Linear Programming: Introduction

Daniel Kane

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

Advanced Algorithms and Complexity
Data Structures and Algorithms

Learning Objectives

- See an example of the type of problem solved by linear programming.

Factory

You are running a widget factory and trying to optimize your production procedures to save money.

Machines vs. Workers

Can use combination of machines and workers.

- Have only 100 machines.
- Unlimited workers.
- Each machine requires 2 workers to operate.

Production

- Each machine makes 600 widgets a day.
- Each worker makes 200 widgets a day.

Limited Demand

Total demand for only $100,000$ widgets a day.

Algebra

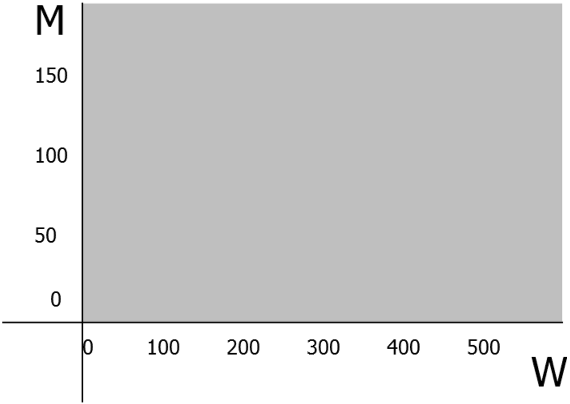
Let W be the number of workers and M the number of machines.

Constraints:

- $W \geq 0$.
- $100 \geq M \geq 0$.
- $W \geq 2M$.
- $100,000 \geq 200(W - 2M) + 600M$.

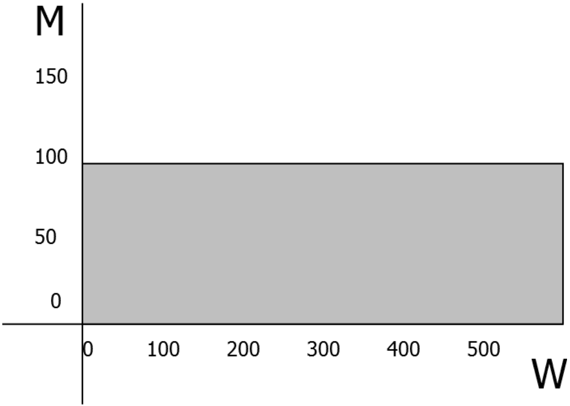
Graph

$$M, W \geq 0$$



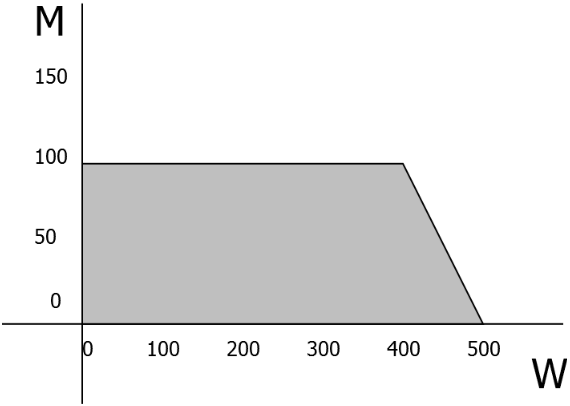
Graph

$M \leq 100$



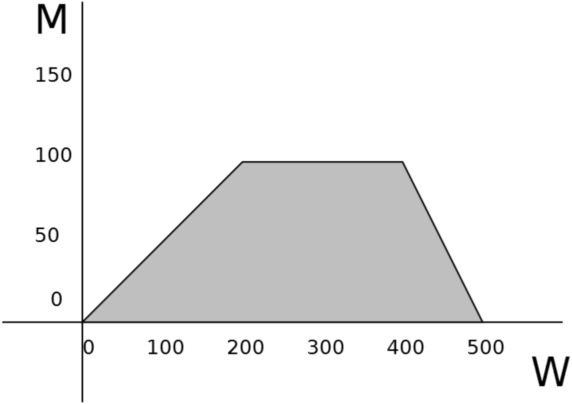
Graph

$$M + W \leq 500$$



Graph

Diagram of possible configurations:



Profits

Profits are determined as follows:

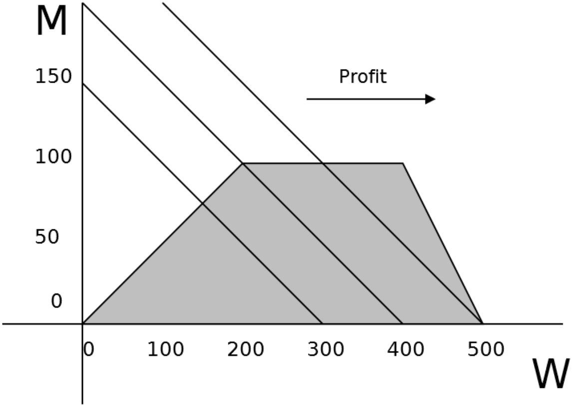
- Each widget earns you \$1.
- Each worker costs you \$100/day.

Total profits (in dollars per day):

$$200(W-2M)+600M-100W = 100W+200M.$$

Graph

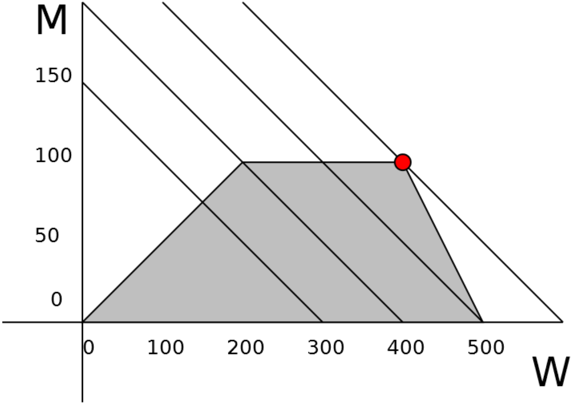
Profit mapped on graph:



Optimum

Best: $M = 100, W = 400$ [NB: A corner]

Profit $= \$60,000/\text{day}$.



Proof of Optimality

$$
\begin{array}{rrcl}
100 \cdot [& 001 \cdot M + 000 \cdot W & \leq & 100] \\
+0.5 \cdot [& 200 \cdot M + 200 \cdot W & \leq & 100,000] \\
\hline
 & 200 \cdot M + 100 \cdot W & \leq & 60,000.
\end{array}
$$

Summary

Maximized:

$$200M + 100W$$

subject to constraints:

$$\begin{aligned}
0M + 1W &\geq 0 \\
1M + 0W &\geq 0 \\
-1M + 0W &\geq -100 \\
-2M + 1W &\geq 0 \\
-1M - 1W &\geq -500
\end{aligned}$$

Linear Programming: Linear Programming

Daniel Kane

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

Advanced Algorithms and Complexity
Data Structures and Algorithms

Learning Objectives

- Understand the formal definition of a linear programming problem.
- Provide some examples of linear programming problems

Last Time

Factory. Set M, W to maximize $200M + 100W$ subject to

- $W \geq 0$.
- $100 \geq M \geq 0$.
- $W \geq 2M$.
- $100,000 \geq 200(W - 2M) + 600M$.

Linear Programming

Linear programming asks for real numbers $x_1, x_2, \ldots, x_n$ satisfying linear inequalities:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &\geq b_1 \\ &\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &\geq b_m \end{aligned}$$

So that a linear objective

$$v_1x_1 + v_2x_2 + \ldots + v_nx_n$$

is as large (or small) as possible.

Notation

Linear Programming

Input: An $m \times n$ matrix A and vectors $b \in \mathbb{R}^m, v \in \mathbb{R}^n$

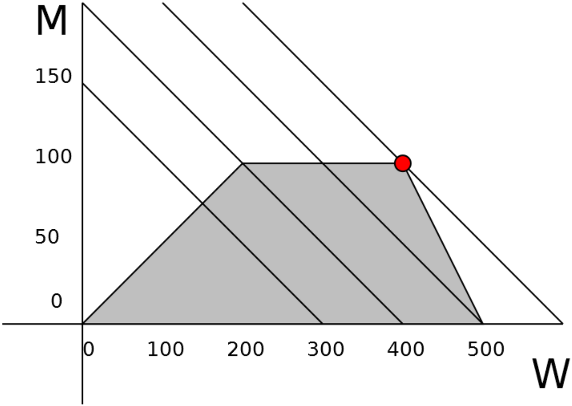
Output: A vector $x \in \mathbb{R}^n$ so that $Ax \geq b$ and $v \cdot x$ is as large (or small) as possible.

Examples

Linear programming is useful because an extraordinary number of problems can be put into this framework.

Factory Example

The factory example we just worked.



The Diet Problem

Studied by George Stigler in the 1930s and 1940s.
How cheaply can you purchase food for a healthy diet?

Variables

You have a number of types of food (bread, milk, apples, etc.).
For each you have a variable giving the number of servings per day.

$$x_{bread}, x_{milk}, x_{apples}, \ldots$$

Constraints

Non-negative number of servings:

$$x_f \geq 0.$$

Constraints

Non-negative number of servings:

$$x_f \geq 0.$$

Sufficient calories/day:

$$\begin{aligned} &(\text{Cal/serving bread})x_{bread} \\ &+(\text{Cal/serving milk})x_{milk} + \ldots \geq 2000. \end{aligned}$$

Constraints

Non-negative number of servings:

$$x_f \geq 0.$$

Sufficient calories/day:

$$\begin{aligned} &(\text{Cal/serving bread})x_{bread} \\ &+(\text{Cal/serving milk})x_{milk} + \ldots \geq 2000. \end{aligned}$$

Similar constraints for other nutritional needs (vitamin C, protein, etc.)

Optimization

Minimize cost.

$$\begin{gathered}(\text{cost of serving bread})x_{bread}\\ +(\text{cost of serving milk})x_{milk} + \ldots\end{gathered}$$

Optimization

Minimize cost.

$$(\text{cost of serving bread})x_{bread}$$
$$+(\text{cost of serving milk})x_{milk} + \ldots$$

Warning: actually doing this can get you some pretty weird diets.

Network Flow

Network flow problems are actually just a special case of linear programming problems!

Network Flow

Variables: f_e for each edge e.

Network Flow

Variables: f_e for each edge e.

Constraints:

$$0 \leq f_e \leq C_e.$$

$$\sum_{e \text{ into } v} f_e - \sum_{e \text{ out of } v} f_e = 0.$$

Network Flow

Variables: f_e for each edge e.

Constraints:

$$0 \leq f_e \leq C_e.$$

$$\sum_{e \text{ into } v} f_e - \sum_{e \text{ out of } v} f_e = 0.$$

Objective:

$$\sum_{e \text{ out of } s} f_e - \sum_{e \text{ into } s} f_e$$

Strange Cases

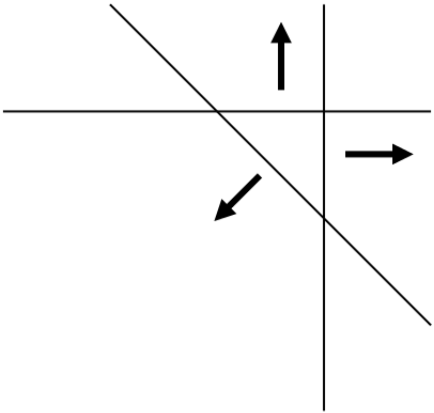
There are a couple of edge cases to keep in mind here.

