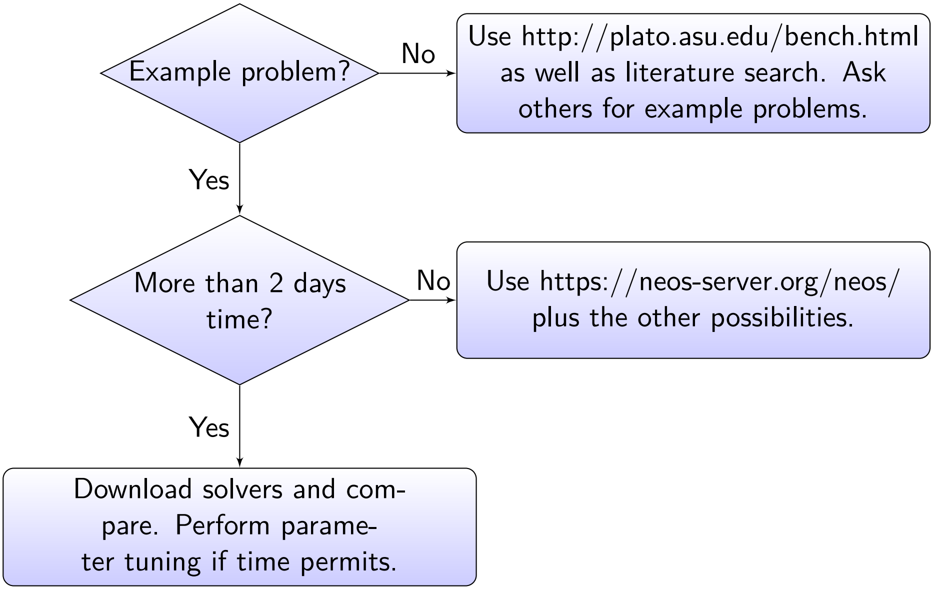
- The "big three": IBM CPLEX, Gurobi, FICO Xpress
- MOSEK, MATLAB, SAS

Free solvers

- **The** free solver: COIN-OR
- GLPK, Google GLOP, lp_solve

Outline

How to choose a solver - Performance


Example problem?
No
Use http://plato.asu.edu/bench.html as well as literature search. Ask others for example problems.
Yes
More than 2 days time?
No
Use https://neos-server.org/neos/ plus the other possibilities.
Yes
Download solvers and compare. Perform parameter tuning if time permits.

How to choose a solver - modelling capabilities

Rule of thumb

Most solvers have similar modelling capabilities. If you don't have very specific requirements, just pick the fastest/most convenient one that you have access to.

1. Is the language fixed? Then consider only those that have an application programming interface (API) for it.
2. How much control do you need over the solver? Do you need a barrier method? Is determinism required?
3. Do you have specific constraints? For example max or $|\cdot|$? It will save you a lot of headache to have those implemented.
4. How "concise" is the API? Can you write your constraints in an elegant way?

A side note on modelling libraries

Modelling libraries

A modelling library provides the ability to describe mathematical models and then hand it off to various solvers. Common examples in Python are pyomo, PuLP and cvxpy.

- Modelling libraries allow you to be solver independent, and easily change between a free and commercial solver
- They are probably the biggest competitor to the domain specific languages such as GAMS
- The ones I have tested however had two downsides:
  - They do not provide a "bare-bones" structure, but high-level interaction, which forces me to write my code in a certain way. I don't like that.
  - They tend to not be performance optimized, and you cannot go directly to the solver level to fix that.
- This is why we are not using them in the course.

How to choose a solver - support

There are always things you don't know about the solver. It is therefore wise to consider the level of support you will be getting. Commonly, this will come in two forms:

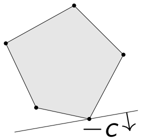
User forums: Most (if not all) solvers have a user forums to ask questions to the community. Check how frequently people post and what the response rate of e.g. the developers is. That will give you an indication of the level of support you can expect.

Customer support: If you have an academic license, customer support may not be open to you. However, for commercial customers it is so much more important because you have a personal line to the vendor of the software.

Outline

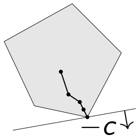
How LP problems are solved nowadays

Simplex: "Check the corners"



- Pick combinations of constraints (active set) that form a vertex
- Use objective function to guide search until optimal is found
- Exponential complexity, but efficient in practice

Interior-point: "Build a wall"



- Start from interior of polytope
- Introduce barrier functions instead of constraints:

$$f(x) - \mu \sum_{i=1}^{m} \log\left(-a_i x + b_i\right)$$

- Polynomial complexity

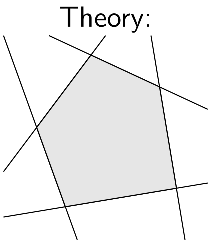
Some remarks on solution algorithms

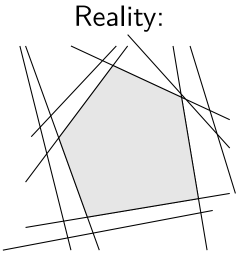
Coding your own solver

Never, ever, ever code your own LP solver!

- There are primal and dual simplex methods: in primal simplex, you start suboptimally and move towards optimality, in dual simplex you start "superoptimally" and move towards feasibility.
- There are also different flavors of interior-point methods (primal-dual etc.)
- It is also common to have "cross-over" algorithms, which start with an interior-point method and then switch over to the simplex method close to optimality.

Redundancy and the meaning of "presolve"





This is called redundancy and is one of the main "clutter" in LP models.

- Redundant **constraints** and **variables** are (almost) necessary in mathematical programming.
- However, getting the "minimal representation" (i.e. all the clutter removed) is computationally very expensive.
- This is why all modern solvers have "presolve" routines. For a nice introduction, check Andersen and Andersen (1995) Presolving in linear programming. Math. Prog. 71, 221 - 245.

Tutorial problem 1 - Batch scheduling

See handout

Tutorial problem 2 - Oil production

See handout