Reminder

- PRA6 due 4/16
- HW6 due 4/25
- Course project presentation 4/23 and 4/25
  - Come to OH for discussions!

Artificial Intelligence Methods for Social Good

Lecture 23

Case Study: Optimizing Kidney Exchange

17-537 (9-unit) and 17-737 (12-unit)

Instructor: Fei Fang

feifang@cmu.edu

Recall: 0-1 Knapsack

- 0-1 Knapsack
  - Maximum weight $= 10$
  - How to select items to maximize total value?

Items	1	2	3	4	5
Weight	5	4	2	6	7
Value	4	3	6	9	5

$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.}\, 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \leq 10$$
$$x_i \in \{0,1\}$$

Recall: LP Relaxation

- LP relaxation of an MILP or BIP is the LP with the same linear constraints

MILP

$$\max_x c^T x$$
$$\text{s.t. } Gx \leq h$$
$$x_i \in \mathbb{Z}, i \in J_z$$

LP Relaxation

$$\max_x c^T x$$
$$\text{s.t. } Gx \leq h$$

BIP

$$\max_x c^T x$$
$$\text{s.t. } Gx \leq h$$
$$x_i \in \{0,1\}, \forall i$$

LP Relaxation

$$\max_x c^T x$$
$$\text{s.t. } Gx \leq h$$
$$x_i \in [0,1]$$

Outline

- Basic Kidney Exchange Problem
- Branch and Bound
- Column Generation
- Extension and Discussion

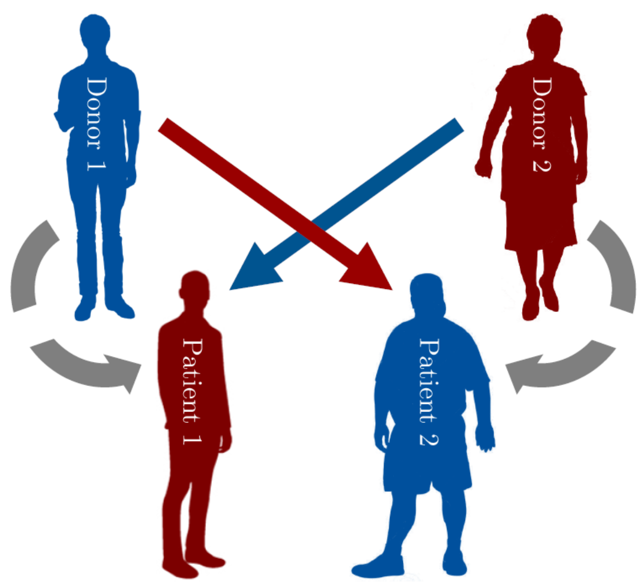
Learning Objectives

- For the kidney exchange problems, briefly describe
  - Significance/Motivation
  - Task being tackled, i.e., what is being predicted/estimated/prescribed
  - Data usage, i.e., what data is used and how it is processed
  - Domain-specific considerations
  - AI method used
  - Evaluation process and criteria
- Describe Branch and Bound and Column Generation methods

Kidney Exchange

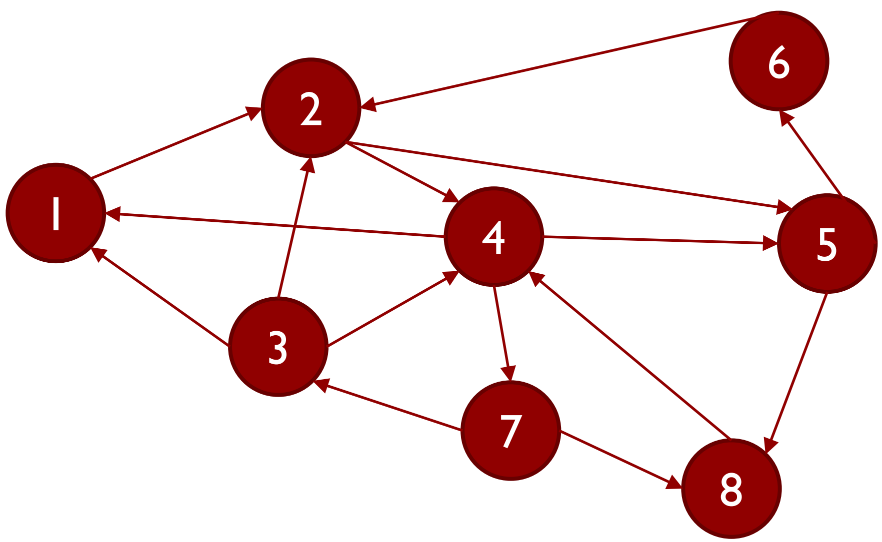
UNOS DONATE LIFE®

UNITED NETWORK FOR ORGAN SHARING



Kidney Exchange Model

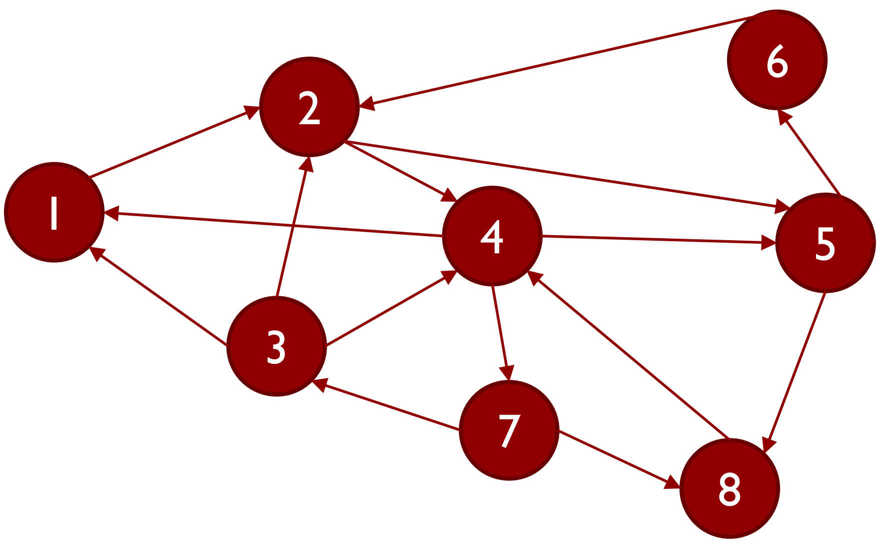
- Given directed graph $G = (V, E)$, where each node represent a patient-donor pair, and an edge $\langle u, v \rangle$ means donor of node u can give one kidney to patient of node v
- The clearing problem: Find a set of disjoint cycles with length $\leq L$ so as to maximize some objective function, e.g., total number of patients matched



What would be a reasonable L?

Poll 1: Kidney Exchange

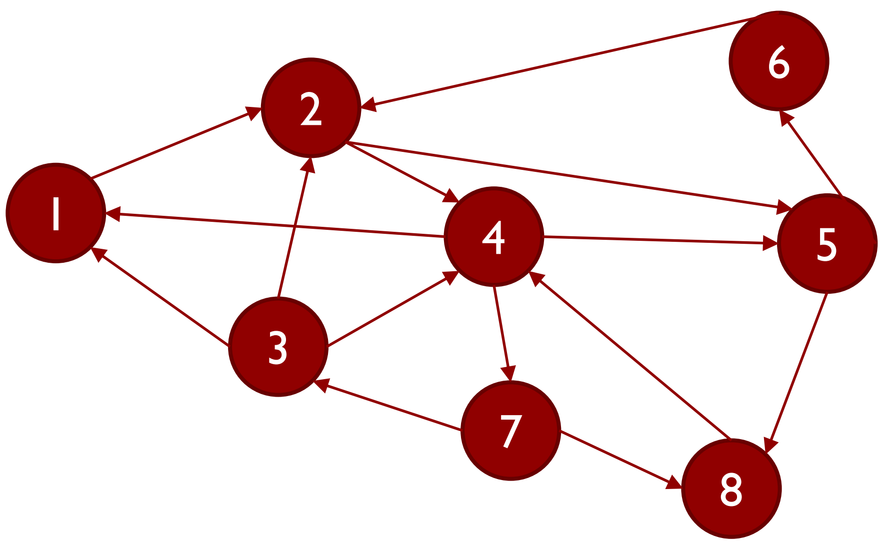
- Given the graph below, what is the maximum number of patients that can get a kidney through kidney exchange assuming the length of each cycle should be less than or equal to 3?
  - A: 3
  - B: 6
  - C: 7
  - D: 8
  - E: None of the above
  - F: I don't know



Cycle-Based ILP Formulation

- Find a set of disjoint cycles with length $\leq L$ so as to maximize the number of patients matched
- Decision variables?
- Constraints?
- Objective function?

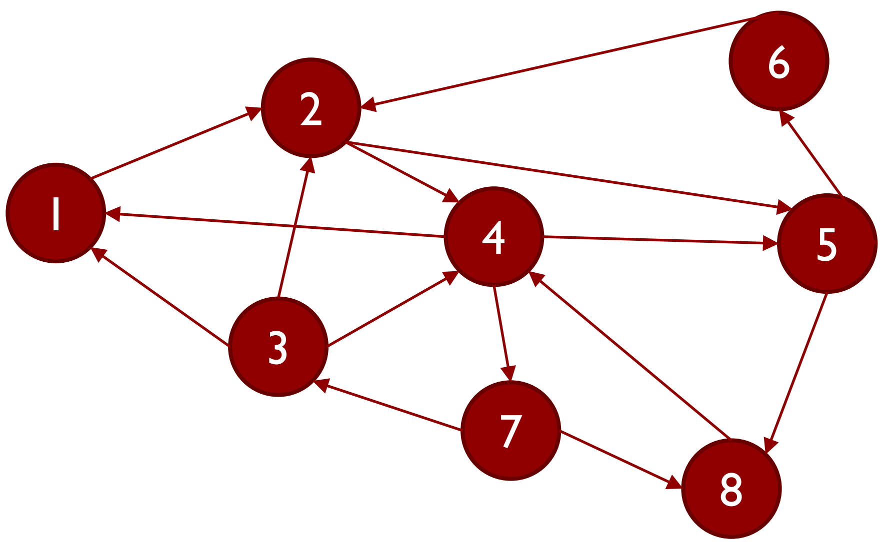
Hint: enumerate all the cycles



Cycle-Based ILP Formulation

- Find a set of disjoint cycles with length $\leq L$ so as to maximize the number of patients matched
- Decision variables?
- Constraints?
- Objective function?

