CS 580: Algorithm Design and Analysis

Jeremiah Blocki
Purdue University
Spring 2018

Recap

- Network Flow Problems
  - Max-Flow Min Cut Theorem
  - Ford Fulkerson
    - Augmenting Paths
    - Residual Flow Graph
    - Integral Solutions (given integral capacities)
  - Capacity Scaling Algorithm
  - Dinic's Algorithm
- Applications of Maximum Flow
  - Maximum Bipartite Matching
  - Marriage Theorem (Hall/Frobenius)
  - Disjoint Paths [Menger's Theorem]
  - Baseball Elimination
  - Circulation with Demands
  - Many Others…

Linear Programming

- Even more general than Network Flow!

- Many Applications
  - Network Flow Variants
    - Taxation
    - Multi-Commodity Flow Problems
  - Supply-Chain Optimization
  - Operations Research
    - Entire Courses Devoted to Linear Programming!
- Our Focus
  - Using Linear Programming as a tool to solve algorithms problems
  - We won't cover algorithms to solve linear programs in any depth

Motivating Example: Time Allocation

168 Hours in Each Week to Allocate as Follows

Studying (S) Partying (P) Everything Else (E)

Constraints:

- [168 Hours] $S + P + E = 168$
- [Maintain Sanity] $P + E \geq 70$
- [Pass Courses 1] $S \geq 60$
- [Pass Courses 2] $2S + E - 3P \geq 150$ (too little sleep, and/or too much partying makes it more difficult to study)

 Credit for Example: Avrim Blum

Motivating Example: Time Allocation

168 Hours in Each Week to Allocate as Follows

Studying (S)

Partying (P)

Everything Else (E)

Constraints:

- [168 Hours] $S + P + E = 168$
- [Maintain Sanity] $P + E \geq 70$
- [Survive] $E \geq 56$
- [Pass Courses 1] $S \geq 60$
- [Pass Courses 2] $2S + E - 3P \geq 150$ (too little sleep, and/or too much partying makes it more difficult to study)

Question 1: Can we satisfy all of the constraints?
(Maintain Sanity + Pass Courses)

Answer: Yes. One feasible solution is S=80, P=20, E=68

Credit for Example: Avrim Blum

Motivating Example: Time Allocation

168 Hours in Each Week to Allocate as Follows

Studying (S)

Partying (P)

Everything Else (E)

Constraints:

- [168 Hours] $S + P + E = 168$
- [Maintain Sanity] $P + E \geq 70$
- [Survive] $E \geq 56$
- [Pass Courses 1] $S \geq 60$
- [Pass Courses 2] $2S + E - 3P \geq 150$ (too little sleep, and/or too much partying makes it more difficult to study)

Objective Function: $2P + E$ [Maximize Happiness]

Question 2: Can we find a feasible solution which maximizes the objective function?

Linear Program Definition

- Variables: $x_1,\dots,x_n$
- m linear inequalities in these variables (equalities are OK)
  - Examples
    - $0 \le x_1 \le 1$
    - $x_1 + x_4 + 3x_{10} - 7x_{11} \le 4$
    - $2S + E - 3P \ge 150$
- [Optional] Linear Objective Function
  - Example:
    - maximize $4x_4 + 3x_{10}$
    - minimize $3x_1 + 3x_2$
    - maximize $2P + E$
- Goal
  - Find values for $x_1,\dots,x_n$ satisfying all constraints, and
  - Maximize the objective
- Feasibility Problem
  - No objective function

Linear Program Definition

- **Variables:** $x_1, \ldots, x_n$
- **Constraints:** m linear inequalities in these variables (equalities are OK)
- [Optional] Linear Objective Function

Requirement:

- All the constrains are linear inequalities in variables (S,P,E)
- The objective function is also linear

Example Non-Linear Constraints:

$$PE \geq 70 \qquad E \in \{0,1\}$$

$$E(1-E) = 1 \qquad \mathbf{Max}\{P, E\} \geq 20$$

Linear Program Example

Goal: Maximize 2P+E

Subject to:

- [168 Hours] $S + P + E = 168$
- [Maintain Sanity] $P + E \geq 70$
- [Survive] $E \geq 56$
- [Pass Courses 1] $S \geq 60$
- [Pass Courses 2] $2S + E - 3P \geq 150$
- [Non-Negativity] $P \geq 0$

Requirement:

- All the constrains are linear inequalities in variables (S,P,E)
- The objective function is also linear

Example Non-Linear Constraints:

$$PE \geq 70 \qquad E \in \{0,1\}$$

 Credit for Example: Avrim Blum

Network Flow as a Linear Program

Given a directed graph G with capacities c(e) on each edge e we can use linear programming to find a maximum flow from source **s** to sink **t**.

Variables: x_e for each directed edge e (represents flow on edge e)

Objective: Maximize $\sum_{e\ out\ of\ s} x_e$

Constraints:

- (Capacity Constraints) For each edge e we have $0 \leq x_e \leq c(e)$
- (Flow Conservation) For each $v \notin \{s, t\}$ we have

$$\sum_{e\ out\ of\ v} x_e = \sum_{e\ into\ v} x_e$$

Network Flow as a Linear Program

Example:

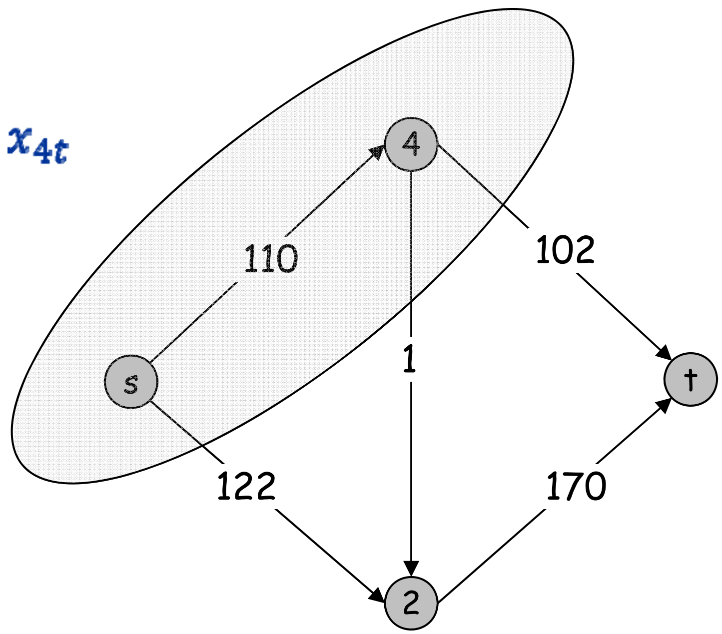
Variables: $x_{s4}, x_{s2}, x_{42}, x_{2t}, x_{4t}$

Goal: maximize $x_{s4} + x_{s2}$

Subject to

- $0 \leq x_{s4} \leq 110$
- $0 \leq x_{s2} \leq 122$
- $0 \leq x_{42} \leq 1$
- $0 \leq x_{2t} \leq 170$
- $0 \leq x_{4t} \leq 102$

capacity



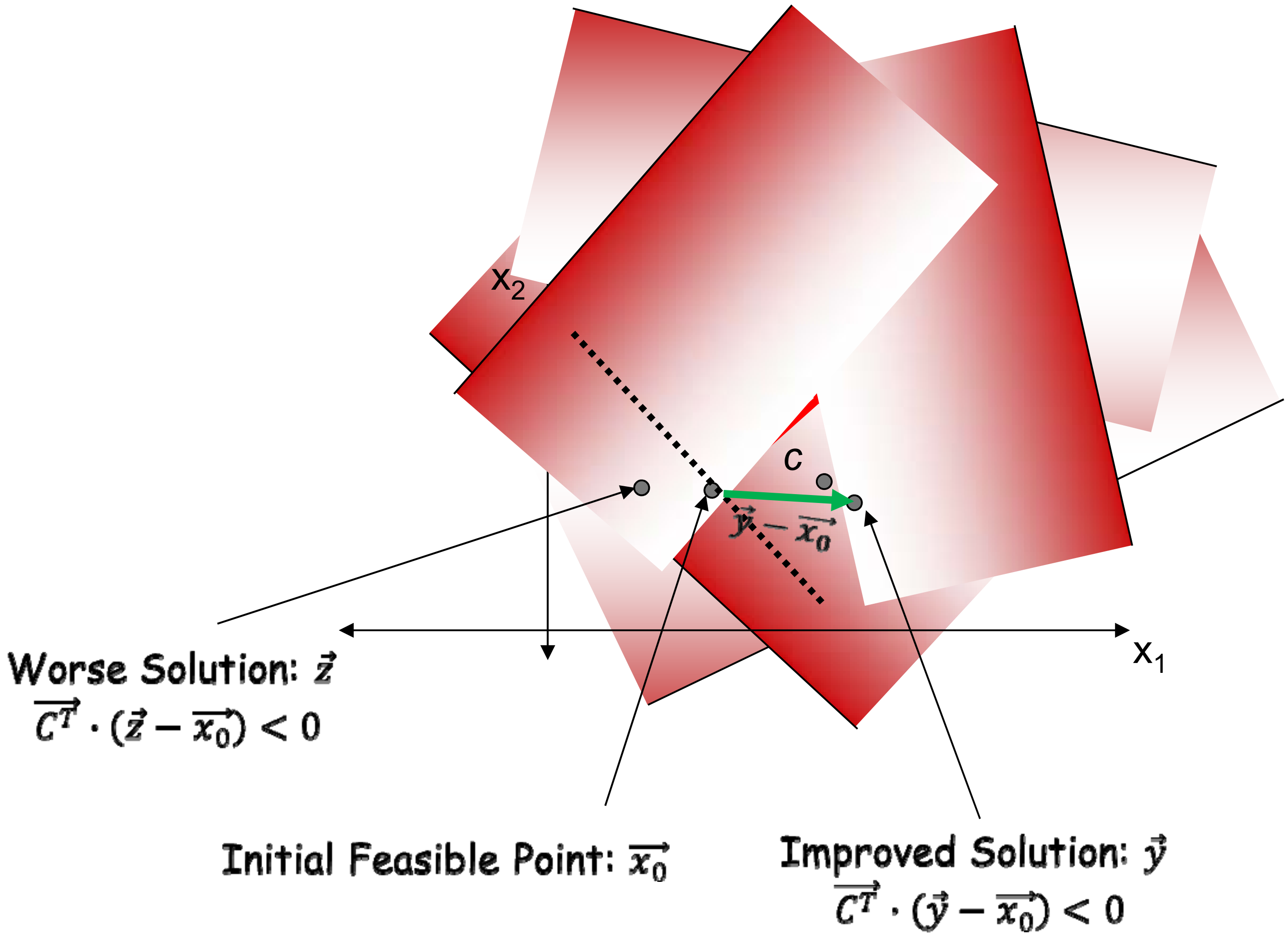
- $x_{s4} = x_{42} + x_{4t}$ [Flow Conservation at node 4]
- $x_{s4} + x_{42} = x_{2t}$ [Flow Conservation at node 2]

Solving a Linear Program

- Simplex Algorithm (1940s)
  - Not guaranteed to run in polynomial time
  - We can find bad examples, but…
  - The algorithm is efficient in practice!
- Ellipsoid Algorithm (1980)
  - Polynomial time (huge theoretical breakthrough), but ….
  - Slow in practice
- Newer Algorithms
  - Karmarkar's Algorithm
    - Competitive with Simplex
    - Polynomial Time

