Data-driven Decision Making

Graph problems: matching

João Pedro Pedroso

2024/2025

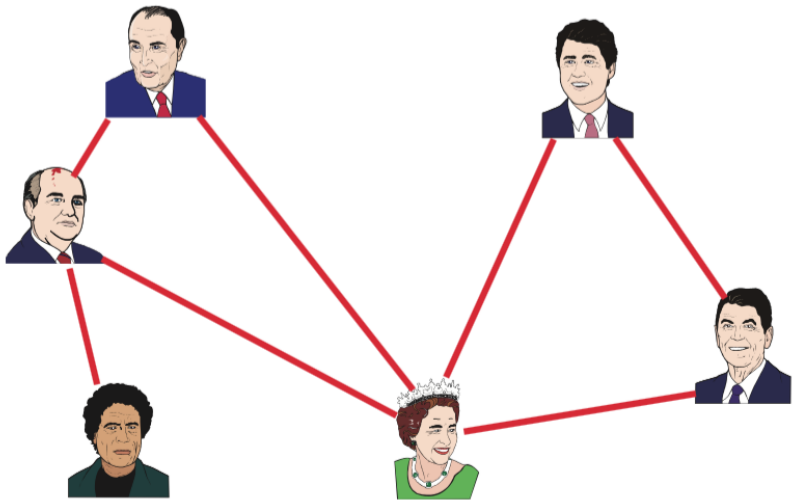
Last class

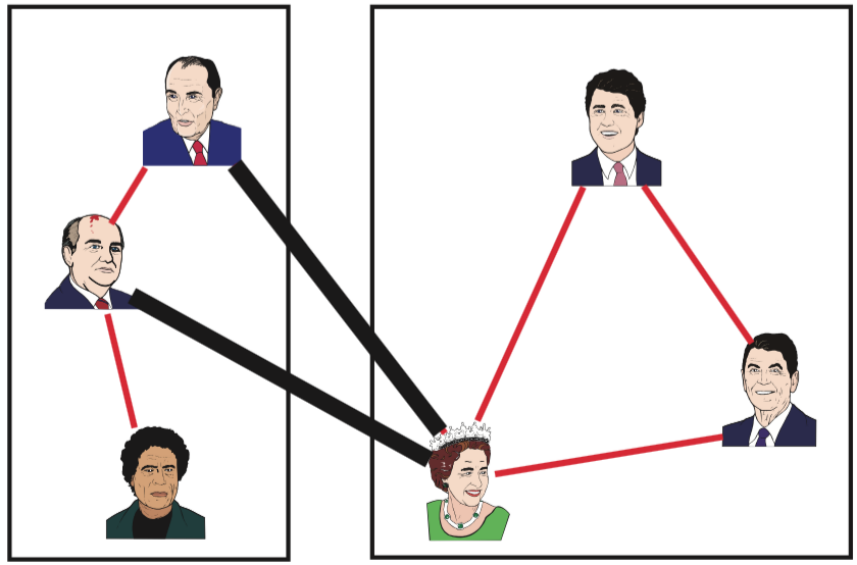
- graphs problems
- graph partitioning
- maximum stable set
- maximum clique

Today's class:

- matching problems

Graph partitioning problem





- Across edges: $\rightarrow$ thick line

Formulation: graph partitioning

- (Binary) variables y_{ij} model the case where edges are incident to different subsets:
  - $y_{ij} = 1$ if the endpoints of edge $\{i, j\}$ are across L and R
  - $y_{ij} = 0$ otherwise
- $x_i, i \in V \rightarrow$ as in previous formulation

$$\text{minimize} \quad \sum_{\{i,j\} \in E} y_{ij}$$

$$\text{subject to} \quad \sum_{i \in V} x_i = n/2$$

$$x_i - x_j \leq y_{ij} \qquad \forall \{i, j\} \in E$$

$$x_j - x_i \leq y_{ij} \qquad \forall \{i, j\} \in E$$

$$x_i \in \{0, 1\} \qquad \forall i \in V$$

$$y_{ij} \in \{0, 1\} \qquad \forall \{i, j\} \in E$$

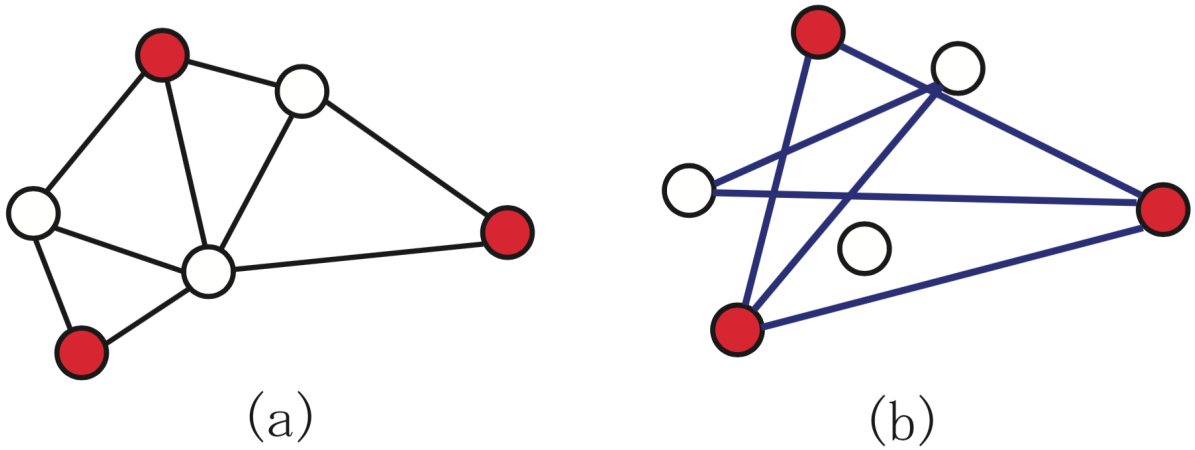
Maximum stable set and maximum clique problems

