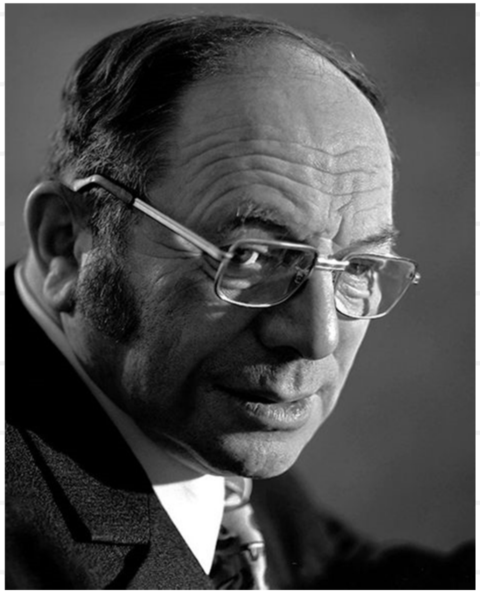
Lecture 17: Column generation

The two problems

The cutting stock problem

- First introduced in 1939
- Led to the invention of Linear programming (and a Nobel price in economics)



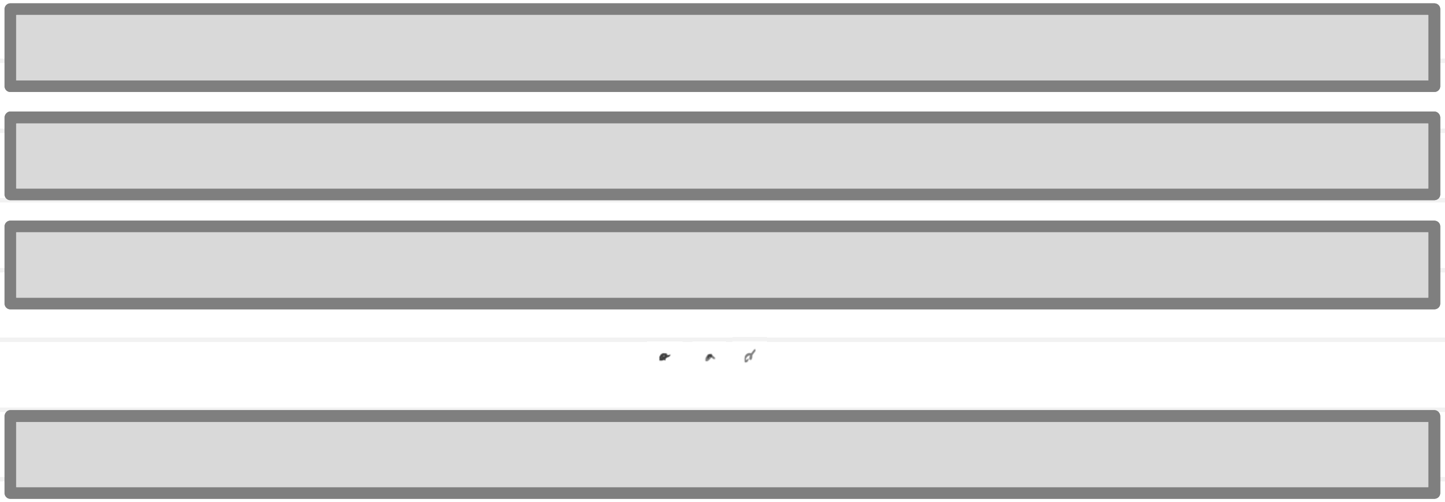
The kidney exchange problem

- Studied (and applied) since early 2000's
- Led to the significant increase of kidney transplants

