Linear Programming: Linear Programming Formulations

Daniel Kane

Department of Computer Science and Engineering
University of California, San Diego

Advanced Algorithms and Complexity
Data Structures and Algorithms

Learning Objectives

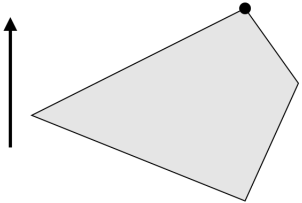
- Distinguish between the different types of linear programming problems.
- Use an algorithm that solves one formulation to solve another formulation.

Formulations

Several different problem types that all go under the heading of "linear programming".

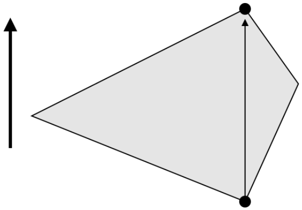
Full Optimization

Minimize or maximize a linear function subject to a system of linear inequality constraints (or say that the constraints have no solution).



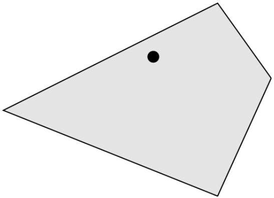
Optimization from Starting Point

Given a system of linear inequalities and a vertex of the polytope they define, optimize a linear function with respect to these constraints.



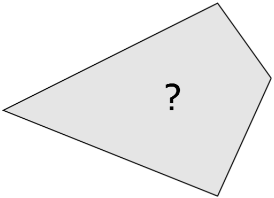
Solution Finding

Given a system of linear inequalities, find some solution.



Satisfiability

Given a system of linear inequalities determine whether or not there is a solution.



Equivalence

Actually, if you can solve any of these problems, you can solve any other!

Full Optimization

Clearly capable of solving all the other versions.

- Start Opt: Ignore starting point.
- Solution Finding: Optimal is a solution.
- Satisfiability: See if finds a solution.

Optimization from Starting Point

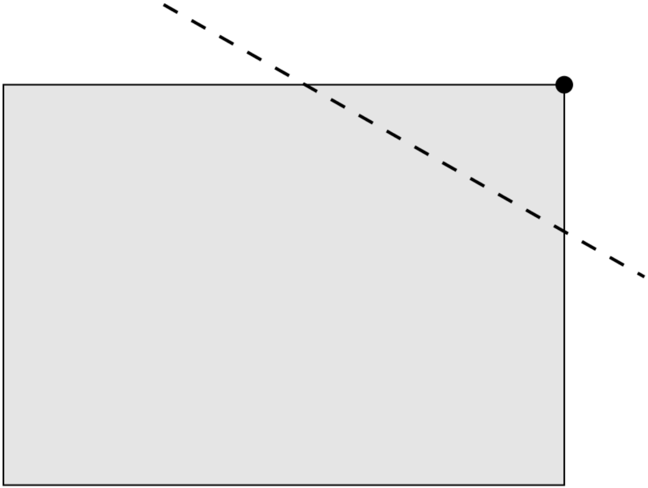
- How do you find starting point?

Optimization from Starting Point

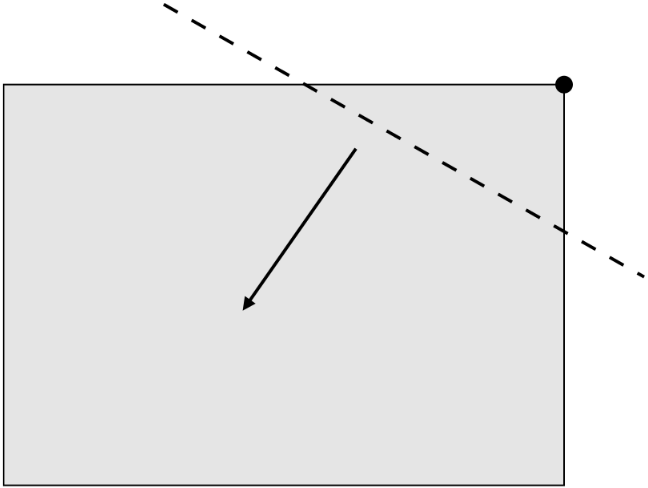
- How do you find starting point?
- Add equations one at a time.
- Optimize left hand side of next equation.