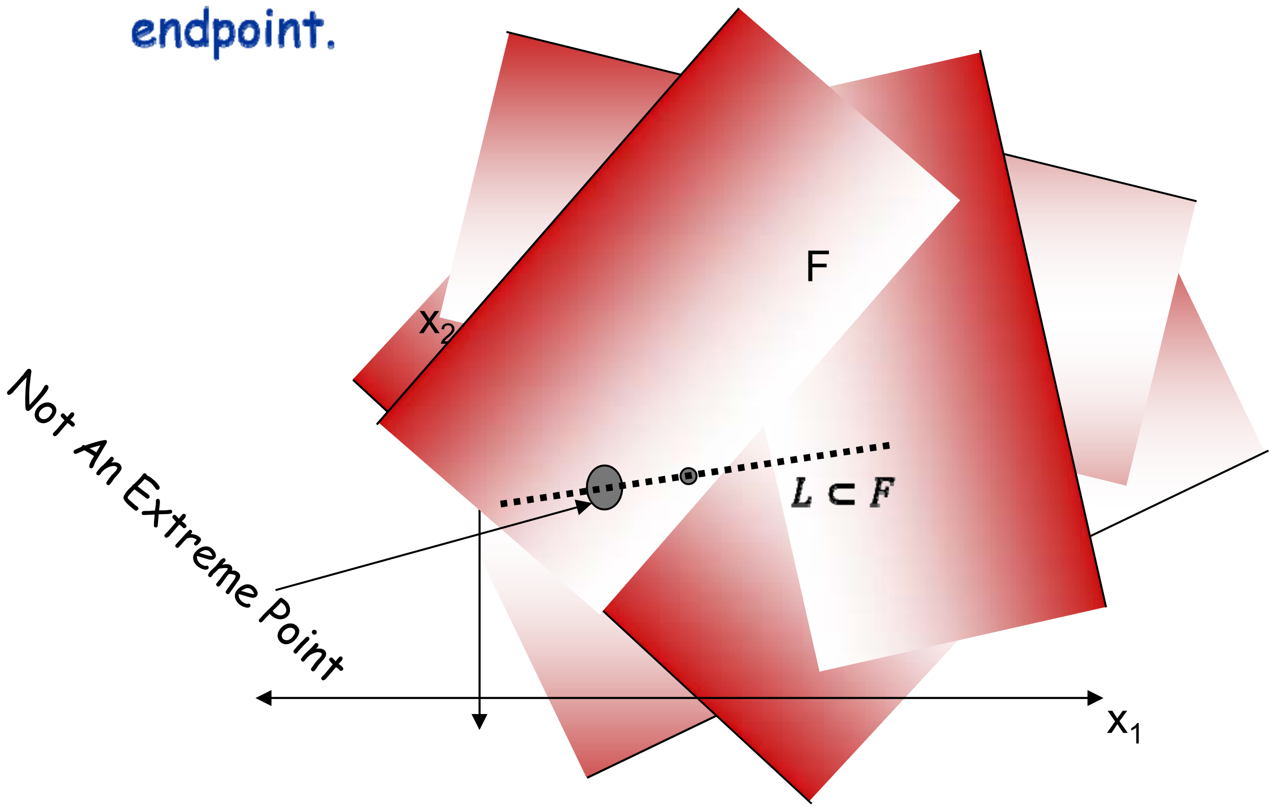
Algorithmic Idea: Direction of Goodness

Goal: Maximize $2x_1+3x_2$ $\quad$ $c=(2,3)$



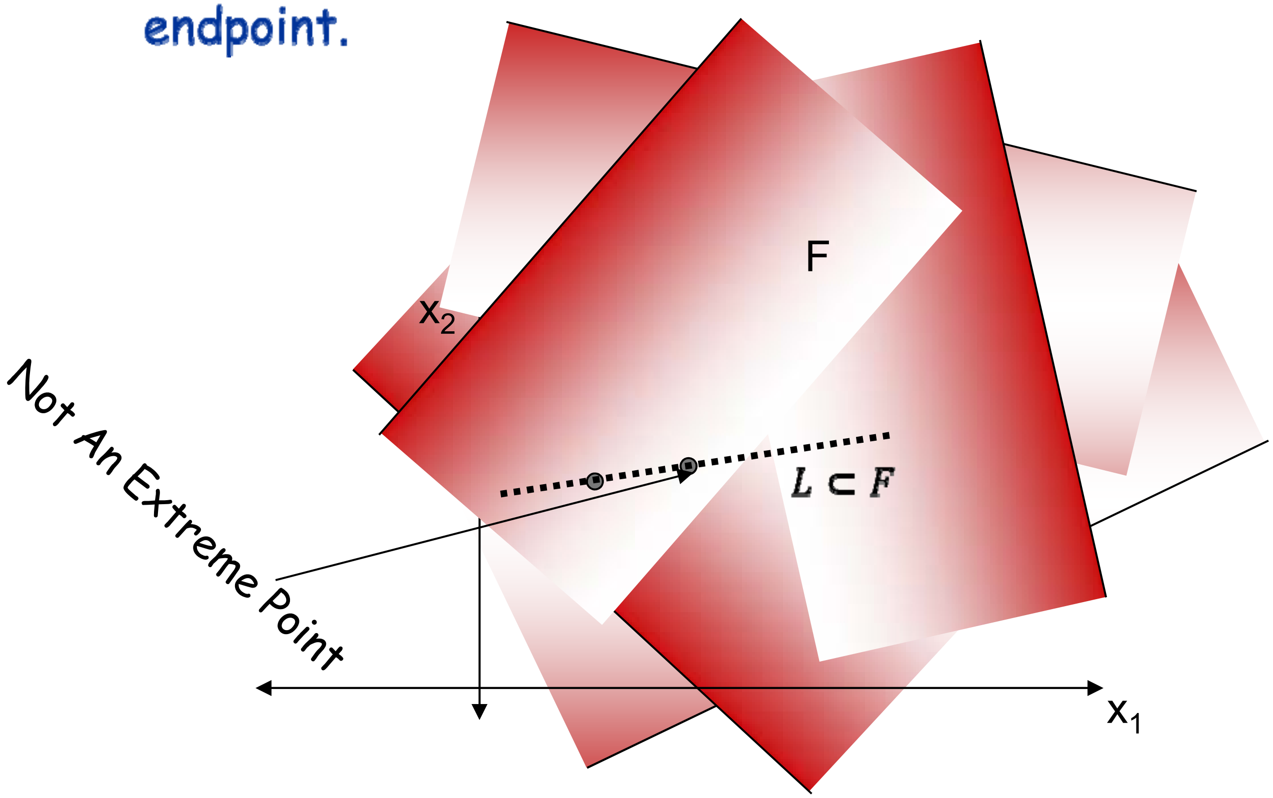
Theorem: Maximum value achieved at vertex (extreme point)

Definition: Let F be the set of all feasible points in a linear program. We say that a point $p \in F$ is an extreme point (vertex) if *every* line segment $L \subset F$ that lies completely in F and contains p has p as an endpoint.



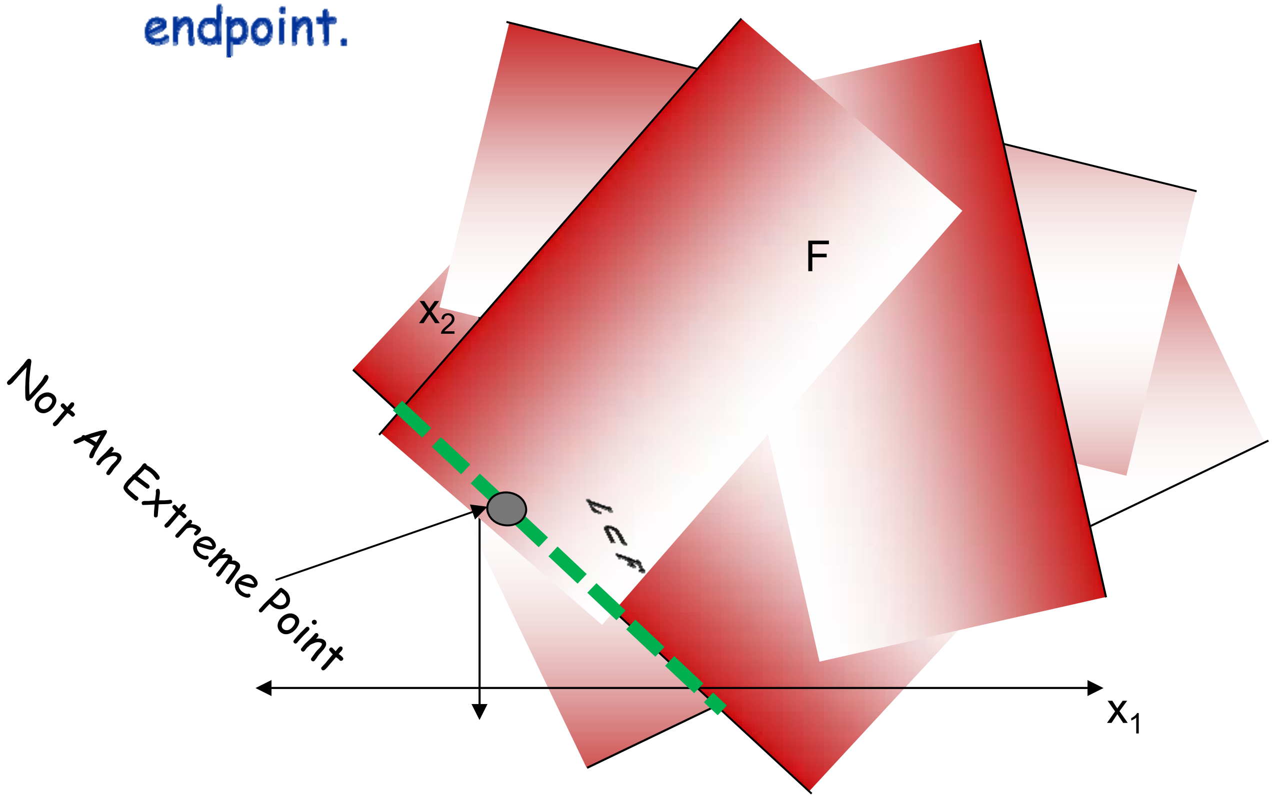
Theorem: Maximum value achieved at vertex (extreme point)

Definition: Let F be the set of all feasible points in a linear program. We say that a point $p \in F$ is an extreme point (vertex) if *every* line segment $L \subset F$ that lies completely in F and contains p has p as an endpoint.



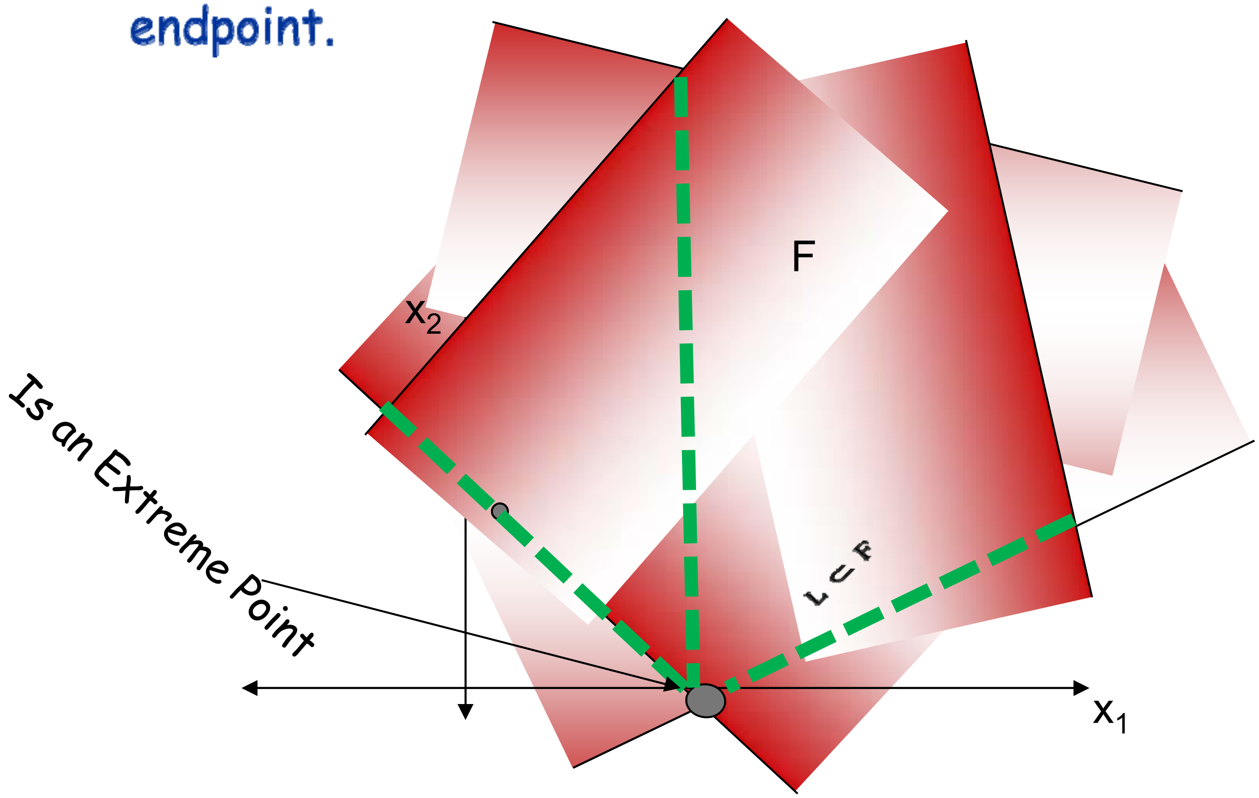
Theorem: Maximum value achieved at vertex (extreme point)

Definition: Let F be the set of all feasible points in a linear program. We say that a point $p \in F$ is an extreme point (vertex) if *every* line segment $L \subset F$ that lies completely in F and contains p has p as an endpoint.