Hint: enumerate all the cycles



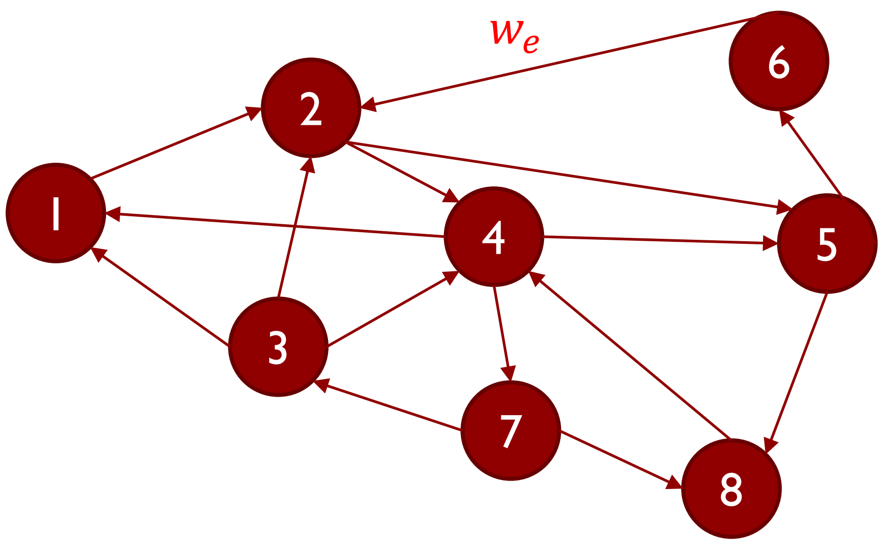
$$\max_{x} \sum_{c} x_c l_c$$

$$\text{s.t. } \textstyle\sum_{c:v \in c} x_c \leq 1, \forall v \in V$$

$$x_c \in \{0,1\}, \forall c$$

Cycle-Based ILP Formulation

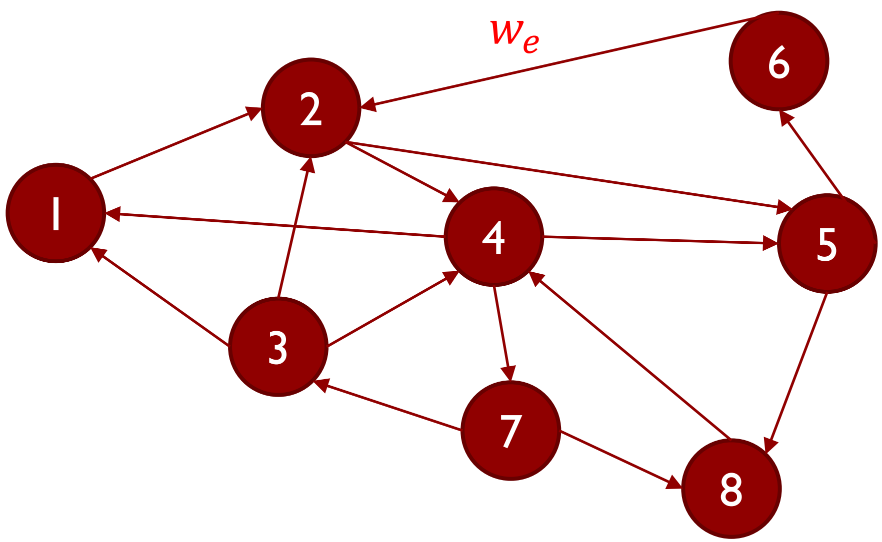
- Find a set of disjoint cycles with length $\leq L$ so as to maximize the total weight if each edge has a weight?
- Decision variables?
- Constraints?
- Objective function?



Max cardinality case is just when all weight $= 1$

Cycle-Based ILP Formulation

- Find a set of disjoint cycles with length $\leq L$ so as to maximize the total weight if each edge has a weight?
- Decision variables?
- Constraints?
- Objective function?



w_c: total weight of the cycle c

$$\max_x \sum_c x_c w_c$$

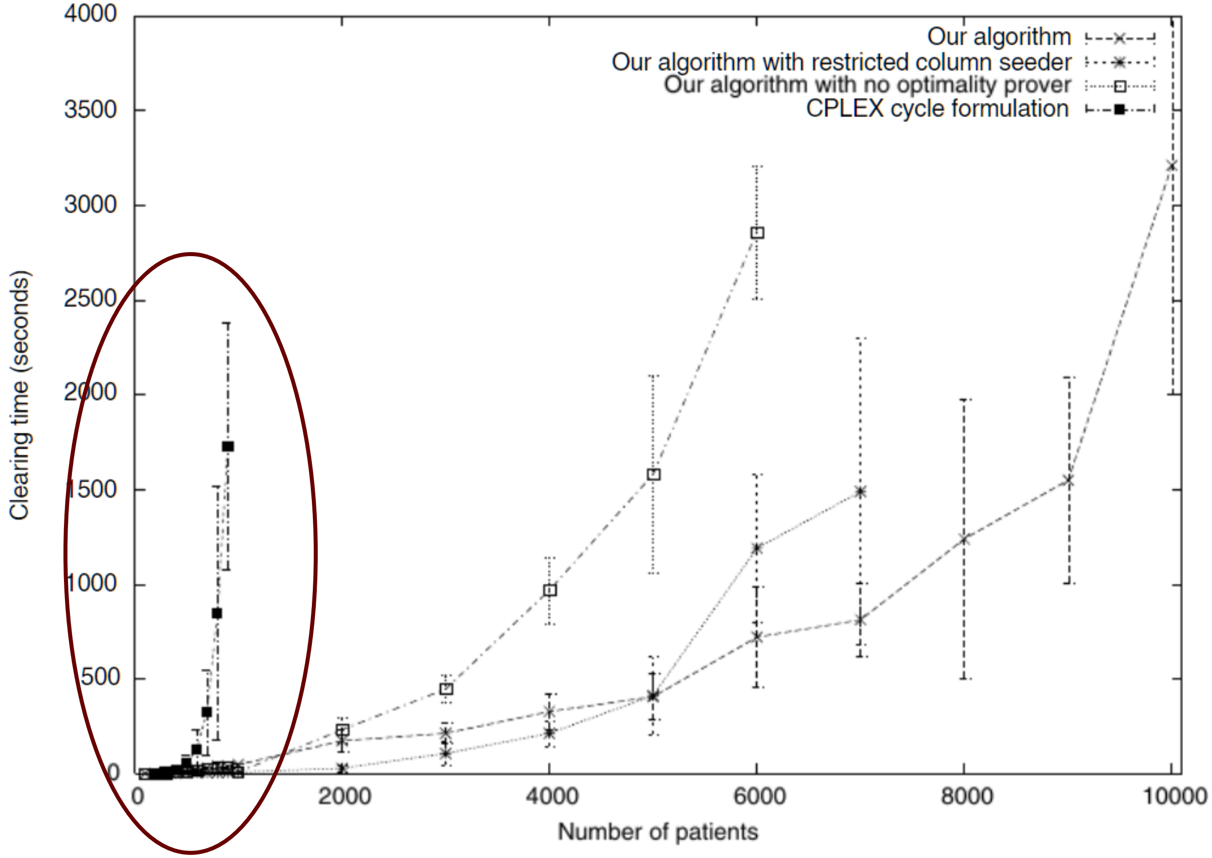
$$\text{s.t.} \sum_{c: v \in c} x_c \leq 1, \forall v \in V$$

$$x_c \in \{0,1\}, \forall c$$

Max cardinality case is just when all weight $= 1$

Cycle-Based ILP Formulation

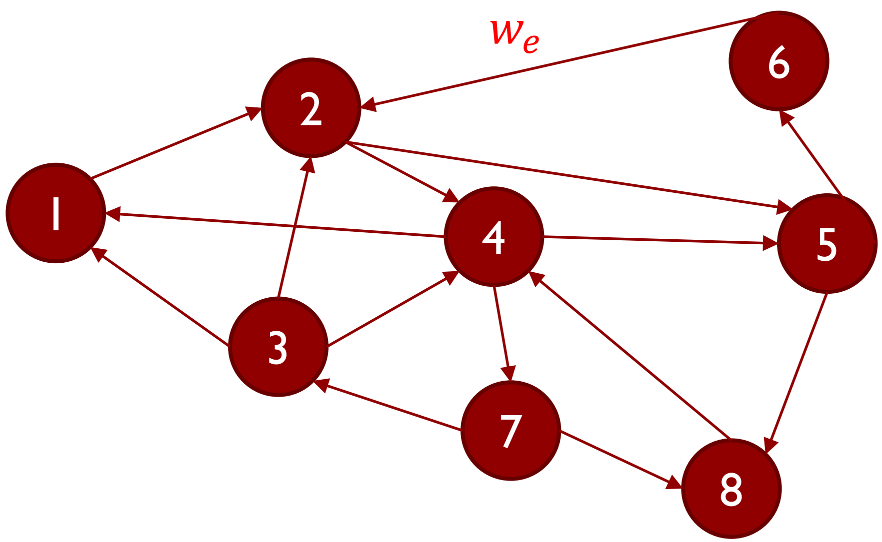
- Limitation: Can only solve for a problem with a few hundred patients   How to improve scalability?



Edge-Based ILP Formulation

- Find a set of disjoint cycles with length $\leq L$ so as to maximize the total weight
- Decision variables?
- Constraints?
- Objective function?

Hint: Use flow conservation constraints



Edge-Based ILP Formulation

- Find a set of disjoint cycles with length $\leq L$ so as to maximize the total weight
- Decision variables?
- Constraints?
- Objective function?