- No Solution
- No Optimum

No Solution

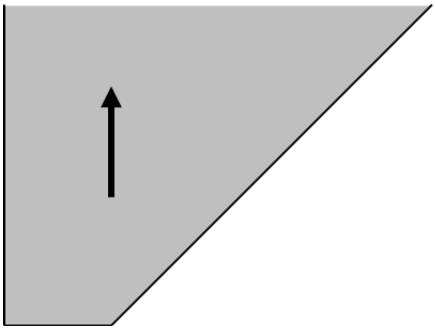
Consider the system:

$x \geq 1, \quad y \geq 1, \quad x + y \leq 1.$



No Optimum

Consider trying to maximize x subject to $x \geq 0$, $y \geq 0$, and $x - y \geq 1$.



Problem

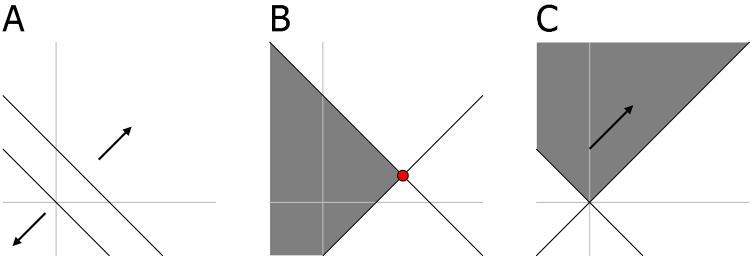
Of these three systems, one has no solution, one has no maximum x value, and one has a maximum. Which is which?

(A) $x + y \geq 1$, $x + y \leq 0$.

(B) $x + y \leq 2$, $x - y \leq 1$.

(C) $x + y \geq 0$, $x - y \leq 0$.

Solution



Linear Programming: Solving Linear Systems

Daniel Kane

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

Advanced Algorithms and Complexity
Data Structures and Algorithms

Learning Objectives

- Solve a system of linear equations.
- Say something about what the set of solution to a system of linear equations looks like.

Last Time

Linear programming: Dealing with systems of linear inequalities.

Linear Algebra

Today, we will deal with the simpler case, of systems of linear equalities.

Linear Algebra

Today, we will deal with the simpler case, of systems of linear equalities.
For example:

$$\begin{aligned} x + y &= 5 \\ 2x + 4y &= 12. \end{aligned}$$

Method of Substitution

- Use first equation to solve for one variable in terms of the others.
- Substitute into other equations.
- Solve recursively.
- Substitute back in to first equation to get initial variable.

Example

$$\begin{aligned} x + y &= 5 \\ 2x + 4y &= 12. \end{aligned}$$

Example

$$\begin{aligned} x + y &= 5 \\ 2x + 4y &= 12. \end{aligned}$$

First equation implies

$$x = 5 - y.$$

Example

$$\begin{aligned} x + y &= 5 \\ 2x + 4y &= 12. \end{aligned}$$

First equation implies

$$x = 5 - y.$$

Substituting into second:

$$12 = 2x + 4y = 2(5 - y) + 4y = 10 + 2y.$$

Example

$$
\begin{aligned}
x + y &= 5 \\
2x + 4y &= 12.
\end{aligned}
$$

First equation implies

$$x = 5 - y.$$

Substituting into second:

$$12 = 2x + 4y = 2(5 - y) + 4y = 10 + 2y.$$

So $y = 1, x = 5 - 1 = 4.$

Problem

What is the value of x in the solution to the following linear system?

$$\begin{aligned} x + 2y &= 6 \\ 3x - y &= -3. \end{aligned}$$

Solution

From the first equation, we get

$$x = 6 - 2y.$$

Solution

From the first equation, we get

$$x = 6 - 2y.$$

Substituting into the second,

$$-3 = 3(6 - 2y) - y = 18 - 7y.$$

Solution

From the first equation, we get

$$x = 6 - 2y.$$

Substituting into the second,

$$-3 = 3(6 - 2y) - y = 18 - 7y.$$

Solving gives, $y = 3$, so $x = 6 - 2 \cdot 3 = 0$.

Another Example

Consider the following system of equations:

$$\begin{aligned} x + y + z &= 5 \\ 2x + y - z &= 1. \end{aligned}$$

Another Example

Consider the following system of equations:

$$\begin{aligned} x + y + z &= 5 \\ 2x + y - z &= 1. \end{aligned}$$

Solve by substitution.

Solution

From first equation:

$$x = 5 - y - z.$$

Solution

From first equation:

$$x = 5 - y - z.$$

Substitute into second.

$$2(5 - y - z) + y - z = 1,$$

or

$$y = 9 + 3z.$$

Cannot Solve for z!

No equations left.

Cannot Solve for z!

No equations left.However, for any z have solution

$$\begin{aligned} y &= 9 + 3z \\ x &= 5 - y - z = -4 - 4z. \end{aligned}$$

Cannot Solve for z!

No equations left.However, for any z have solution

$$\begin{aligned} y &= 9 + 3z \\ x &= 5 - y - z = -4 - 4z. \end{aligned}$$

Have entire family of solutions. z is a free variable.

Degrees of Freedom

- Your solution set will be a subspace.
- Dimension = number of free variables.
- Each equation gives one variable in terms of others.
- Generally, dimension equals

$$\text{num. variables} - \text{num. equations}.$$

Summary

- Can solve systems using method of substitution.
- Each equation reduces degrees of freedom by one.

Next Time

Systematize this to simplify notation and make into an algorithm.

Linear Programming: Gaussian Elimination

Daniel Kane

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

Advanced Algorithms and Complexity
Data Structures and Algorithms

Learning Objectives

- Translate between systems of equations and augmented matrices.
- Row reduce a matrix.
- Write an algorithm to solve linear systems.

Last Time

Solving systems of linear equations by substitution.

Notation

To simplify notation, instead of writing full equations like

$$\begin{aligned} x + y &= 5 \\ 2x + 4y &= 12. \end{aligned}$$

Notation

To simplify notation, instead of writing full equations like

$$\begin{aligned} x + y &= 5 \\ 2x + 4y &= 12. \end{aligned}$$

We just store coefficients of equations in an (augmented) matrix, like so:

$$\begin{array}{ccc} x & y & = 1 \end{array}$$

$$\left[\begin{array}{cc|c} 1 & 1 & 5 \\ 2 & 4 & 12 \end{array}\right]$$

Substitution

How do we solve for x?

Substitution

How do we solve for x? Can't directly have row correspond to $x = 5 - y$. But row $[1\ 1|5]$ corresponds to $x + y = 5$, which is almost as good.

Substitution

How do we solve for x? Can't directly have row correspond to $x = 5 - y$. But row $[1\ 1|5]$ corresponds to $x + y = 5$, which is almost as good.

How do we substitute into second equation?

Substitution

How do we solve for x? Can't directly have row correspond to $x = 5 - y$. But row $[1\ 1|5]$ corresponds to $x + y = 5$, which is almost as good.

How do we substitute into second equation? By subtracting. Subtracting $2(x + y = 5)$ from $(2x + 4y = 12)$ gives $2y = 2$.

Substitution

How do we solve for x? Can't directly have row correspond to $x = 5 - y$. But row $[1\ 1|5]$ corresponds to $x + y = 5$, which is almost as good.

How do we substitute into second equation?

By subtracting. Subtracting $2(x + y = 5)$ from $(2x + 4y = 12)$ gives $2y = 2$.

Subtract twice first row from second.

Basic Row Operations

There are three basic ways to manipulate our matrix. These are called Basic row operations. Each of them gives us an equivalent system of equations.

Adding

Add/subtract a multiple of one row to another.

Adding

Add/subtract a multiple of one row to another.Subtracting twice the first row from second,

$$\left[\begin{array}{cc|c} 1 & 1 & 5 \\ 2 & 4 & 12 \end{array}\right] \to \left[\begin{array}{cc|c} 1 & 1 & 5 \\ 0 & 2 & 2 \end{array}\right]$$

Scaling

Multiply/divide a row by a non-zero constant.

Scaling

Multiply/divide a row by a non-zero constant. Dividing the second row by 2:

$$\left[\begin{array}{cc|c} 1 & 1 & 5 \\ 0 & 2 & 2 \end{array}\right] \to \left[\begin{array}{cc|c} 1 & 1 & 5 \\ 0 & 1 & 1 \end{array}\right]$$

Swapping

Sometimes you want to change the ordering of rows.

Swapping

Sometimes you want to change the ordering of rows. For example, swapping the first and second rows we get

$$\left[\begin{array}{cc|c} 1 & 1 & 5 \\ 0 & 1 & 1 \end{array}\right] \to \left[\begin{array}{cc|c} 0 & 1 & 1 \\ 1 & 1 & 5 \end{array}\right]$$

Row Reduction

Row reduction uses row operations to put a matrix into a simple standard form. The idea is to simulate the substitution method.

Example

Consider the system given by the matrix:

$$\left[\begin{array}{rrrr|r} 2 & 4 & -2 & 0 & 2 \\ -1 & -2 & 1 & -2 & -1 \\ 2 & 2 & 0 & 2 & 0 \end{array}\right]$$

Example

Use first for to solve for first variable.

$$\left[\begin{array}{cccc|c} 2 & 4 & -2 & 0 & 2 \\ -1 & -2 & 1 & -2 & -1 \\ 2 & 2 & 0 & 2 & 0 \end{array}\right]$$

Example

Divide first row by 2.

$$\left[\begin{array}{cccc|c} 1 & 2 & -1 & 0 & 1 \\ -1 & -2 & 1 & -2 & -1 \\ 2 & 2 & 0 & 2 & 0 \end{array}\right]$$

Example

Substitute into other equations.

$$\left[\begin{array}{cccc|c} 1 & 2 & -1 & 0 & 1 \\ -1 & -2 & 1 & -2 & -1 \\ 2 & 2 & 0 & 2 & 0 \end{array}\right]$$

Example

Add first row to second.

$$\left[\begin{array}{rrrr|r} 1 & 2 & -1 & 0 & 1 \\ 0 & 0 & 0 & -2 & 0 \\ 2 & 2 & 0 & 2 & 0 \end{array}\right]$$

Example

Subtract twice first row from third.

$$\left[\begin{array}{rrrr|r} 1 & 2 & -1 & 0 & 1 \\ 0 & 0 & 0 & -2 & 0 \\ 0 & -2 & 2 & 2 & -2 \end{array}\right]$$

Example

Need to solve for next variable.

$$\left[\begin{array}{rrrr|r} 1 & 2 & -1 & 0 & 1 \\ 0 & 0 & 0 & -2 & 0 \\ 0 & -2 & 2 & 2 & -2 \end{array}\right]$$

Example

Cannot use second row.

$$\left[\begin{array}{cccc|c} 1 & 2 & -1 & 0 & 1 \\ 0 & 0 & 0 & -2 & 0 \\ 0 & -2 & 2 & 2 & -2 \end{array}\right]$$

Example

Swap second and third rows.

$$\left[\begin{array}{rrrr|r} 1 & 2 & -1 & 0 & 1 \\ 0 & -2 & 2 & 2 & -2 \\ 0 & 0 & 0 & -2 & 0 \end{array}\right]$$

Example

Solve for second variable.

$$\left[\begin{array}{rrrr|r} 1 & 2 & -1 & 0 & 1 \\ 0 & -2 & 2 & 2 & -2 \\ 0 & 0 & 0 & -2 & 0 \end{array}\right]$$

Example

Divide second row by -2.

$$\left[\begin{array}{rrrr|r} 1 & 2 & -1 & 0 & 1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & -2 & 0 \end{array}\right]$$