Given an undirected graph $G = (V, E)$, a subset $S \subseteq V$ is called a stable set when there isn't any edge among vertices of S. The problem is to find a stable set S such that its cardinality is maximum.

Given an undirected graph $G = (V, E)$, a subset $C \subseteq V$ is called a clique when the subgraph induced by C is complete. The problem is to find a clique C which maximizes cardinality $|C|$.

Maximum stable set and Maximum clique

- Maximum clique problem: equivalent to maximum stable set:
  - can be converted through a simple transformation, and
  - solution is the same



- Red vertices are both:
  - Maximum stable set on the original graph
  - Maximum clique on the complementary graph

Maximum stable set: formulation in integer optimization

- (Binary) Variables: for each vertex $i \in V$:
  - $x_i = 1$ if i is included in the stable set
  - $x_i = 0$ otherwise
- Model:

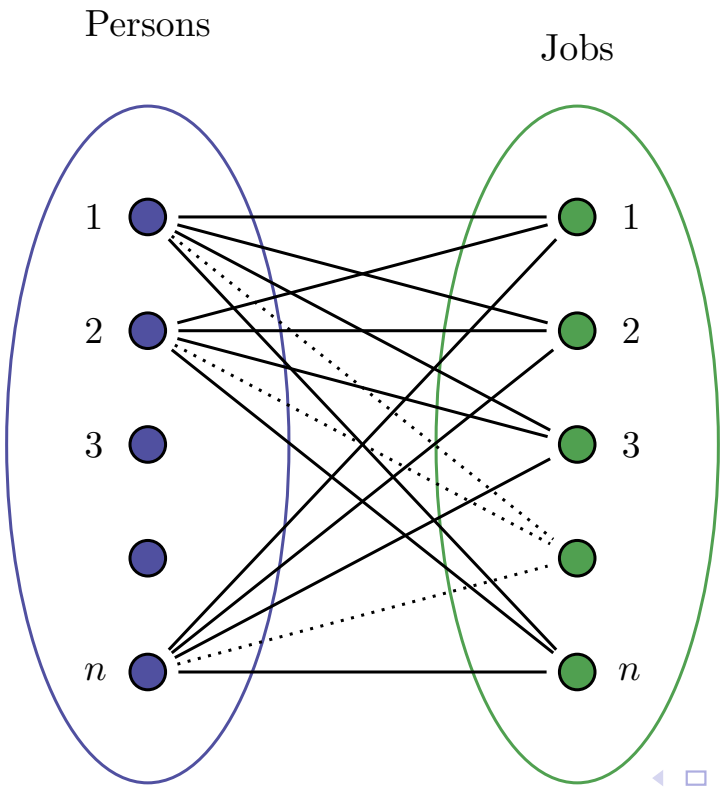
$$\begin{aligned} \text{maximize} \quad & \sum_{i \in V} x_i & \\ \text{subject to} \quad & x_i + x_j \leq 1 & \forall \{i,j\} \in E \\ & x_i \in \{0,1\} & \forall i \in V \end{aligned}$$

Today's class

Preliminaries: Assignment problem

Assignment problem

- There are n people available to carry out n jobs
- Each person is assigned to carry out exactly one job
- Some persons are better suited to particular jobs than others
  - cost c_{ij} if person i is assigned to job j
- Problem: find a minimum cost / maximum reward assignment



Assignment problem

Formulation:

- $x_{ij} = 1$ if person i does job j, 0 otherwise
- all persons are assigned to a job
- all jobs are assigned to a person

$$\text{minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

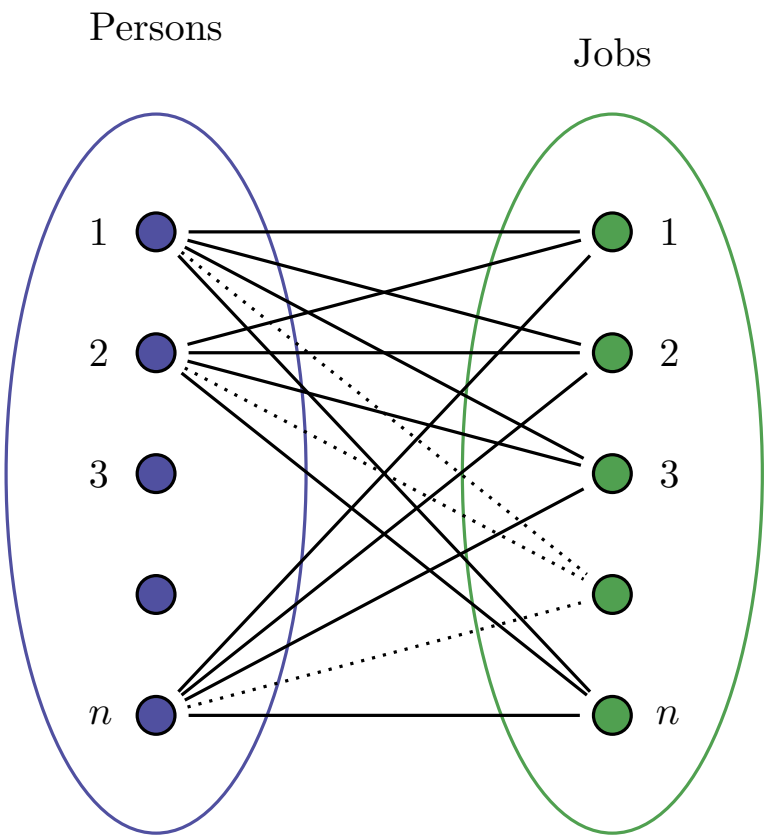
$$\text{subject to} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, \ldots, n$$