Hint: Use flow conservation constraints

y_e: whether edge e will be selected

$$\max_{y} \sum_{e} y_e w_e$$

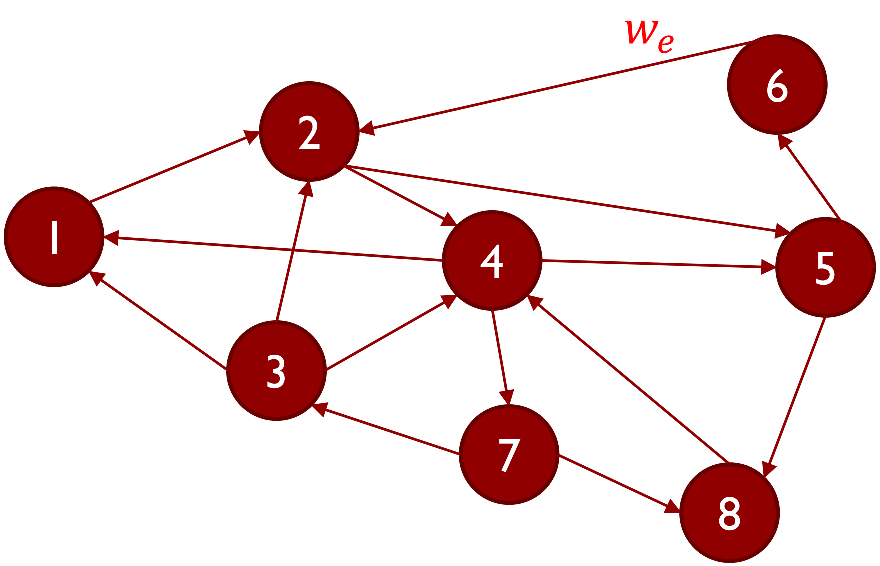
$$\text{s.t.} \sum_{e \in v\rightarrow} y_e - \sum_{e \in \rightarrow v} y_e = 0, \forall v \in V$$

$$\sum_{e \in v\rightarrow} y_e \leq 1, \forall v \in V$$

$$y_e \in \{0,1\}, \forall e$$

$$\sum_{e \in P} y_e \leq L - 1, \forall P \in$$

{Acyclic paths with length L}



Complexity of the Clearing Problem

$$\max_x \sum_c x_c w_c$$

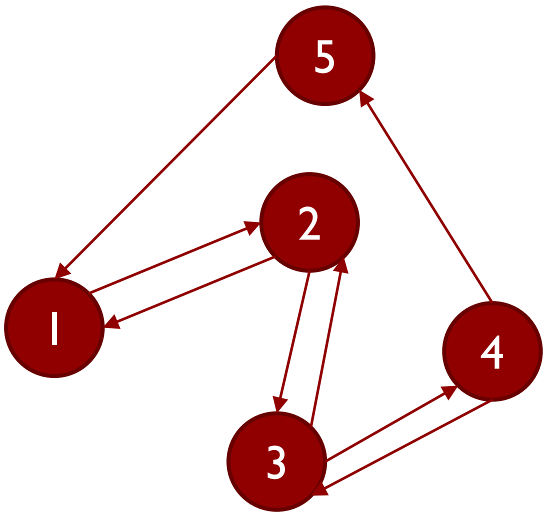
$$\text{s.t.} \sum_{c:v\in c} x_c \leq 1, \forall v \in V$$

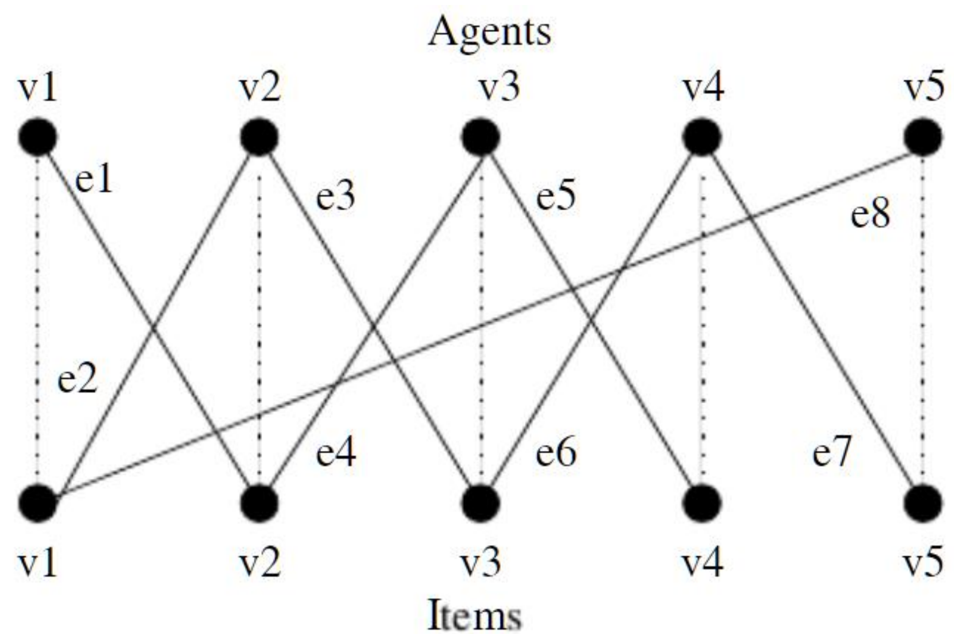
$$x_c \in \{0,1\}, \forall c$$

- When $L = 2$, the clearing problem can be solved in polynomial time
  - Satisfy total unimodulaity, can solve the LP relaxation directly
- The clearing problem with $2 < L < +\infty$ is NP-complete

Complexity of the Clearing Problem

- When $L = +\infty$, i.e., no length constraint, the clearing problem can be solved in polynomial time (maximum weight bipartite matching, Hungarian Maximum Matching Algorithm)



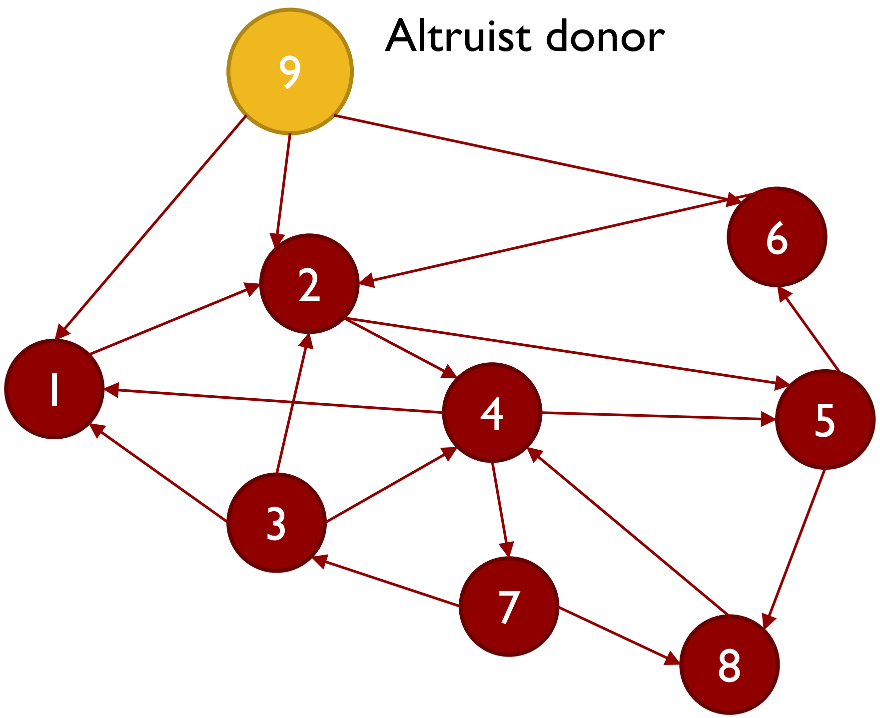


How would max length make a difference?

- Significantly better solutions can be obtained by just allowing cycles of length 3 instead of allowing 2-cycles only. In practice, a cycle length cap of 3 is typically used.

Kidney Exchange with Chains

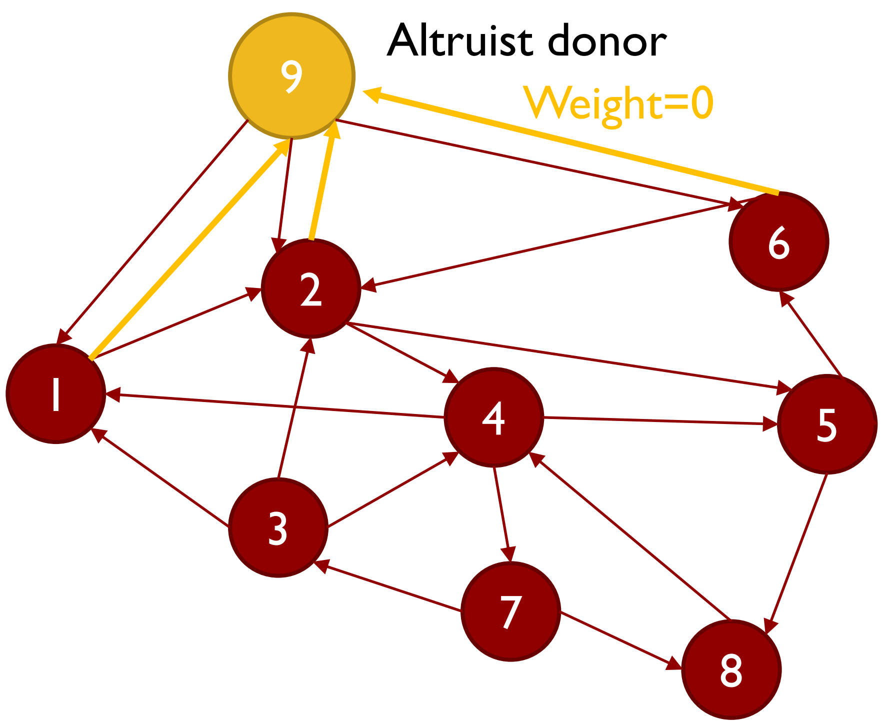
- What if an altruist donor enters the pool offering to donate a kidney to any needy candidate in the pool without a candidate patient?



Does not scale when L is too large.