Example

Substitute into other equations.

$$\left[\begin{array}{cccc|c} 1 & 2 & -1 & 0 & 1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & -2 & 0 \end{array}\right]$$

Example

Subtract twice second row from first.

$$\left[\begin{array}{cccc|c} 1 & 0 & 1 & 2 & -1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & -2 & 0 \end{array}\right]$$

Example

Can't solve for third variable.

$$\left[\begin{array}{rrrr|r} 1 & 0 & 1 & 2 & -1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & -2 & 0 \end{array}\right]$$

Example

Solve for fourth instead.

$$\left[\begin{array}{cccc|c} 1 & 0 & 1 & 2 & -1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & -2 & 0 \end{array}\right]$$

Example

Divide last row by -2.

$$\left[\begin{array}{cccc|c} 1 & 0 & 1 & 2 & -1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Example

Substitute into other equations.

$$\left[\begin{array}{cccc|c} 1 & 0 & 1 & 2 & -1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Example

Subtract twice third row from first.

$$\left[\begin{array}{cccc|c} 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Example

Add third row to second.

$$\left[\begin{array}{cccc|c} 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Example

Done.

$$\left[\begin{array}{rrrr|r} 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Answer

Our matrix

$$\left[\begin{array}{cccc|c} 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

corresponds to equations:

$$\begin{aligned} x + z &= -1 \\ y - z &= 1 \\ w &= 0. \end{aligned}$$

Solution

So for any value of z, we have solution:

$$\begin{aligned} x &= -1 - z \\ y &= 1 + z \\ w &= 0. \end{aligned}$$

RowReduce(A)

Leftmost non-zero
Swap row to top
Make entry pivot
Rescale to make pivot 1
Subtract row from others to make
other entries in column 0
Repeat

RowReduce(A)

Leftmost non-zero in non-pivot row

Swap row to top of non-pivot rows

Make entry pivot

Rescale to make pivot 1

Subtract row from others to make

other entries in column 0

Repeat until no more non-zero

entries outside of pivot rows

Reading off Answer

- Each row has one pivot and a few other non-pivot entries.
- Gives equation writing pivot variable in terms of non-pivot variables.
- If pivot in units column, have equation $0 = 1$, so no solutions.
- Otherwise, set non-pivot variables to anything, gives answer.

Runtime

- m equations in n variables.
- $\min(n, m)$ pivots.
- For each pivot, need to subtract multiple of row from each other row $O(nm)$ time.
- Total runtime: $O(nm \min(n, m))$.

Linear Programming: Gaussian Elimination

Daniel Kane

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

Advanced Algorithms and Complexity
Data Structures and Algorithms

Learning Objectives

- Solve a system of linear equations.
- Implement a row reduction algorithm.
- Say something about what the set of solution to a system of linear equations looks like.

Last Time

Linear programming: Dealing with systems of linear inequalities.

Linear Algebra

Today, we will deal with the simpler case, of systems of linear equalities.

Linear Algebra

Today, we will deal with the simpler case, of systems of linear equalities.
For example:

$$\begin{aligned} x + y &= 5 \\ 2x + 4y &= 12. \end{aligned}$$

Method of Substitution

- Use first equation to solve for one variable in terms of the others.
- Substitute into other equations.
- Solve recursively.
- Substitute back in to first equation to get initial variable.

Example

$$
\begin{aligned}
x + y &= 5 \\
2x + 4y &= 12.
\end{aligned}
$$

Example

$$\begin{aligned} x + y &= 5 \\ 2x + 4y &= 12. \end{aligned}$$

First equation implies

$$x = 5 - y.$$

Example

$$\begin{aligned} x + y &= 5 \\ 2x + 4y &= 12. \end{aligned}$$

First equation implies

$$x = 5 - y.$$

Substituting into second:

$$12 = 2x + 4y = 2(5 - y) + 4y = 10 + 2y.$$

Example

$$\begin{aligned} x + y &= 5 \\ 2x + 4y &= 12. \end{aligned}$$

First equation implies

$$x = 5 - y.$$

Substituting into second:

$$12 = 2x + 4y = 2(5 - y) + 4y = 10 + 2y.$$

So $y = 1, x = 5 - 1 = 4$.

Problem

What is the value of x in the solution to the following linear system?

$$\begin{aligned} x + 2y &= 6 \\ 3x - y &= -3. \end{aligned}$$

Solution

From the first equation, we get

$$x = 6 - 2y.$$

Solution

From the first equation, we get

$$x = 6 - 2y.$$

Substituting into the second,

$$-3 = 3(6 - 2y) - y = 18 - 7y.$$

Solution

From the first equation, we get

$$x = 6 - 2y.$$

Substituting into the second,

$$-3 = 3(6 - 2y) - y = 18 - 7y.$$

Solving gives, $y = 3$, so $x = 6 - 2 \cdot 3 = 0$.

Notation

To simplify notation, instead of writing full equations like

$$\begin{aligned} x + y &= 5 \\ 2x + 4y &= 12. \end{aligned}$$

Notation

To simplify notation, instead of writing full equations like

$$\begin{aligned} x + y &= 5 \\ 2x + 4y &= 12. \end{aligned}$$

We just store coefficients of equations in an (augmented) matrix, like so:

$$\begin{array}{cc} & \begin{array}{cccc} x & y & = & 1 \end{array} \\ & \left[\begin{array}{cc|c} 1 & 1 & 5 \\ 2 & 4 & 12 \end{array}\right] \end{array}$$

Substitution

How do we solve for x?

Substitution

How do we solve for x? Can't directly have row correspond to $x = 5 - y$. But row $[1\ 1|5]$ corresponds to $x + y = 5$, which is almost as good.

Substitution

How do we solve for x? Can't directly have row correspond to $x = 5 - y$. But row $[1\ 1|5]$ corresponds to $x + y = 5$, which is almost as good.

How do we substitute into second equation?

Substitution

How do we solve for x? Can't directly have row correspond to $x = 5 - y$. But row $[1\ 1|5]$ corresponds to $x + y = 5$, which is almost as good.

How do we substitute into second equation? By subtracting. Subtracting $2(x + y = 5)$ from $(2x + 4y = 12)$ gives $2y = 2$.

Substitution

How do we solve for x? Can't directly have row correspond to $x = 5 - y$. But row $[1\ 1|5]$ corresponds to $x + y = 5$, which is almost as good.

How do we substitute into second equation? By subtracting. Subtracting $2(x + y = 5)$ from $(2x + 4y = 12)$ gives $2y = 2$.

Subtract twice first row from second.

Basic Row Operations

There are three basic ways to manipulate our matrix. These are called Basic row operations. Each of them gives us an equivalent system of equations.

Adding

Add/subtract a multiple of one row to another.

Adding

Add/subtract a multiple of one row to another.Subtracting twice the first row from second,

$$\left[\begin{array}{cc|c} 1 & 1 & 5 \\ 2 & 4 & 12 \end{array}\right] \rightarrow \left[\begin{array}{cc|c} 1 & 1 & 5 \\ 0 & 2 & 2 \end{array}\right]$$

Scaling

Multiply/divide a row by a non-zero constant.

Scaling

Multiply/divide a row by a non-zero constant. Dividing the second row by 2:

$$\left[\begin{array}{cc|c} 1 & 1 & 5 \\ 0 & 2 & 2 \end{array}\right] \rightarrow \left[\begin{array}{cc|c} 1 & 1 & 5 \\ 0 & 1 & 1 \end{array}\right]$$

Swapping

Sometimes you want to change the ordering of rows.

Swapping

Sometimes you want to change the ordering of rows. For example, swapping the first and second rows we get

$$\left[\begin{array}{cc|c} 1 & 1 & 5 \\ 0 & 1 & 1 \end{array}\right] \to \left[\begin{array}{cc|c} 0 & 1 & 1 \\ 1 & 1 & 5 \end{array}\right]$$

Row Reduction

Row reduction uses row operations to put a matrix into a simple standard form. The idea is to simulate the substitution method.

Example

Consider the system given by the matrix:

$$\left[\begin{array}{cccc|c} 2 & 4 & -2 & 0 & 2 \\ -1 & -2 & 1 & -2 & -1 \\ 2 & 2 & 0 & 2 & 0 \end{array}\right]$$

Example

Use first for to solve for first variable.

$$\left[\begin{array}{rrrr|r} 2 & 4 & -2 & 0 & 2 \\ -1 & -2 & 1 & -2 & -1 \\ 2 & 2 & 0 & 2 & 0 \end{array}\right]$$

Example

Divide first row by 2.

$$\left[\begin{array}{rrrr|r} 1 & 2 & -1 & 0 & 1 \\ -1 & -2 & 1 & -2 & -1 \\ 2 & 2 & 0 & 2 & 0 \end{array}\right]$$

Example

Substitute into other equations.

$$\left[\begin{array}{rrrr|r} 1 & 2 & -1 & 0 & 1 \\ -1 & -2 & 1 & -2 & -1 \\ 2 & 2 & 0 & 2 & 0 \end{array}\right]$$

Example

Add first row to second.

$$\left[\begin{array}{cccc|c} 1 & 2 & -1 & 0 & 1 \\ 0 & 0 & 0 & -2 & 0 \\ 2 & 2 & 0 & 2 & 0 \end{array}\right]$$

Example

Subtract twice first row from third.

$$\left[\begin{array}{rrrr|r} 1 & 2 & -1 & 0 & 1 \\ 0 & 0 & 0 & -2 & 0 \\ 0 & -2 & 2 & 2 & -2 \end{array}\right]$$

Example

Need to solve for next variable.

$$\left[\begin{array}{rrrr|r} 1 & 2 & -1 & 0 & 1 \\ 0 & 0 & 0 & -2 & 0 \\ 0 & -2 & 2 & 2 & -2 \end{array}\right]$$

Example

Cannot use second row.

$$\left[\begin{array}{rrrr|r} 1 & 2 & -1 & 0 & 1 \\ 0 & 0 & 0 & -2 & 0 \\ 0 & -2 & 2 & 2 & -2 \end{array}\right]$$

Example

Swap second and third rows.

$$\left[\begin{array}{cccc|c} 1 & 2 & -1 & 0 & 1 \\ 0 & -2 & 2 & 2 & -2 \\ 0 & 0 & 0 & -2 & 0 \end{array}\right]$$

Example

Solve for second variable.

$$\left[\begin{array}{rrrr|r} 1 & 2 & -1 & 0 & 1 \\ 0 & -2 & 2 & 2 & -2 \\ 0 & 0 & 0 & -2 & 0 \end{array}\right]$$

Example

Divide second row by -2.

$$\left[\begin{array}{cccc|c} 1 & 2 & -1 & 0 & 1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & -2 & 0 \end{array}\right]$$

Example

Substitute into other equations.

$$\left[\begin{array}{cccc|c} 1 & 2 & -1 & 0 & 1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & -2 & 0 \end{array}\right]$$

Example

Subtract twice second row from first.

$$\left[\begin{array}{rrrr|r} 1 & 0 & 1 & 2 & -1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & -2 & 0 \end{array}\right]$$

Example

Can't solve for third variable.

$$\left[\begin{array}{rrrr|r} 1 & 0 & 1 & 2 & -1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & -2 & 0 \end{array}\right]$$

Example

Solve for fourth instead.

$$\left[\begin{array}{cccc|c} 1 & 0 & 1 & 2 & -1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & -2 & 0 \end{array}\right]$$

Example

Divide last row by -2.

$$\left[\begin{array}{rrrr|r} 1 & 0 & 1 & 2 & -1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Example

Substitute into other equations.

$$\left[\begin{array}{rrrr|r} 1 & 0 & 1 & 2 & -1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Example

Subtract twice third row from first.

$$\left[\begin{array}{rrrr|r} 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Example

Add third row to second.

$$\left[\begin{array}{cccc|c} 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Example

Done.

$$\left[\begin{array}{cccc|c} 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Answer

Our matrix

$$\left[\begin{array}{cccc|c} 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

corresponds to equations:

$$\begin{aligned} x + z &= -1 \\ y - z &= 1 \\ w &= 0. \end{aligned}$$

Solution

So for any value of z, we have solution:

$$\begin{aligned} x &= -1 - z \\ y &= 1 + z \\ w &= 0. \end{aligned}$$

RowReduce(A)

Leftmost non-zero

Swap row to top

Make entry pivot

Rescale to make pivot 1

Subtract row from others to make other entries in column 0

Repeat

RowReduce(A)

Leftmost non-zero in non-pivot row

Swap row to top of non-pivot rows

Make entry pivot

Rescale to make pivot 1

Subtract row from others to make other entries in column 0

Repeat until no more non-zero entries outside of pivot rows

Reading off Answer

- Each row has one pivot and a few other non-pivot entries.
- Gives equation writing pivot variable in terms of non-pivot variables.
- If pivot in units column, have equation $0 = 1$, so no solutions.
- Otherwise, set non-pivot variables to anything, gives answer.