Example

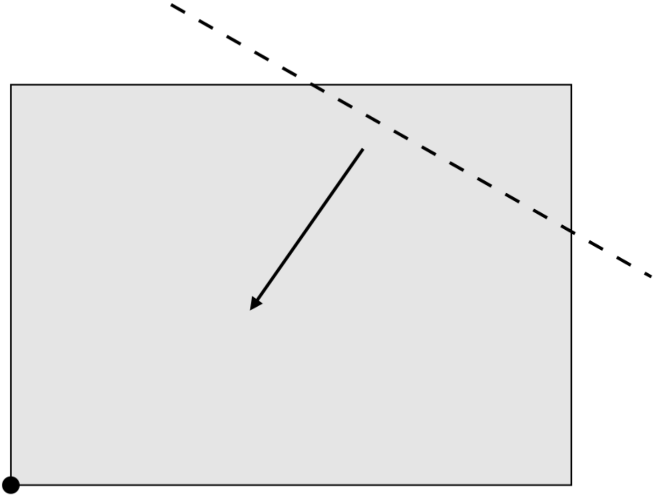
Example



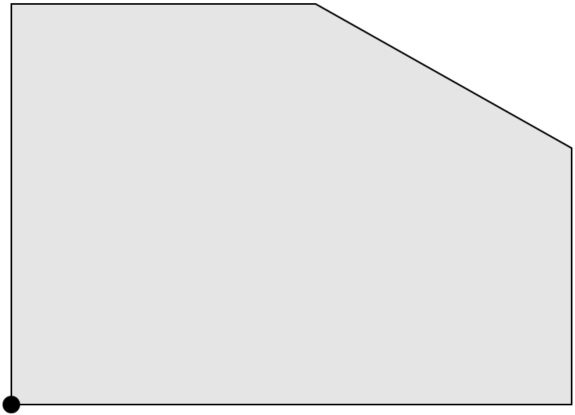
Example



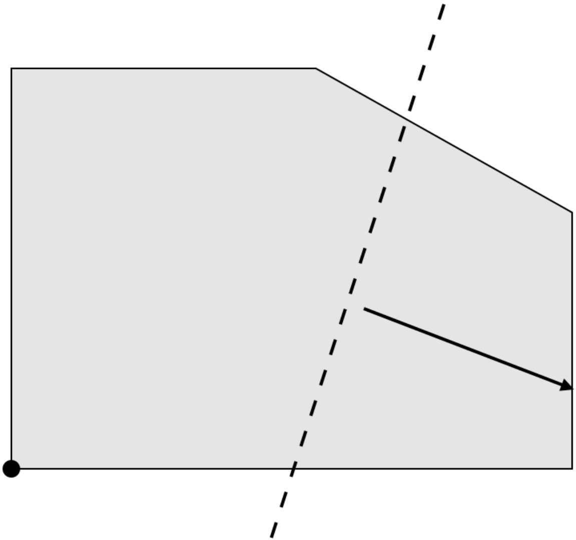
Example



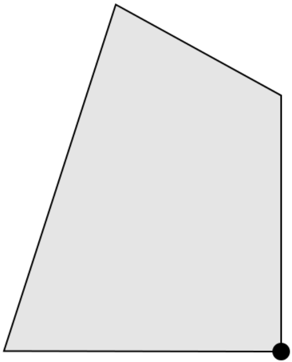
Example



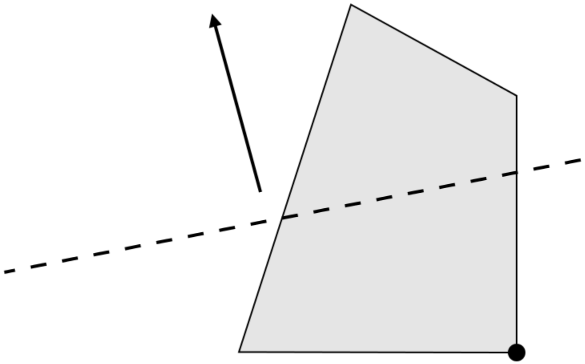
Example



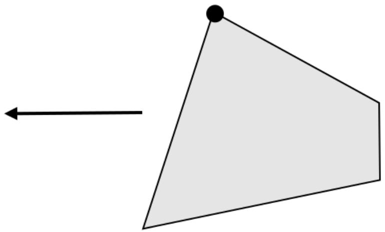
Example



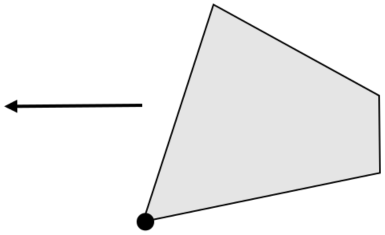
Example



Example



Example



Technical Point

Things are a bit messier if some intermediate systems don't have optima.

Technical Point

Things are a bit messier if some intermediate systems don't have optima.

Fix: start with n constraints (gives a single vertex). Then while trying to add constraint $v \cdot x \geq t$, don't just maximize $v \cdot x$. Also add $v \cdot x \leq t$ as a constraint (so that maximum will exist).

Solution Finding

Q: How do we go from being able to find a solution to finding the best one?