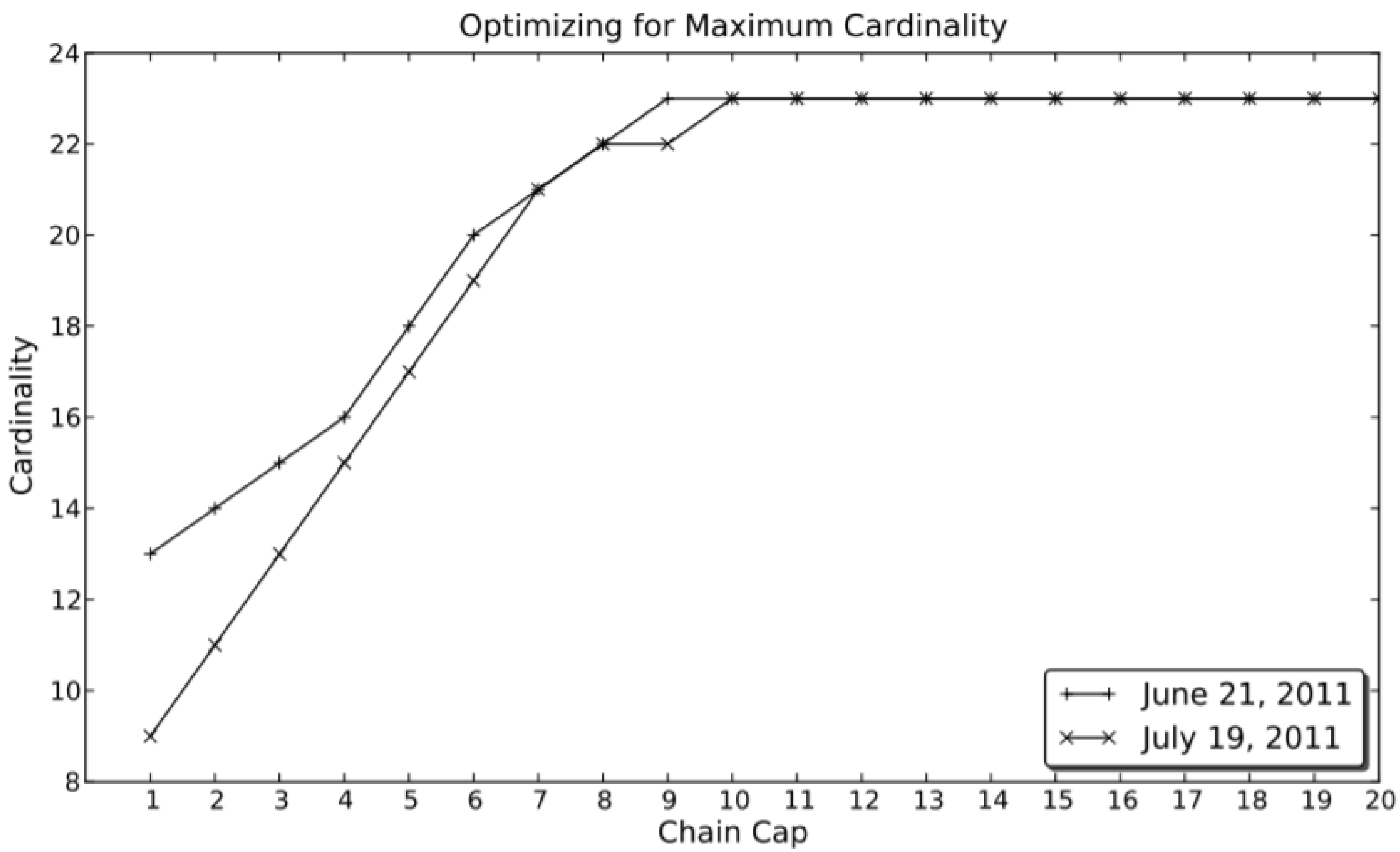
Kidney Exchange with Chains

- What if an altruist donor enters the pool offering to donate a kidney to any needy candidate in the pool without a candidate patient?



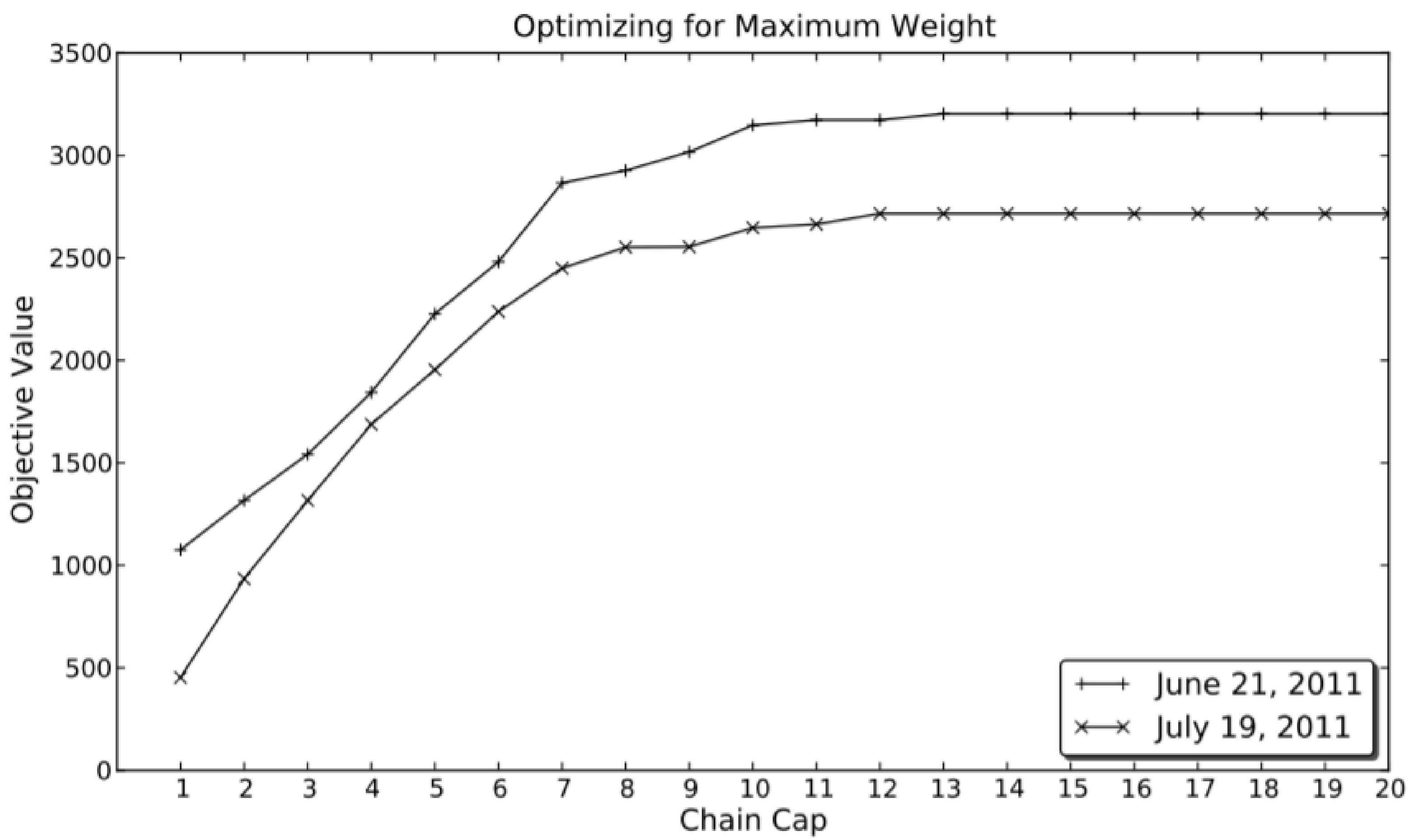
Does not scale when L is too large.

How would max length make a difference?



With UNOS data

How would max length make a difference?



With UNOS data

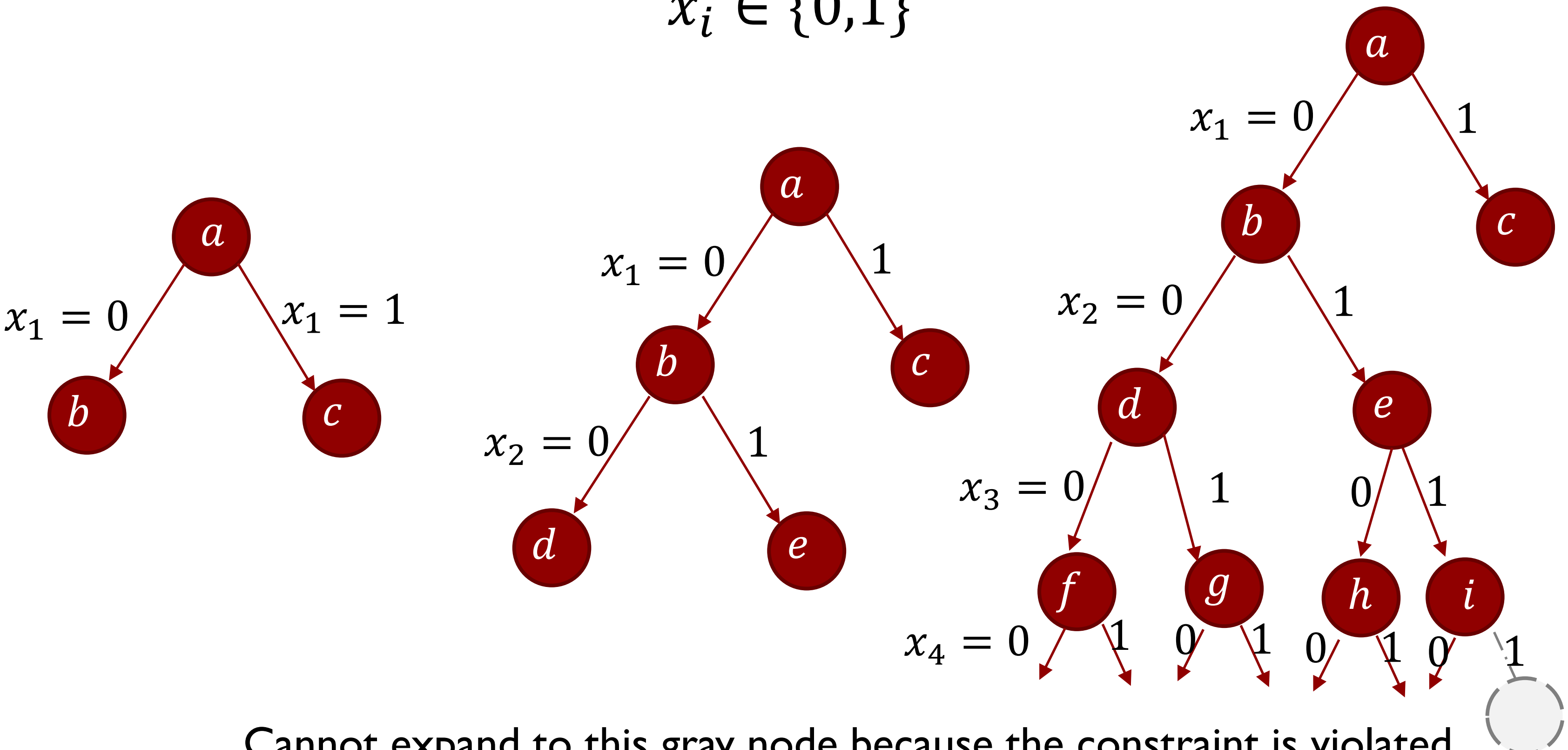
Outline

- Basic Kidney Exchange Problem
- Branch and Bound
- Column Generation
- Extension and Discussion