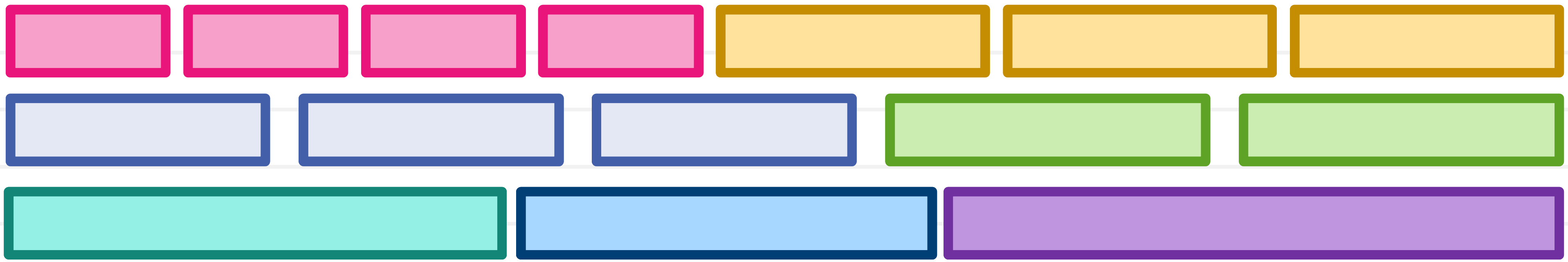
Same computation framework: linear programming with excessive number of variables.

The cutting stock problem

Input: rolls of width W.



Output: set of M rolls with width $w_1, w_2, \ldots w_M$ (each $\leq W$).



Objective: minimize the number of used input rolls.

"Naïve" formulation

$y_i \quad i = 1 \ldots N$

$x_{ij} \quad i = 1 \ldots N, \; j = 1 \ldots M$

$$\min_{x,y} \sum_{i=1}^{N} y_i$$ minimize the number of used big rolls

s.t.

$$\forall j \quad \sum_{i=1}^{N} x_{ij} = 1$$ each small roll has to be produced

$$\forall i \quad \sum_{j=1}^{M} x_{ij} w_j \le W$$ small rolls should fit

$$\forall i,j \quad x_{ij} \le y_i$$ consistency

$$x_{ij}, y_i \in \{0; 1\}$$

"Naïve" formulation

$$\min_{x,y} \sum_{i=1}^{N} y_i$$

$$\text{s.t.} \quad \forall j \ \sum_{i=1}^{N} x_{ij} = 1$$

$$\forall i \ \sum_{j=1}^{M} x_{ij} w_j \le W$$

$$\forall i,j \quad x_{ij} \le y_i$$

$$0 \le x_{ij} \le 1$$

$$0 \le y_i \le 1$$

LP relaxation is "small", but has a useless solution:

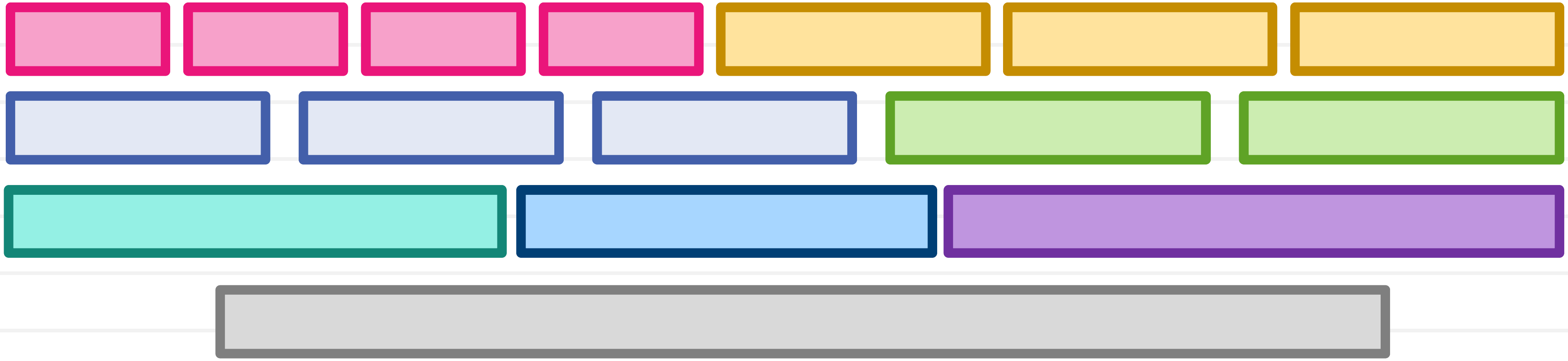
$$y_i = \frac{\sum_j w_j}{NW}$$

$$x_{ij} = \frac{w_j}{NW}$$

$$obj = \sum_j w_j / W$$

Branch-and Bound very inefficient. (Why?)