Degrees of Freedom

- Your solution set will be a subspace.
- Dimension = number of non-pivot variables.
- Or n minus the number of pivot variables.
- Generally, dimension equals

$$\text{num. variables} - \text{num. equations}.$$

Runtime

- m equations in n variables.
- $\min(n, m)$ pivots.
- For each pivot, need to subtract multiple of row from each other row $O(nm)$ time.
- Total runtime: $O(nm \min(n, m))$.

Linear Programming: Convex Polytopes

Daniel Kane

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

Advanced Algorithms and Complexity
Data Structures and Algorithms

Learning Objectives

- Understand what a convex polytope is and why it is relevant to linear programming.
- Get a feel for what a convex polytope looks like.
- Prove some basic facts about convex polytopes.

Linear Programs

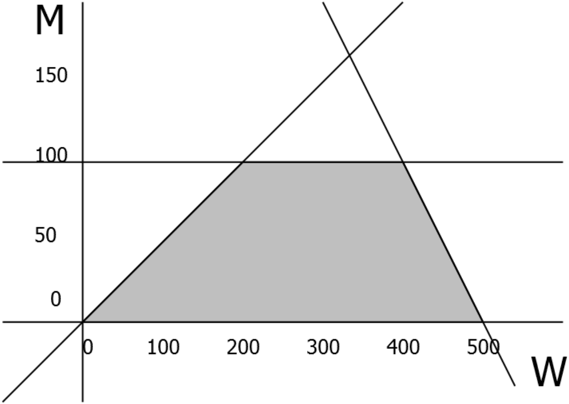
Optimize linear function given linear inequality constraints.

Linear Programs

Optimize linear function given linear inequality constraints.
Want to understand region of points defined by inequalities.

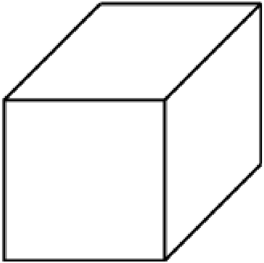
Example

From factory example



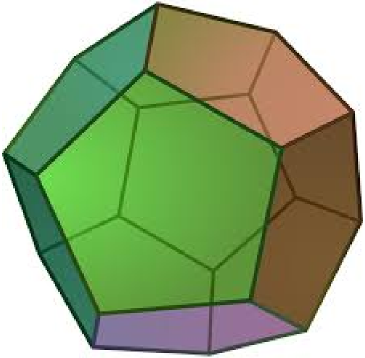
Example II

Equations: $0 \le x, y, z \le 1$ give cube.



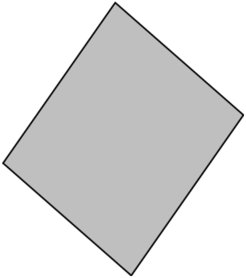
In General

Get what's called a convex polytope



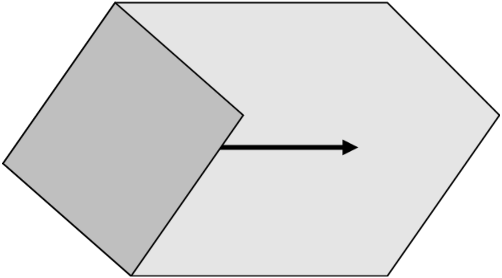
Hyperplanes

A single linear equation defines a hyperplane.



Hyperplanes

A single linear equation defines a hyperplane.



An inequality, defines a halfspace.

Polytopes

So a system of linear inequalities, defines a region bounded by a bunch of hyperplanes.

Definition

A polytope is a region in $\mathbb{R}^n$ bounded by finitely many flat surfaces.

Polytopes

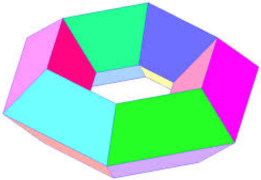
So a system of linear inequalities, defines a region bounded by a bunch of hyperplanes.

Definition

A polytope is a region in $\mathbb{R}^n$ bounded by finitely many flat surfaces. These surfaces may intersect in lower dimensional facets (like edges), with zero-dimensional facets called vertices.

More Conditions

But not every polytope is possible.

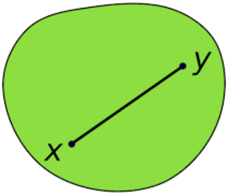


Everything must be on one side of each face.

Convexity

Definition

A region $\mathcal{C} \subset \mathbb{R}^n$ is convex, if for any $x, y \in \mathcal{C}$, the line segment connecting x and y is contained in $\mathcal{C}$.



Convexity

Lemma

An intersection of halfspaces is convex.

Proof

- Defined by $Ax \geq b$.
- Need for $x, y \in \mathcal{C}$ and $t \in [0, 1]$, $tx + (1-t)y \in \mathcal{C}$.

$$
\begin{aligned}
A(tx + (1-t)y) &= tAx + (1-t)Ay \\
&\geq tb + (1-t)b \\
&= b.
\end{aligned}
$$

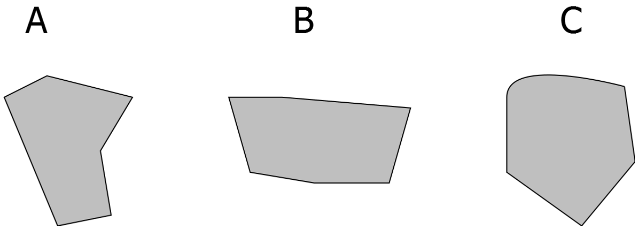
Convex Polytope

Theorem

The region defined by a system of linear inequalities is always a convex polytope.

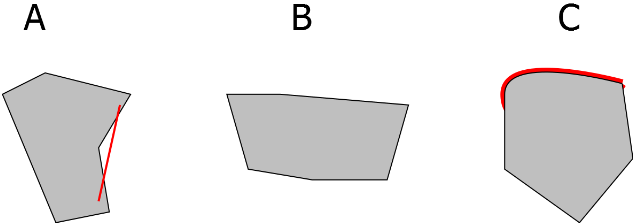
Problem

Which of these figures is a convex polytope?



Solution

Only B.



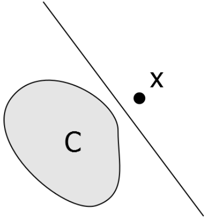
Lemmas

We will conclude with a couple important lemmas about convex polytopes.

Separation

Lemma

Let $\mathcal{C}$ be a convex region and $x \notin \mathcal{C}$ a point. Then there is a hyperplane H separating x from $\mathcal{C}$.



Separation

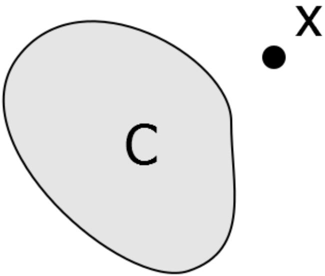
Lemma

Let $\mathcal{C}$ be a convex region and $x \notin \mathcal{C}$ a point. Then there is a hyperplane H separating x from $\mathcal{C}$.

Note that if $\mathcal{C}$ is given by a system of linear inequalities, we can just find one of the defining inequalities that x violates.

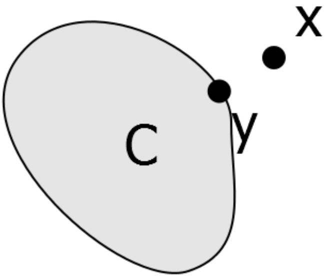
Proof (Optional)

Start with x.



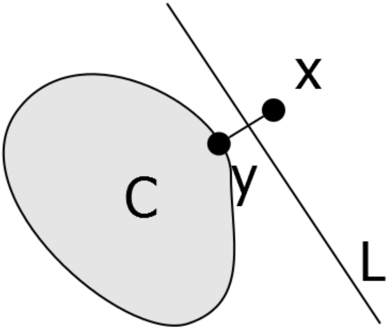
Proof (Optional)

Let y be closest point in $\mathcal{C}$



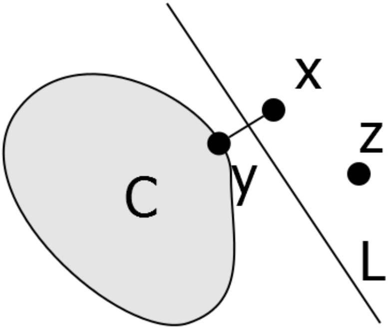
Proof (Optional)

Let L be the perpendicular bisector of xy.



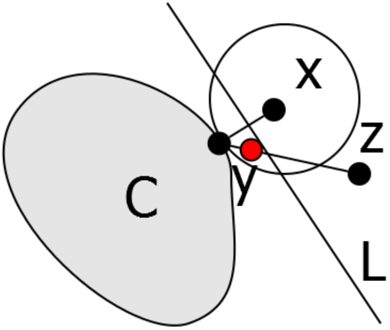
Proof (Optional)

If $z \in \mathcal{C}$ on wrong side of L,



Proof (Optional)

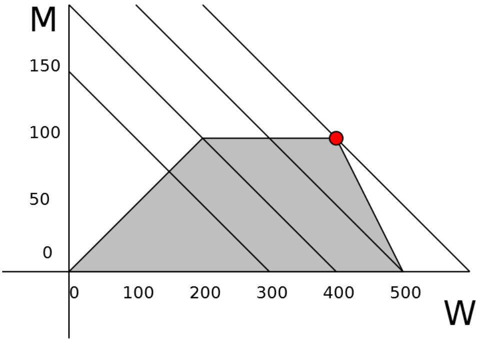
yz contains point closer to x. Contradiction.



Extreme Points

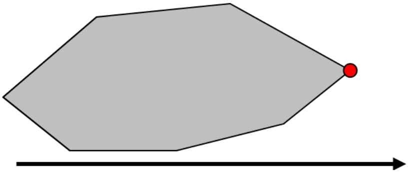
Lemma

A linear function on a polytope takes its minimum/maximum values on vertices.



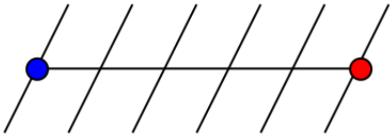
Intuition

The corners are the only extreme points.
Optima must be there.