Improve the scalability

Recall: Depth-First Search for BIP

$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.} 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \le 10$$
$$x_i \in \{0,1\}$$



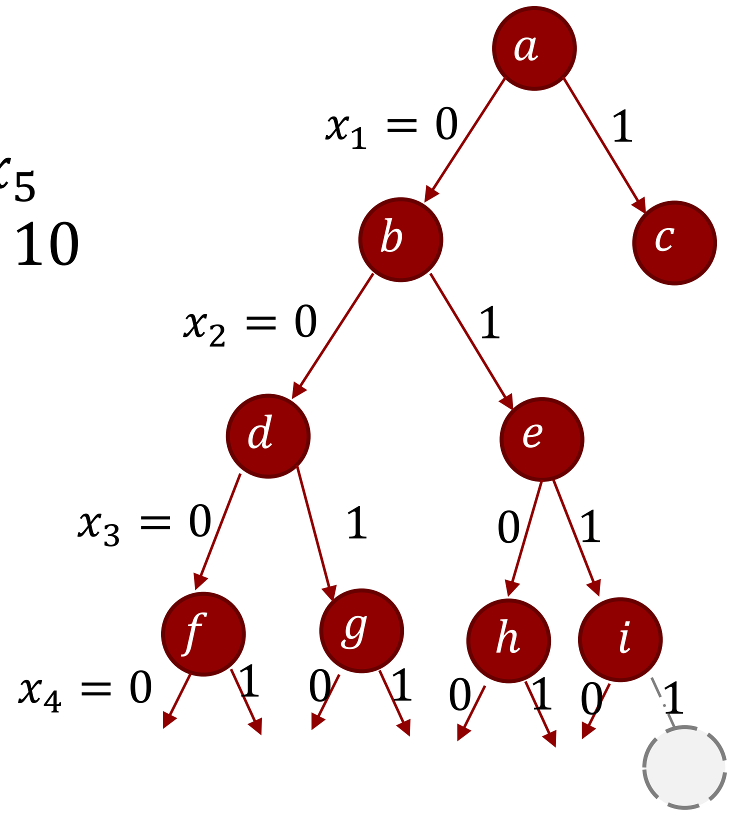
Cannot expand to this gray node because the constraint is violated

Recall: Depth-First Search for BIP

- Can we prune the branches and make search more efficient?

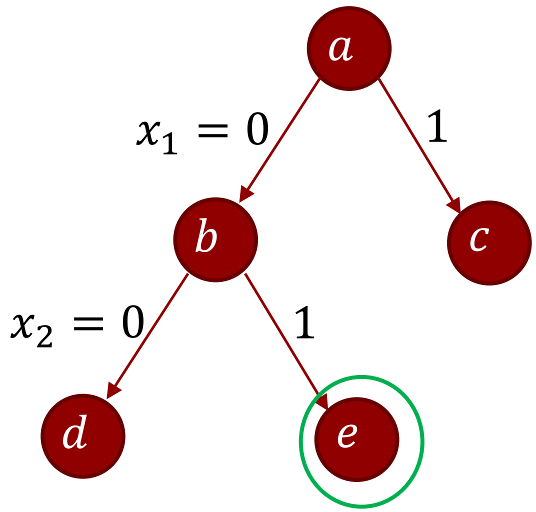
$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.} 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \leq 10$$
$$x_i \in \{0,1\}$$

Estimate upper bound!



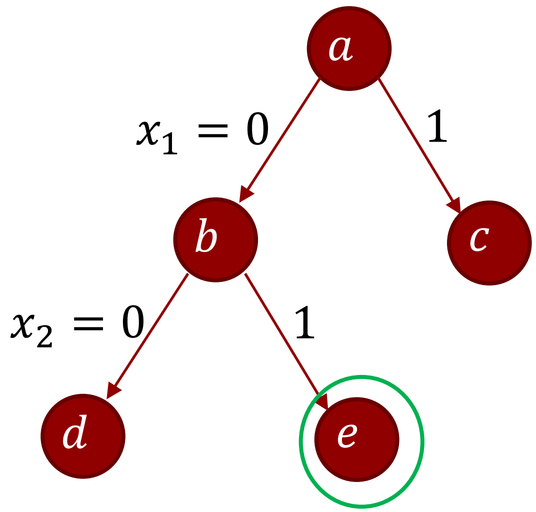
Upper Bound (if maximization): LP Relaxation

$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.} 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \le 10$$
$$x_i \in \{0,1\}$$



Upper Bound (if maximization): LP Relaxation

$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.} 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \leq 10$$
$$x_i \in [0,1], i = 1..5$$
$$x_1 = 0$$
$$x_2 = 1$$

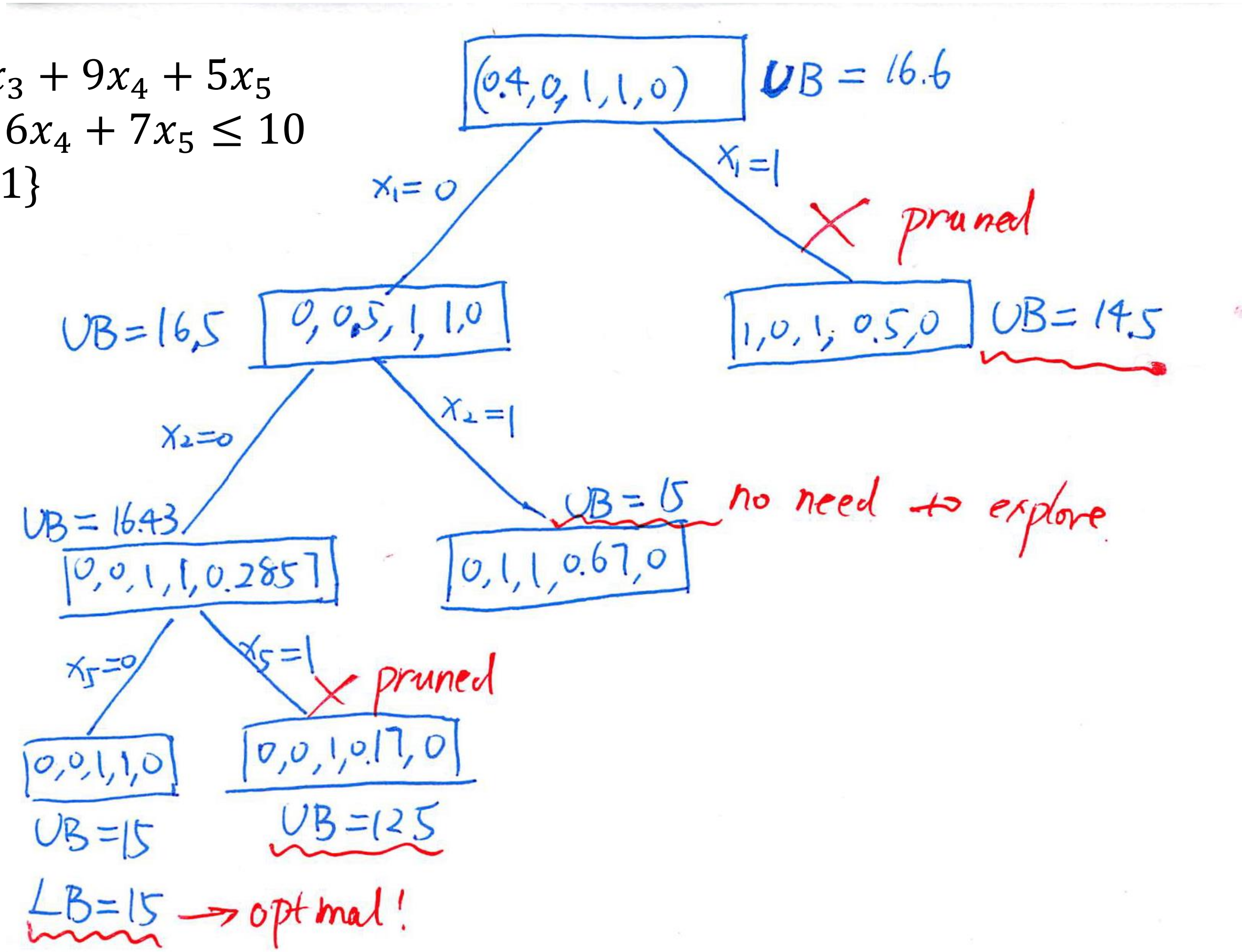


Branch and Bound for BIP

- Branch and Bound overview (assuming maximization)
  - Heuristic search
  - Use optimal objective value of LP relaxation (upper bound) as the heuristic function
  - Always expand the node with the best upper bound first (poly-time computable)
  - Terminate early when best upper bound of remaining nodes is worse than the current best solution

Branch and Bound for BIP: Example

$$\max 4x_1 + 3x_2 + 6x_3 + 9x_4 + 5x_5$$
$$\text{s.t.}\, 5x_1 + 4x_2 + 2x_3 + 6x_4 + 7x_5 \leq 10$$
$$x_i \in \{0,1\}$$



Branch and Bound for BIP

- Solve-LP($\mathcal{C}$) returns (f, x), the optimal objective value and the optimal solution for the LP relaxation of the original problem with additional constraints $\mathcal{C}$

Algorithm: Branch and Bound for BIP

Input: A BIP with $x_i, i = 1..n$ as variables

Initialize $nodelist$ with *Solve-LP*($\{\}$)
Repeat
    Remove a node with best f from $nodelist$: $(f, x, \mathcal{C})$
    If x are all integer valued, return (f, x)
    Get a feasible integer solution $\hat{x}$ based on x, update current best $(\bar{f}, \bar{x})$
    If $\bar{f} \leq f + \epsilon$, return $(\bar{f}, \bar{x})$
    Choose a variable x_i that is not integer valued and add two nodes *Solve-LP*($\mathcal{C} \cup \{x_i = 0\}$) and *Solve-LP*($\mathcal{C} \cup \{x_i = 1\}$) to $nodelist$
Until $nodelist$ is empty

