

# CS 164 & CS 266: Computational Geometry

## Lecture 9

### LP-type problems

**David Eppstein**

University of California, Irvine

Fall Quarter, 2025



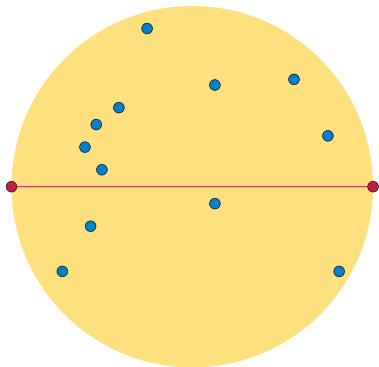
This work is licensed under a Creative Commons Attribution 4.0 International License

**Minimum enclosing circle**

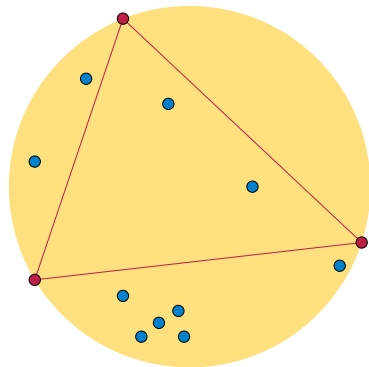
## Minimum enclosing circle

Problem: Given  $n$  points in the plane, find min-radius circle containing them

May be either of two cases:



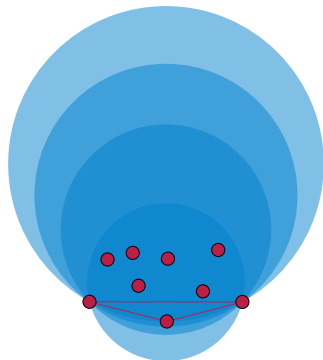
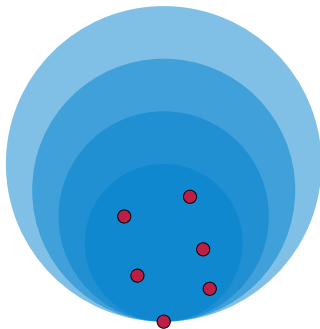
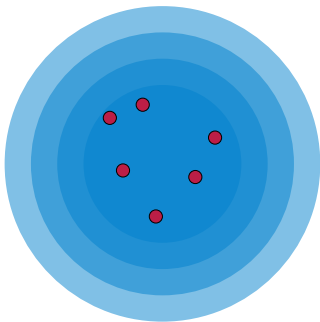
Two points form its diameter



Circle through an acute triangle

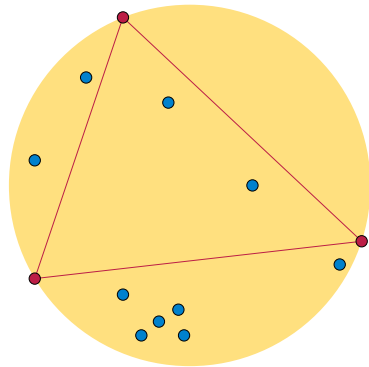
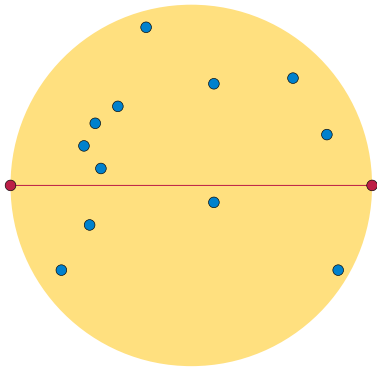
## Why only those two cases?