Theorem: Maximum value achieved at vertex (extreme point)

Definition: Let F be the set of all feasible points in a linear program. We say that a point $p \in F$ is an extreme point (vertex) if *every* line segment $L \subset F$ that lies completely in F and contains p has p as an endpoint.



Linear Programming

Theorem: Maximum value achieved at vertex (extreme point)

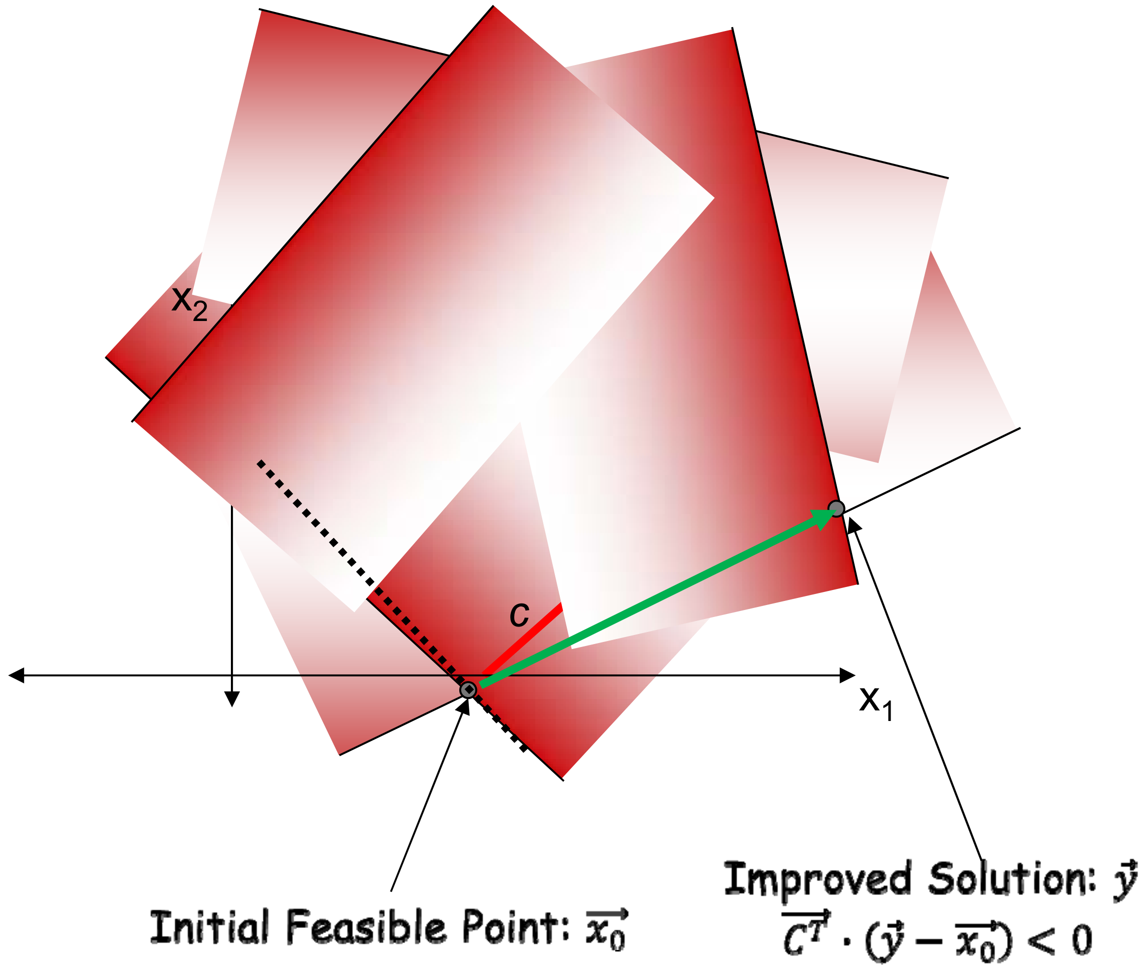
Definition: Let F be the set of all feasible points in a linear program. We say that a point $p \in F$ is an extreme point (vertex) if *every* line segment $L \subset F$ that lies completely in F and contains p has p as an endpoint.

Observation: Each extreme point lies at the intersection of (at least) two constraints.

Theorem: a vertex is an optimal solution if there is no better *neighboring vertex*.