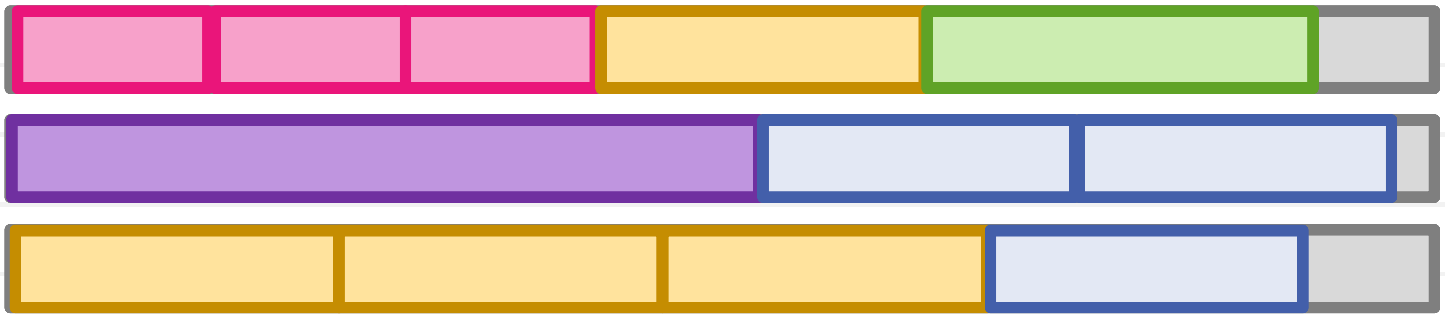
Cutting patterns



A pattern a_k is a subset of output rolls that fit into one output roll:

$$a_{i,k} \in \{0;1\} \qquad a_k^T \cdot w = \sum_{i=1}^{M} a_{i,k} \cdot w_i \leq W$$

Examples of *maximal* patterns:



New formulation

$$\underbrace{A = [a_1 \; a_2 \ldots a_T]}$$

all possible maximal patterns (one column per pattern)

$$\min_x \sum_{k=1}^{T} x_k$$

x_k whether we use a pattern

$$\text{s.t.:} \quad A \cdot x \geq 1$$

$$x \in \{0; 1\}$$

- Same idea as before, blow up the program in order to make the relaxation tighter.
- However, rather then adding new constraints, we are adding more variables

New LP formulation

$$\min_{x} \sum_{k=1}^{T} x_k$$

$$\text{s.t.:} \quad A \cdot x \geq 1 \qquad \sum_{k=1}^{T} a_{i,k} \cdot x_k \geq 1$$

$$x_k \geq 0$$

- Much tighter relaxation
- Rounding up in most cases would give good (and feasible) solution
- Branch-and-Bound works well (no symmetric minima)
- The number of variables is huge

Duality to the rescue

Reminder: the Lagrange dual of an LP problem "swaps" constraints and variables

$$\min_x \sum_{k=1}^{T} x_k \quad \text{s.t. } Ax \geq \mathbb{1} \quad (y), \quad x_k \geq 0 \quad (z)$$

$$L(x,y,z) = \sum x_k - y^T A x + \sum_{j=1}^{M} y_j - z^T x = \left(\mathbb{1}^T - y^T A - z^T\right) x + \sum_{j=1}^{M} y_j$$

$$\max_{y,z} \sum_{j=1}^{M} y_j \quad \text{s.t. } A^T y + z = \mathbb{1}, \quad y \geq 0, \quad z \geq 0$$

$$\max_y \sum_{j=1}^{M} y_j \quad A^T y \leq \mathbb{1}, \quad y \geq 0$$

Solving the dual problem

$$\max_{y} \sum_{j=1}^{M} y_j$$
$$A^T y \le 1$$
$$y \ge 0$$

- The dual has M variables and a huge number of constraints
- Use delayed constraint generation to solve it :

1. Start with a subset of constraints
2. Solve the program
3. Find the most violated inactive constraint
4. Activate it

How?