Any other circle can be shrunk



## Cannot be a linear program

In a  $d$ -dimensional linear program, optimal solution is determined by exactly  $d$  constraints, but here solution sometimes comes from pairs of inputs and sometimes comes from triples



# Circle primitives

## Representation of circle

center, radius<sup>2</sup>

## Does it contain a given point?

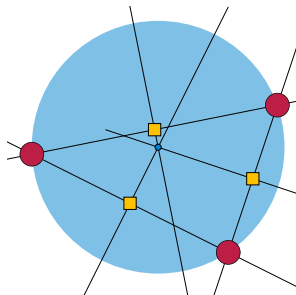
compare point-to-center dist<sup>2</sup>  
to radius<sup>2</sup>

## Circle from two points

center = average coords  
radius<sup>2</sup> = dist<sup>2</sup>/4

## Circle from three points

Construct lines through pairs  
Rotate 90° at midpoints  
They meet at circle center  
radius<sup>2</sup> = dist<sup>2</sup> to any point



## As a nonlinear program

Given points  $x_i, y_i$ :

Find center  $X, Y$   
and squared radius  $R$

Obey nonlinear constraints  
 $(x_i - X)^2 + (y_i - Y)^2 \leq R$

Minimize linear objective  $R$

Define  $S = R - X^2 - Y^2$

Find  $X, Y, S$  with linear  
constraints

$$x_i^2 - 2x_iX + y_i^2 - 2y_iY \leq S$$

Minimize nonlinear objective  
 $S + X^2 + Y^2$

## LP-type problems

# Key properties of circle problem

- ▶ We can describe it as a function mapping **sets** of points to their optimal solution  $(X, Y, R)$
- ▶ Monotonic: We can compare solutions, and if  $A \subset B$  are two sets of points then  $A$ 's solution is at least as good as  $B$ 's
- ▶ Local: If adding  $p$  or  $q$  separately to a set doesn't change its solution, then neither does adding both at once
- ▶ Low-dimensional: Every set has a **basis** of  $\leq 3$  points with same solution (diameter points or acute triangle)
- ▶ Primitives: Find circle defined by two or three points  
"Violation test": is point inside circle?

## LP-type problem

Define a class of problems with the same properties:

- ▶ We can describe it as a function mapping **sets** of inputs to their optimal solution
- ▶ Monotonic: We can compare solutions, and if  $A \subset B$  are two sets of inputs then  $A$ 's solution is at least as good as  $B$ 's
- ▶ Local: If adding  $p$  or  $q$  separately to a set doesn't change its solution, then neither does adding both at once
- ▶ Low-dimensional: Every set has a **basis** of  $O(1)$  inputs with same solution ("dimension": maximum size of a basis)
- ▶ Primitives: Find solution for basis  
"Violation test": does adding  $p$  to  $B$  change its solution?

Goal: Find optimal solution using a small number of primitives

# Linear programming is LP-type

Fix objective function, and consider subsets of constraints. Then:

- ▶ We can describe it as a function mapping **sets** of constraints to their optimal solution points and its objective value
  - ▶ Monotonic: We can compare solutions by their objective value, and if  $A \subset B$  are two sets of constraints then  $A$ 's solution is at least as good as  $B$ 's
  - ▶ Local: If a solution point obeys constraints  $p$  and  $q$  when one of them is added to the set of constraints, it still obeys them when both of them are added.
  - ▶ Low-dimensional: Every set has a **basis** of  $d$  constraints with same solution
  - ▶ Primitives: Solving a basis means turning those inequalities to equalities  $\Rightarrow$  solve system of linear inequalities  $\Rightarrow$  Gaussian elimination
- Violation test: check linear inequality on solution point

**More example problems**

# Closest distance between disjoint convex hulls

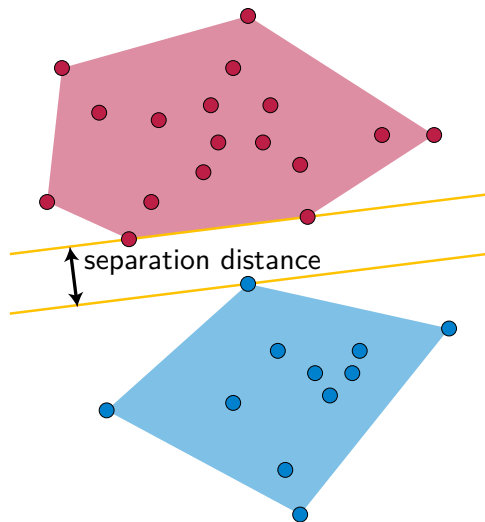
Map subsets to hull distance  
= max separation of parallel lines  
between hulls

