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.

max $c_1 x_1 + c_2 x_2 + \cdots + c_d x_d$

st. $a_{11} x_1 + a_{12} x_2 + \cdots + a_{1d} x_d \leq b_1$

$\vdots$

$a_{n1} x_1 + \cdots + a_{nd} x_d \leq b_n$

i.e. $\max c^T x$

$A x \leq b$

$c$ 1x$d$ vector

$x$  $d$x1 vector

$A$  $n$x$d$ matrix

$b$  $n$x1
### An application: planning menus.

<table>
<thead>
<tr>
<th>Foods</th>
<th>Apple</th>
<th>Broccoli</th>
<th>Milk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Costs</td>
<td>$c_1$</td>
<td>$c_2$</td>
<td>$c_d$</td>
</tr>
<tr>
<td>Nutrients</td>
<td>Protein</td>
<td>Vitamin D</td>
<td></td>
</tr>
<tr>
<td>Requirements</td>
<td>$b_1$</td>
<td>$b_2$</td>
<td>...</td>
</tr>
</tbody>
</table>

- $a_{ij}$ = amount of nutrient $i$ in food $j$

Buy food to meet daily requirements, minimizing cost.

\[
\begin{align*}
\text{min } & \quad c^T x \\
\text{subject to } & \quad Ax \geq b \\
\end{align*}
\]

Variables $x_1, \ldots, x_d$

$x_j$ = amount of food $j$ to buy.
picture in 2D

Each constraint \( a_1 x_1 + a_2 x_2 \leq b \) is a half-space (bounded by a line).

Feasible region

Note: opt. soln need not be unique.

May be unbounded.
Straightforward algorithm:

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

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

$O(n \log n)$ in 2D, 3D

$O\left(n \frac{\log n}{\log \log n}\right)$ 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
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 = worst case over all $s$ and $t$

**Hirsch conjecture.**

The diameter of a convex polyhedron is $\leq n - d$

where $n =$ number of inequalities, $d =$ 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 — Khachiyan, 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

\[ \text{https://en.wikipedia.org/wiki/Smoothed_analysis} \]

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 Dimensions

Applications

1. Separating red and blue points by a line

\[ y = ax + b \]

variables \( a, b \)

\[ py \leq apx + b \quad \forall (px, py) \text{ blue} \]

\[ py \geq apx + b \quad \forall \text{ red} \]

Solve for \( a, b \) (just feasibility, no max.)

... and can check red below, blue above... vertical line...

Can also ask for strict separation

\[ py < apx + b \implies py \leq apx + b + \delta \]

\( \delta \) a variable, \( \delta \geq 0 \)

maximize \( \delta \).
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 $2^d$ vs $d!$

take logs $2^d$ vs $d \log d$
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. \( v \in h_i \) - no update required
2. \( v \notin h_i \) - we must update \( v \).
   - Claim: new opt will lie on line \( l_i \) of \( h_i \)
   - Solve 1-dimensional linear program (LP) on line \( l_i \)
We reduced to 1D Linear Programming.
What is 1D Linear Programming?

\[
\begin{align*}
\max \ & x \\
\text{subject to} \ & x \leq 2 \\
\ & -1 \leq x \\
\ & x \leq 5 \\
\end{align*}
\]

Solution is easy \( O(n) \) where \( n = \# \text{ constraints} \):
- max of lower bounds
- min of upper bounds.
Some issues:

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

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

**Solution**

Add a large box — containing all vertices.

```
\begin{align*}
\text{if final } v \text{ on box} \\
\text{then original was unbounded.}
\end{align*}
```

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 x₁, then max x₂, . . )
Algorithm LP\(_2\) \((H, c)\) \(H_n = \{h_1, \ldots, h_n\}\), 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 \(a_1x_1 + a_2x_2 \leq b\)
5. if \(v \not\in h_i\) (i.e. \(a_1v_1 + a_2v_2 > b\)) then
6.       # solve the problem restricted to the line \(a_1x_1 + a_2x_2 = b\)
7. \(\{h'_1, \ldots, h'_{i-1}\}, c' := \text{use the equation to eliminate one variable from}\)
   \(\{h_1, \ldots, h_{i-1}\}, c\)
8. \(v := LP_1(\{h'_1, \ldots, h'_{i-1}\}, c')\)

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

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

use backwards analysis

Suppose adding \( h_i \) causes work - we update to \( u' \)

\[ h' \quad \rightarrow \quad u' \quad \rightarrow \quad h'' \]

Then \( u' \) is at intersection of two lines of halfplanes (else we would not update)

one of \( h' \) or \( h'' \) is \( h_i \)

We've processed from \( 1 \ldots i \)

\( * \) \( h_i \) is equally likely to any of them

We did update only if \( h_i \) is \( h' \) or \( h'' \)

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

Expected total work of updates

\[ \sum_{i=1}^{n} \frac{2}{i} \cdot O(i) = O(n) \]
In higher dimensions

\[ \frac{2}{i} \] becomes \( \frac{d}{i} \) because it takes \( d \) hyperplanes to specify a vertex for expected runtime

run time recurrence:

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

solution is:

\[ T_d(n) = O(d! n) \] (prove by induction)
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 \( R = \emptyset \)

**FACT:** updated disc goes through \( p \)

New problem: min disc enclosing \( P \) and going through \( p \)

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

- brief intro to linear programming

- linear programming in fixed dimension — randomized algorithm with expected run time $O(n)$

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

- general Linear Programming

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

  https://ocul-wtl.exlibrisgroup.com/permalink/01OCUL_WTL/5ob3ju/alma9953153109505162