Branch and Bound for MILP

- For MILP
  - BnB: For each integer variable, branching a node by considering $x_i \leq \lfloor \widetilde{x_i} \rfloor$ and $x_i \geq \lceil \widetilde{x_i} \rceil$ where $\widetilde{x_i}$ is a non-integer value
- Standard BnB has already been integrated into existing (M)ILP solvers in Cplex and Gurobi
- Extension: Branch and Cut
  - On top of branch and bound, use cutting planes (which are essentially linear constraints) to separate current non-integer solution and integer solutions

Outline

- Basic Kidney Exchange Problem
- Branch and Bound
- Column Generation
- Extension and Discussion

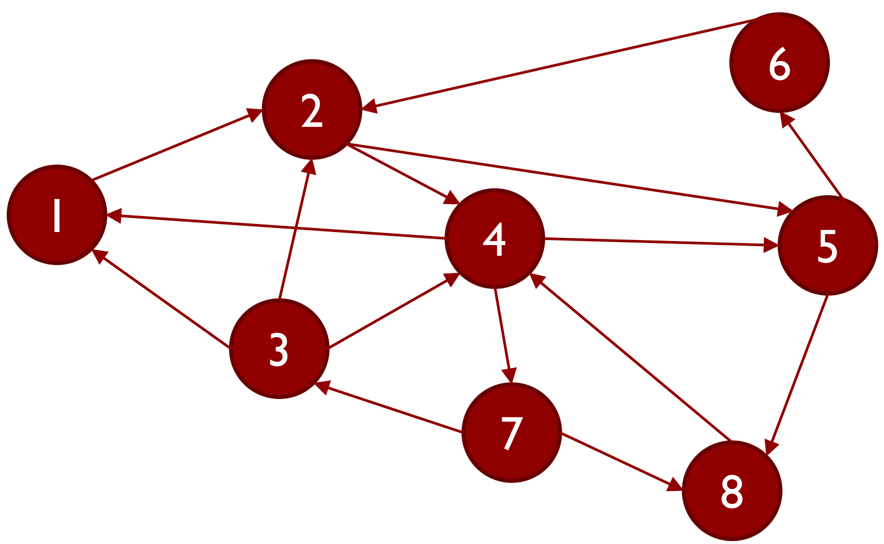
Column Generation

- In kidney exchange: too many edges and cycles
  - Even solving the relaxed LPs is challenging
  - Too many variables (cycle-based formulation) or constraints (edge-based formulation)

	Edges		Length 2 & 3 cycles	
Patients	Mean	Max	Mean	Max
100	2.38e+3	2.79e+3	2.76e+3	5.90e+3
500	6.19e+4	6.68e+4	3.96e+5	5.27e+5
1000	2.44e+5	2.68e+5	3.31e+6	4.57e+6
2000	9.60e+5	1.02e+6	2.50e+7	3.26e+7
3000	2.19e+6	2.28e+6	8.70e+7	9.64e+7
4000	3.86e+6	3.97e+6	1.94e+8	2.14e+8
5000	5.67e+6	6.33e+6	3.60e+8	4.59e+8
6000	8.80e+6	8.95e+6		
7000	1.19e+7	1.21e+7		
8000	1.56e+7	1.59e+7		
9000	1.98e+7	2.02e+7		
10000	2.44e+7	2.51e+7		

Column Generation for Solving LPs

- Start with a restricted LP containing only a small number of columns (variables, i.e., cycles)
- Repeatedly add columns until an optimal solution to this partially formulated LP is an optimal solution to the original LP



$$\max_x \sum_{c \in C'} x_c w_c$$

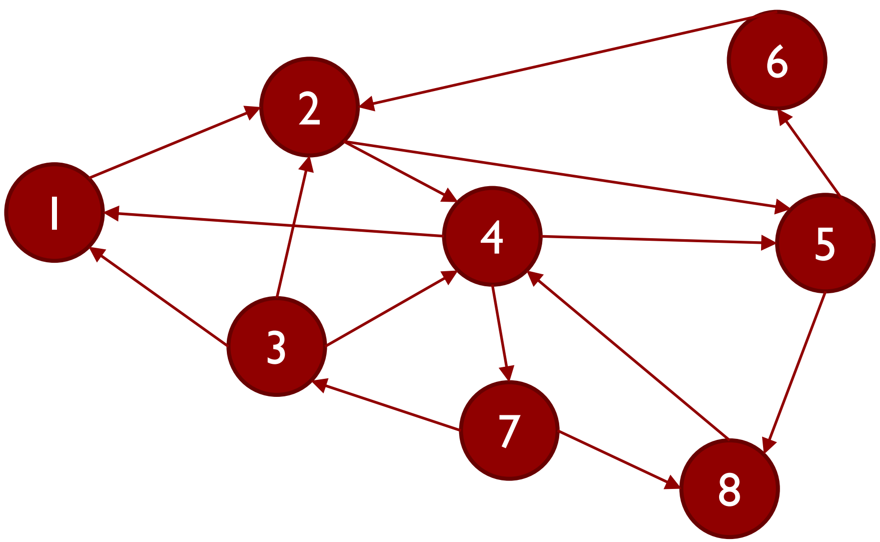
$$\text{s.t.} \sum_{c \in C' : v \in c} x_c \leq 1, \forall v \in V$$

$$x_c \in [0,1], \forall c \in C'$$

Column Generation

- $C' = \{124, 256\}$, solution?
- Add 347 to C'
- $C' = \{124, 256{,}347\}$, solution?

How to determine which cycle to add to C'? (Pricing Problem)



$$\max_x \sum_{c \in C'} x_c w_c$$

s.t. $\sum_{c \in C' : v \in c} x_c \leq 1, \forall v \in V$

$$x_c \in [0,1], \forall c \in C'$$

Pricing Problem for Kidney Exchange

- Goal: Find a new cycle to be added (for cycle-based formulation)
  - Rely on dual LP to get the dual value of each vertex
  - Ideally a feasible path with highest total dual value
  - Depth-first search with several pruning rules

$$\max_x \sum_{c \in C'} x_c w_c$$
$$\text{s.t.} \sum_{c \in C' : v \in c} x_c \le 1, \forall v \in V$$
$$x_c \in [0,1], \forall c \in C'$$

$\Rightarrow$

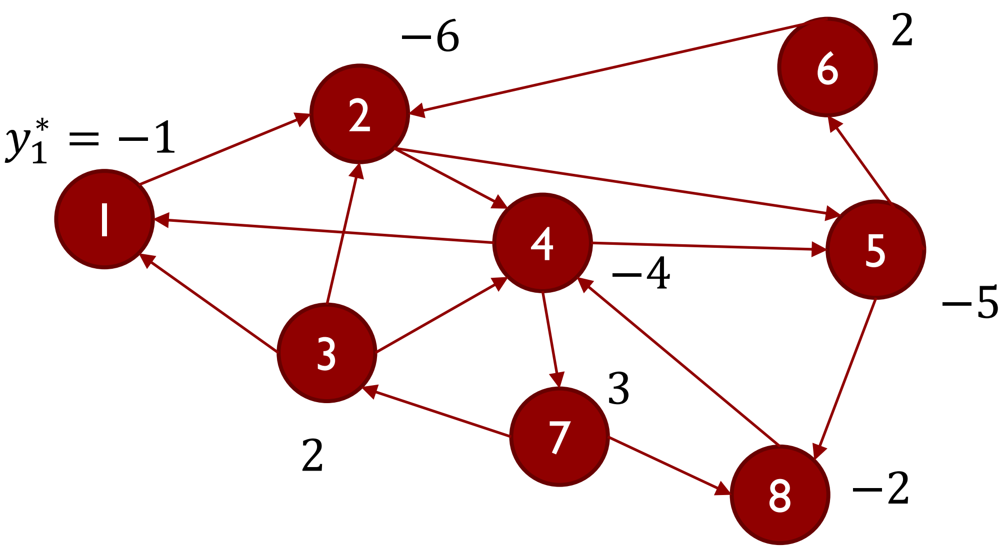
Dual LP

$$\min_y \sum_{v \in V} y_v$$
$$\text{s.t.} \sum_{v \in c} y_v \ge w_c, \forall c \in C'$$
$$y_v \ge 0$$

Optimal dual solution $\{y_v^*\}$

Pricing Problem for Kidney Exchange

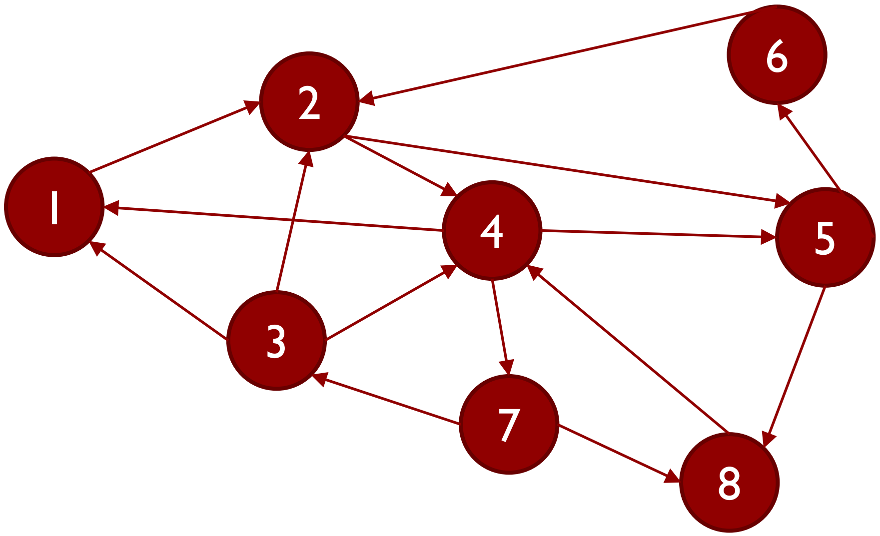
- Goal: Find a new cycle to be added (for cycle-based formulation)
  - Rely on dual LP to get the dual value of each vertex
  - Ideally a feasible path with highest total dual value
  - Depth-first search with several pruning rules



Q: Which cycle should be added?