Finding the most violated constraint

$\hat{y}_1 \ldots \hat{y}_M$ - current dual solution

$$A^T = \begin{bmatrix} 0\,1\,1\,0 \ldots 1\,0 \\ \ldots \\ \\ \end{bmatrix} \cdot \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \vdots \\ \hat{y}_M \end{bmatrix}$$

a cutting pattern

Task: find a cutting pattern with the largest sum of dual variables

constraint violation

$$\max_{\alpha} \sum_{j=1}^{M} \alpha_j \hat{y}_j$$

$$\text{s.t.:} \quad \sum \alpha_j w_j \le W$$

$$\alpha_j \in \{0; 1\}$$

α must define a cutting pattern

Overview: column generation

$$\min_x \sum_{k=1}^{T} x_k$$
$$Ax \geq 1$$
$$x \geq 0$$

primal

$$\max_y \sum_{j=1}^{M} y_j$$
$$A^T y \leq 1$$
$$y \geq 0$$

dual

Identifying active patterns from the dual solution:

$$z = \mathbb{1} - A^T y$$
$$z_k \cdot x_k = 0$$
$$a_k \cdot y < 1 \Rightarrow x_k = 0$$

- As we run delayed constraint generation in the dual, we do *column generation* in the primal
- Once, we are done with the dual, we know the non-zero variables in the primal (KKT!) and can solve it (e.g. by solving linear equations)

Branch-and-Price

- For any pattern, we determine a *price*, that is a total value of dual variables corresponding to rolls participating in a pattern
- The process of choosing the pattern with the highest price is called *pricing*
- Column generation can be combined with branch-and-bound (*branch-and-price*) to achieve integral solutions
- In branch-and-price, the generated variables are propagated down the branches

Practical implementation

- Conceptually, it is easier (arguably) to understand the problem in the dual form
- Algorithmically, solving the primal problem via the simplex algorithm is better:
  - Run the simplex algorithm with a subset of variables
  - Identify the dual variables at the optimal vertex
  - Run knapsack and add a new variable (no need to restart from scratch!)

