Small-dimensional Linear Programming and Convex Hulls Made Easy

Raimund Seidel

Teng Liu

April 3rd 2025

Background

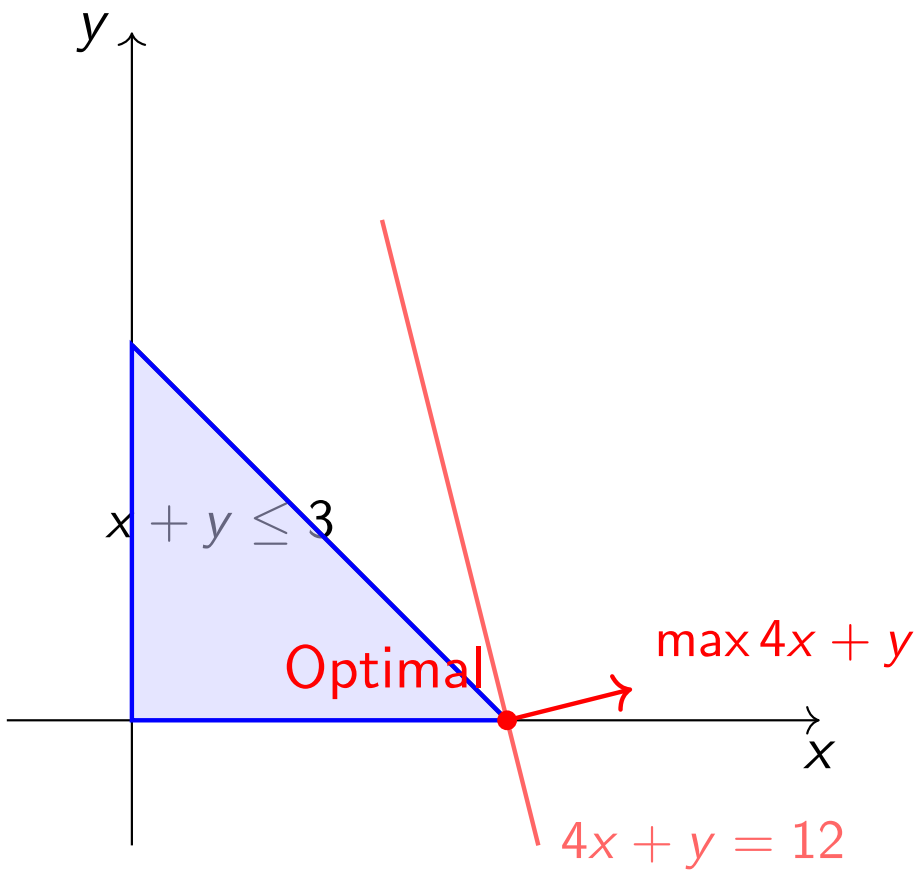
Linear programming captures one of the most canonical and influential constrained optimization problems.