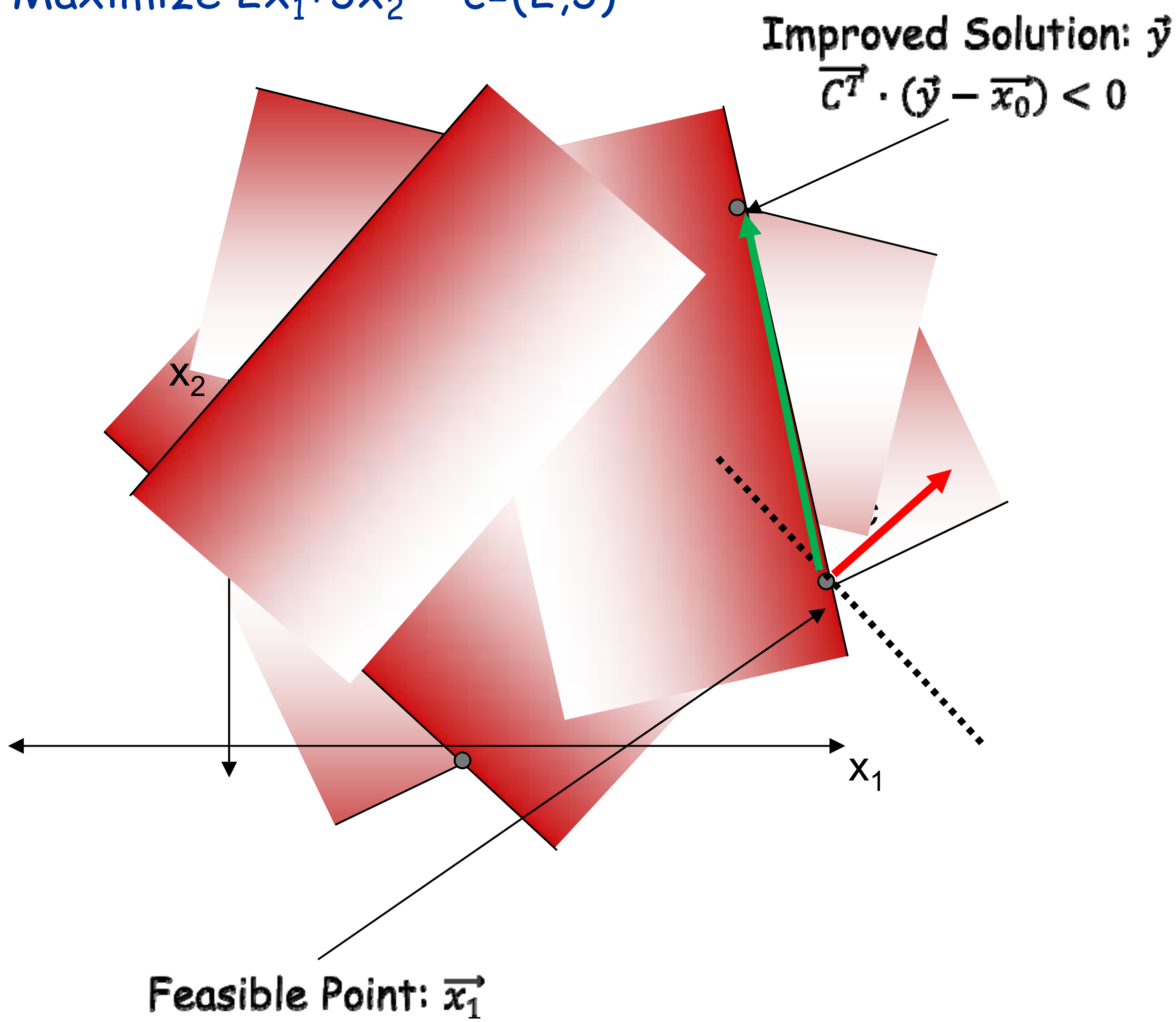
Algorithmic Idea: Vertex Walking

Goal: Maximize $2x_1+3x_2$ $c=(2,3)$



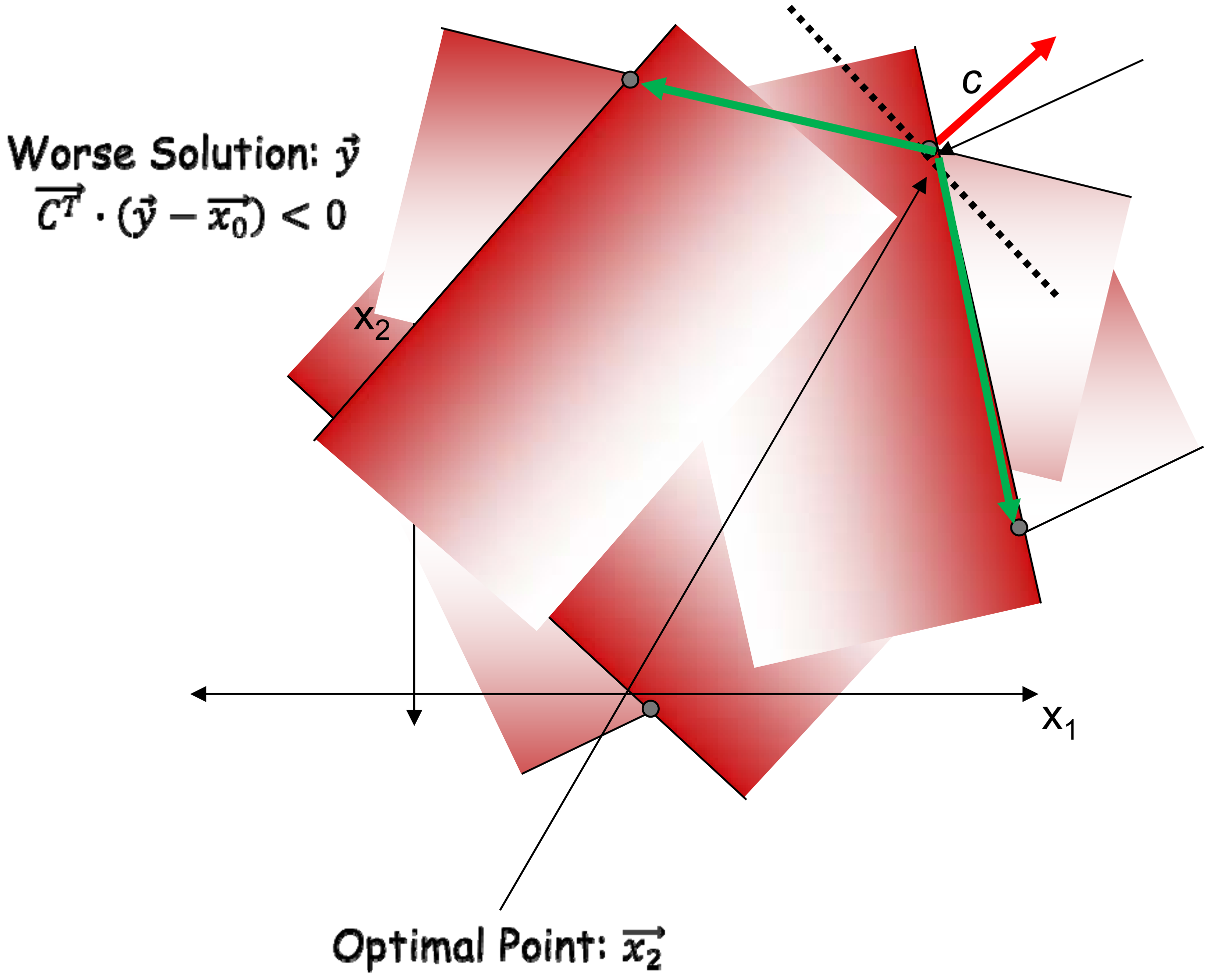
Algorithmic Idea: Vertex Walking

Goal: Maximize $2x_1+3x_2$ $c=(2,3)$



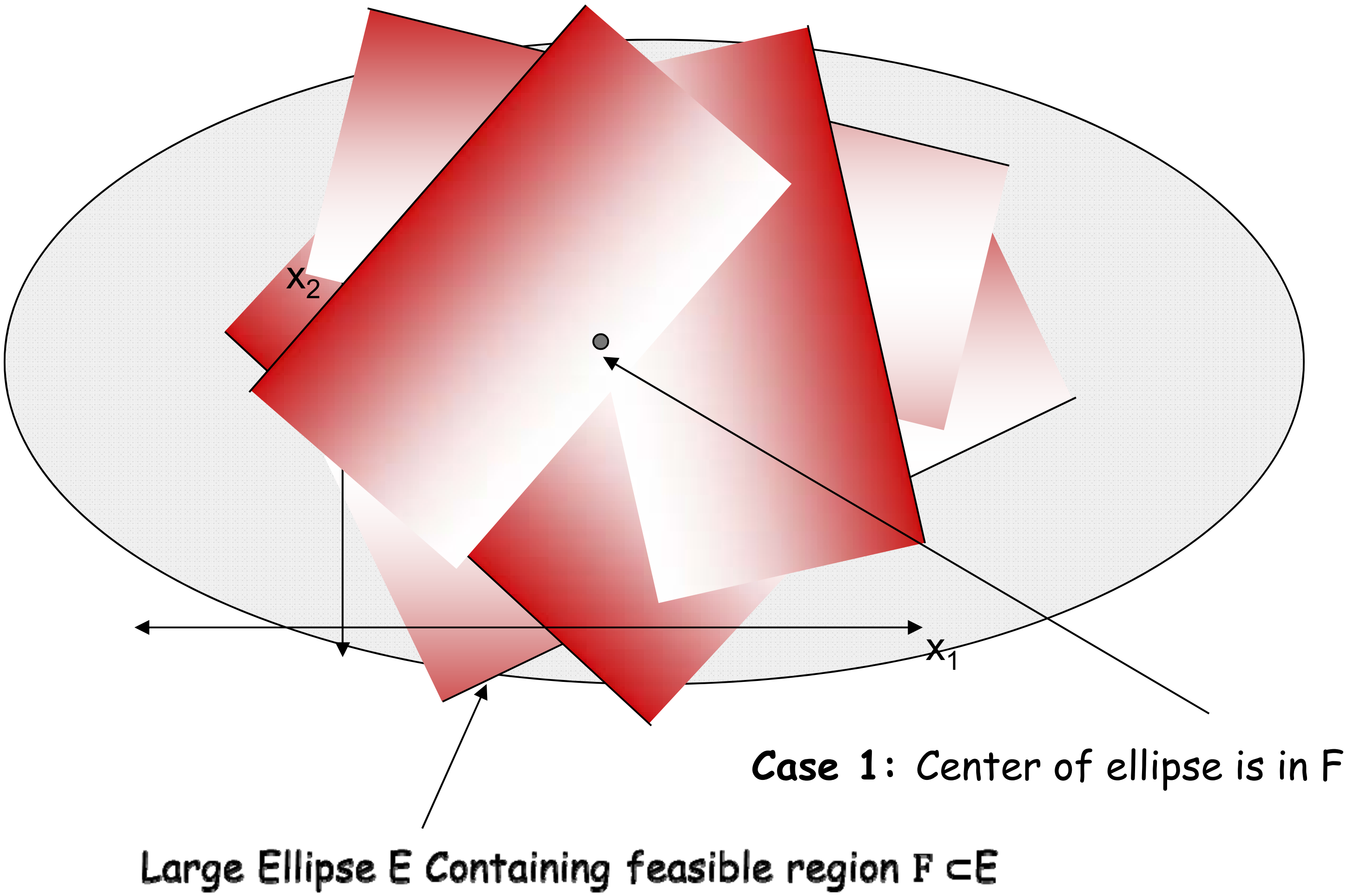
Algorithmic Idea: Vertex Walking

Goal: Maximize $2x_1+3x_2$ $c=(2,3)$



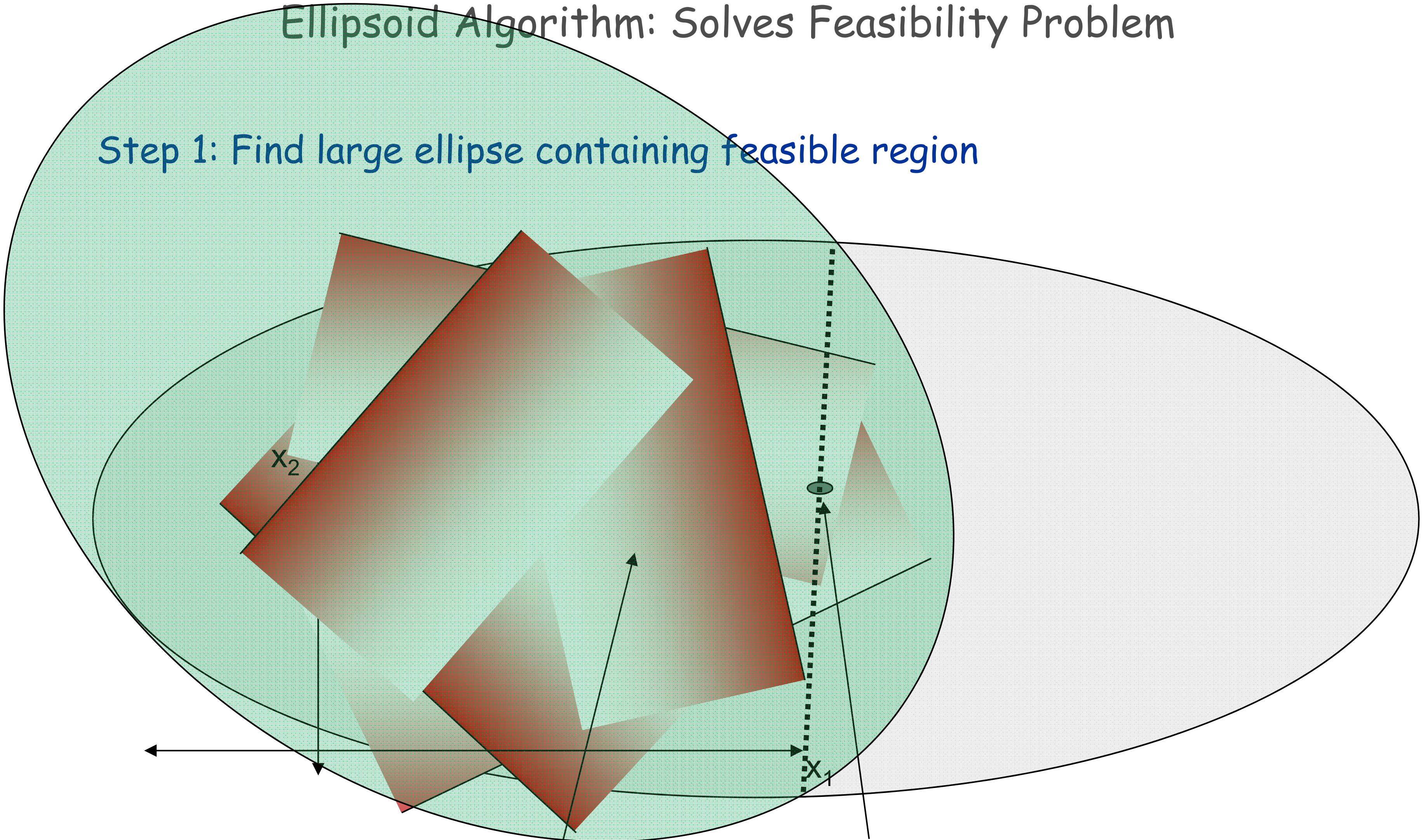
Ellipsoid Algorithm: Solves Feasibility Problem

Step 1: Find large ellipse containing feasible region



Ellipsoid Algorithm: Solves Feasibility Problem

Step 1: Find large ellipse containing feasible region

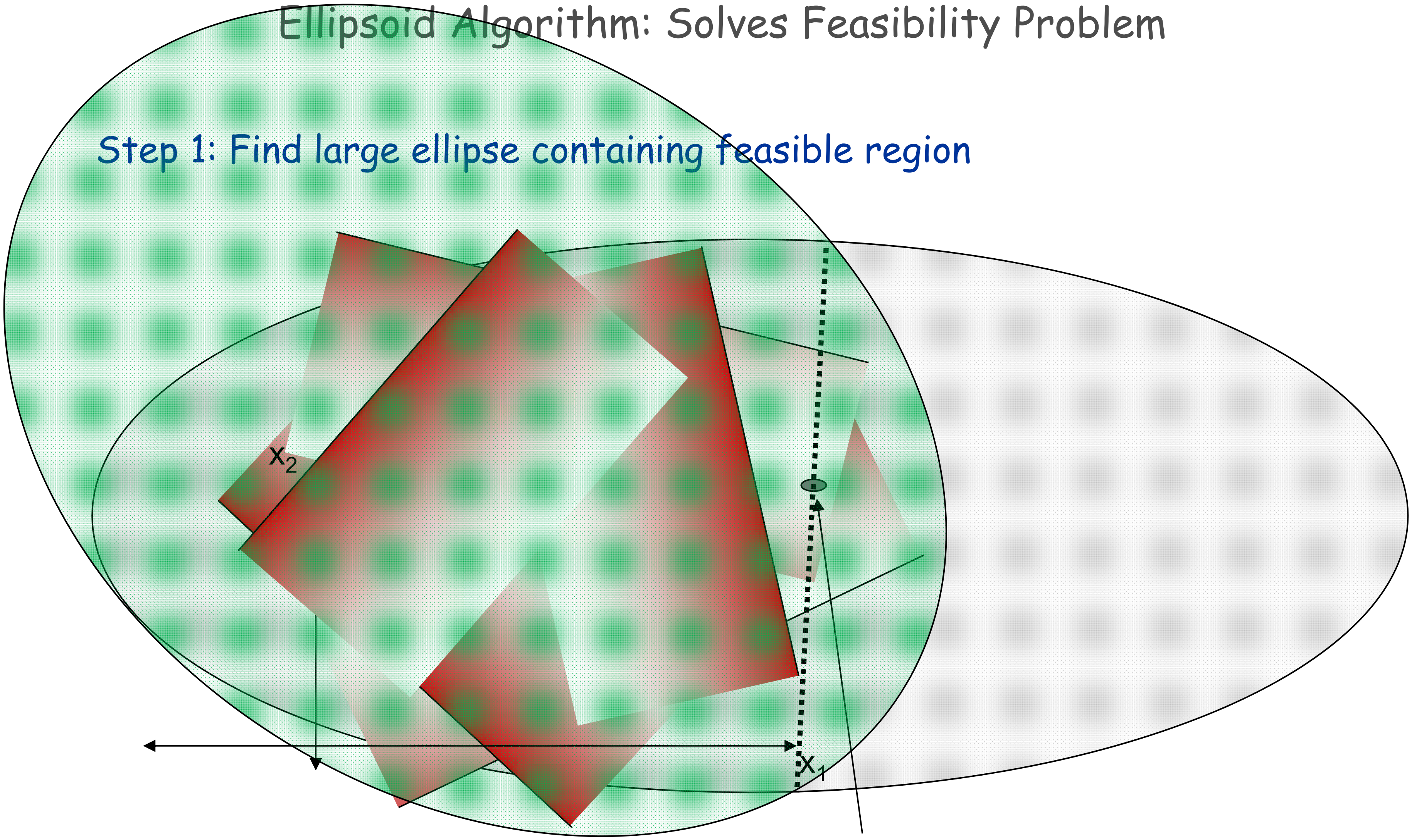


Case 2: Center of ellipse not in F

F contained in one half of the ellipsoid
➔ Can find smaller ellipsoid containing F
smaller by *at least* a $\left(1-\frac{1}{n}\right)$-factor.

Ellipsoid Algorithm: Solves Feasibility Problem

Step 1: Find large ellipse containing feasible region



Case 2: Center of ellipse not in F

smaller by *at least* a $\left(1-\frac{1}{n}\right)$-factor

➔ Every n steps volume drops by factor (1/e)

➔ poly(n) iterations to find feasible point (or reject)

Finding the Optimal Point with Ellipsoid Algorithm

Goal: maximize $\sum_i w_i x_i$ (where each w_i is a constant)

Key Idea: Binary Search for value of Optimal Solution!