Solution Finding

Q: How do we go from being able to find a solution to finding the best one?
A: Duality.

Solution Finding

Q: How do we go from being able to find a solution to finding the best one?
A: Duality. Find a solution and a matching dual solution.

Setup

Want to minimize $x \cdot v$ subject to $Ax \geq b$.

Setup

Want to minimize $x \cdot v$ subject to $Ax \geq b$.
Instead find solution to:

$$\begin{aligned} Ax &\geq b \\ y &\geq 0 \\ y^T A &= v \\ x \cdot v &= y \cdot b. \end{aligned}$$

Will give optimal solution to original problem.

Satisfiability

How does just knowing when you have a solution help?

Satisfiability

How does just knowing when you have a solution help?

Can always find solution at a vertex. Means n equations are tight.

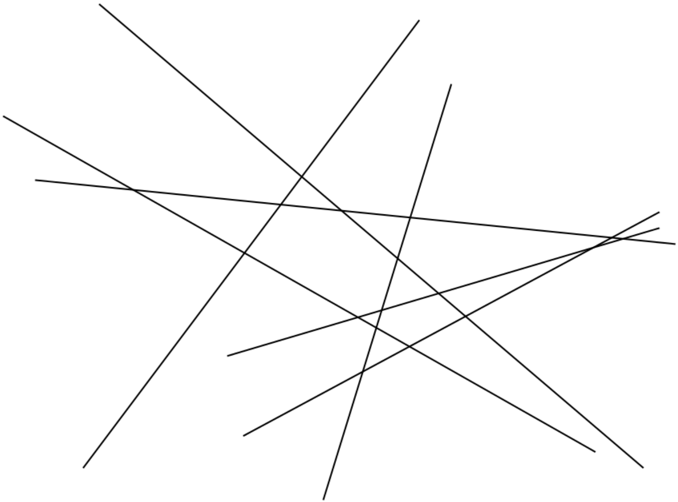
Satisfiability

How does just knowing when you have a solution help?