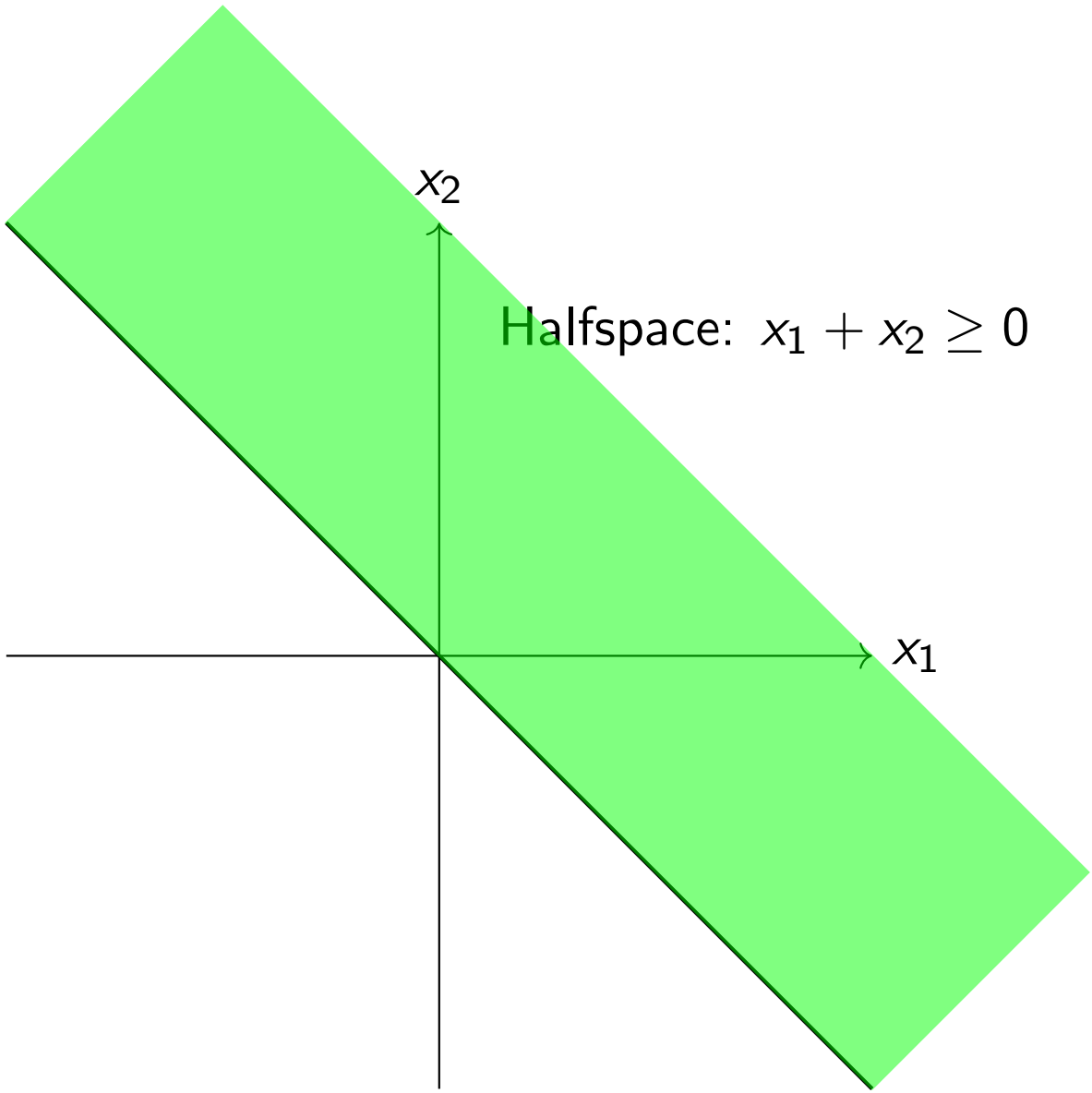
More precisely, it asks to maximize or minimize a linear objective under linear inequality and equality constraints.

$$\begin{aligned} \max / \min \ & c^T x \\ \text{subject to} \quad & Ax \leq e \\ & Bx \geq f \\ & Cx = g \end{aligned}$$

Background - Geometrical Interpretation

Every constraint corresponds to a hyperplane (a halfspace, precisely). The intersection of several halfspaces gives us a polyhedron.





Hyperplane: $x_1 + x_2 = 0$

Background - Simplex Method

Fact

Optimal solution can always be achieved by some vertex.

Fact

If there exists any vertex with better objective value than current vertex, we always have a neighbour vertex with better value.

Background - Simplex Method

Fact

Optimal solution can always be achieved by some vertex.

Fact

If there exists any vertex with better objective value than current vertex, we always have a neighbour vertex with better value.

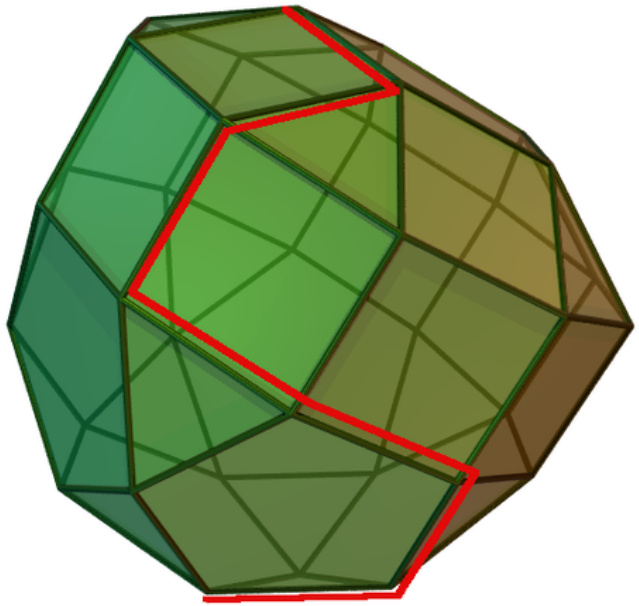
Firstly select any starting vertex.