Idea

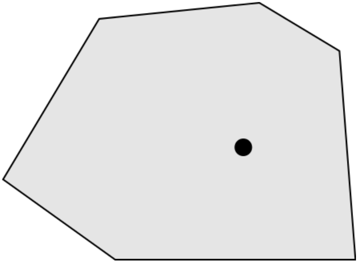
Linear function on segment takes extreme values on ends.



Use to push towards corners.

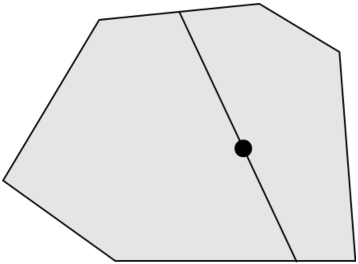
Proof

Start at any point.



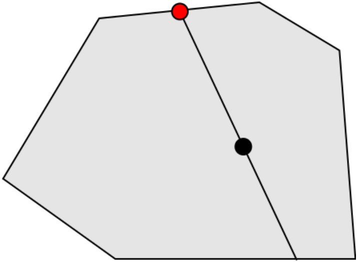
Proof

Pick line through point.



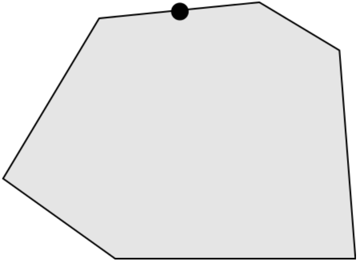
Proof

Extreme values at endpoint.



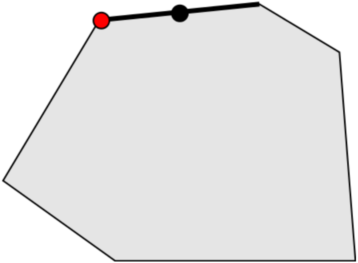
Proof

Point is on a facet.



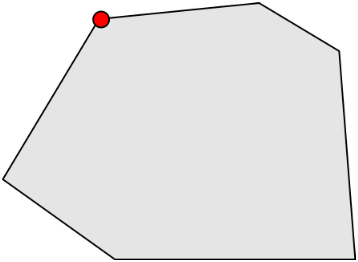
Proof

Repeat to push to lower dimensional facet.



Proof

Eventually on a vertex.



Summary

- Region determined by LP always convex polytope.
- Optimum always at vertex.
- Can separate from outside points by hyperplanes.

Linear Programming: Duality

Daniel Kane

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

Advanced Algorithms and Complexity
Data Structures and Algorithms

Learning Objectives

- Write the dual program of a linear program.
- Understand the duality theorem.

Example

Recall our first example:

Maximize $200M + 100W$ subject to:

- $W \geq 0$.
- $100 \geq M \geq 0$.
- $W \geq 2M$.
- $100,000 \geq 200(W - 2M) + 600M$.

Upper Bound

The best you could do was 60000, but we proved it by combining constraints

$$
\begin{array}{rlcl}
100 \cdot [& 001 \cdot M + 000 \cdot W & \leq 100] \\
+0.5 \cdot [& 200 \cdot M + 200 \cdot W & \leq 100,000] \\
\hline
 & 200 \cdot M + 100 \cdot W & \leq 60,000.
\end{array}
$$

General Technique

Try to prove bound by combining the constraints together.

Linear Program

Say you have the linear program where you want to minimize

$$v_1 x_1 + v_2 x_2 + \ldots + v_n x_n$$

subject to constraints

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &\geq b_1 \\ &\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &\geq b_m \end{aligned}$$

Combine Constraints

If we have $c_i \geq 0$, we can combine constraints:

$$
\begin{array}{rcl}
c_1 \cdot [a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n & \geq & b_1] \\
\ldots & & \\
+c_m \cdot [a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n & \geq & b_m] \\
\hline
w_1x_1 + w_2x_2 + \ldots + w_nx_n & \geq & t,
\end{array}
$$

$w_i = \sum c_j a_{ji}$, $t = \sum c_j b_j$.

Bound

If $w_i = v_i$ for all i, have

$$v_1 x_1 + v_2 x_2 + \ldots + v_n x_n \geq t.$$

Bound

If $w_i = v_i$ for all i, have

$$v_1 x_1 + v_2 x_2 + \ldots + v_n x_n \geq t.$$

Want to find $c_i \geq 0$ so that $v_i = \sum_{j=1}^{m} c_j a_{ji}$ for all i, and $t = \sum_{j=1}^{m} c_j b_j$ is as large as possible.

Linear Program

Note that this is another linear program.
Find $c \in \mathbb{R}^m$ so that $\sum_{j=1}^{m} c_j b_j$ is as large as possible, subject to the linear inequalities $c_i \geq 0$, and equalities

$$v_i = \sum_{j=1}^{m} c_j a_{ji}.$$

Dual Program

Definition

Given the linear program (the primal):

Minimize $v \cdot x$

Subject to $Ax \geq b$

The dual linear program is the linear program:

Maximize $y \cdot b$

Subject to $y^T A = v$, and $y \geq 0$.

Bounds

It is not hard to show that a solution to the dual bounds the optimum for the primal.

Bounds

It is not hard to show that a solution to the dual bounds the optimum for the primal.
If $y \geq 0$ and $y^T A = v$,
then for any x with $Ax \geq b$,

$$x \cdot v = y^T A x \geq y^T b = y \cdot b.$$

Bounds

It is not hard to show that a solution to the dual bounds the optimum for the primal.
If $y \geq 0$ and $y^T A = v$,
then for any x with $Ax \geq b$,

$$x \cdot v = y^T Ax \geq y^T b = y \cdot b.$$

The surprising thing is that these two linear programs always have the same solution.

Duality

Theorem

A linear program and its dual always have the same (numerical) answer.

Duality

Theorem

A linear program and its dual always have the same (numerical) answer.

This means that one can instead solve the dual problem. This is sometimes easier, and often provides insight into the solution.

Example: Flows

The size of our flow is

$$\sum_{e \text{ out of a source}} f_e - \sum_{e \text{ into a source}} f_e.$$

Example: Flows

The size of our flow is

$$\sum_{e \text{ out of a source}} f_e - \sum_{e \text{ into a source}} f_e.$$

By adding multiples of the conservation of flow equation, this is

$$\sum_v c_v \left(\sum_{e \text{ out of } v} f_e - \sum_{e \text{ into } v} f_e \right)$$

where $c_s = 1, c_t = 0$.

Example: Flows

This is
$$\sum_{e=(v,w)} (c_v - c_w) f_e.$$

Example: Flows

This is

$$\sum_{e=(v,w)} (c_v - c_w) f_e.$$

We can bound this using capacity constraints as

$$\sum_{e=(v,w)} C_e \max(c_v - c_w, 0).$$

Example: Flows

It is not hard to show that minimum attained when $c_v \in \{0, 1\}$.

Example: Flows

It is not hard to show that minimum attained when $c_v \in \{0, 1\}$.
Letting $\mathcal{C}$, be the set of vertices where $c_v = 1$, our bound is

$$\sum_{e=(v,w), v \in \mathcal{C}, w \notin \mathcal{C}} C_e = |\mathcal{C}|.$$

Example: Flows

It is not hard to show that minimum attained when $c_v \in \{0,1\}$.
Letting $\mathcal{C}$, be the set of vertices where $c_v = 1$, our bound is

$$\sum_{e=(v,w),v\in\mathcal{C},w\notin\mathcal{C}} C_e = |\mathcal{C}|.$$

The dual program just finds the minimum cut!

Example: Diet Problem

Recall the diet problem:

- Minimize the cost of the foods you need to buy subject to:
- Meets daily requirements for various nutrients
- Non-negative amount of each type of food

Example: Diet Problem

Recall the diet problem:

- Minimize the cost of the foods you need to buy subject to:
- Meets daily requirements for various nutrients
- Non-negative amount of each type of food

What is the dual program?

Example: Diet Problem

For each nutrient N, use a multiple C_N of the equation for that nutrient.
Can then add multiples of the constraint that you get a non-negative amount of each food.

Example: Diet Problem

Think of C_N as a cost of nutrient N. We pick values so that for each food item, f, we have

$$\mathrm{Cost}(f) \geq \sum_N C_N \cdot (\text{Amount of nutrient } N \text{ in } f).$$

Example: Diet Problem

Think of C_N as a cost of nutrient N. We pick values so that for each food item, f, we have

$$\mathrm{Cost}(f) \geq \sum_N C_N \cdot (\text{Amount of nutrient } N \text{ in } f).$$

Costs C_N to get a unit of nutrient N. This means total cost of a balanced diet is at least

$$\sum_N C_N \cdot (\text{Required amount of nutrient } N).$$

Observation

Note that if you want to actually obtain this lower bound, you cannot buy overpriced foods. Can only afford to buy foods with

$$\mathrm{Cost}(f) = \sum_N C_N \cdot (\text{Amount of nutrient } N \text{ in } f).$$

Observation

Note that if you want to actually obtain this lower bound, you cannot buy overpriced foods. Can only afford to buy foods with

$$\text{Cost}(f) = \sum_N C_N \cdot (\text{Amount of nutrient } N \text{ in } f).$$

This is an example of a general phenomena called complementary slackness.

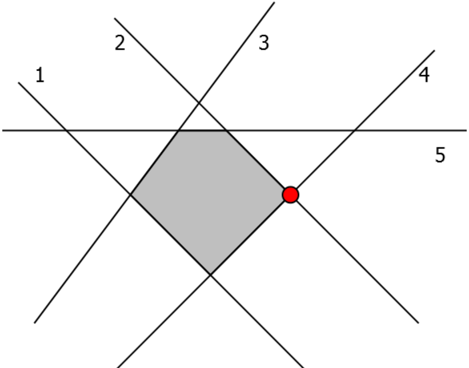
Complementary Slackness

Theorem

Consider a primal LP:
Minimize $v \cdot x$ subject to $Ax \geq b$,
and its dual LP:
Maximize $y \cdot b$ subject to $y^T A = v$, $y \geq 0$.
Then in the solutions, $y_i > 0$ only if the i^{th} equation in x is tight.

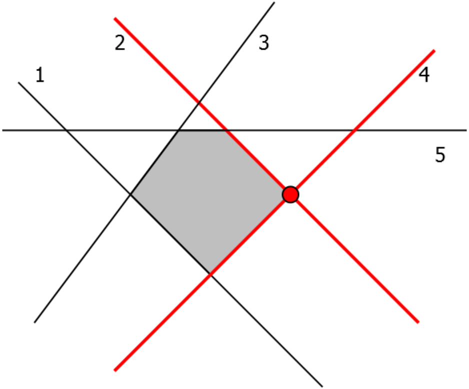
Problem

Assuming that the highlighted point is the optimum to the linear program below, which equations might have non-zero coefficients in the solution to the dual program?



Solution

Only 2 and 4.



Summary

- Every LP has dual LP.
- Solutions to dual bound solutions to primal.
- LP and dual have same answer!
- Complementary slackness.

Linear Programming: (Optional) Duality Proofs

Daniel Kane

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

Advanced Algorithms and Complexity
Data Structures and Algorithms

Learning Objectives

- Prove the duality and complementary slackness theorems.

Last Time

To each linear program, associate a dual program. Find non-negative combination of constraints to put bound on objective.

Duality

Theorem

A linear program and its dual always have the same (numerical) answer.