- Add Constraint $\sum_i w_i x_i \geq B$
  - Infeasible?
    - ➔Value of optimal solutions is less than B
  - Feasible?
    - ➔Value of optimal solution is at least B

Linear Programming in Practice

Many optimization packages available

- Solver (in Excel)
- LINDO
- CPLEX
- GUROBI (free academic license available)
- Matlab, Mathematica

More Linear Programming Examples

Typical Operations Research Problem

Brewer's Problem: Maximize Profit

- (1 Barrel) of Ale sells for \$13, but recipe requires
  - 6 pounds corn,
  - 5 ounces of hops and
  - 33 pounds of malt.
- (1 Barrel) of Beer sells for \$23, but recipe requires
  - 16 pounds of corn
  - 4 ounces of hops and
  - 21 pounds of malt
- Suppose we start off with C= 480 pounds of corn, H=160 ounces of hops and M=1190 pounds of malt.

- Let A (resp. B) denote number of barrels of Ale (resp. Beer)

More Linear Programming Examples

Typical Operations Research Problem

Brewer's Problem: Maximize Profit

- (1 Barrel) of Ale sells for \$15, but recipe requires
  - 6 pounds corn,
  - 5 ounces of hops and
  - 33 pounds of malt.
- (1 Barrel) of Beer sells for \$27, but recipe requires
  - 16 pounds of corn
  - 4 ounces of hops and
  - 21 pounds of malt
- Suppose we start off with C= 480 pounds of corn, H=160 ounces of hops and M=1190 pounds of malt.
- Let A (resp. B) denote number of barrels of Ale (resp. Beer)
- **Goal:** maximize 15A+27B

More Linear Programming Examples

Brewer's Problem: Maximize Profit

- (1 Barrel) of Ale sells for \$15, but recipe requires
  - 6 pounds corn, 5 ounces of hops and 33 pounds of malt.
- (1 Barrel) of Beer sells for \$27, but recipe requires
  - 16 pounds of corn, 4 ounces of hops and 21 pounds of malt
- Suppose we start off with C= 480 pounds of corn, H=160 ounces of hops and M=1190 pounds of malt.
- Let A (resp. B) denote number of barrels of Ale (resp. Beer)
- **Goal:** maximize 15A+27B (subject to)
  - $A \geq 0, B \geq 0$ (positive production)
  - $6\mathrm{A} + 16B \leq C$ (Must have enough CORN)
  - $5\mathrm{A} + 4B \leq H$ (Must have enough HOPS)
  - $33\mathrm{A} + 21B \leq M$ (Must have enough HOPS)

Solving in Mathematica

```
Maximize[{15 A + 27 B,A>= 0, B>= 0, 6A+16B <= 480, 5A + 4B <= 160,
    33A+21 B <= 1190},{A,B}]
```

```
{6060/7,{A->80/7,B->180/7}}
```

Profit: $865.71

2-Player Zero-Sum Games

Example: Rock-Paper-Scissors

Alice/Bob	Rock	Paper	Scissors
Rock	(0,0)	(-1,1)	(1,-1)
Paper	(1,-1)	(0,0)	(-1,1)
Scissors	(1,-1)	(1,-1)	(0,0)

Alice wins ➔ Bob loses (and vice-versa)

Minimax Optimal Strategy (possibly randomized) best strategy you can find given that opponent is rational (and knows your strategy)

Minimax Optimal for Rock-Paper-Scissors: play each action with probability 1/3.

2-Player Zero-Sum Games

Example: Rock-Paper-Scissors

Alice's View of Rewards
(Bob's are reversed)

Alice/Bob	Rock	Paper	Scissors
Rock	0	-1	1
Paper	1	0	-1
Scissors	-1	1	0

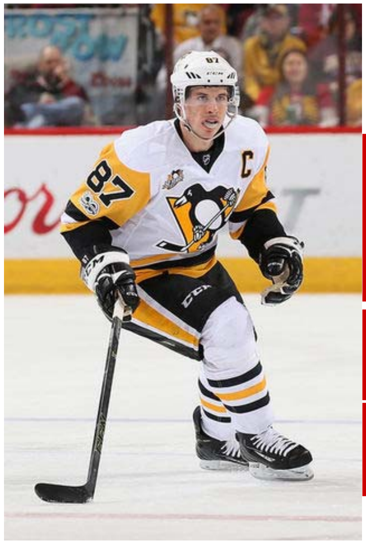
Alice wins ➔ Bob loses (and vice-versa)

Minimax Optimal Strategy (possibly randomized) best strategy you can find given that opponent is rational (and knows your strategy)

Minimax Optimal for Rock-Paper-Scissors: play each action with probability 1/3.

2-Player Zero-Sum Games

Example: Shooter-Goalie



	Block Left	Block Right
Kick Left	1/2	0.9
Kick Right	0.8	1/3

Shooter scores 80% of time when shooter aims right and goalie blocks left

Minimax Optimal Strategy (possibly randomized) best strategy you can find given that opponent is rational (and knows your strategy)

How can we find Minimax Optimal Strategy?

Finding Minimax Optimal Solution using Linear Programming

Variables: $p_1,\ldots p_n$ and v (p_i is probability of action i)
Goal: Maximize v (our expected reward).

Constraints:

- $p_1, \ldots, p_n \geq 0$
- $p_1 + \ldots + p_n = 0$
- For all columns j we have

Expected reward when player 2 takes action j

$$\sum_i p_i m_{ij} \geq v$$

m_{ij} denotes reward when player 1 takes action i and player 2 takes action j.

Extra Slides

Circulation with Demands

Circulation with demands.

- Directed graph $G = (V, E)$.
- Edge capacities $c(e)$, $e \in E$.
- Node supply and demands $d(v)$, $v \in V$.

demand if $d(v) > 0$; supply if $d(v) < 0$; transshipment if $d(v) = 0$

Def. A **circulation** is a function that satisfies:

- For each $e \in E$: $\quad 0 \leq f(e) \leq c(e)$ (capacity)
- For each $v \in V$: $\quad \sum_{e \text{ in to } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$ (conservation)

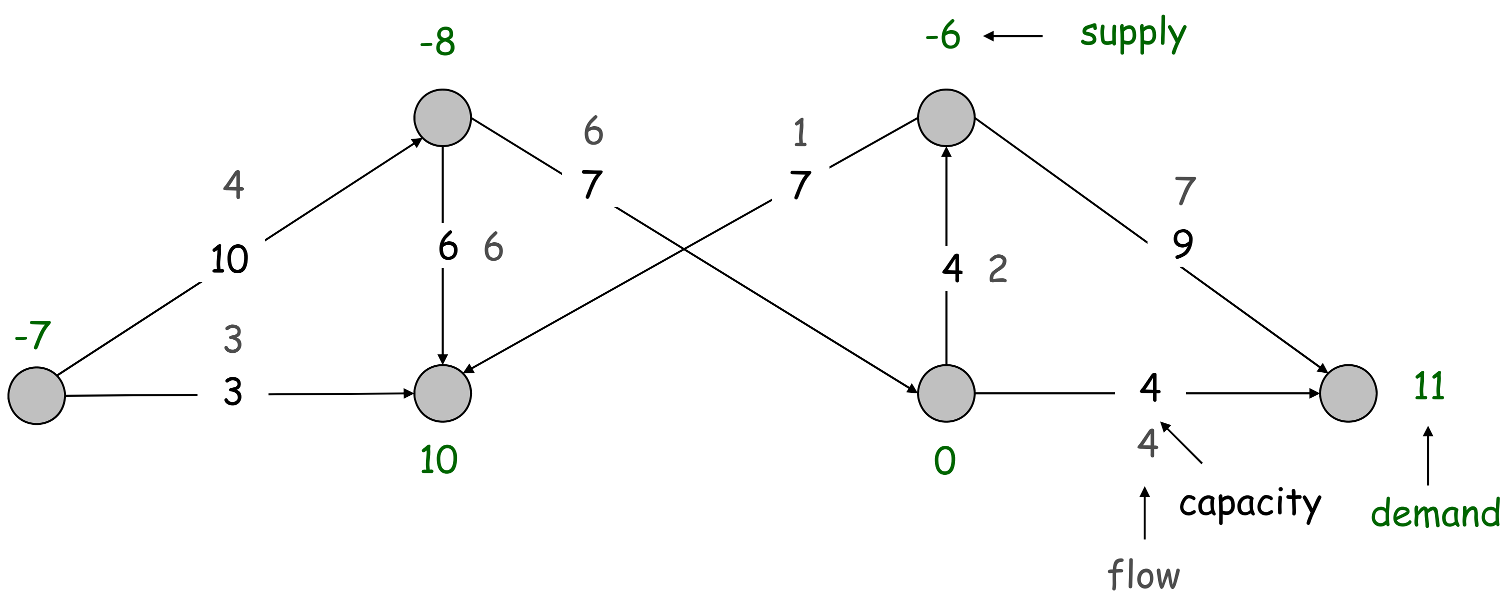
Circulation problem: given (V, E, c, d), does there exist a circulation?

Circulation with Demands

Necessary condition: sum of supplies = sum of demands.

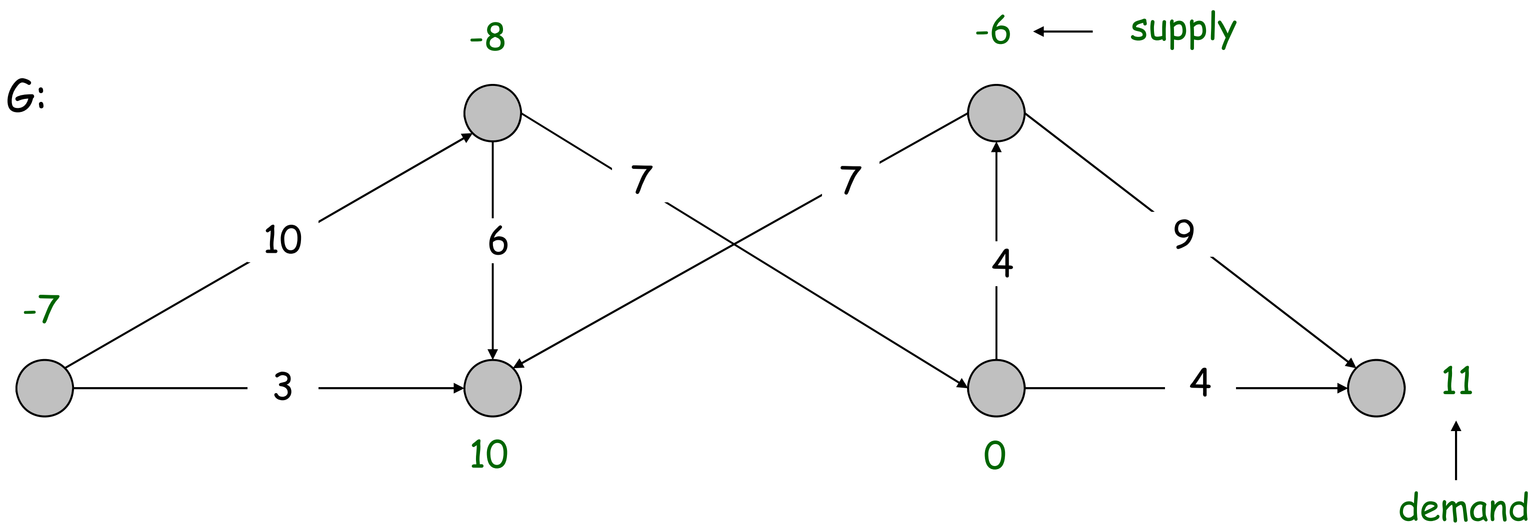
$$\sum_{v\,:\,d(v)>0} d(v) \;=\; \sum_{v\,:\,d(v)<0} -d(v) \;=:\; D$$

Pf. Sum conservation constraints for every demand node v.



Circulation with Demands

Max flow formulation.