Each step in Simplex Method move from one vertex to its neighbour with larger objective.

Background - Simplex Method

Firstly select any starting vertex.

Each step in Simplex Method move from one vertex to its neighbour with larger objective.



Background - Simplex Method Running Time

Algorithm	**Complexity**	**Speed**	**Path**
Simplex	Exponential	Fast	Edges of polyhedron
Ellipsoid	Polynomial	Slow	Ellipsoid shrinking
Interior Point	Polynomial	Fast	Through the interior

Table: Comparison of Famous LP Algorithms

Previous Work

When d, the dimension of LP (i.e. the number of variables), is considered to be constant, then LP can be solved in time linear in m, the number of constraints.

These methods[2] are complicated and running time superexponential to d: e.g. 3^{d^2}.

We also have a simple algorithm[1] with complexity $O(d^2 m) + (\log m) O(d)^{d/2+O(1)} + O(d^4 \sqrt{m} \log m)$ but the analysis is involved.

The presented algorithm gives a simple procedure and complexity $O(d!m)$.

Algorithm

This algorithm makes use of a very simple idea:

Suppose we delete one constraint $\vec{a}x \geq b$, and somehow managed to solve the new LP. Then we check if the optimum x^* conforms with the deleted constraint.

- If it doesn't violate the constraint, i.e. $\vec{a}x^* \geq b$, then x^* is also the optimum for original LP, because deleting one constraint gives us a larger feasible area.
- If it violates the constraint, then the opt value lies on the hyperplane corresponding to this constraint. We can project the feasible area onto the corresponding hyperplane, thus reducing dimension by 1. Then solve the new LP problem with lower dimension.

Algorithm

Let original feasible region be D, feasible region after deleting constraint be D', the optimum for D' be x^* . Clearly $D \subseteq D'$ and in this case $x^* \in D$.

$$u^T x^* \underset{\text{(i)}}{\leq} \max_D \underset{\text{(ii)}}{\leq} \max_{D'} = u^T x^*$$

(i): $x^* \in D$, by definition of max.

(ii): Optimizing on a larger area gives no worse optimal value than the original one.

Algorithm

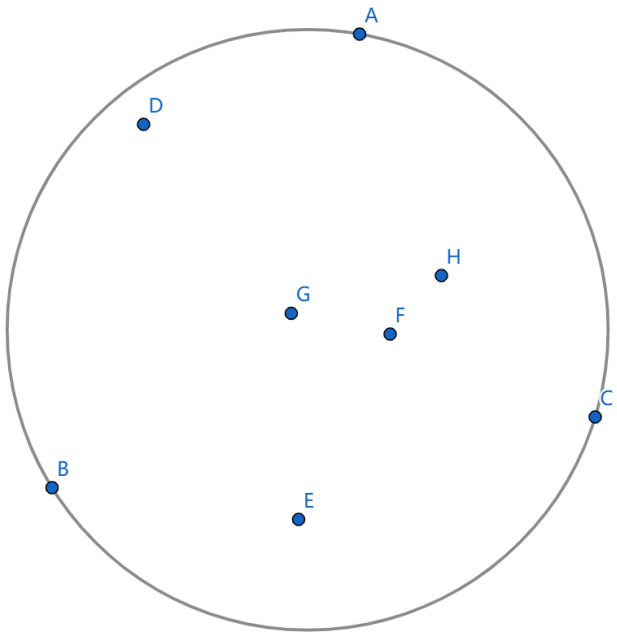
Let original feasible region be D, feasible region after deleting constraint be D', the optimum for D' be x^* . In this case $x^* \notin D$. If it violates the constraint, then the opt value lies on the hyperplane corresponding to this constraint.
This is because if we consider the points along the line connecting new LP's opt point and original LP's opt point, by convexity we have that we have a better point on the hyperplane. This is also not bad, at least we decrease the dimension (we restrict the problem on a hyperplane, which has dimension $d-1$), which gives us the intuition to recurssively analyse.