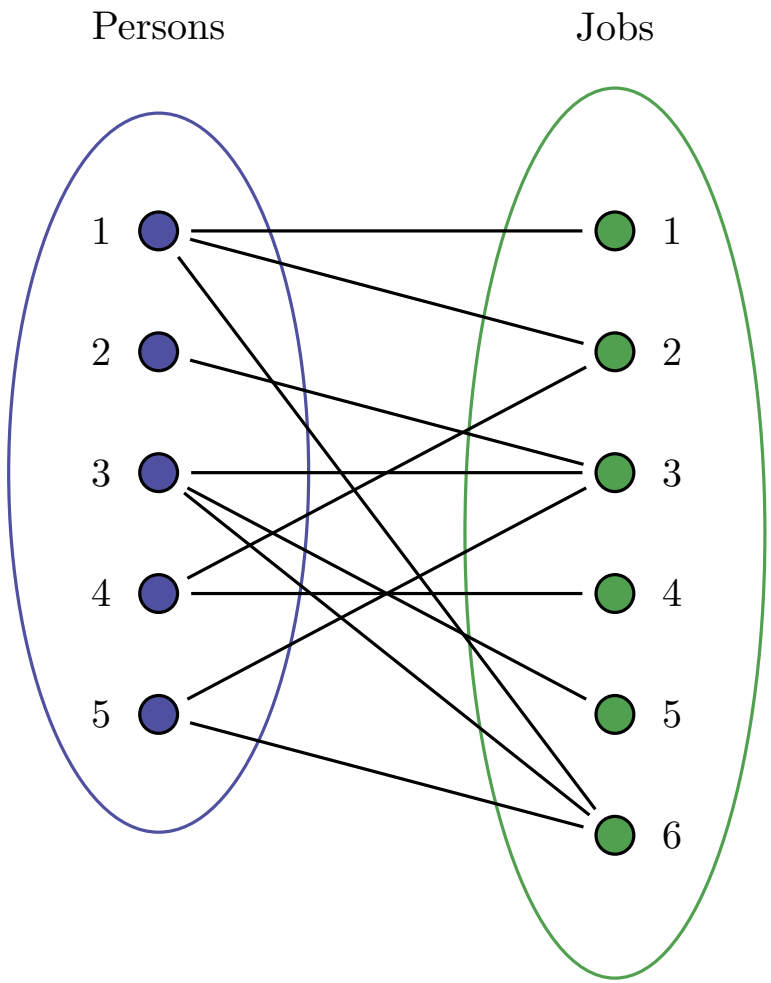
$$x_{ij} \in \{0, 1\} \qquad i = 1, \ldots, n, j = 1, \ldots, n$$