Monotonic: More points  $\Rightarrow$  closer hulls

Local: if two points stay outside  
parallel lines when added separately,  
they stay outside when added together

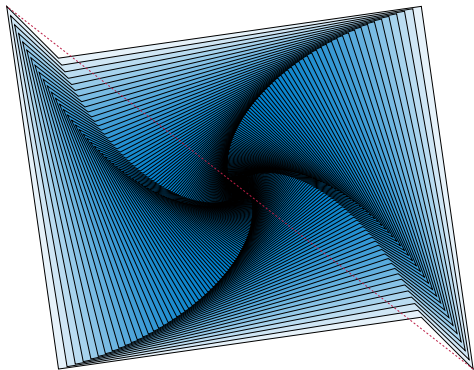
Low-dim: In  $\mathbb{R}^d$ , solution determined  
by  $\leq d + 1$  points

[Matoušek et al. 1996]



Like red-blue separation but Euclidean not vertical distance

# Existence of continuous shrinking motion of polygon



Like star-shaped but more complicated

LP-type with dimension 3

Variables: center point and slope angle  
of logarithmic spirals traced by  
polygon vertices

Hard part: proving that dimension is 3

[Eppstein 2023]

# Integer programming

Find best **integer** solution to a linear program

NP-complete when number of variables can be large,  
even when all variables are restricted to  $\{0, 1\}$

[Karp 1972]

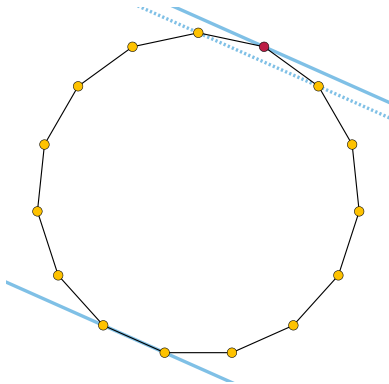
For  $k$  variables, LP-type with dimension  $2^k$

[Bell 1977; Scarf 1977]

## A problem that is not LP-type

Enclose  $n$  points between parallel lines as close together as possible

Like  $L_\infty$  regression, but Euclidean distance not vertical distance



It is monotonic and local

For input = regular  
 $(2n + 1)$ -sided polygon, optimal  
solutions use lines through one  
side and opposite vertex

If any point is removed,  
solution gets better

So all  $2n + 1$  points are needed  
to determine the solution

⇒ not low-dimensional

# Algorithms

## Seidel's algorithm

Instead of restricting recursive subproblems to a linear subspace, we pass them a subset of known basis elements, reducing the LP-type-dimension of the remaining problem by the number of these known elements

Call the following with Seidel(input set, empty set):

```
def Seidel(Inputs, Known):
```

```
    CurrentSolution = solution(Known)
```

```
    AlreadyProcessed = empty set
```

```
    For each element  $x$  of Inputs in random order:
```

```
        If  $x$  fails violation test for CurrentSolution:
```

```
            CurrentSolution = Seidel(AlreadyProcessed, Known  $\cup$   $\{x\}$ )
```

```
        Add  $x$  to AlreadyProcessed
```

```
    Return CurrentSolution
```

# Random sampling

Choose and solve a random subset  $R$  of the items

Let  $V(R)$  = items that fail violation test for solution

- ▶ Each  $x$  in  $V$  must be part of basis for  $V \cup \{x\}$
- ▶ Probability that this is true for  $x$  is  $\leq d/(|R| + 1)$
- ▶ So expected size of  $V(R)$  is  $O(nd/|R|)$
- ▶ If  $V(R)$  is non-empty, it includes at least one member of basis of whole problem

