Linear Programming

“program” as in “exercise program” or “spending program”, not “C program”

optimization problem with linear inequalities

variables $x_1, \ldots, x_d$ in $d$-dimensions

\[
\begin{align*}
\text{max} & \quad c_1 x_1 + c_2 x_2 + \cdots + c_d x_d \\
\text{st.} & \quad a_{11} x_1 + a_{12} x_2 + \cdots + a_{1d} x_d \leq b_1 \\
& \quad \vdots \\
& \quad a_{n1} x_1 + \cdots + a_{nd} x_d \leq b_n
\end{align*}
\]

i.e. \[ \max c^T x \] \quad $c$ is a $1 \times d$ vector
\[ A \quad x \leq b \] \quad $A$ is an $n \times d$ matrix
\[ b \quad n \times 1 \]
An application: planning menus.

\[
\begin{align*}
\text{d foods: } & \quad \text{apple} & \quad \text{broccoli} & \quad \text{milk} \\
\text{each with cost: } & \quad c_1 & \quad c_2 & \quad \cdots & \quad c_d \\
\text{n nutrients: } & \quad \text{protein} & \quad \text{vitamin D} \\
\text{each with daily requirement: } & \quad b_1 & \quad b_2 & \quad \cdots & \quad b_n \\
\text{\(a_{ij}\) - amount of nutrient i in food j }
\end{align*}
\]

Buy food to meet daily requirements, \[\min \text{ cost} \]
\[\min cx \quad \text{variables } x_1 \ldots x_d \]
\[Ax \geq b \]
\[x_j = \text{amount of food j to buy} \]
picture in 2D

Each constraint $a_1x_1 + a_2x_2 \leq b$

is a half-space

Push as far as possible

Optimum solution.

Opt. solution is at a vertex except

Use tie-breaking to avoid this

Unbounded
Straightforward algorithm:

try **all** vertices, see which gives max

From last day: this is the dual problem to Convex Hull and can be solved by same algorithms

\[
O(n \log n) \text{ in 2D, 3D} \\
O(n^{d-1/2}) \text{ for } d \geq 4
\]

But we don’t really want all the vertices, so we can do better.
History

early 40’s, 50’s

George Dantzig

- simplex method in the ’40’s

Simplex Method

- geometrically — walk from one vertex of the feasible region to an adjacent one

- Simplex pivot rule

  - which inequality to remove
  - which one to add

a great intro to linear programming:

Understanding and using linear programming
J Matousek, B Gärtner - 2007

https://ocul-wt1.promo.exlibrisgroup.com/permalink/01OCUL_WTL/5ob3ju/alma9953153109505162
**History**

OPEN: is there a pivot rule that gives a polynomial time algorithm?

But the simplex algorithm is very good in practice.

Related question:

Given initial vertex $s$ and final vertex $t$ (on a convex polyhedron), how many edges on the shortest path from $s$ to $t$?

* diameter of the polyhedron $= \text{worst case over all } s \text{ and } t$

**Hirsch conjecture.**  
The diameter of a convex polyhedron is $\leq n - d$ where $n = \text{number of inequalities, } d = \text{dimension}$

disproved in 2012, $d = 43$.

But there could still be a polynomial (or even linear) bound.
History

Polynomial time algorithms for Linear Programming:

'80 — Katchian, ellipsoid method

'84 — Karmarkar, interior point method

these operate on the bit representations of the numbers

OPEN: an LP algorithm that uses number of arithmetic operations polynomial in $n$ and $d$, “strongly polynomial time”

“smoothed analysis” explains the good behaviour of the simplex method

https://en.wikipedia.org/wiki/Smoothed_analysis

The simplex algorithm is NP-mighty

Y Disser, M Skutella - ACM Transactions on Algorithms (TALG), 2018 - dl.acm.org

We show that the Simplex Method, the Network Simplex Method—both with Dantzig’s original pivot rule—and the Successive Shortest Path Algorithm are NP-mighty. That is, each of these algorithms can be used to solve, with polynomial overhead, any problem in NP implicitly during the algorithm’s execution. This result casts a more favorable light on these algorithms’ exponential worst-case running times. Furthermore, as a consequence of our approach, we obtain several novel hardness results. For example, for a given input to the …
Linear programming in small (fixed) dimension $d$

Application: Casting (from [CGAA]). Make a 3D object in a mold.

Pour liquid into a mold, harden, and then remove by straight line motion in some direction. Find a direction that works.

For a given top face, this can be expressed as linear programming. Try all top faces.
Linear programming in small (fixed) dimension

Megiddo 1984, algorithm with runtime $O(n)$

but the dependence on $d$ is bad $O(2^d n)$

Seidel 1991, randomized algorithm with expected runtime $O(n)$

dependence on $d$ $O(d! n)$

comparing $2^{2d}$ vs $d!$

- take logs $2^d \quad d \log d$
Randomized Incremental Algorithm in 2D, Seidel

Idea: add the halfplanes one by one in random order, updating the optimum solution vertex $v$ each time

To add $h_i$. Two cases:

1. \( v \notin h_i \) - no update to $v$.