Backward Analysis

Sometimes it's hard to determine the probability if thinking *forward*, while surprisingly straightforward if thinking *backward*.

Smallest Enclosing Circle

Given a set of points P in Euclidean plane, compute the smallest circle that contains them all.



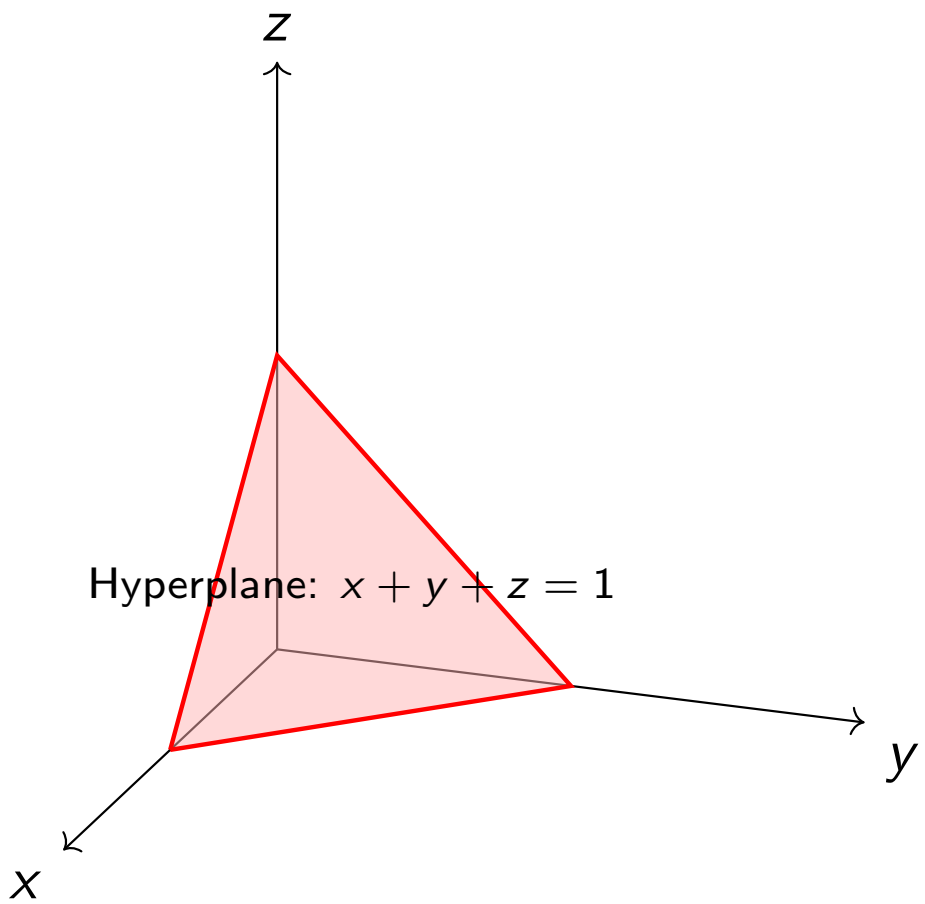
Backward Analysis - Smallest Enclosing Circle

Emo Welzl gave an efficient random algorithm with surprisingly simple procedure:

```
def Smallest_Enclosing_Circle(P, R):
    ```input : P and R are finite sets of points in the plane, |R| <= 3
    ```output: Smallest circle enclosing P with R on the boundary

    if P is empty or |R| == 3 then:
        return trivial(R)

    p = random_select(P)
    D = Smallest_Enclosing_Circle(P - {p}, R)
    if p in D then:
        return D
    else:
        return Smallest_Enclosing_Circle(P - {p}, R + {p})
```


z
Hyperplane: $x + y + z = 1$
y
x

References I

Kenneth L. Clarkson.
Las vegas algorithms for linear and integer programming when the dimension is small.
J. ACM, 42(2):488–499, March 1995.

Nimrod Megiddo.
Linear programming in linear time when the dimension is fixed.

J. ACM, 31(1):114–127, January 1984.