Simplex-a: changing basis

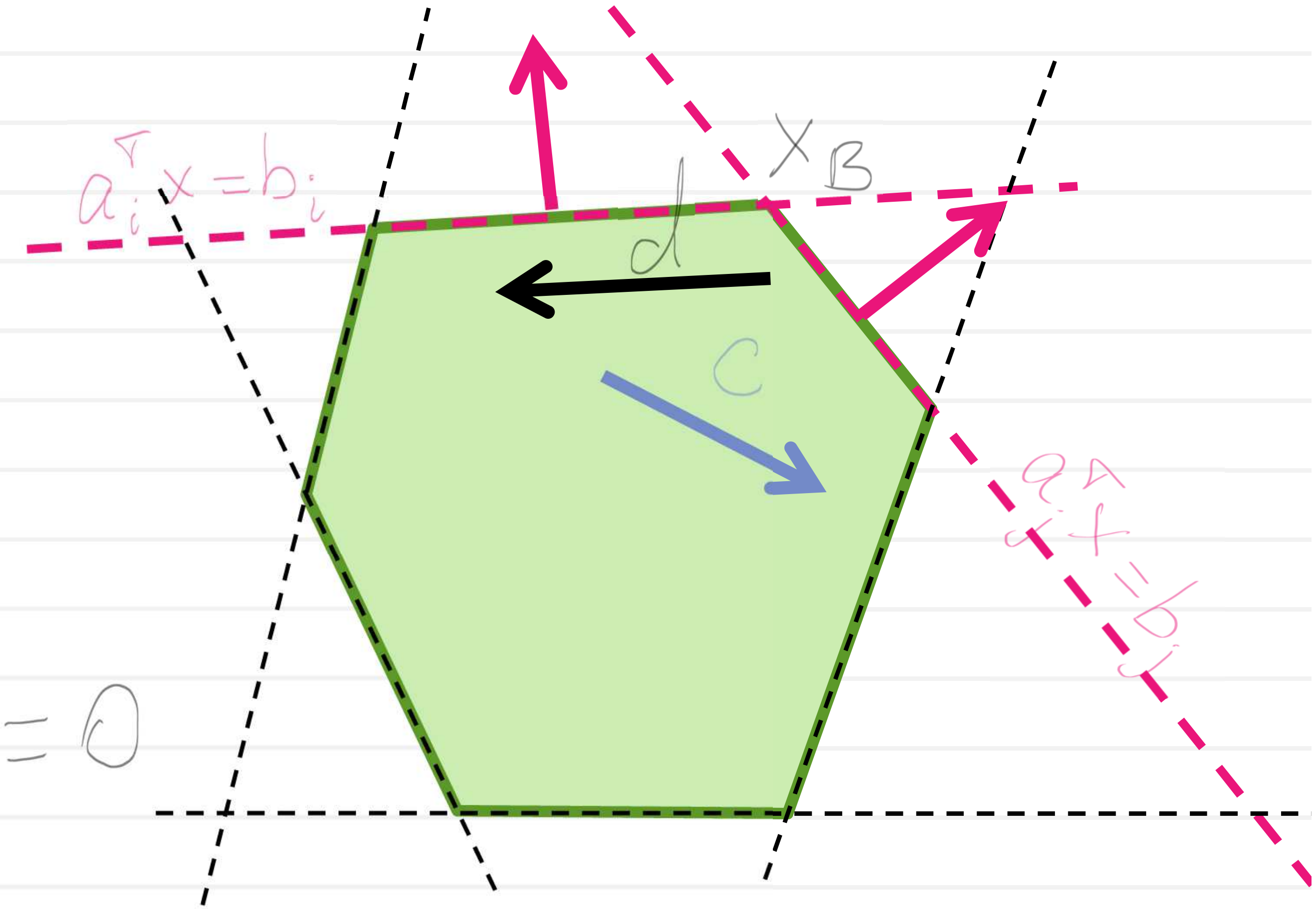
$$\lambda_B^T A_B = c^T$$

$$\lambda_B^T = c^T A_B^{-1}$$

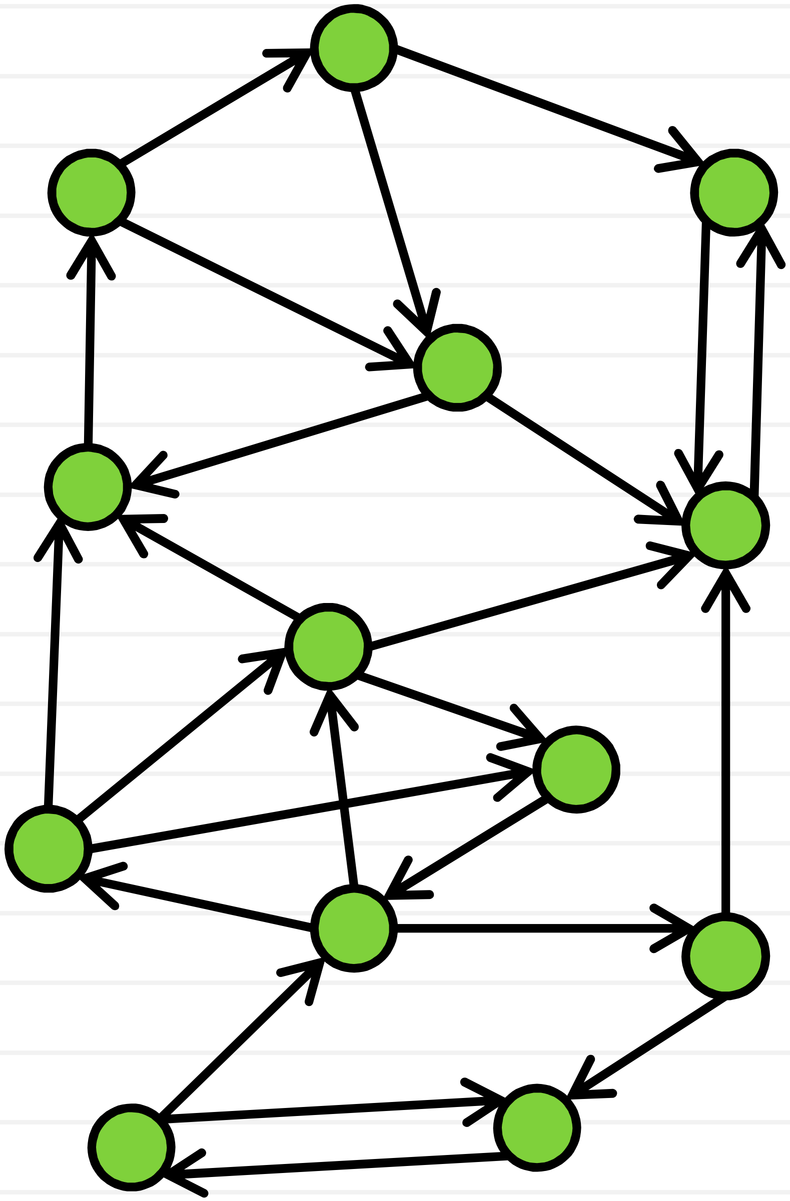
$$k : \lambda_B^k > 0$$

$$\begin{cases} A_{B \setminus \{B_k\}} \cdot d = 0 \\ a_{B_k}^T \cdot d = -1 \end{cases}$$