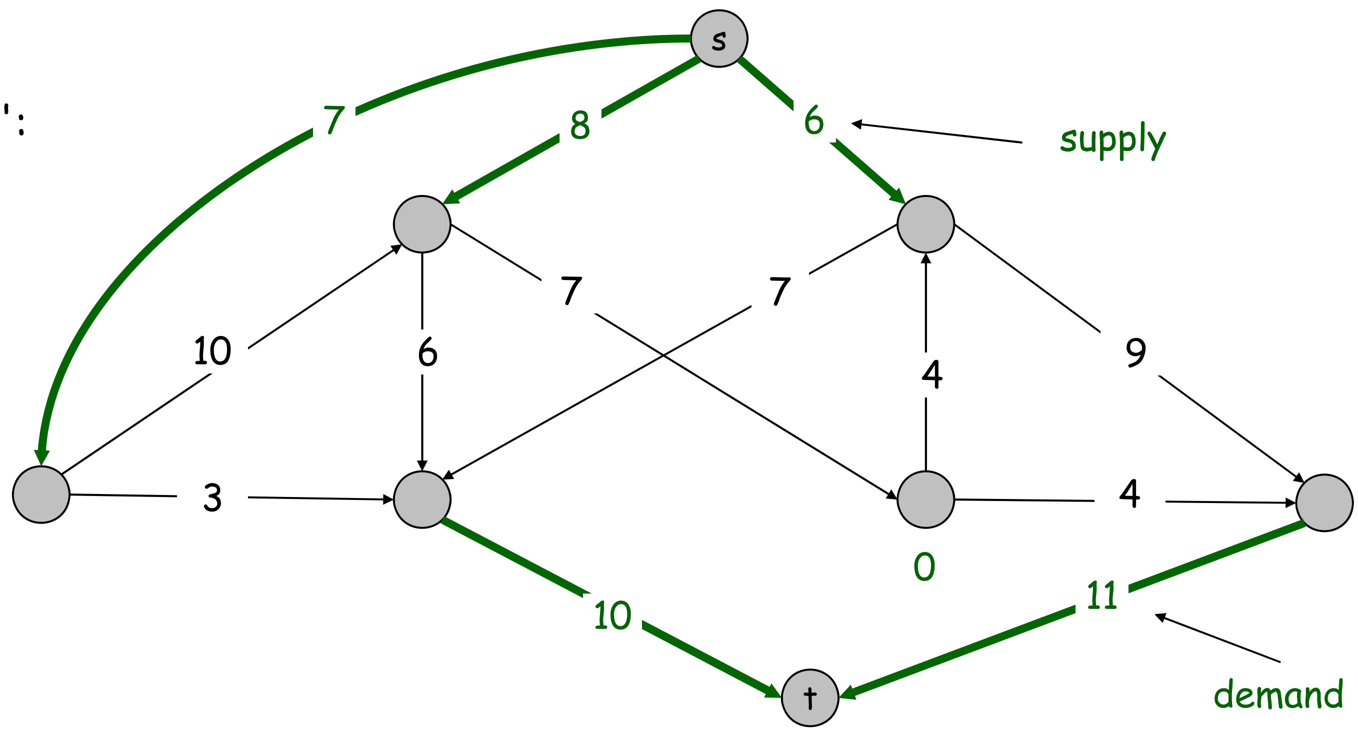
Circulation with Demands

Max flow formulation.

- Add new source s and sink t.
- For each v with d(v) < 0, add edge (s, v) with capacity -d(v).
- For each v with d(v) > 0, add edge (v, t) with capacity d(v).
- Claim: G has circulation iff G' has max flow of value D.

saturates all edges leaving s and entering t

G':



Circulation with Demands

Integrality theorem. If all capacities and demands are integers, and there exists a circulation, then there exists one that is integer-valued.

Pf. Follows from max flow formulation and integrality theorem for max flow.

Characterization. Given (V, E, c, d), there does not exists a circulation iff there exists a node partition (A, B) such that $\Sigma_{v \in B} d_v > cap(A, B)$

demand by nodes in B exceeds supply of nodes in B plus max capacity of edges going from A to B

Pf idea. Look at min cut in G'.

Circulation with Demands and Lower Bounds

Feasible circulation.

- Directed graph $G = (V, E)$.
- Edge capacities $c(e)$ and lower bounds $\ell(e)$, $e \in E$.
- Node supply and demands $d(v)$, $v \in V$.

Def. A circulation is a function that satisfies:

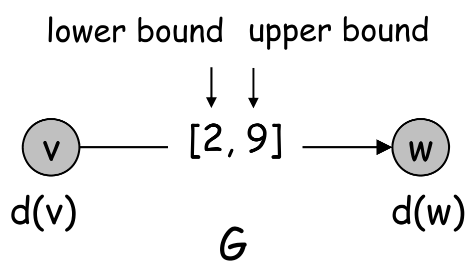
- For each $e \in E$: $\ell(e) \leq f(e) \leq c(e)$ (capacity)
- For each $v \in V$: $\sum_{e \text{ in to } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$ (conservation)

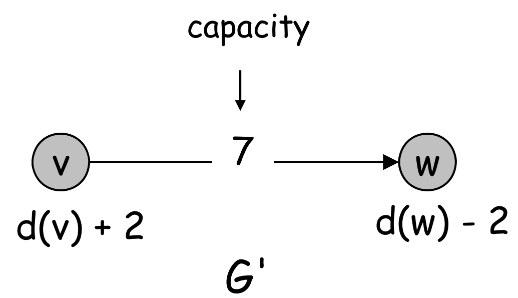
Circulation problem with lower bounds. Given (V, E, ℓ, c, d), does there exists a a circulation?

Circulation with Demands and Lower Bounds

Idea. Model lower bounds with demands.

- Send $\ell(e)$ units of flow along edge e.
- Update demands of both endpoints.





Theorem. There exists a circulation in G iff there exists a circulation in G'. If all demands, capacities, and lower bounds in G are integers, then there is a circulation in G that is integer-valued.

Pf sketch. $f(e)$ is a circulation in G iff $f'(e) = f(e) - \ell(e)$ is a circulation in G'.

7.8 Survey Design

Survey Design

one survey question per product

Survey design.

- Design survey asking n_1 consumers about n_2 products.
- Can only survey consumer i about product j if they own it.
- Ask consumer i between c_i and c_i' questions.
- Ask between p_j and p_j' consumers about product j.

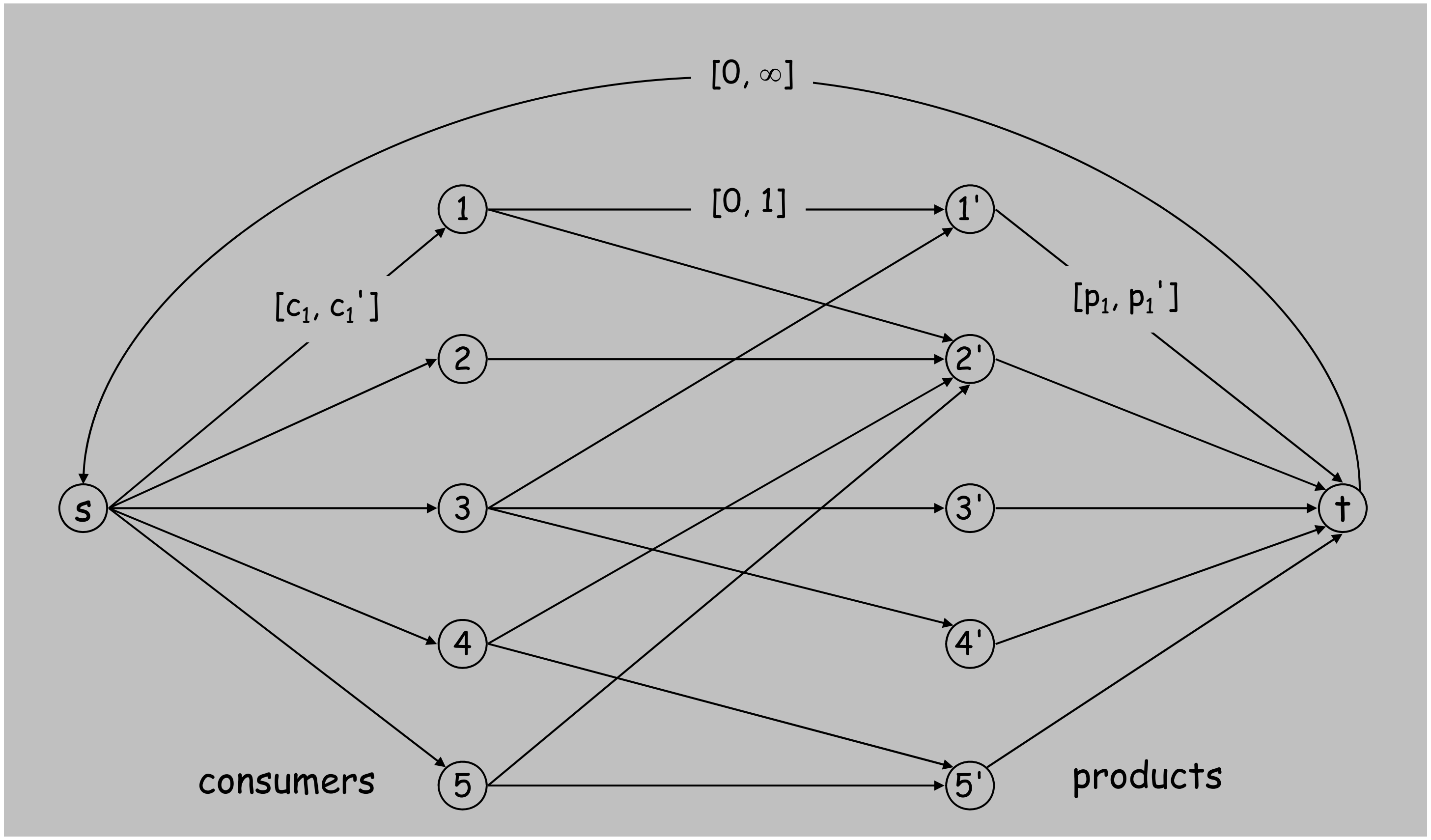
Goal. Design a survey that meets these specs, if possible.

Bipartite perfect matching. Special case when $c_i = c_i' = p_i = p_i' = 1$.

Survey Design

Algorithm. Formulate as a circulation problem with lower bounds.

- Include an edge (i, j) if consumer j owns product i.
- Integer circulation $\Leftrightarrow$ feasible survey design.



7.10 Image Segmentation

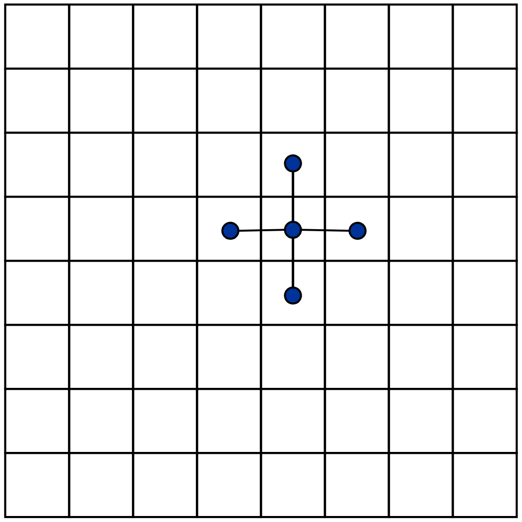
Image Segmentation

Image segmentation.

- Central problem in image processing.
- Divide image into coherent regions.

Ex: Three people standing in front of complex background scene. Identify each person as a coherent object.

Image Segmentation



Foreground / background segmentation.

- Label each pixel in picture as belonging to foreground or background.
- V = set of pixels, E = pairs of neighboring pixels.
- $a_i \geq 0$ is likelihood pixel i in foreground.
- $b_i \geq 0$ is likelihood pixel i in background.
- $p_{ij} \geq 0$ is separation penalty for labeling one of i and j as foreground, and the other as background.

Goals.

- Accuracy: if $a_i > b_i$ in isolation, prefer to label i in foreground.
- Smoothness: if many neighbors of i are labeled foreground, we should be inclined to label i as foreground.
- Find partition (A, B) that maximizes: $\sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}| = 1}} p_{ij}$

foreground background

Formulate as min cut problem.

- Maximization.
- No source or sink.
- Undirected graph.

Turn into minimization problem.

- Maximizing $$\sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}| = 1}} p_{ij}$$

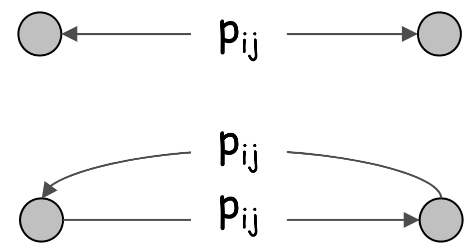
  is equivalent to minimizing $$\underbrace{\left(\sum_{i \in V} a_i + \sum_{j \in V} b_j\right)}_{\text{a constant}} - \sum_{i \in A} a_i - \sum_{j \in B} b_j + \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}| = 1}} p_{ij}$$

- or alternatively $$\sum_{j \in B} a_j + \sum_{i \in A} b_i + \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}| = 1}} p_{ij}$$

Image Segmentation

Formulate as min cut problem.

