Small-dimensional Linear Programming and Convex Hulls Made Easy

Raimund Seidel

Teng Liu

April 3rd 2025

Background

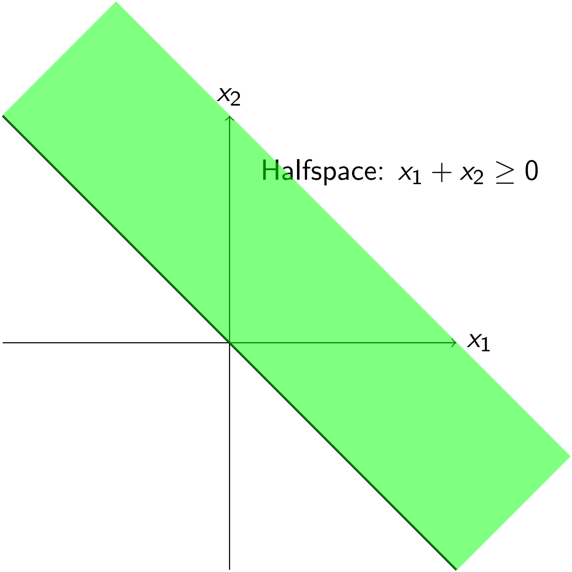
Linear programming captures one of the most canonical and influential constrained optimization problems.

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

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

Background - Geometrical Interpretation

Every constraint corresponds to a hyperplane (a halfspace, precisely).
The intersection of several halfspaces gives us a polyhedron.
We tend to optimize over this feasible area.



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.

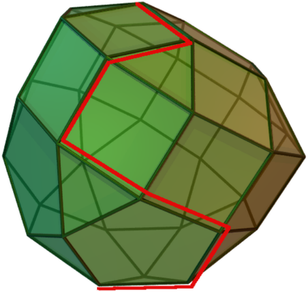
Firstly select any starting vertex.

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

Add here the complexity and running time tables for three methods.

The three most famous algorithms for LP: Interior Method, Ellipsoid Method, Simplex Method.
Even though we have Interior Method and Ellipsoid Method, the most widely used algorithm is still Simplex Method due to its efficiency in practice.

Previous Work

there are some previous work showing that if d, the dimension or i.e. the number of variables in LP, is considered as a constant, then LP can be solved in time linear to m, the number of constraints. However these methods are: complicated; and running time superexponential to d: e.g. 3^{d^2}.

we also have a simple algorithm 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 $O(d!m)$.

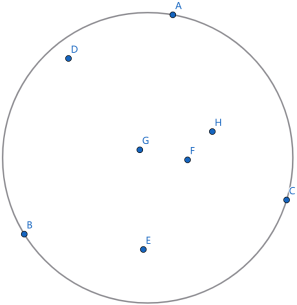
We were talking about other algorithms for LP but what do these have to do with this paper?
and we talked so much about simplex method.
But unfortunately it has nothing to do with what we are going to present.
This algorithm makes use of a very simple idea:
Suppose we delete one constraint, and somehow managed to solve the new LP. Then we check if the opt position conforms with the deleted constraint.
If it doesn't violate the constraint, then we are done. (this is obvious, since we have a larger feasible area, we certainly have better opt value)
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

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.

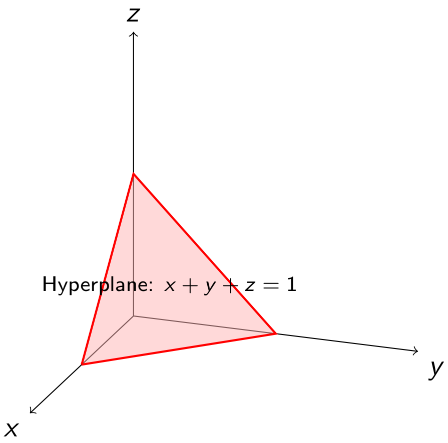


Emo Welzl gave an efficient random algorithm with surprisingly simple procedure:

```
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