Duality

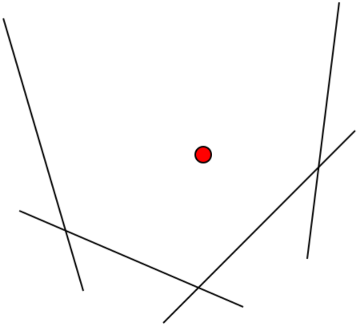
Theorem

A linear program and its dual always have the same (numerical) answer.

Today we prove it.

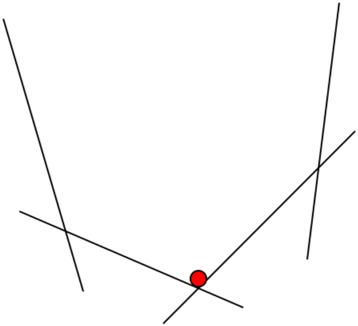
Proof of Duality (intuition)

Ball in a well.



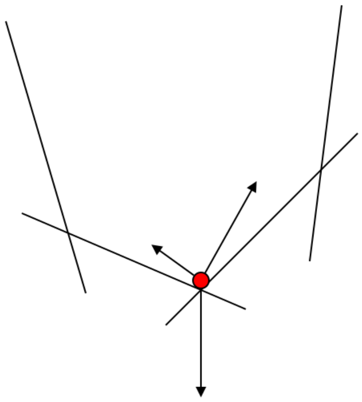
Proof of Duality (intuition)

Gravity pulls to lowest point.



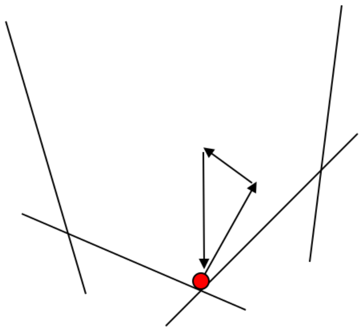
Proof of Duality (intuition)

Force of gravity cancels normal forces.



Proof of Duality (intuition)

Linear combination of normal vectors equals downward vector.



Duality Proof

- Consider solvability of system $Ax \geq b, x \cdot v \leq t$.
- Claim solution unless combination of equations yields $0 \geq 1$.
- Equivalent to original problem.

Duality Proof

Consider set $\mathcal{C}$ of combinations

$$c_1 E_1 + c_2 E_2 + \ldots + c_m E_m$$

with $c_i \geq 0$.

Duality Proof

Consider set $\mathcal{C}$ of combinations

$$c_1E_1 + c_2E_2 + \ldots + c_mE_m$$

with $c_i \geq 0$. Note that $\mathcal{C}$ is convex.

Duality Proof

Consider set $\mathcal{C}$ of combinations

$$c_1 E_1 + c_2 E_2 + \ldots + c_m E_m$$

with $c_i \geq 0$. Note that $\mathcal{C}$ is convex.
If equation $0 \geq 1$ not in $\mathcal{C}$, there is a separating hyperplane.

Duality Proof

Consider set $\mathcal{C}$ of combinations

$$c_1 E_1 + c_2 E_2 + \ldots + c_m E_m$$

with $c_i \geq 0$. Note that $\mathcal{C}$ is convex.
If equation $0 \geq 1$ not in $\mathcal{C}$, there is a separating hyperplane.
This hyperplane correspond to a solution to the original system!

Example

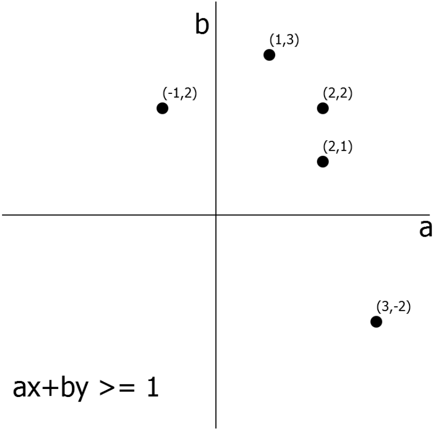
Consider system:

$$\begin{aligned} 3x - 2y &\geq 1 \\ -x + 2y &\geq 1 \\ 2x + 1y &\geq 1 \\ x + 3y &\geq 1 \\ 2x + 2y &\geq 1. \end{aligned}$$

Is there a solution?

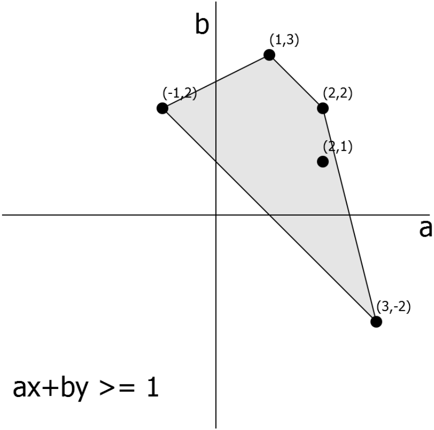
Example

Plot equations.



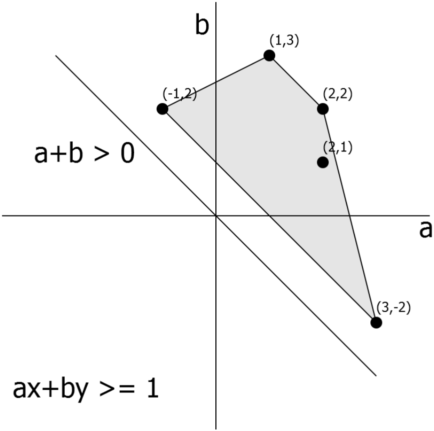
Example

Consider linear combinations.



Example

Find separator.



Example

- All equations of form $ax + by \geq 1$ with $a + b > 0$.
- If $x = y = \mathrm{Big}$, $ax + by = (a + b)\mathrm{Big} > 1$.
- Have solution $x = y = 1$!

Complementary Slackness

Theorem

Consider a primal LP:
Minimize $v \cdot x$ subject to $Ax \geq b$,
and its dual LP:
Maximize $y \cdot b$ subject to $y^T A = v$, $y \geq 0$.
Then in the solutions, $y_i > 0$ only if the i^{th} equation in x is tight.

Proof

- Solution x to primal matching solution y to the dual.
- $x \cdot v = t$.
- Combination of equations $\sum y_i E_i$ yields $x \cdot v \geq t$.
- Since final equation is tight, cannot use non-tight equations in sum.
- For each i, either E_i is tight or $y_i = 0$.

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

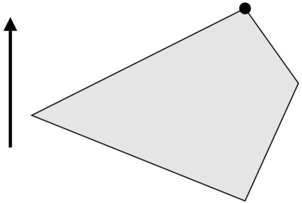
- Distinguish between the different types of linear programming problems.
- Use an algorithm that solves one formulation to solve another formulation.

Formulations

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

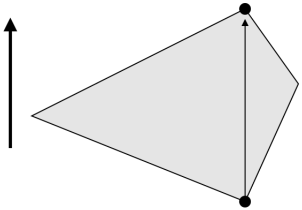
Full Optimization

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



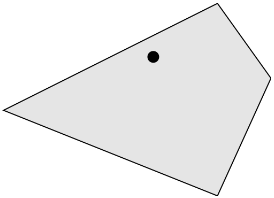
Optimization from Starting Point

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



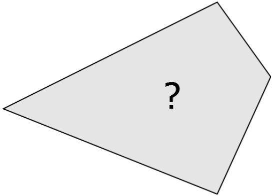
Solution Finding

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

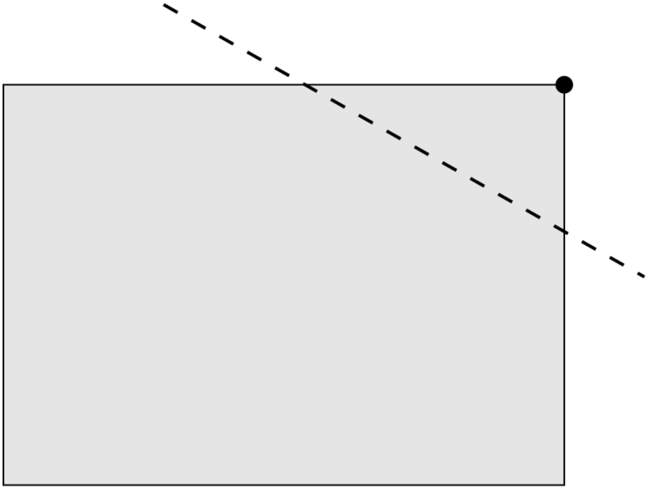
- How do you find starting point?

Optimization from Starting Point

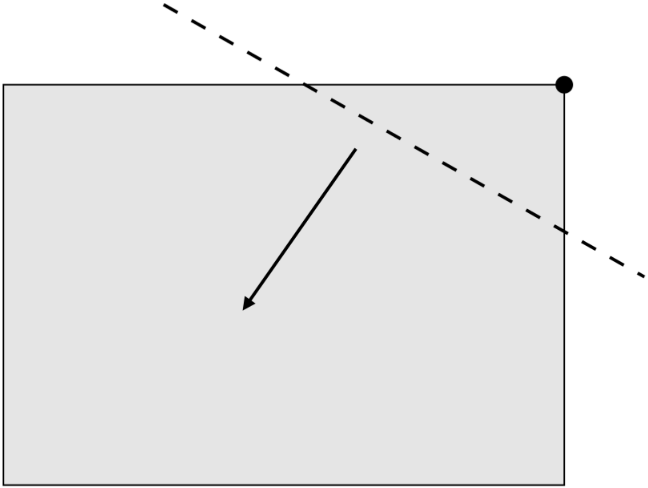
- How do you find starting point?
- Add equations one at a time.
- Optimize left hand side of next equation.