- $G' = (V', E')$.
- Add source to correspond to foreground; add sink to correspond to background
- Use two anti-parallel edges instead of undirected edge.



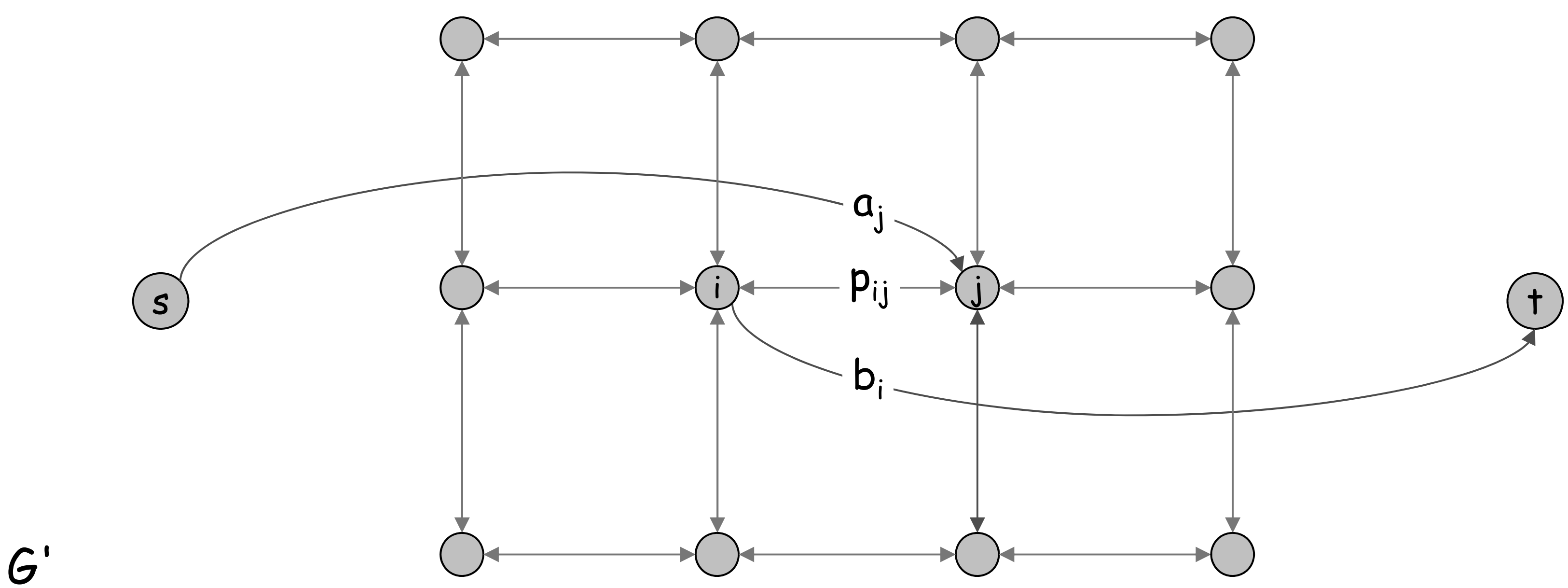


Image Segmentation

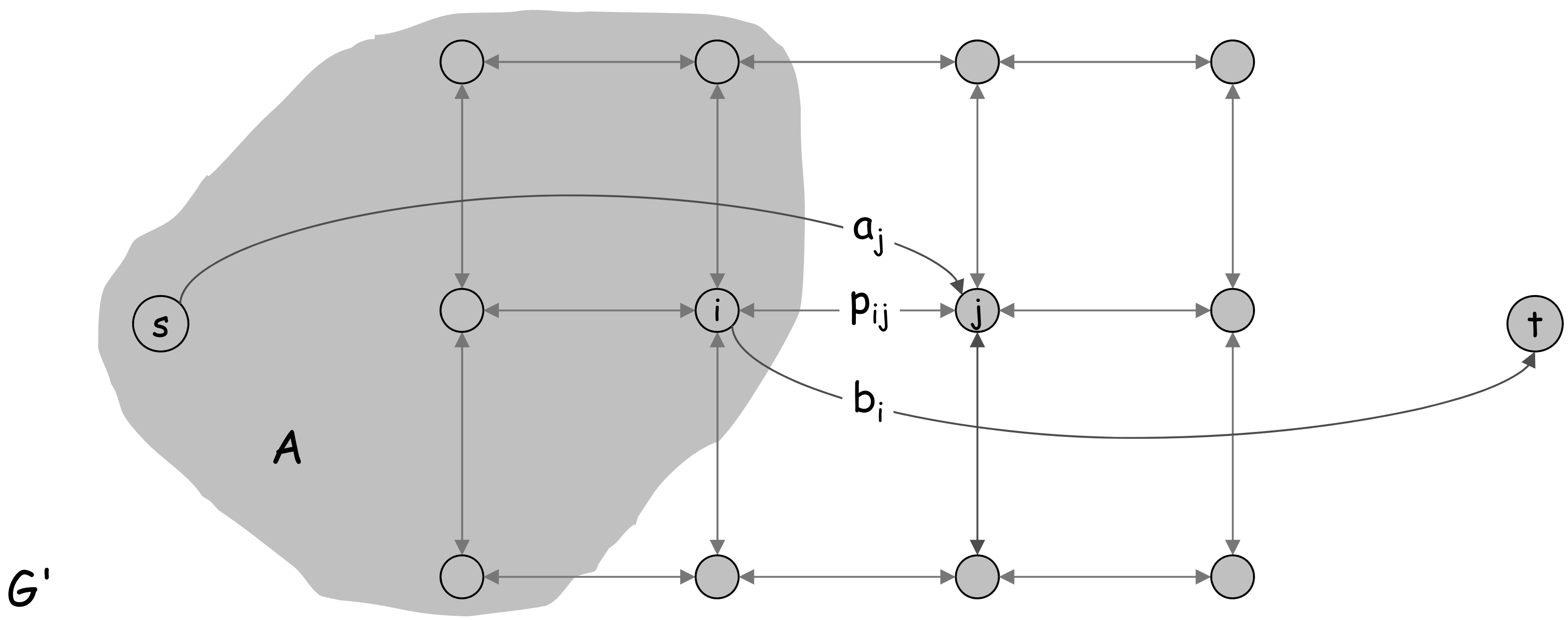
Consider min cut (A, B) in G'.

- A = foreground.

$$cap(A,B) \;=\; \sum_{j \in B} a_j + \sum_{i \in A} b_i \;+\; \sum_{\substack{(i,j) \in E \\ i \in A,\; j \in B}} p_{ij}$$

if i and j on different sides, p_{ij} counted exactly once

- Precisely the quantity we want to minimize.



7.11 Project Selection

Project Selection

can be positive or negative

Projects with prerequisites.

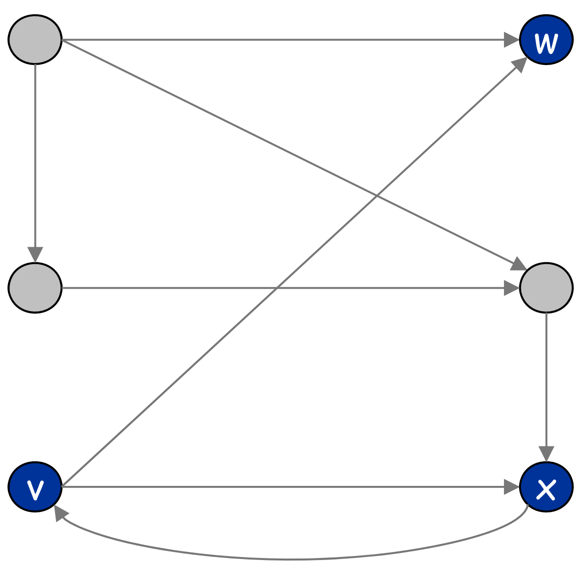
- Set P of possible projects. Project v has associated revenue p_v.
  - some projects generate money: create interactive e-commerce interface, redesign web page
  - others cost money: upgrade computers, get site license
- Set of prerequisites E. If $(v, w) \in E$, can't do project v and unless also do project w.
- A subset of projects $A \subseteq P$ is feasible if the prerequisite of every project in A also belongs to A.

Project selection. Choose a feasible subset of projects to maximize revenue.

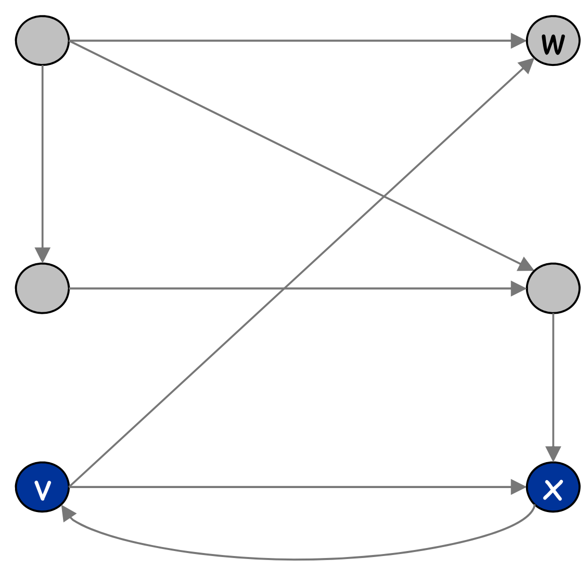
Project Selection: Prerequisite Graph

Prerequisite graph.

- Include an edge from v to w if can't do v without also doing w.
- {v, w, x} is feasible subset of projects.
- {v, x} is infeasible subset of projects.



feasible

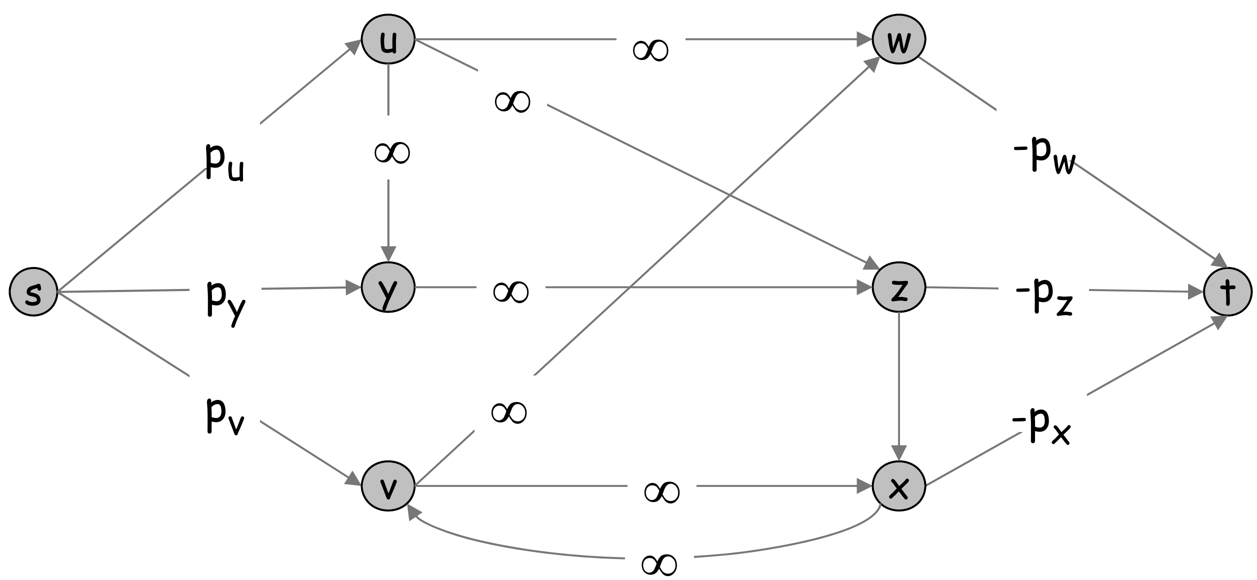


infeasible

Project Selection: Min Cut Formulation

Min cut formulation.

- Assign capacity ∞ to all prerequisite edge.
- Add edge (s, v) with capacity p_v if $p_v > 0$.
- Add edge (v, t) with capacity $-p_v$ if $p_v < 0$.
- For notational convenience, define $p_s = p_t = 0$.