$$x_B \Longrightarrow x_B + \mu \cdot d$$



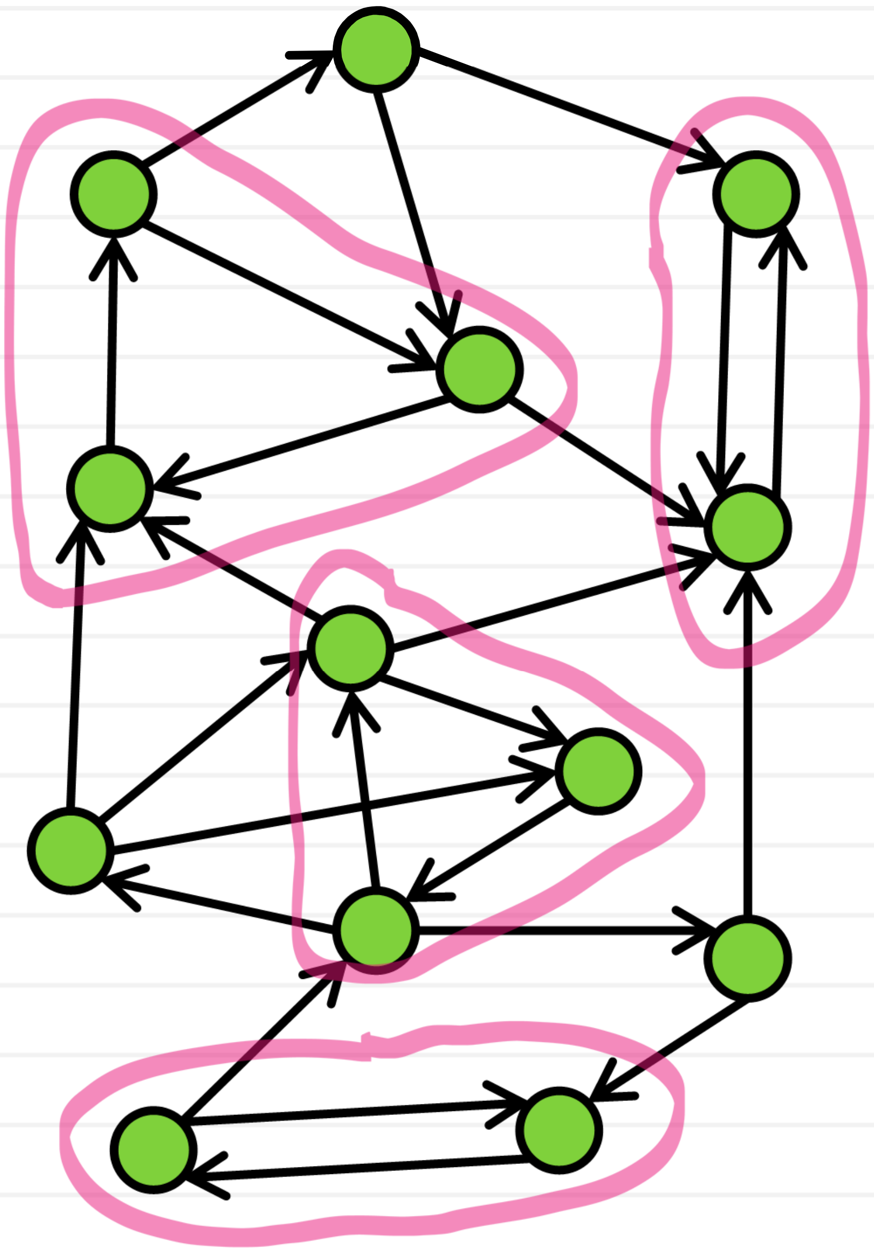
The kidney exchange graph



In the US alone there are thousands of people needing kidney transplant, each having an “incompatible” person willing to donate a kidney (e.g. the spouse)

We can arrange such patient-donor couples into a directed graph, where vertices are couples and arcs indicate biological compatibilities (the donor in the tail vertex can donate to the partient in the head vertex)