[Clarkson 1995]

## Recursive sampling

$V$  = empty set

repeat  $d$  times or until no more violators are found:

Choose sample  $R$  of size  $d\sqrt{n}$

Compute solution for  $R \cup V$  (recursively using this or another algorithm)

Add its violators to  $V$

Each time through the loop, adds  $O(\sqrt{n})$  elements to  $V$  including at least one more basis element

Solves whole problem in  $O(dn)$  violation tests and  $O(d)$  recursive calls on problems of size  $O(d\sqrt{n})$

When  $n = O(d^2)$ , sample size is already  $n$ , does not make progress

Branching factor of  $d$  quickly blows up; better to use this only once and solve subproblems using a different algorithm

# Iterated reweighting

Give all elements weight 1

Repeat:

Select a random subset of  $2d^2$  elements,  
with probability proportional to their weights

Compute solution for subset and its set  $V$  of violated elements

Double the weights of the members of  $V$

Total expected weight of all items increases by  $(1 + 1/2d)$  factor

Weight of optimal basis increases by larger  $(1 + 1/d)$  factor

Can only happen  $O(d \log n)$  times until subset has bigger weight than whole set  
(impossible) or we find optimal solution

Solves whole problem in  $O(dn \log n)$  violation tests and  $O(d \log n)$  recursive calls on  
problems of size  $O(d^2)$

## Putting it together

Outer level of algorithm: use random sampling

- ▶  $O(dn)$  violation tests
- ▶  $O(d)$  second-level calls on subproblems of size  $O(d\sqrt{n})$

Second-level calls: use iterated reweighting

- ▶ Each subproblem has size  $O(d\sqrt{n})$
- ▶ It does  $O(d^2\sqrt{n} \log n)$  violation tests
- ▶  $O(d \log n)$  third-level calls on subproblems of size  $O(d^2)$

Third-level calls: use Seidel's algorithm

- ▶ Each subproblem has size  $O(d^2)$
- ▶  $O(d! d^2) \Rightarrow d^{O(d)}$  solution primitives

Total:  $O(dn)$  violation tests,  $d^{O(d)} \log n$  solution primitives

Can reduce  $d^{O(d)}$  to  $d^{O(\sqrt{d})}$ : use algorithm of Matoušek et al. [1996] in place of Seidel

## References

- David E. Bell. A theorem concerning the integer lattice. *Studies in Applied Mathematics*, 56(2):187–188, 1977. doi: 10.1002/sapm1977562187.
- Kenneth L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM*, 42(2):488–499, 1995. doi: 10.1145/201019.201036.
- David Eppstein. Locked and unlocked smooth embeddings of surfaces. *Computing in Geometry and Topology*, 2(2):5.1–5.20, 2023. doi: 10.57717/CGT.V2I2.28.
- Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, editors, *Complexity of Computer Computations*, pages 85–103. Plenum, New York, 1972. doi: 10.1007/978-1-4684-2001-2\_9.
- Jiří Matoušek, Micha Sharir, and Emo Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4–5):498–516, 1996. doi: 10.1007/BF01940877.
- Herbert E. Scarf. An observation on the structure of production sets with indivisibilities. *Proc. National Academy of Sciences*, 74(9):3637–3641, 1977. doi: 10.1073/pnas.74.9.3637.
- Micha Sharir and Emo Welzl. A combinatorial bound for linear programming and related problems. In *STACS '92: 9th Annual Symposium on Theoretical Aspects of Computer Science, Cachan, France, February 13-15, 1992, Proceedings*, volume 577 of *Lecture Notes in Computer Science*, pages 567–579. Springer-Verlag, 1992. ISBN 978-3-540-55210-9. doi: 10.1007/3-540-55210-3\_213.