BnB + Column Generation for Cyble-Based ILP Formulation

- Use BnB to solve the ILP
- When solving a LP relaxation, use column generation
  - 1. Start with a small number of cycles (variables)
  - 2. Solve the LP with the subset of cycles
  - 3. Check if a cycle can be added to the subset to improve the objective function (the most). If so, add it to the subset and go back to 2



$$\max_{x} \sum_{c} x_c w_c$$

$$\text{s.t.} \sum_{c:v \in c} x_c \leq 1, \forall v \in V$$

$$x_c \in \{0,1\}, \forall c$$

Similar ideas can be applied to edge-based ILP

Outline

- Basic Kidney Exchange Problem
- Branch and Bound
- Column Generation
- Discussion for Extensions (optional)

Discussion for Extensions

- Real-world settings can be much more complex than what the basic model describes

Deal with Uncertainty

- Uncertainty always exists in practice
- Which part of the basic model can be extended to consider uncertainty in real-world settings?

Deal with Uncertainty

- How do we deal with uncertainty?
  - Probabilistic
    - Compute expectation
  - Non-probabilistic
    - Maximin Criterion (Wald's Maximin Model)
    - Minimax Regret Criterion

A Simple Example

- Uncertainty in the existence of some edges
- Maximin: Maximize the worst case utility (Conservative)

$$\max_{x \in X} \min_{\mathrm{u} \in U(x)} f(x, u)$$

- Solution under the maximin paradigm:

Ignore the uncertain edges

Minimax regret

- Minimize maximum regret (Less conservative)

$$\min_{x \in X} \max_{u \in U(x)} f(x^*(u), u) - f(x, u)$$

- Let $\tilde{f}(x, u) = f(x, u)\ \forall x, u \in U(x)$ and $\tilde{f}(x, u) = M, \forall x, u \notin U(x)$

$$\min_{x \in X} v$$

$$\text{s.t. } v \geq \tilde{f}(x^*(u), u) - \tilde{f}(x, u), \forall u \in U$$

May still use column generation!

Discussion for Extensions

- What AI methods and paradigms have we learned so far? Can we leverage them to deal with problems in kidney exchange?
  - LP, MILP
  - Linear Regression, Kernel Regression, Decision Trees, Neural Networks
  - Multi-armed Bandit, Monte Carlo Tree Search, Markov Decision Process, Reinforcement Learning
  - Game theory, Stackelberg security games, Human Behavior Modeling

Reference and Related Work

- Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges
- *FutureMatch: Combining Human Value Judgments and Machine Learning to Match in Dynamic Environments [Extended version]*
- *Position-Indexed Formulations for Kidney Exchange [Extended version]*
- Optimizing Kidney Exchange with Transplant Chains: Theory and Reality

Linear Program Duality

- Dual problem of an LP: also a linear program
  - Each dual variable corresponds to a constraint in primal LP

Primal LP	Dual LP
$\max_{x} c^T x$	$\min_{y} b^T y$
s.t. $Ax \leq b$	s.t. $A^T y = c$
	$y \geq 0$
$\boxed{x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, m < n}$	$\boxed{y \in \mathbb{R}^m}$

Linear Program Duality

- Strong duality holds (if feasible and bounded)
  - Primal and dual have the same optimal objective value
- The dual of the dual of a problem is itself

Primal LP

$$\max_x c^T x$$
$$\text{s.t. } Ax \le b$$

$$\boxed{x \in \mathbb{R}^n, A \in \mathbb{R}^{m\times n}, m < n}$$

Dual LP

$$\min_y b^T y$$
$$\text{s.t. } A^T y = c$$
$$y \ge 0$$

$$\boxed{y \in \mathbb{R}^m}$$

Weak duality: $c^T x^* \le b^T y^*$
Strong duality: $c^T x^* = b^T y^*$

Linear Program Duality

- Prove weak duality: $c^T x^* \leq b^T y^*$

Primal LP

$$\max_x c^T x$$
$$\text{s.t. } Ax \leq b$$

$$\boxed{x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, m < n}$$

Dual LP

$$\min_y b^T y$$
$$\text{s.t. } A^T y = c$$
$$y \geq 0$$

$$\boxed{y \in \mathbb{R}^m}$$

Linear Program Duality

- Prove weak duality: $c^T x^* \leq b^T y^*$

$$c^T x^* = \left(A^T y^*\right)^T x^* = y^{*T} A x^* = y^{*T}(A x^*)$$
$$\leq y^{*T} b$$

Primal LP

$$\max_x c^T x$$
s.t. $Ax \leq b$

$$\boxed{x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, m < n}$$

Dual LP

$$\min_y b^T y$$
s.t. $A^T y = c$
$y \geq 0$

$$\boxed{y \in \mathbb{R}^m}$$

Write the Dual of an LP

Maximize	Minimize
ith constraint ≤	ith variable ≥ 0
ith constraint ≥	ith variable ≤ 0
ith constraint =	ith variable unrestricted
jth variable ≥ 0	jth constraint ≥
jth variable ≤ 0	jth constraint ≤
jth variable unrestricted	jth constraint =

Linear Program Duality

- Let LP-1 denote the original LP, LP-2 denote the dual of LP-1, and LP-3 denote the dual of LP-2. Then LP-1 and LP-3 are the same (or can be converted to each other with variable substitution)

Lp-1

$$\min_x c^T x$$
$$\text{s.t. } Ax = b$$
$$x \geq 0$$

Lp-2

$$\max_y b^T y$$
$$\text{s.t. } A^T y \leq c$$

$$y = y^+ - y^-$$

Lp-2 (Standard form)

$$\min_{y^+, y^-, z} \; b^T y^+ - b^T y^-$$
$$\text{s.t. } A^T y^+ - A^T y^- + z = c$$
$$y^+, y^-, z \geq 0$$

dual

LP-3

$$\max_w c^T w$$
$$\text{s.t. } Aw \leq b^T$$
$$Aw \leq -b^T$$
$$w \leq 0$$

$$x = w$$

Proof of strong duality theorem

- Farkas' lemma: Let $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Then exactly one of the following two statements is true
  - I. There exists an $x \in \mathbb{R}^n$ such that $Ax = b$ and $x \geq 0$
  - II. There exists a $y \in \mathbb{R}^m$ such that $A^T y \geq 0$ and $b^T y < 0$
  - Proof:
    - If (I) is true, i.e., $Ax = b$ holds for some x. If $A^T y \geq 0$ for some y, then $b^T y = (Ax)^T y = x^T(A^T y) \geq x^T\mathbf{0} = 0$. So (I)(II) cannot both be true.
    - If (I) is false, then define $C = \{q \in \mathbb{R}^m : \exists x \geq 0, Ax = q\}$. We know $b \neq C$. Notice that C is convex. From separating hyperplane theorem, we know for some $y \in \mathbb{R}^m \backslash \mathbf{0}$ s.t. $q^T y \geq 0\ \forall q \in C$ and $b^T y < 0$. Then we can show that for this y, $A^T y \geq 0$. If not, i.e., if $A^T y < 0$, then choose any $q \in C$, and choose any $x \geq 0$ such that $Ax = q$, we have $0 \leq q^T y = (Ax)^T y = x^T A^T y = x^T(A^T y) < x^T\mathbf{0} = 0$. Contradiction. So this y satisfies $A^T y \geq 0$ and $b^T y < 0$. Therefore (II) is true.
    - So exactly one of (I) and (II) is true

Proof of strong duality theorem

- Second variant of Farkas' lemma: Let $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Then system $Ax \leq b$ has a solution if and only if $\lambda^T b \geq 0$ holds for all λ that satisfies $\lambda \geq 0$ and $\lambda^T A = 0$
  - Proof:
    - If $Ax \leq b$ has a solution, denote the solution as x^*. If $\lambda \geq 0$ and $\lambda^T A = 0$, then $\lambda^T b \geq \lambda^T(Ax^*) = \left(\lambda^T A\right)x^* = 0$
    - If $Ax \leq b$ does not have a solution, then $Ax^+ - Ax^- + z = b, x^+, x^-, z \geq 0$ does not have a solution (otherwise you can easily construct a solution for $Ax \leq b$). By Farkas' lemma, there exists a λ such that $[A \ \ -A \ \ I]^T\lambda \geq 0$ and $b^T\lambda < 0$. Then for this λ, we know $A^T\lambda = 0$ (and therefore $\lambda^T A = 0$) and $\lambda \geq 0$

Proof of strong duality theorem

- Suppose the primal has an optimal solution x^*, leading to optimal value $\text{f}^* = c^T x^*$, $(y^*, g^* = b^T y^*)$ is the optimal solution and the optimal value of the dual, and $f^* > g^*$. Then for any $\epsilon > 0$, we know that $\nexists y, b^T y \geq g^* + \epsilon, A^T y \leq c$, i.e., $\begin{bmatrix} A^T \\ -b^T \end{bmatrix} y \leq \begin{bmatrix} c \\ -g^* - \epsilon \end{bmatrix}$ does not have a solution. Based on the variant of the Farkas' lemma, there exists a $\lambda \in \mathbb{R}^{n+1}$ satisfying $\lambda \geq 0$, $\lambda^T \begin{bmatrix} A^T \\ -b^T \end{bmatrix} = 0$, and $\lambda^T \begin{bmatrix} c \\ -g^* - \epsilon \end{bmatrix} < 0$. Write this λ as $\lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}$ where $\lambda_1 \in \mathbb{R}^n, \lambda_2 \in \mathbb{R}, \lambda_1 \geq 0, \lambda_2 \geq 0$.
- If $\lambda_2 = 0$, then $\lambda_1^T A^T = 0, \lambda_1^T c < 0, \lambda_1 \geq 0$. According to the variant of the Farkas' lemma, $A^T y \leq c$ should not have a solution. But y^* is a solution of the dual and therefore $A^T y^* \leq c$. Contradiction.
- If $\lambda_2 > 0$, then we can scale every the parameters in the problem so that $\lambda_2 = 1$. Then $\lambda_1^T A^T = b^T$ and $\lambda_1^T c < g^* + \epsilon$. Therefore λ_1 is a feasible solution of the primal and has a corresponding objective value lower than $g^* + \epsilon$. Since primal is minimization, we have $f^* \leq c^T \lambda_1 < g^* + \epsilon$. According to weak duality theorem, $f^* \geq g^*$. So $g^* \leq f^* < g^* + \epsilon$ for any $\epsilon > 0$. Then the only possibility is $f^* = g^*$.