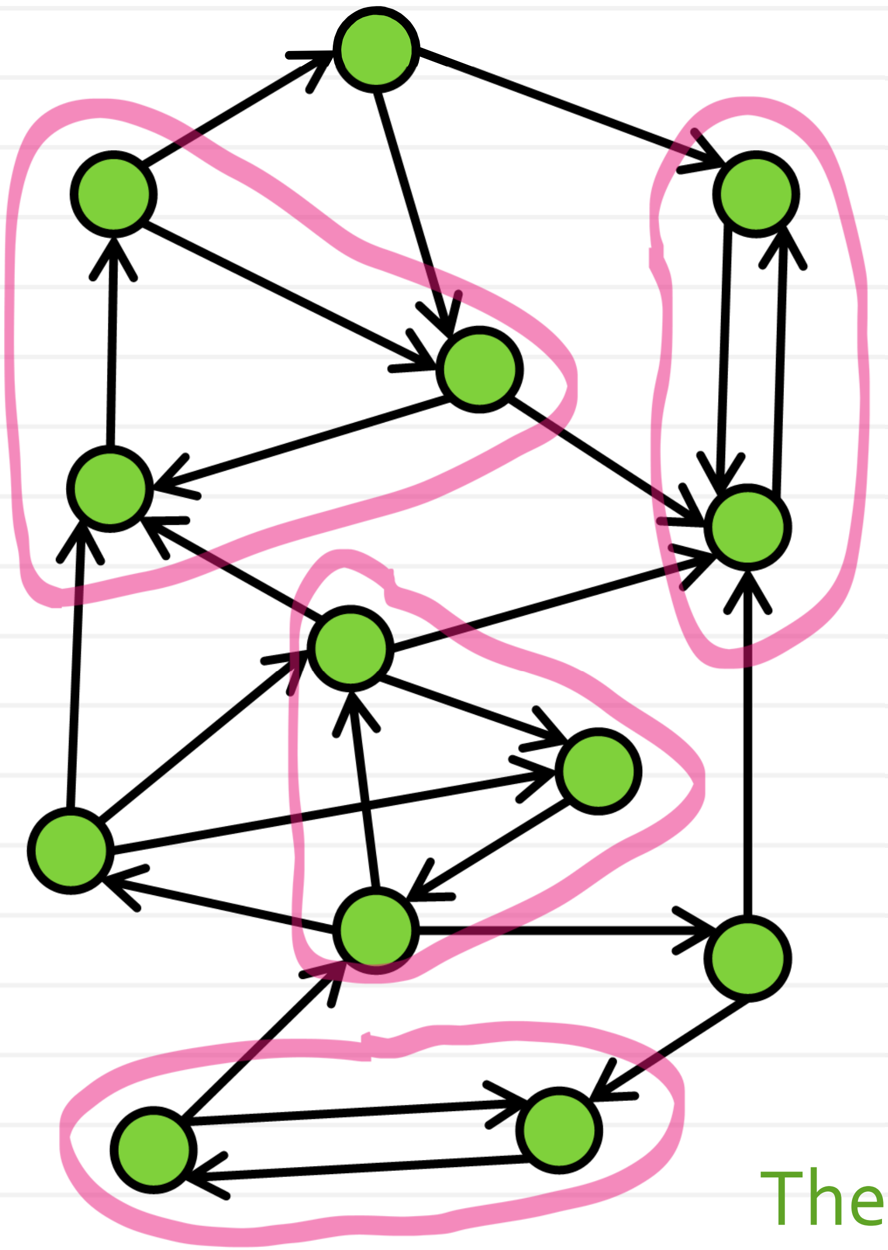
The kidney exchange graph



- Each cycle allows transplantation for all patients in the cycle
- For logistical reasons, only short cycles (2-3) are considered

Abraham, Blum, Sandholm. **Clearing Algorithms for Barter Exchange Markets: Enabling Nationwide Kidney Exchanges,** 2007

Kidney exchange via Set Packing



Essentially, a weighted set packing problem (the weight for a 2-cycle is 2, the weight for a 3-cycle is 3-ε)

$$\max \sum_{i=1}^{M} \omega_i c_i$$

$$\text{s.t.: } Ac \leq 1$$

$a_{j,i}=1$

The cycle *i* contains vertex *j*

$\cancel{c \in \{0;1\}}$

$c \geq 0$

The primal and the dual

For a graph with 5000 vertices there are 400 million of 2- and 3-cycles (cannot plug into the LP solver!)