Example

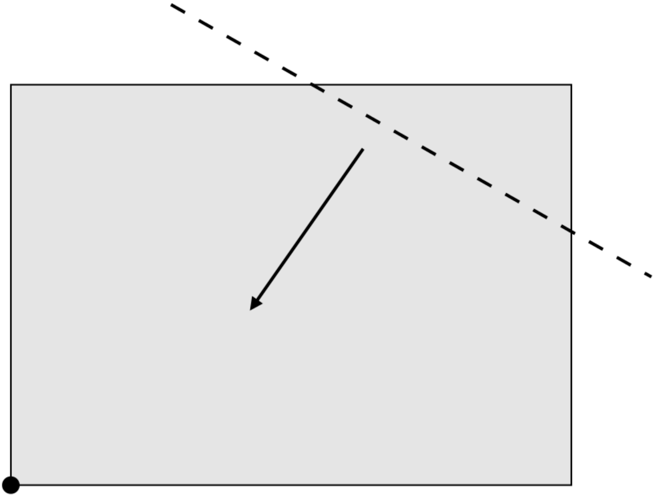
Example



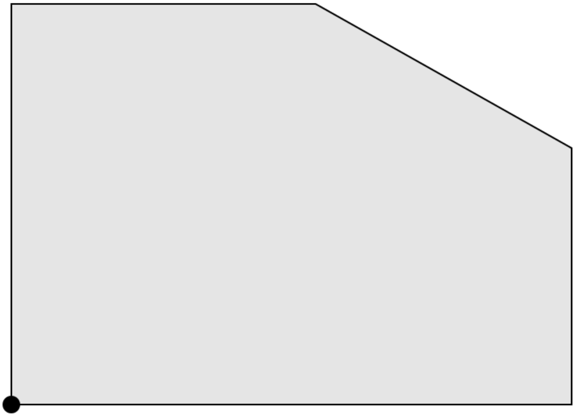
Example



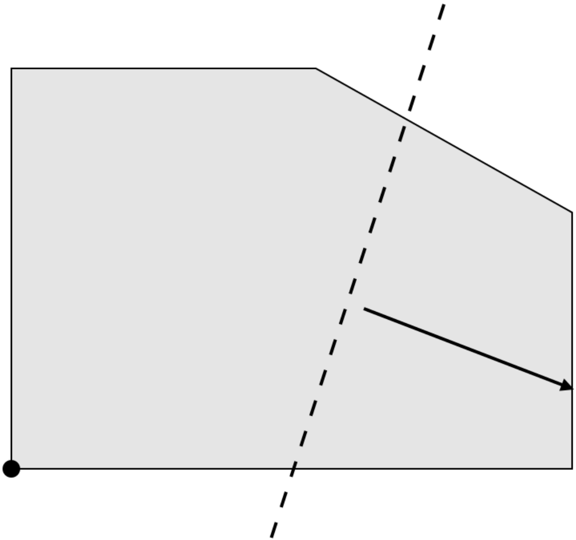
Example



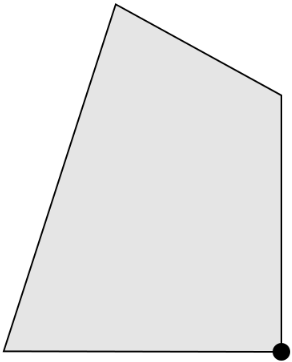
Example



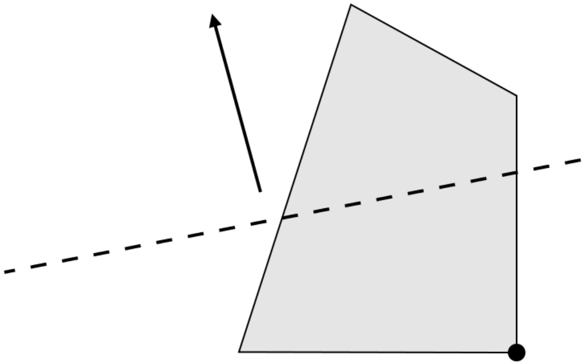
Example



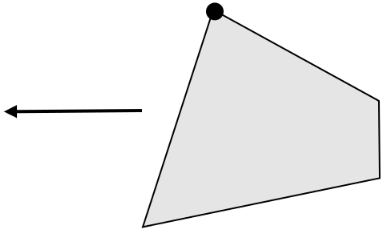
Example



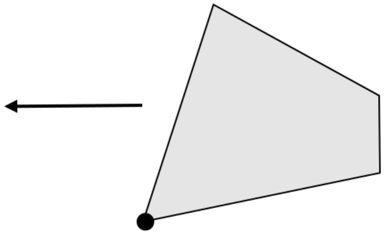
Example



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.

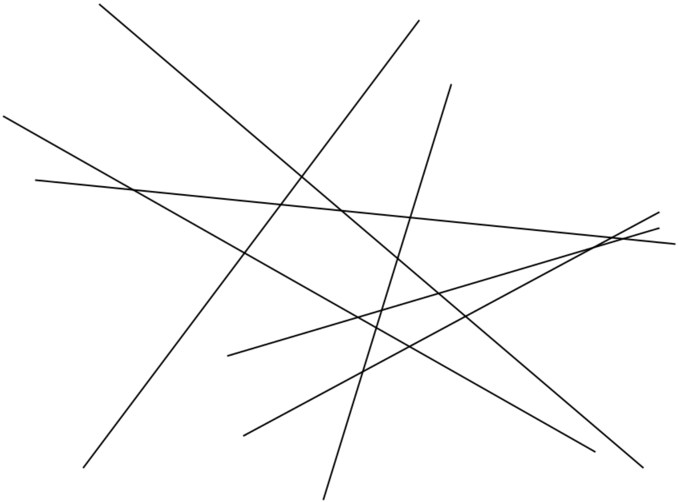
Satisfiability

How does just knowing when you have a solution help?

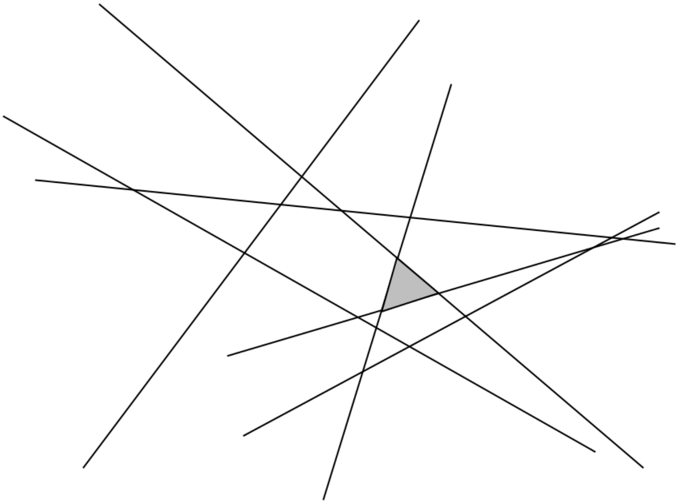
Can always find solution at a vertex. Means n equations are tight.

Figure out which equations to use.

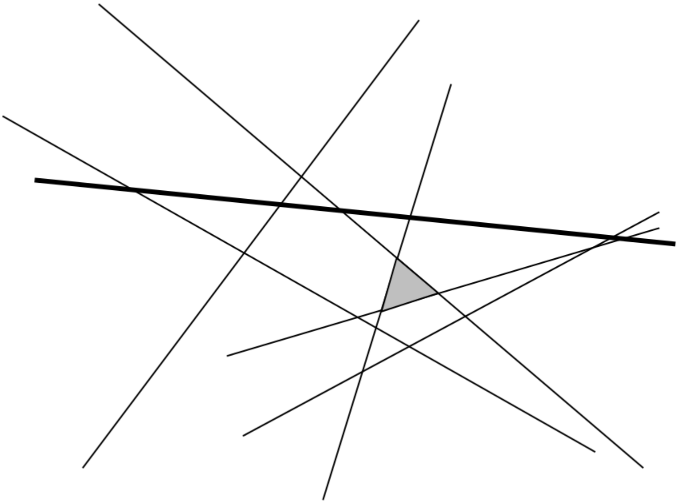
Example



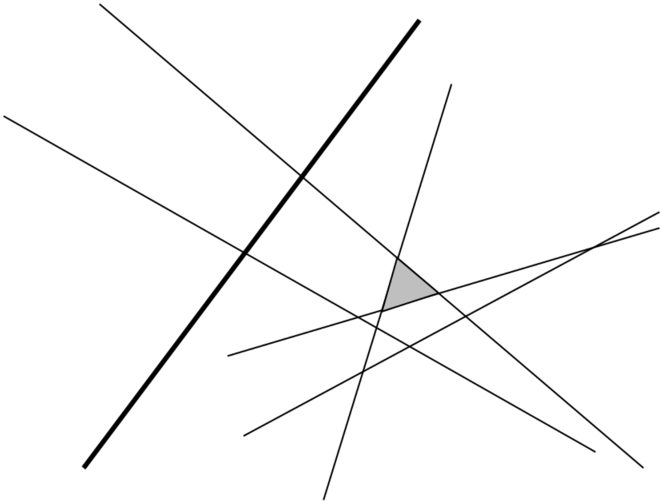
Example



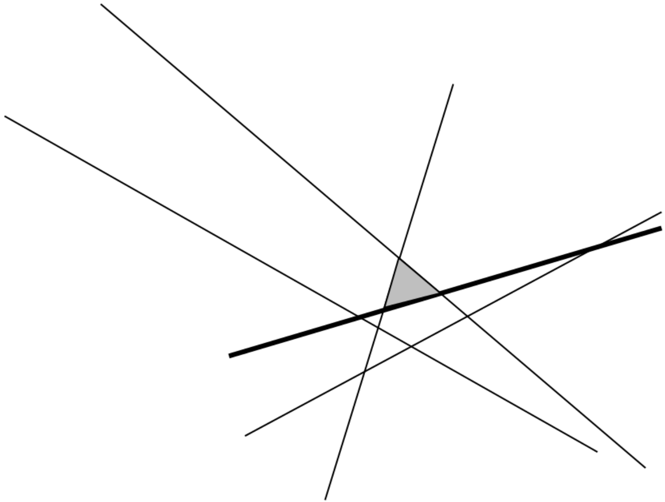
Example



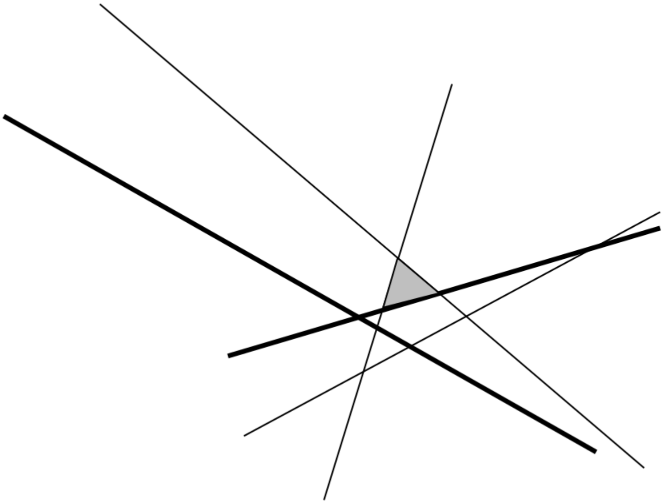
Example



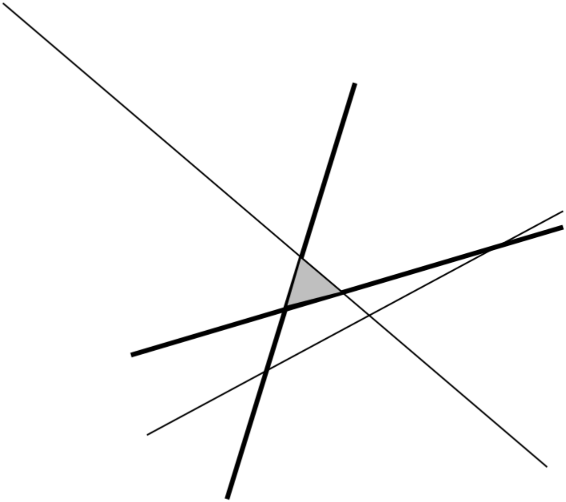
Example



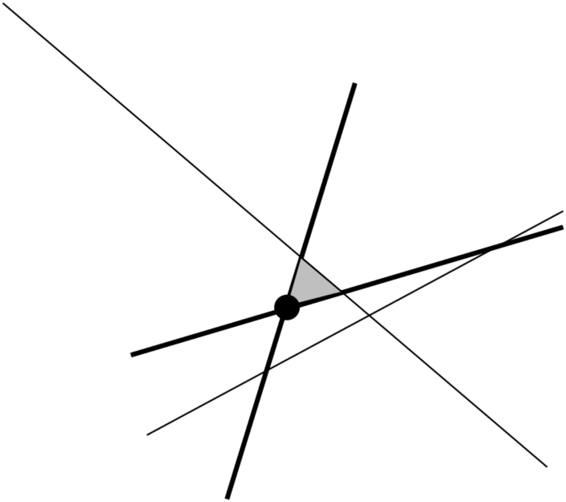
Example



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.