Bipartite graphs

Complete

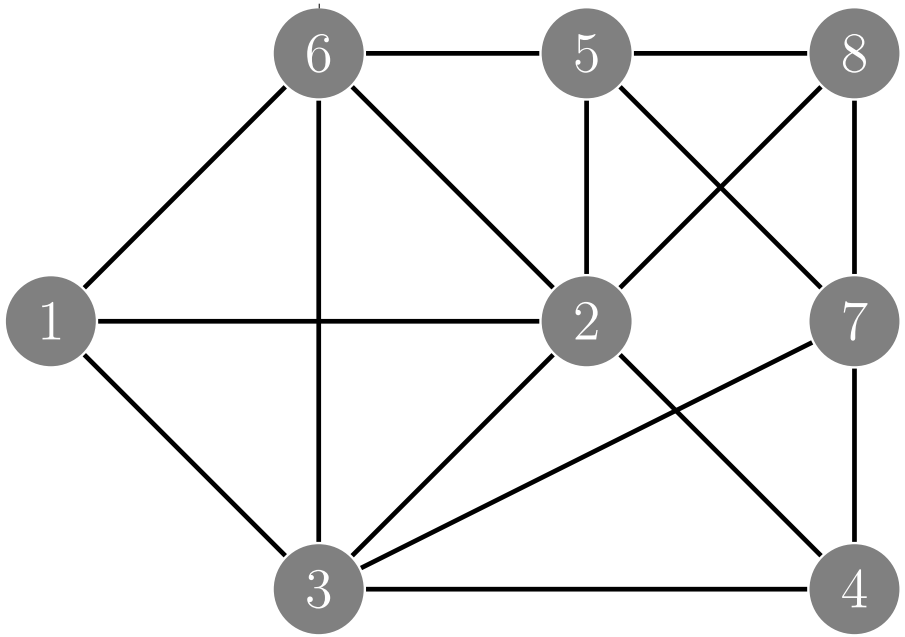


Arbitrary



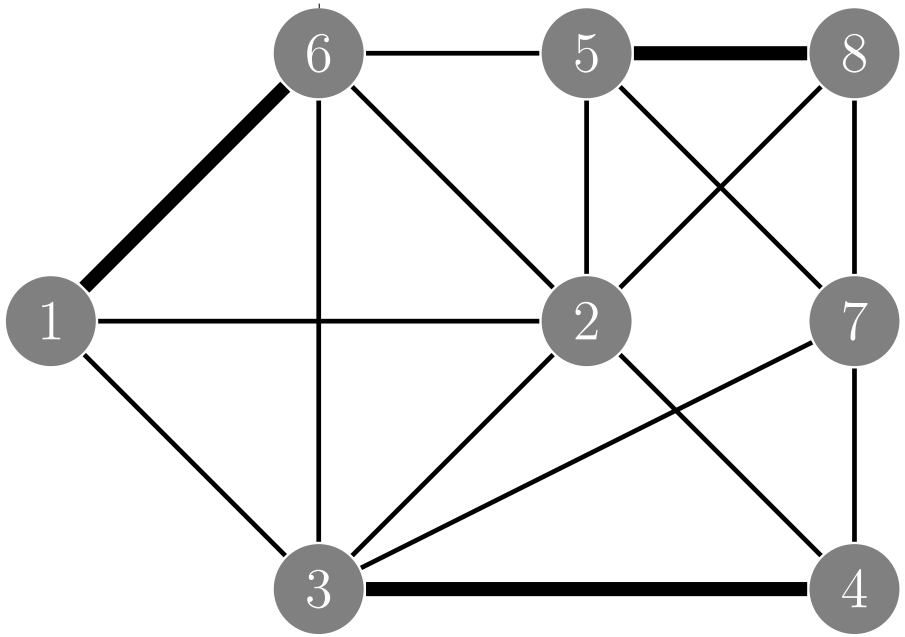
Matching

- Given a graph $G = (V, E)$ a matching is a set of disjoint edges



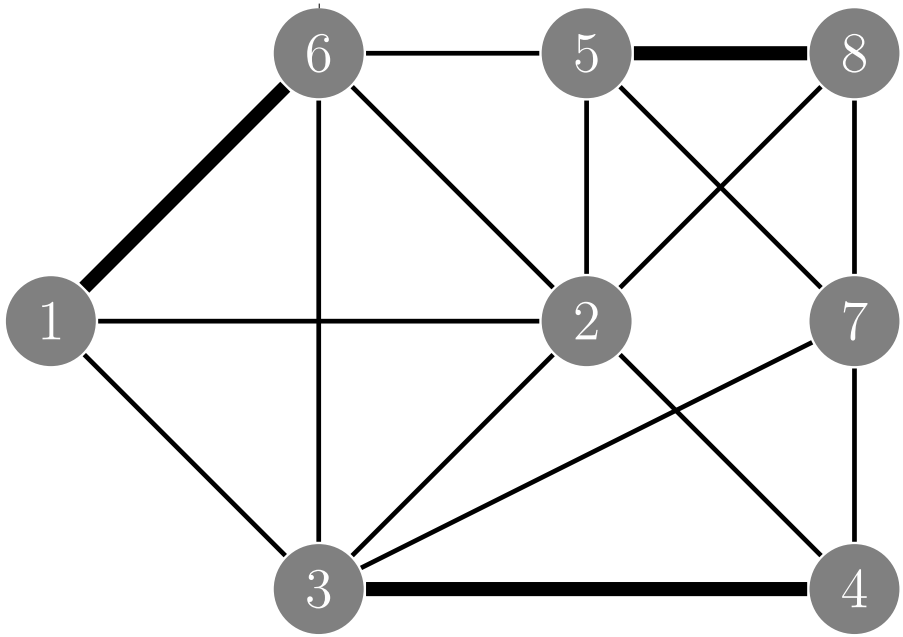
Matching

- Given a graph $G = (V, E)$ a matching is a set of disjoint edges



Matching

- Given a graph $G = (V, E)$ a matching is a set of disjoint edges



- Maximum cardinality matching:
  - select the maximum number of edges that are a matching
- Maximum weight matching:
  - select edges that are a matching with maximum total weight
- Perfect matching:
  - for a graph with $2k$ nodes: matching with k edges

Maximum weight matching

- Simple, polynomial algorithms for the bipartite case
- Not so simple, still polynomial for the general case

Maximum cardinality matching: all weights c_{ij} are 1

Maximum weight matching

Given:

- graph $G = (V, E)$
- weight c_{ij} for all $ij \in E$

Formulation:

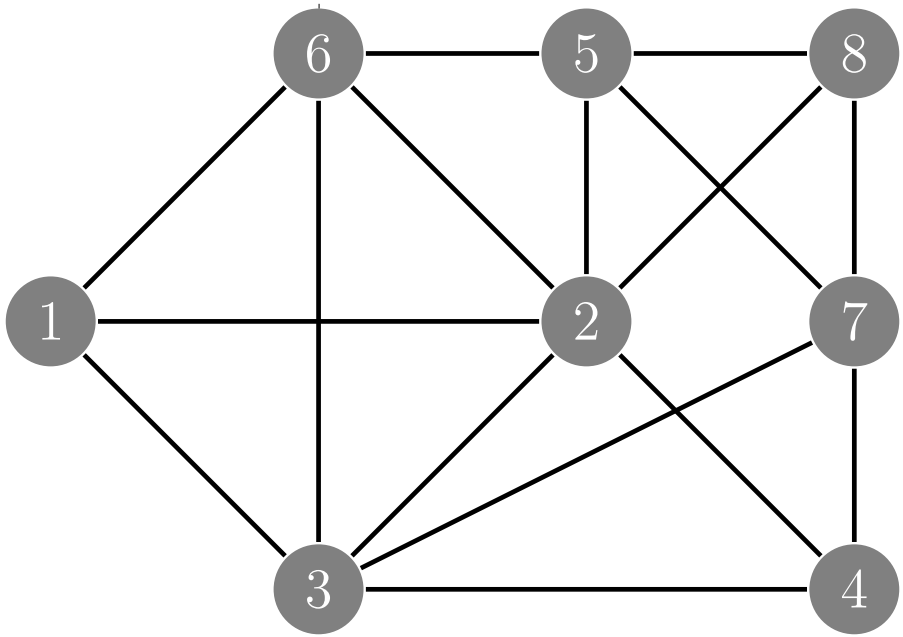
- $x_{ij} = 1$ if edge $\{i, j\}$ is in the matching, 0 otherwise $\quad \forall \{i, j\} \in E$

$$\begin{aligned} \text{maximize} \quad & \sum_{ij \in E} c_{ij} x_{ij} \\ \text{subject to} \quad & \sum_{j : ij \in E} x_{ij} \leq 1 && \forall i \in V \\ & x_{ij} \in \{0, 1\} && \forall ij \in E \end{aligned}$$

Note: I'm abusing notation, and representing an edge $\{i, j\}$ simply by ij

Example: maximum cardinality matching

Consider the graph



Find the maximum cardinality matching

Model

Recall:

$$\begin{aligned} \text{maximize} \quad & \sum_{ij \in E} c_{ij} x_{ij} & \\ \text{subject to} \quad & \sum_{j:ij \in E} x_{ij} \leq 1 & \forall i \in V \\ & x_{ij} \in \{0, 1\} & \forall ij \in E \end{aligned}$$

Model

Recall:

$$\begin{aligned} \text{maximize} \quad & \sum_{ij \in E} c_{ij} x_{ij} & \\ \text{subject to} \quad & \sum_{j:ij \in E} x_{ij} \leq 1 & \forall i \in V \\ & x_{ij} \in \{0,1\} & \forall ij \in E \end{aligned}$$

```
set V;
set E within {V,V};

var x {E} binary;

maximize z: sum {(i,j) in E} x[i,j];

subject to
Matched {i in V}:
    sum {j in V : (i,j) in E} x[i,j] +
    sum {j in V : (j,i) in E} x[j,i] <= 1;
```