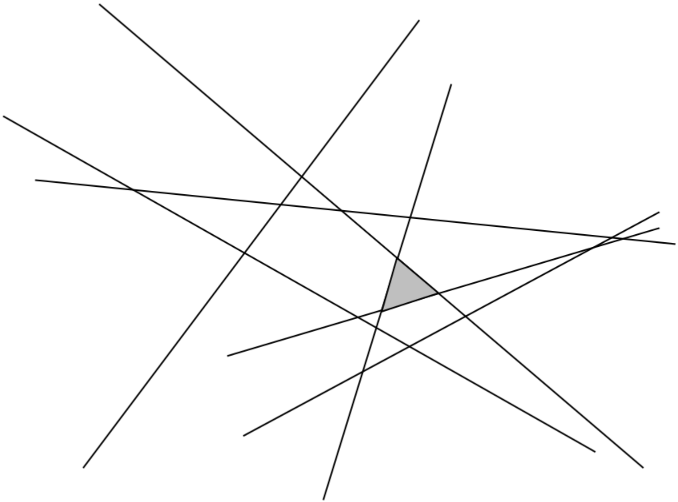
Can always find solution at a vertex. Means n equations are tight.

Figure out which equations to use.

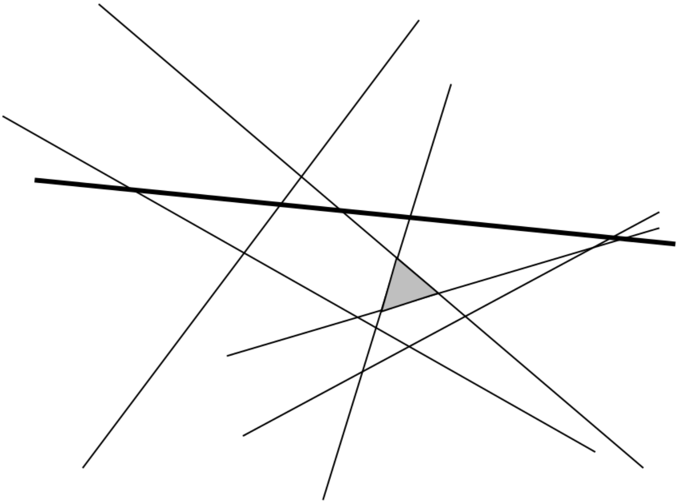
Example



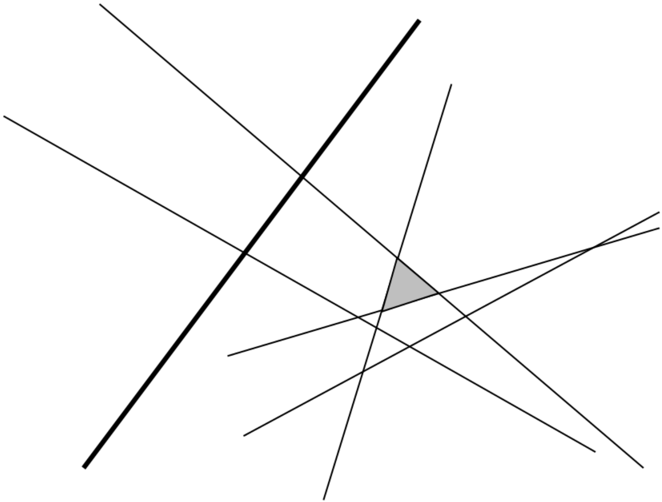
Example



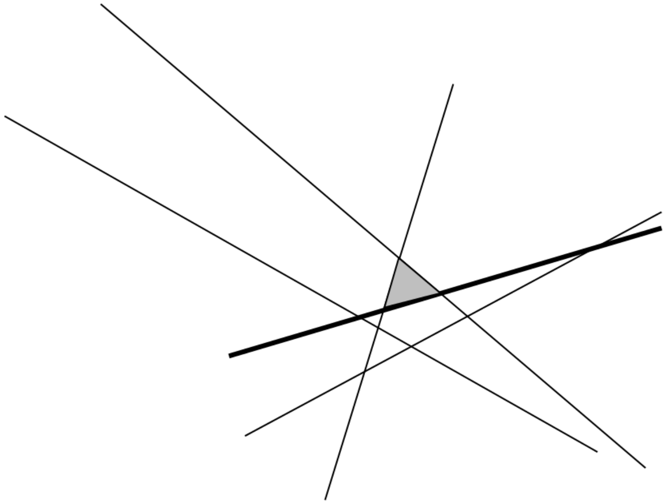
Example



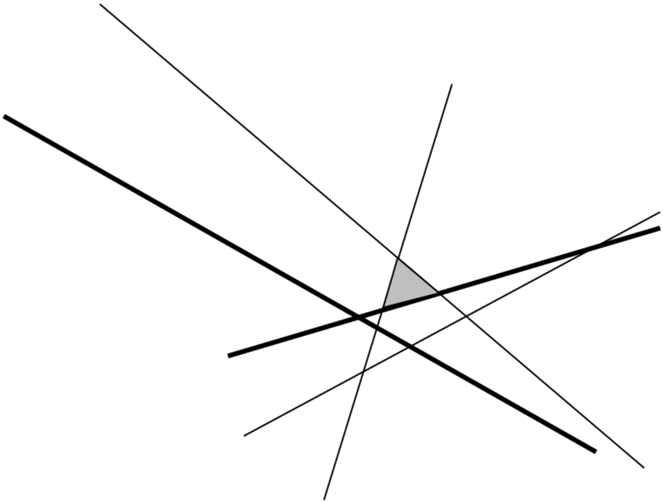
Example



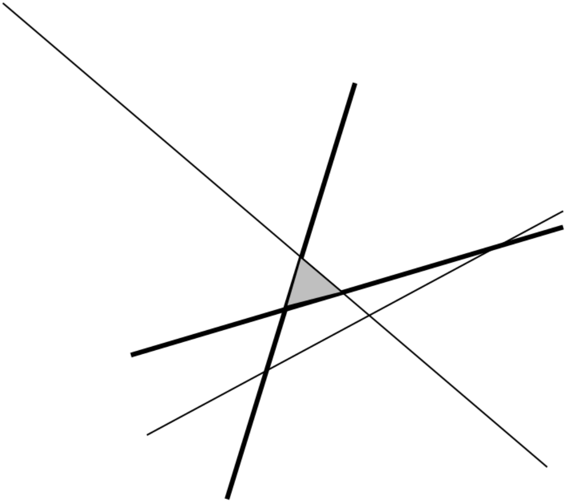
Example



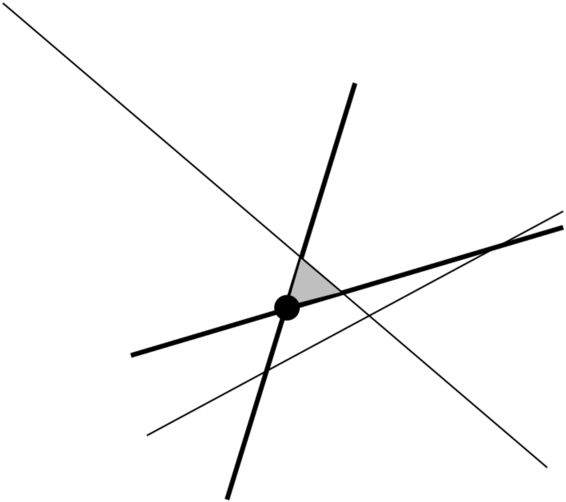
Example



Example



Example



Problem

In order to find a solution to a linear program with m equations in n variables, how many times would one have to call a satisfiability algorithm?

Solution

In order to find a solution to a linear program with m equations in n variables, how many times would one have to call a satisfiability algorithm?

m times. You need to test each equation once, keeping the ones that work.