Linear Programming: Simplex Method

Daniel Kane

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

Advanced Algorithms and Complexity
Data Structures and Algorithms

Learning Objectives

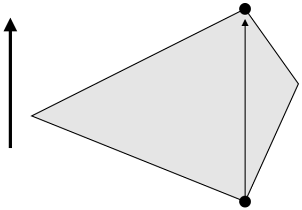
- Understand the idea of moving between vertices of a polytope.
- Implement the simplex method.

Simplex Method

- Oldest algorithm for solving linear programs.
- Still one of the most efficient.
- Not quite the runtime we would like.

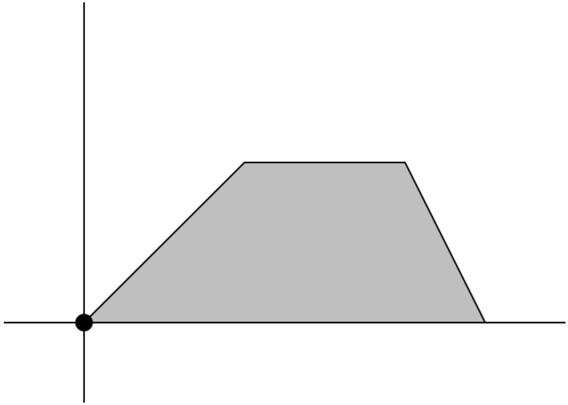
Formulation

Solves the optimization from starting point formulation.



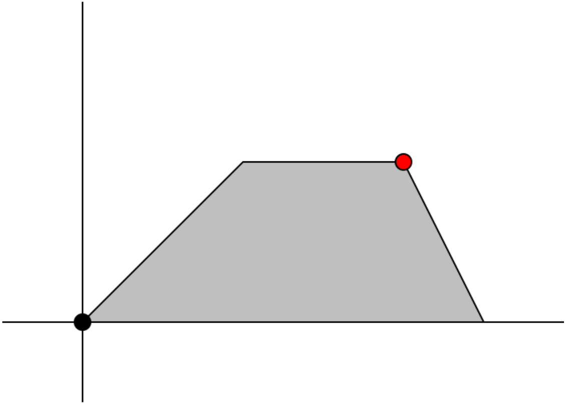
Idea

Start at vertex.



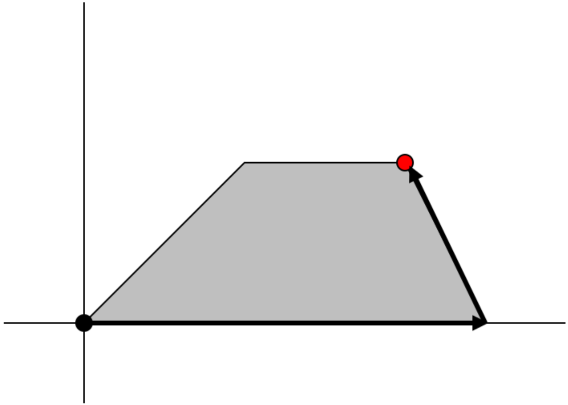
Idea

Optimum at another vertex.



Idea

Path of vertices to find optimum.



Vertices and Edges

- Vertex p when n defining equations are tight (solve with Gaussian elimination).

Vertices and Edges

- Vertex p when n defining equations are tight (solve with Gaussian elimination).
- Relax one equation to get an edge. Points of form $p + tw$, $t \geq 0$.
- Edge continues until it violates some other constraint.

Vertices and Edges

- Vertex p when n defining equations are tight (solve with Gaussian elimination).
- Relax one equation to get an edge. Points of form $p + tw$, $t \geq 0$.
- Edge continues until it violates some other constraint.
- If $v \cdot w > 0$, can follow edge to find larger value of objective.

Pseudocode

Simplex

```
Start at vertex p
repeat
  for each equation through p
    relax equation to get edge
    if edge improves objective:
      replace p by other end
      break
  if no improvement:  return p
```

OtherEndOfEdge

Vertex p defined by n equations
Relax one, write general solution as $p + tw$ (Gaussian elimination)
Relaxed inequality requires $t \geq 0$
For each other inequality in system:
  Largest t so $p + tw$ satisfies
Let t_0 be smallest such t
return $p + t_0 w$.

Correctness

Theorem

If p is a vertex that is not optimal, there is some adjacent vertex that does better.

Proof (sketch)

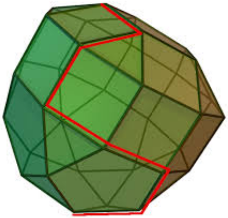
- p at intersection of equations $E_1, \dots, E_n$.
- Can always write $x \cdot v \leq \dots$ as linear combination.
- If positive coefficients, p is an optimum.
- Otherwise, relax equation with negative coefficient.

Analysis

How long does simplex take?

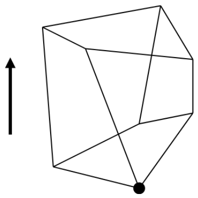
Analysis

How long does simplex take? Must follow path to optimum.



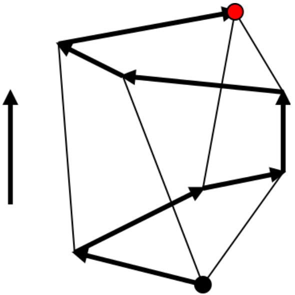
Problem

What is the largest number of steps that the simplex method might require to find the optimum in the following situation? Assume that points drawn higher on the screen have larger values of the objective.



Solution

As many as 7 steps (though potentially as few as 3).



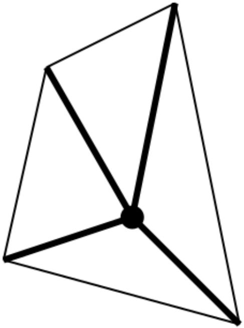
Runtime

- Runtime proportional to path length.
- Path length usually pretty reasonable.
- However, there are examples where path is exponential!

Degeneracy

One technical problem:

If more than n hyperplanes intersect at a vertex, don't know which to relax.



Fix

Tweak equations a tiny bit to avoid these intersections.

Fix

Tweak equations a tiny bit to avoid these intersections.

You can actually make this work nicely with infinitesimal changes.

Fix

Tweak equations a tiny bit to avoid these intersections.

You can actually make this work nicely with infinitesimal changes.

Number constraints. Strengthen first by ε, next by ε^2, etc.

Fix

- Don't need to actually change equations.
- At degenerate point, keep track of which n you are "really" on.
- When travelling along an edge to a degenerate corner, add the lowest-numbered constraint at the new corner.
- Edges from degenerate corner to itself: change "corner" if edge "improves" objective.

Summary

- Solve LP by moving between adjacent vertices towards optimum.
- Works well in practice.
- Potentially exponential time.

Linear Programming: (Optional) The Ellipsoid Algorithm

Daniel Kane

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

Advanced Algorithms and Complexity
Data Structures and Algorithms

Learning Objectives

- Understand the basic ideas behind the ellipsoid algorithm.
- Explain the practical differences between the ellipsoid and simplex algorithms.

Last Time

Simplex algorithm.

- Solves LPs.
- Works well most of the time.
- Exponential in some cases.

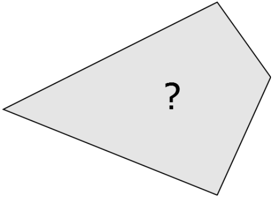
Today

Ellipsoid algorithm.

- Solves LPs.
- Polynomial time in all cases.
- Often not as good in practice.

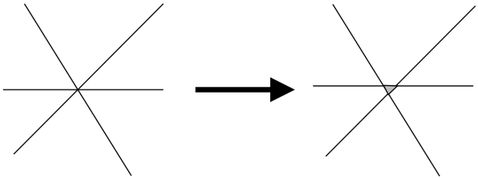
Formulation

Ellipsoid solves the satisfiability version of a Linear Program.



Step I

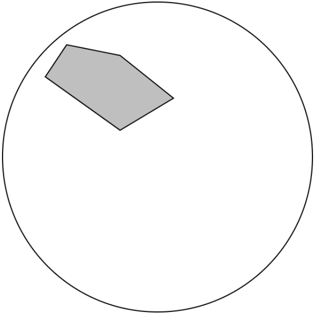
Relax all equations a tiny bit.



Solution set (if exists) has positive volume.

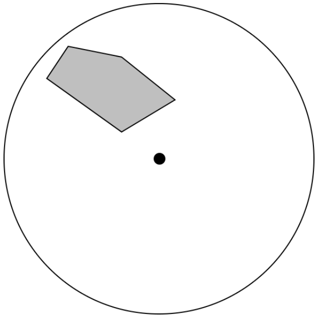
Step II

Bound solution set in large ball.



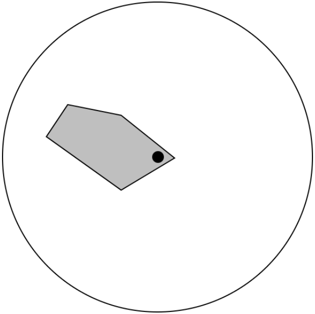
Step III

Have ellipsoid that contains all solutions. See if center of ellipsoid is a solution.



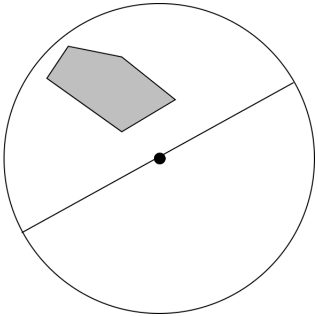
Step IIIa

If it is a solution, system is satisfiable.



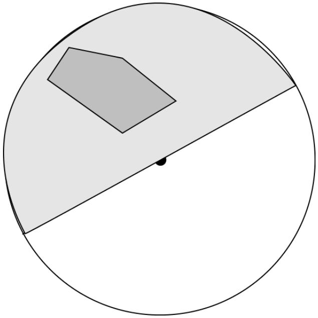
Step IIIb

If not solution, find separating hyperplane.



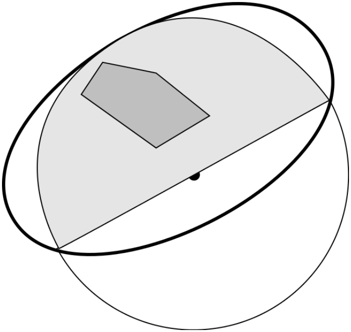
Step IIIb

Solutions now contained in half-ellipsoid.



Step IIIb

Half-ellipsoid contained in new ellipsoid of smaller volume.



Step IV

Repeat until ellipsoid is too small to contain solution set. In this case, there must be no solutions.

Runtime

Ellipsoid runs in polynomial time

$$O((m + n^2)n^5 \log(nU))$$

where

$$\begin{aligned} n &= \text{Dimension} \\ m &= \text{Number of Equations} \\ U &= \text{Numerical size of coefficients} \end{aligned}$$

Runtime

Runtime is:

- Polynomial!

Runtime

Runtime is:

- Polynomial!
- But a bad polynomial.
- That also depends (logarithmically) on the coefficient size.

Oracles

Also, note that the Ellipsoid algorithm doesn't really need the equations. It just needs a separation oracle. That is an algorithm that given an $x \notin \mathcal{C}$ gives a hyperplane that separates x from $\mathcal{C}$. This lets you employ Ellipsoid in contexts when you cannot necessarily enumerate the constraints.

Summary

The ellipsoid algorithm

- Is another way to solve LPs.
- Has better worst-case performance than simplex.
- However, is usually slower.
- Can run with only a separation oracle.