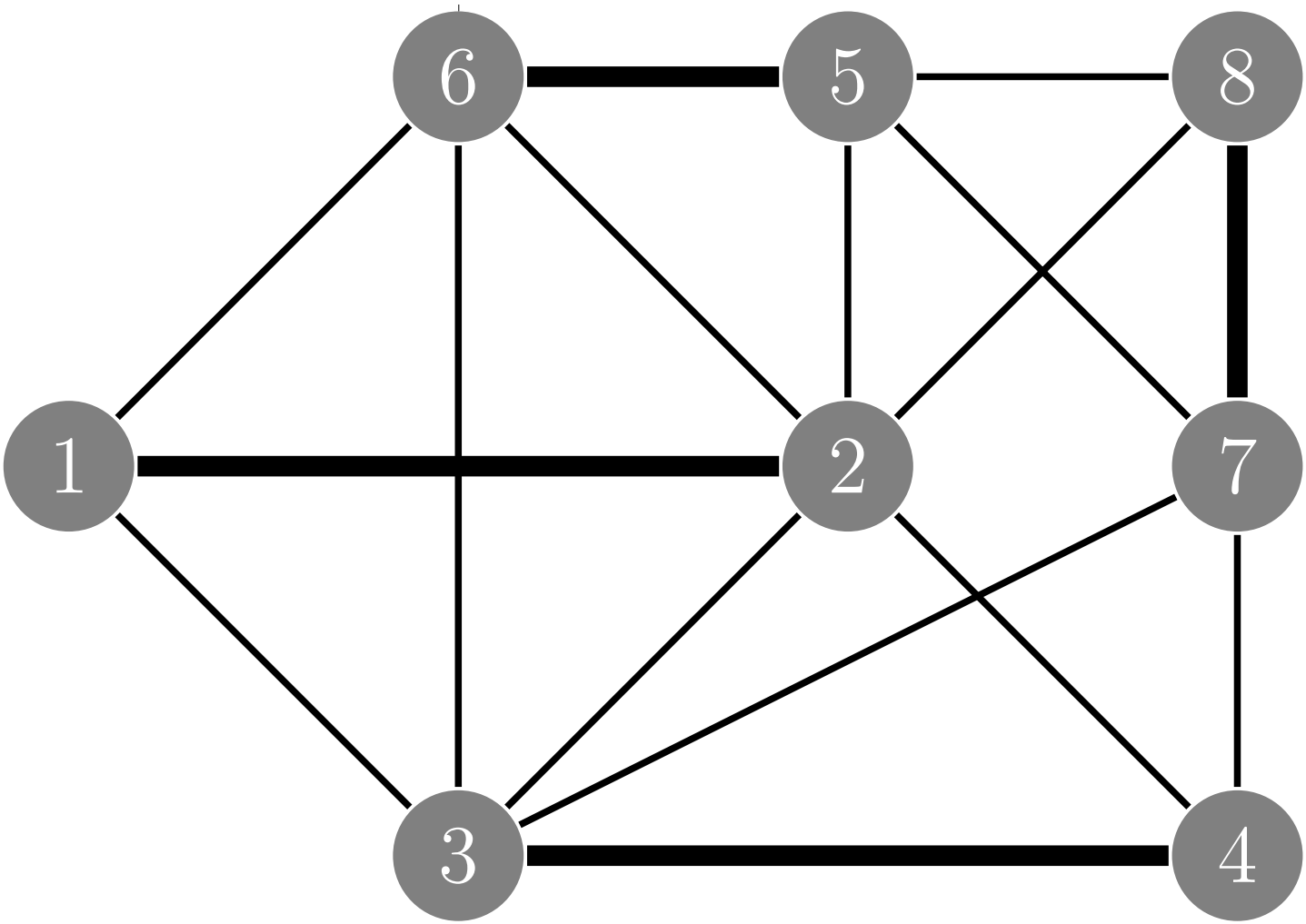
Data

```
set V;
set E within {V,V};

var x {E} binary;

maximize z: sum {(i,j) in E} x[i,j];

subject to
Matched {i in V}:
    sum {j in V : (i,j) in E} x[i,j] +
    sum {j in V : (j,i) in E} x[j,i] <= 1;
```

```
set V := 1 2 3 4 5 6 7 8;
set E :=
      1 2
      1 3
      1 6
      2 3
      2 4
      2 5
      2 6
      2 8
      3 4
      3 6
      3 7
      4 7
      5 6
      5 7
      5 8
      7 8
;
```

Solution

```
z = 4

x :=
1 2    1
1 3    0
1 6    0
2 3    0
2 4    0
2 5    0
2 6    0
2 8    0
3 4    1
3 6    0
3 7    0
4 7    0
5 6    1
5 7    0
5 8    0
7 8    1
;
```



Related notions: covering and packing

Given:

- graph $G = (V, E)$

we can define:

- covering by nodes: a set $R \subseteq V$ such that every edge $e \in E$ has at least one endpoint in R
- packing by edges: a set $F \subseteq E$ such that every node $i \in V$ is the endpoint of at most one $e \in F$
  - i.e., *a matching*
  - but we may want to pack with more general structures
    $\rightarrow$ kidney exchange problems

Kidney exchange programs

Kidney exchange programs

- Motivation
- Models

Information systems for health care

- Expected to be one of the areas where more resources will be applied in the next few years
- Has issues involving the many disciplines, including operations research, computer science, informatics, ...
- Information systems have a huge impact in terms of
  - economy
  - social benefits
  - work rationalization
  - reliability

Kidney Failure Treatments

- Kidney failure
  - One kidney $\longrightarrow$ OK
  - Both kidneys $\longrightarrow$ Dialysis or Transplantation
- Dialysis vs Transplantation
  - Transplantation yields longer survivability
  - Transplantation yields a better quality of life
  - Dialysis is more expensive than transplantation; values for Portugal:
    - Hemodialysis $\longrightarrow$ 30K euro per year per person
    - Transplantation: 30K euro once + 10K euro year

Kidney Failure Treatments

I don't care what day it is.
Four hours is four hours.

- Objective: $\longrightarrow$ carry out the *maximum* possible number of (successful) transplants

Sources of kidneys for transplantation

- Deceased donors
  - very large waiting lists (5 years or more waiting)
- Living donors:
  - relatives, spouse, friends, altruistic donors
  - many ethical and legal issues (varies with country)
    - e.g. no commercial transaction of kidneys is generally accepted

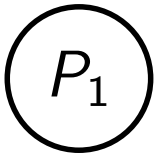
Sources of incompatibility

- Blood type compatibilities

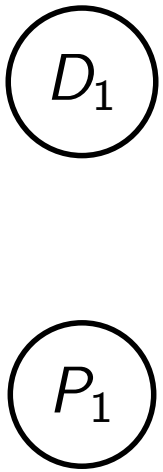
Donor	Recipient			
	O	A	B	AB
O	✓	✓	✓	✓
A	✗	✓	✗	✓
B	✗	✗	✓	✓
AB	✗	✗	✗	✓

- Tissue type incompatibility
  - HLA (Human Leukocyte Antigens)
  - ...