$$L(c,y,z) = -\omega^T c + y^T A c - \sum y_j - z^T c$$

$$\min -\sum_{i=1}^{M} \omega_i c_i$$

s.t.: $Ac \le 1$ y

$c \geq 0$ z

$$\min \sum y_j$$

s.t.: $y^T A - z^T = \omega^T$

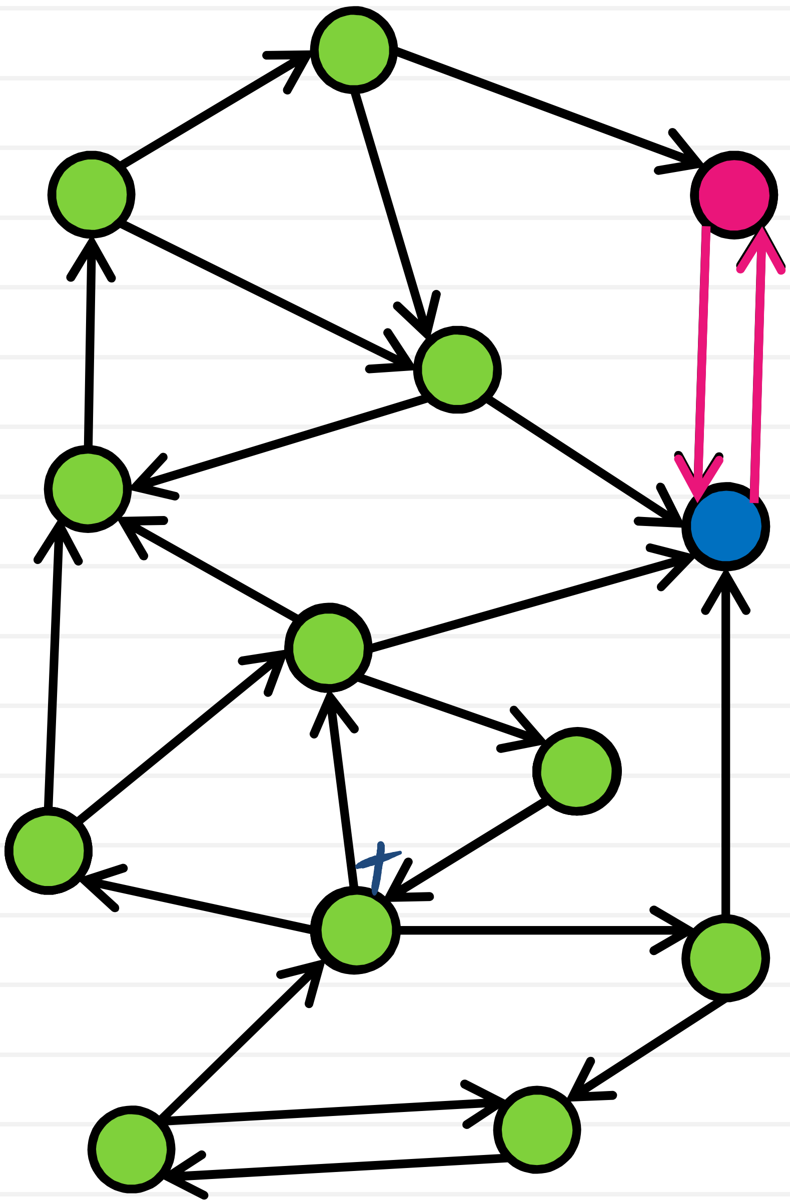
$y \geq 0$ $z \geq 0$

$$\min \sum y_j$$

s.t.: $A^T y \geq \omega$

$y \geq 0$

Column generation



$$\min \sum y_j$$
$$\text{s.t.: } A^T y \geqslant \omega$$
$$y \geqslant 0$$

Best-first search:

- Order the vertices in increasing order by their dual values
- Start from the top and look for 2-or-3 cycles
- Prune out when we hit vertices with higher dual values

Important details/output

- "Seeding" the column matrix with good cycles is important to speed up convergence
- Branch-and-Price search is not deep
- For some instances, the optimal solution is obtained way before the optimality is proven (the columns keep being generated but the objective does not change)
- By 2007, the system has been fielded for a 10,000+ "*market*"

More details: Abraham, Blum, Sandholm. **Clearing Algorithms for Barter Exchange Markets: Enabling Nationwide Kidney Exchanges,** 2007

(Mixed) Integer programming