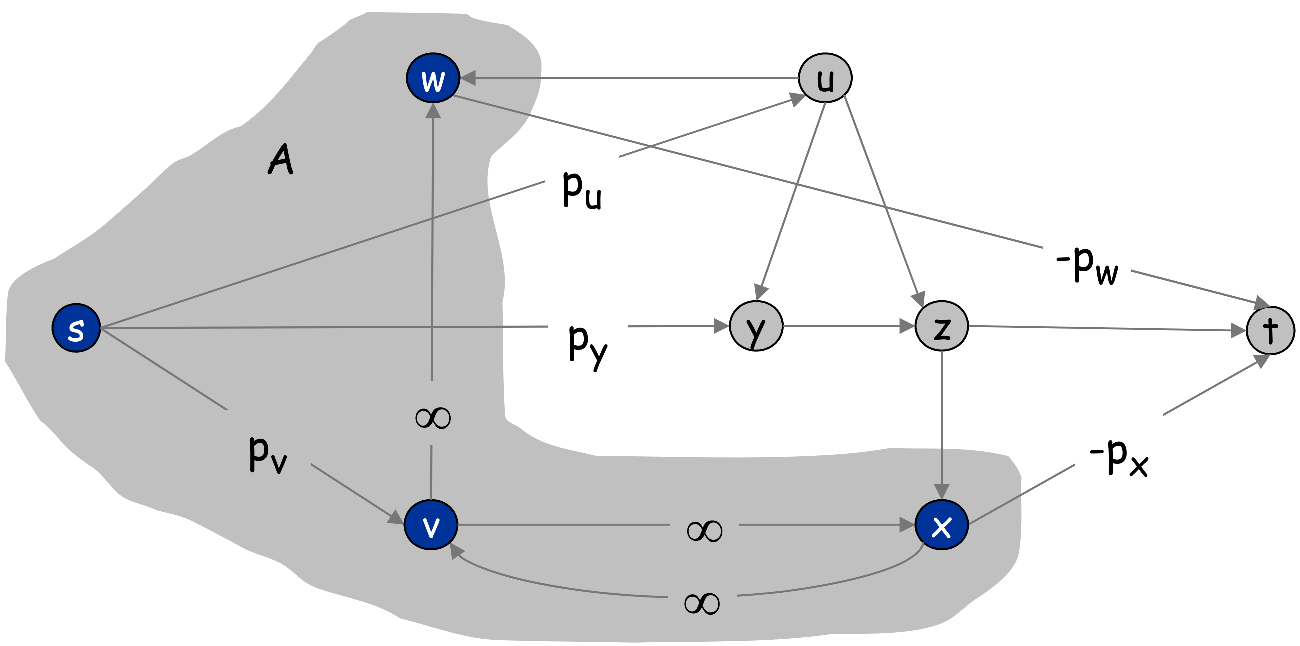


Project Selection: Min Cut Formulation

Claim. (A, B) is min cut iff $A - \{ s \}$ is optimal set of projects.

- Infinite capacity edges ensure $A - \{ s \}$ is feasible.
- Max revenue because:

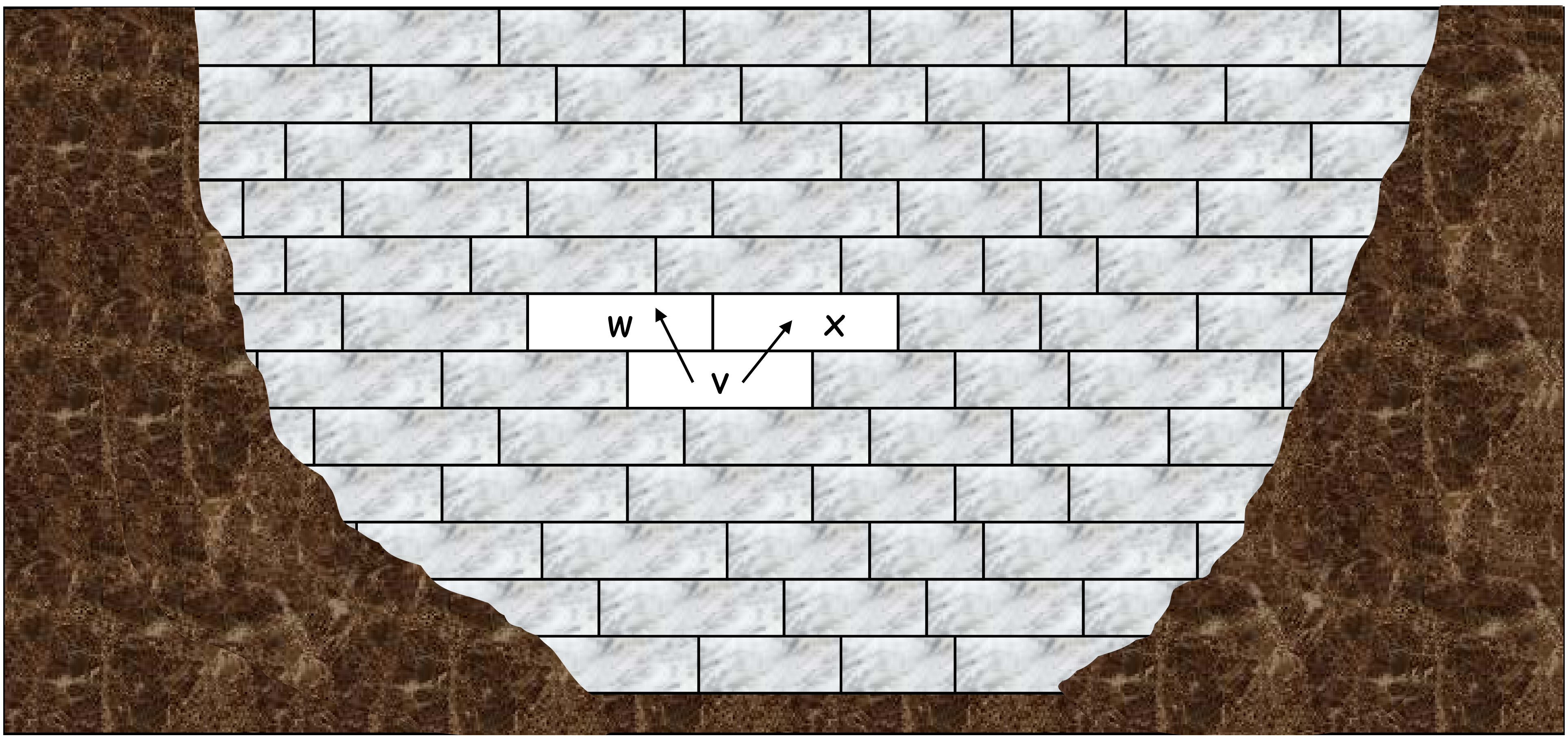
$$\begin{aligned} cap(A, B) &= \sum_{v \in B:\ p_v > 0} p_v + \sum_{v \in A:\ p_v < 0} (-p_v) \\ &= \underbrace{\sum_{v:\ p_v > 0} p_v}_{\text{constant}} - \sum_{v \in A} p_v \end{aligned}$$



Open Pit Mining

Open-pit mining. (studied since early 1960s)

- Blocks of earth are extracted from surface to retrieve ore.
- Each block v has net value p_v = value of ore - processing cost.
- Can't remove block v before w or x.



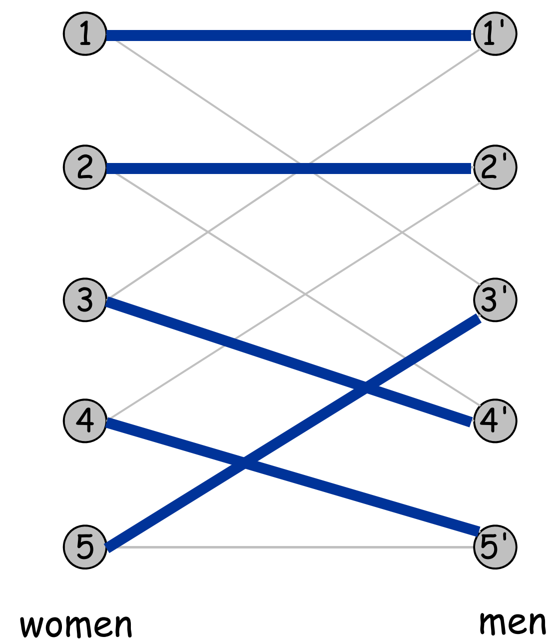
k-Regular Bipartite Graphs

Dancing problem.

- Exclusive Ivy league party attended by n men and n women.
- Each man knows exactly k women; each woman knows exactly k men.
- Acquaintances are mutual.
- Is it possible to arrange a dance so that each woman dances with a different man that she knows?

Mathematical reformulation. Does every k-regular bipartite graph have a perfect matching?

Ex. Boolean hypercube.



k-Regular Bipartite Graphs Have Perfect Matchings

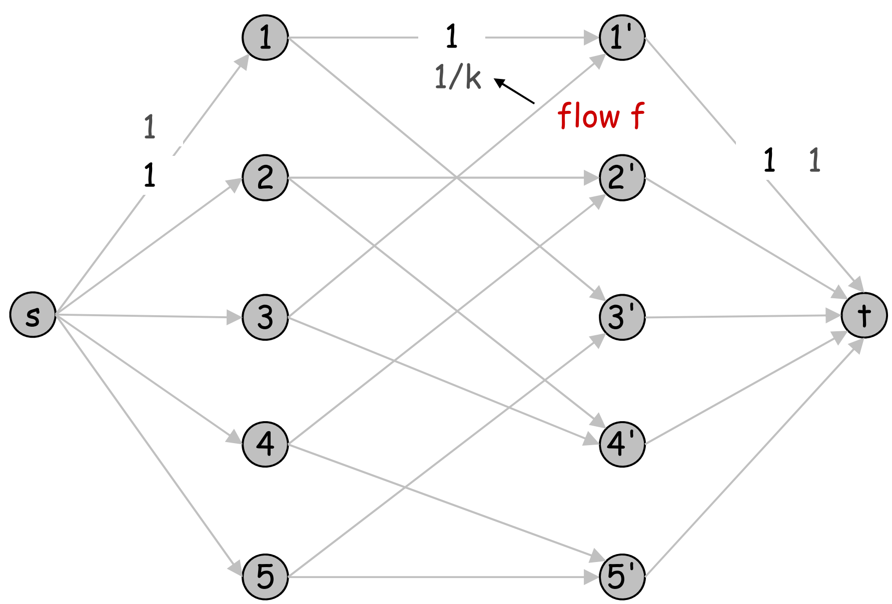
Theorem. [König 1916, Frobenius 1917] Every k-regular bipartite graph has a perfect matching.

Pf. Size of max matching = value of max flow in G'. Consider flow:

$$f(u,v) = \begin{cases} 1/k & \text{if } (u,v) \in E \\ 1 & \text{if } u = s \text{ or } v = t \\ 0 & \text{otherwise} \end{cases}$$

- f is a flow and its value = n $\Rightarrow$ perfect matching. ▪

G'



Census Tabulation (Exercise 7.39)

Feasible matrix rounding.

- Given a p-by-q matrix $D = \{d_{ij}\}$ of real numbers.
- Row i sum = a_i, column j sum b_j.
- Round each d_{ij}, a_i, b_j up or down to integer so that sum of rounded elements in each row (column) equals row (column) sum.
- Original application: publishing US Census data.

Goal. Find a feasible rounding, if one exists.

3.14	6.8	7.3	17.24
9.6	2.4	0.7	12.7
3.6	1.2	6.5	11.3
16.34	10.4	14.5	

original matrix

3	7	7	17
10	2	1	13
3	1	7	11
16	10	15	

feasible rounding

Census Tabulation

Feasible matrix rounding.

- Given a p-by-q matrix $D = \{d_{ij}\}$ of real numbers.
- Row i sum = a_i, column j sum b_j.
- Round each d_{ij}, a_i, b_j up or down to integer so that sum of rounded elements in each row (column) equals row (column) sum.
- Original application: publishing US Census data.

Goal. Find a feasible rounding, if one exists.
Remark. "Threshold rounding" can fail.

0.35	0.35	0.35	1.05
0.55	0.55	0.55	1.65
0.9	0.9	0.9	

original matrix

0	0	1	1
1	1	0	2
1	1	1	

feasible rounding

Census Tabulation

Theorem. Feasible matrix rounding always exists.
Pf. Formulate as a circulation problem with lower bounds.

- Original data provides circulation (all demands = 0).
- Integrality theorem $\Rightarrow$ integral solution $\Rightarrow$ feasible rounding. ▪

3.14	6.8	7.3	17.24
9.6	2.4	0.7	12.7
3.6	1.2	6.5	11.3
16.34	10.4	14.5