Column Generation for Linear Programs

- Column generation is an approach to solving large-scale linear programs with a massive number of variables

- Recall:

$$\max_x c^T x$$
$$\text{s.t. } Ax \le b$$

  - $c \in \mathbb{R}^n$
  - $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$
  - Optimal solution is at a vertex
  - Simplex algorithm: Iteratively move to a neighboring vertex

Column Generation for Linear Programs

- Consider LP in the following form (all LPs can be converted into this form)

$$\max_x c^T x$$
$$\text{s.t. } Ax \le b$$
$$x \ge 0$$

If a variable , say z is unrestricted in the original problem, then introduce two non-negative variables z_+ and z_- substitute z with $z_+ - z_-$

  - $c \in \mathbb{R}^n$
  - $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$

Column Generation for Linear Programs

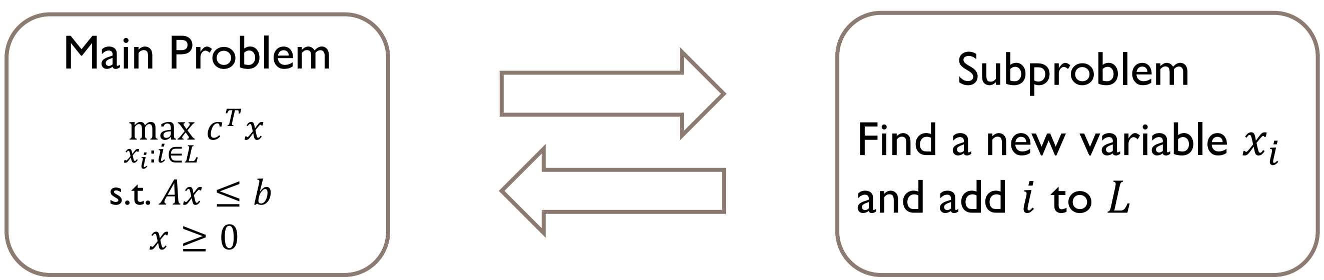
- If $n \gg m$, many variables will be zero at the optimal solution

  Why? The optimal solution is at a vertex. A vertex in the feasible space (which is a subset of $\mathbb{R}^n$) is determined by n equalities. We can get at most m equalities from boundary hyperplanes of constraints in $Ax \leq b$. So we need to use at least $n - m$ boundary lines of the inequality constraints $x \geq 0$, which means those corresponding variables are 0.

- What if $n \ll m$? Then the dual problem would have a lot of zero-valued variables. We can then try to solve the dual problem using column generation, which is called constraint generation.

Column Generation for Linear Programs

- Column generation: Iteratively solve a main problem and a subproblem
- Main problem: The original LP but with a subset of variables (assuming all other variables are zero)
- Subproblem: Identify a new variable to be added to the subset of variables considered by the main problem



Column Generation for Linear Programs

- What is the goal of the subprolem?
- Add a variable that can increase the objective function the most

$$\max_x c^T x$$
$$\text{s.t. } Ax \leq b$$
$$x \geq 0$$

Dual LP

$$\min_y b^T y$$
$$\text{s.t. } A^T y \geq c$$
$$y \geq 0$$

- Assume the optimal solution with only a set L of variables considered is x_L^*, the corresponding optimal dual solution is y_L^*
- The new variable chosen, say x_i, should have the highest "reduced cost", calculated as $c_i - A_i^T y_L^*$ where A_i is the ith column of A, i.e., coefficients w.r.t. to x_i. If the highest reduced cost is non-positive, then no variable will be added, x_L^* is the optimal solution of the original problem with all variables

Reduced Cost Explained

- Reduced cost is an important quantity in LP
- First, convert the LP into "canonical form" by adding slack variables $x_{n+1}, \dots, x_{n+m}$

$$\max_x c^T x \quad \text{s.t. } Ax \le b, \quad x \ge 0$$

$$\begin{aligned} \max_{x_1,\dots,x_{n+m}} \quad & c_1 x_1 + \dots + c_n x_n \\ \text{s.t. } & a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n + x_{n+1} = b_1 \\ & a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n + x_{n+2} = b_2 \\ & \dots \\ & a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n + x_{n+m} = b_m \\ & x_i \ge 0, \forall i \in \{1..n+m\} \end{aligned}$$

- Assume we choose a set of "basic variables" from $\{1..n+m\}$ of size m, called J. Set all variables not in J as 0. The constraints will then be simplified to constraints w.r.t. basic variables only. Then solve this linear system with the m basic variables and m constraints. The solution corresponds to a vertex of the feasible region of the LP in the canonical form shown above. Subselect $x_1, \dots, x_n$ from the solution + the zero-valued non-basic variables lead to a vertex of the feasible region of the original LP.

Reduced Cost Explained

- Formally, denote the new coefficient matrix with slack variables as $\tilde{A} = [A \;\; I], \tilde{c} = \begin{bmatrix} c \\ \mathbf{0} \end{bmatrix}$
- Let $\tilde{A}_J$ be the submatrix of $\tilde{A}$ containing only columns corresponding to variables in J
- Then $x_J = \tilde{A}_J^{-1} b$ and $x_j = 0, \forall j \notin J$ represents a vertex of the feasible region of the following LP

$$
\begin{aligned}
\max_{x_1,\dots,x_{n+m}} \quad & c_1 x_1 + \cdots + c_n x_n \\
\text{s.t. } & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + x_{n+1} = b_1 \\
& a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n + x_{n+2} = b_2 \\
& \cdots \\
& a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n + x_{n+m} = b_m \\
& x_i \geq 0, \forall i \in \{1..n+m\}
\end{aligned}
\quad \Longleftrightarrow \quad
\begin{aligned}
\max_{x \in \mathbb{R}^{n+m}} \quad & \tilde{c}^T x \\
\text{s.t. } & \tilde{A} x = b \\
& x \geq 0
\end{aligned}
$$

Reduced Cost Explained

- Given $x = (x_1, \dots x_{n+m})$ with $x_J = \tilde{A}_J^{-1} b$ and $x_j = 0, \forall j \notin J$
- Consider adjusting x to x' by setting $x_j' = \alpha > 0$ for some $j \notin J$ while ensuring $x_i' = 0 \ \forall i \notin J, i \neq j$ and $\tilde{A}x' = b, x' \geq 0$, i.e., introducing one variable to the current basic variable set
- All $x_i, i \in J$ has to change accordingly
- Denote $x_J' = x_J + \alpha d_J$, then

$$
\begin{gathered}
\tilde{A}x' = b \Rightarrow \tilde{A}_J(x_J + \alpha d_J) + \alpha \tilde{A}_j = b \\
\Rightarrow \tilde{A}_J\left(\cancel{\tilde{A}_J^{-1} b} + \alpha d_J\right) + \alpha \tilde{A}_j \cancel{= b} \\
\Rightarrow \alpha \tilde{A}_J d_J + \alpha \tilde{A}_j = 0 \\
\Rightarrow d_J = -\tilde{A}_J^{-1} \tilde{A}_j
\end{gathered}
$$

Reduced Cost Explained

$$\max_x c^T x \quad \text{s.t. } Ax \le b \quad x \ge 0$$

$$\min_y b^T y \quad \text{s.t. } A^T y \ge c \quad y \ge 0$$

- If $j \in [1..n]$, the new objective value is

$$f(x') = \tilde{c}^T x' = \tilde{c}^T x + \alpha(\tilde{c}_j + \tilde{c}_J^T d_J)$$

- Rewritten as $f(x') = \tilde{c}^T x + \alpha \bar{c}_j$ where

$$\bar{c}_j = \tilde{c}_j + \tilde{c}_J^T d_J = \tilde{c}_j - \tilde{c}_J^T \tilde{A}_J^{-1} \tilde{A}_j$$

Therefore $f(x') > \tilde{c}^T x$ if $\bar{c}_j > 0$

For $j \in \{1..n\}$, $\bar{c}_j$ is called *reduced cost*

Reduced Cost Explained

$$f(x') = \tilde{c}^T x + \alpha \bar{c}_j$$

$$\bar{c}_j = \tilde{c}_j - \tilde{c}_J^T \tilde{A}_J^{-1} \tilde{A}_j$$

- If $\bar{c}_j$ is non-positive for all non-basic variables of a vertex corresponding to basic variable set J, then the vertex is the optimal solution
- If $\bar{c}_j$ is positive for some j, then moving from x to x' can lead to a higher objective value, the higher the value of $\bar{c}_j$, the higher the increase rate. The Simplex algorithm move towards the neighboring vertex with the highest $\bar{c}_j$

Reduced Cost Explained

- If $x^* \in \mathbb{R}^{n+m}$ is the optimal solution of the primal LP in canonical form, and it corresponds to a set of basis J, then consider the corresponding optimal dual solution $y^* \in \mathbb{R}^m$
  - According to complementary slackness, if x_j is in J, then the corresponding dual constraint is tight, i.e., $A_j^T y^* = c_j$ if $j \in \{1..n\}$ and $y_{j-n}^* = 0$ if $j \in \{n+1, \dots, n+m\}$
- Together with the fact $\tilde{A} = [A \;\; I], \tilde{c} = \begin{bmatrix} c \\ \mathbf{0} \end{bmatrix}$, we have $\tilde{A}_J^T y^* = \tilde{c}_J$
- We can conclude: at optimal solution, $\bar{c}_j = \tilde{c}_j - \tilde{c}_J^T \tilde{A}_J^{-1} \tilde{A}_j$ can be rewritten as $\bar{c}_j = c_j - A_j^T y^*$ for $j \in \{1..n\}$

Reduced Cost Explained

- Assume that after you solved an LP and get x^* and the corresponding y^*, you are asked to add a new variable x_j to the LP with coefficient c_j and matrix column A_j
- x^* still corresponds to a vertex in the augmented LP, but it may not be the optimal solution
- We need to check if we introduce j to the basis, whether the objective value will increase
- This can be done by directly checking the reduced cost

Subproblem and Reduced Cost

- Now consider the column generation process.
- It can be viewed as add variables one by one.
- Again, whether and how much a new variable x_j will improve the objective value depends on its reduced cost, computed as $c_i - A_i^T y_L^*$ where y_L^* is the optimal dual solution (without slack variables) before x_j is added