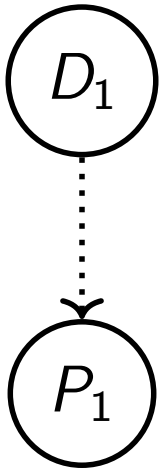
Introduction: background



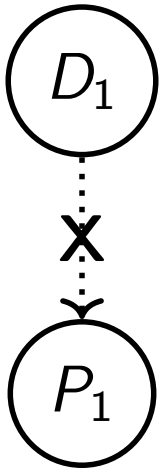
Introduction: background



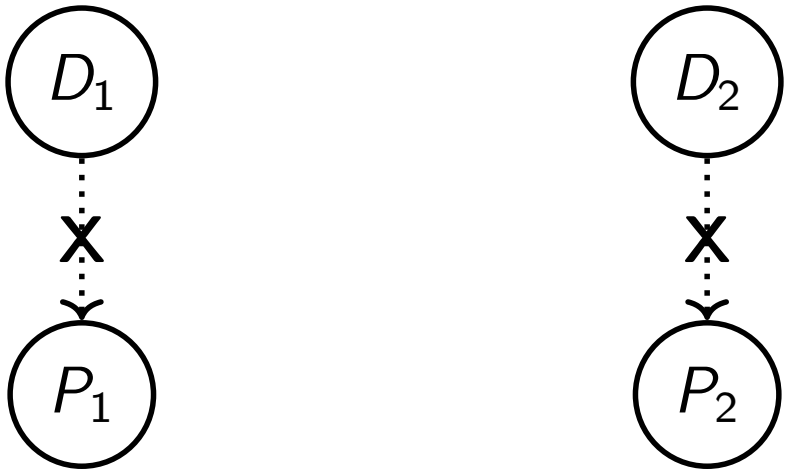
Introduction: background



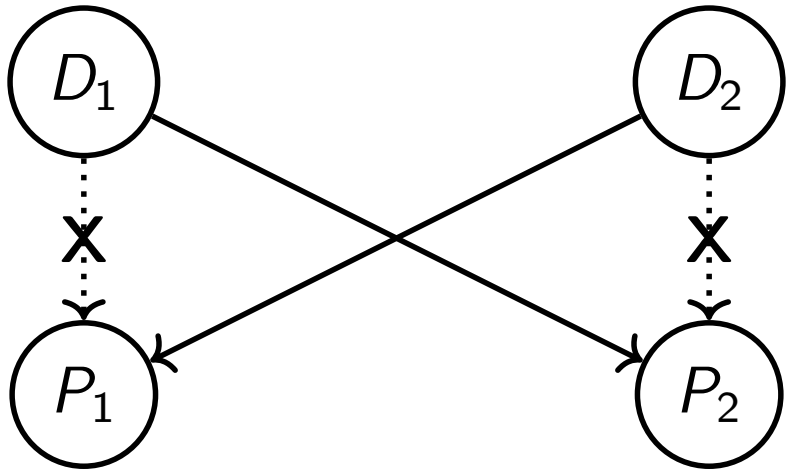
Introduction: background



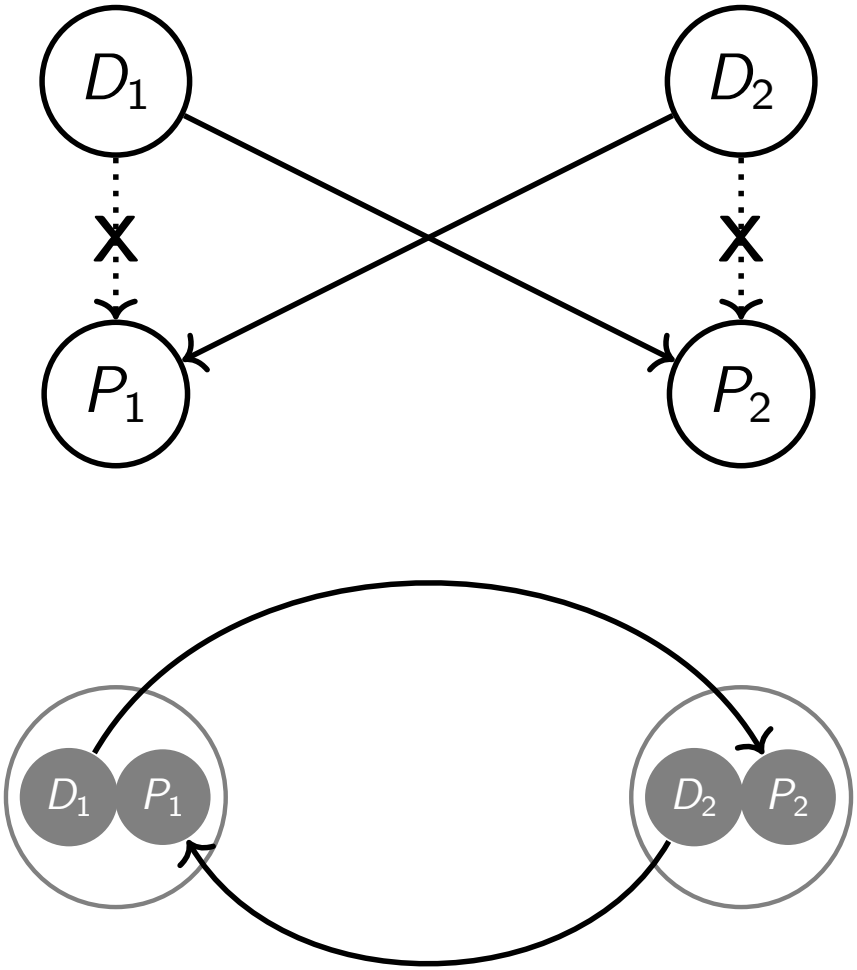
Introduction: background