2. \( v \notin h_i \) - new opt. vertex will lie on the line $l_i$ of $h_i$.
   So solve a 1-dim. LP problem along line $l_i$.
We reduced to 1D Linear Programming.
What is 1D Linear Programming?

\[
\text{max } x \quad \text{(the constant is irrelevant)}
\]

Subject to
inequalities like
\[ x \leq 2 \]
\[ -1 \leq x \leq 5 \]

Feasible region is intersection of intervals = one interval
easy to do \( O(n) \)

Solution is \( x = 2 \)
Some issues:

What is the initial vertex (when there are no halfplanes)?

What if the LP is “unbounded”, (e.g. max $x$, $x \geq 0$)

Add a large box.
Initialize vertex $v$ to optimum vertex of box
If final $v$ on boundary of box then original LP was unbounded

The method also needs the optimum to be unique — handle this by asking for optimum, then (to break ties) the lexicographically largest (i.e. max $x_1$, then max $x_2$, ...)
Algorithm \( \text{LP}_2 (H, c) \) \( \quad H_n = \{h_1, \ldots, h_n\} \), a set of halfplanes, \( c = \text{objective} \)

1. add large box; initialize \( v \) to optimum vertex of box (wrt \( c \))
2. take random order \( h_1, \ldots, h_n \)
3. for \( i = 1 \ldots n \) \# add \( h_i \)
4. suppose \( h_i \) is \( a_1 x_1 + a_2 x_2 \leq b \)
5. if \( v \not\in h_i \) (i.e. \( a_1 v_1 + a_2 v_2 > b \) ) then
6. \# solve the problem restricted to the line \( a_1 x_1 + a_2 x_2 = b \)
7. \( \{h'_1, \ldots, h'_{i-1}\} \), \( c' := \text{use the equation to eliminate one variable from} \quad \{h_1, \ldots, h_{i-1}\}, c \)
8. \( v := \text{LP}_1 (\{h'_1, \ldots, h'_{i-1}\}, c') \)

Worst case run time: \( O(n^2) \) line 7, 8 each take \( O(n) \)

Exercise: Show that the worst case can happen.
**Expected runtime**

Prove expected run time is $O(n)$

use backwards analysis

After we add $h_i$. Suppose opt. is vertex $v'$

at intersection of $h'$, $h''$

We already $h_1, \ldots, h_i$

So $h', h'' \subset \{ h_1, \ldots, h_i \}$

How much work for $h_i$

was it case 1 (no work) or case 2 (call LP$_i$)

Case 2 iff $h_i$ is $h'$ or $h''$.

Prob $\{ h_i = h' \text{ or } h_i = h'' \} = \frac{2}{i}$

because $h_i$ is equally likely to be any of the $i$ inequalities inserted so far

Expected total work $\sum_{i=1}^{n} \frac{2}{i} O(i) = O(n)$

Note: degeneracy ok.
In higher dimensions

\[ \frac{2}{i} \text{ becomes } \frac{d}{i} \text{ because it takes } d \text{ hyperplanes to specify a vertex} \]

run time recurrence: \[ T_d(n) = T_d(n - 1) + \frac{d}{n} O(T_{d-1}(n)) \]

solution is: \[ T_d(n) = O(d! n) \]
**Smallest enclosing disc**

Not a linear programming problem, but amenable to the same approach

Given points \( p_1, \ldots, p_n \in \mathbb{R}^d \)

find the smallest enclosing sphere.

This is a facility location problem — the center of the disc minimizes the maximum distance to all points.

Natural formulation gives quadratic programming.

Megiddo's approach gives \( O(n) \) but bad constant.

Randomized incremental approach, Welzl, 1991.

note that the smallest disc will go through 3 points
Smallest enclosing disc. Randomized incremental algorithm. Suppose we have the solution for \( n - 1 \) points.

\[ W(P, R) \] — find smallest disc enclosing points \( P \) with points \( R \) on the boundary. \( |R| \leq 3 \). Initially \( P \) is the whole set of points and \( R = \emptyset \)

\[
\text{if } |R| = 3 \text{ or } P = \emptyset \quad \text{— easy}
\]

\[
D := W(P - \{p\}, R) \quad \text{— } p \text{ chosen at random}
\]

\[
\text{if } p \in D \text{ return } D
\]

\[
\text{else return } W(P - \{p\}, R \cup \{p\})
\]

Expected run time \( O(n) \) (no details)
Summary

- brief intro to linear programming
- linear programming in fixed dimension — randomized algorithm with expected run time $O(n)$
- smallest enclosing disc

References

- [CGAA] Chapter 4
- [Zurich] Appendix E, F, G
- Seidel’s paper
  Small-dimensional linear programming and convex hulls made easy
  R. Seidel - Discrete & Computational Geometry, 1991 - Springer
  [https://doi.org/10.1007/BF02574699](https://doi.org/10.1007/BF02574699)
- general Linear Programming

Understanding and using linear programming
J. Matousek, B. Gärtner - 2007

[https://ocul-wtl.primo.exlibrisgroup.com/permalink/01OCUL_WTL/5ob3ju/alma9953153109505162](https://ocul-wtl.primo.exlibrisgroup.com/permalink/01OCUL_WTL/5ob3ju/alma9953153109505162)