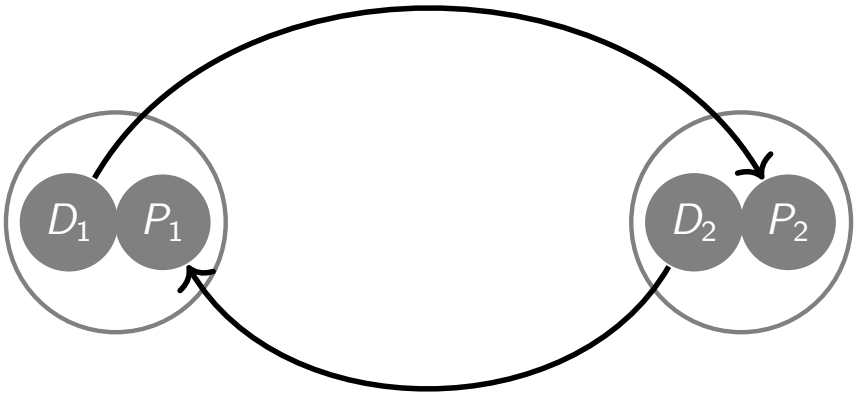
Introduction: background

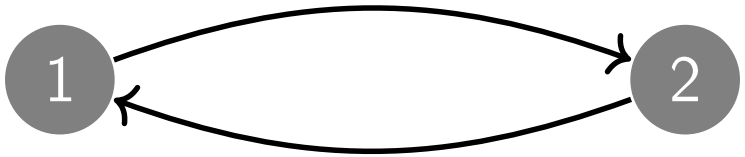


Introduction: background

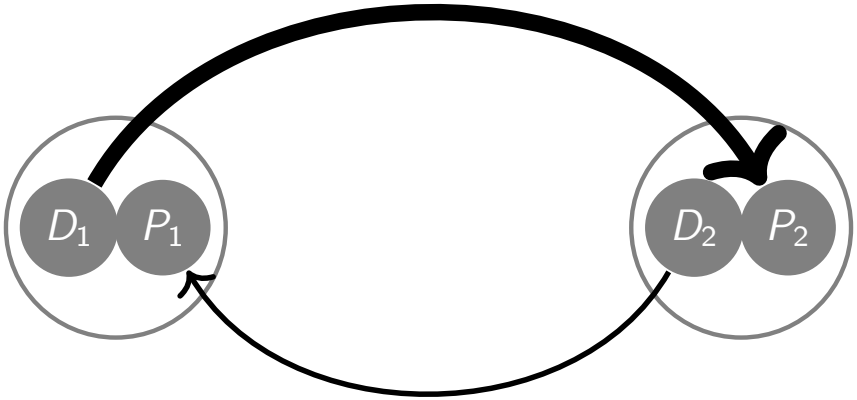


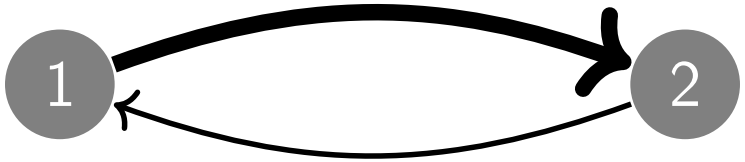
Introduction: background



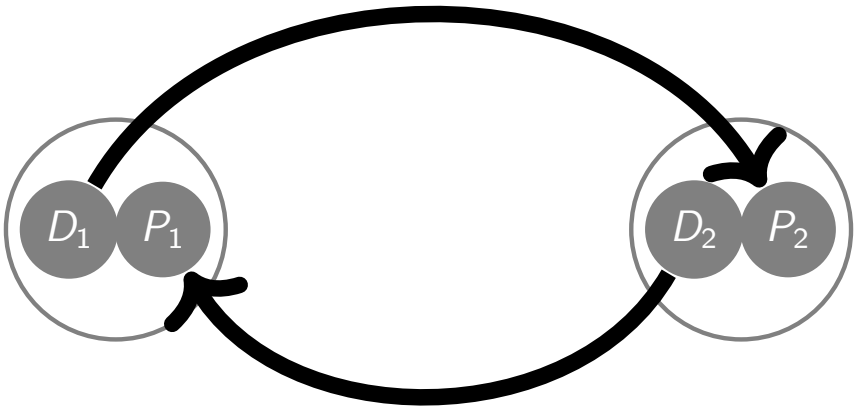


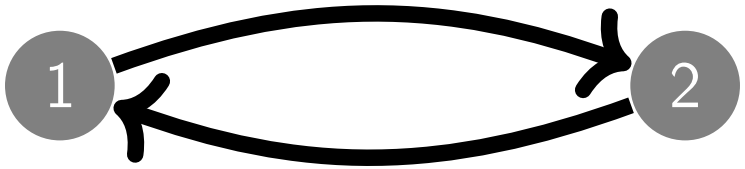
Introduction: background



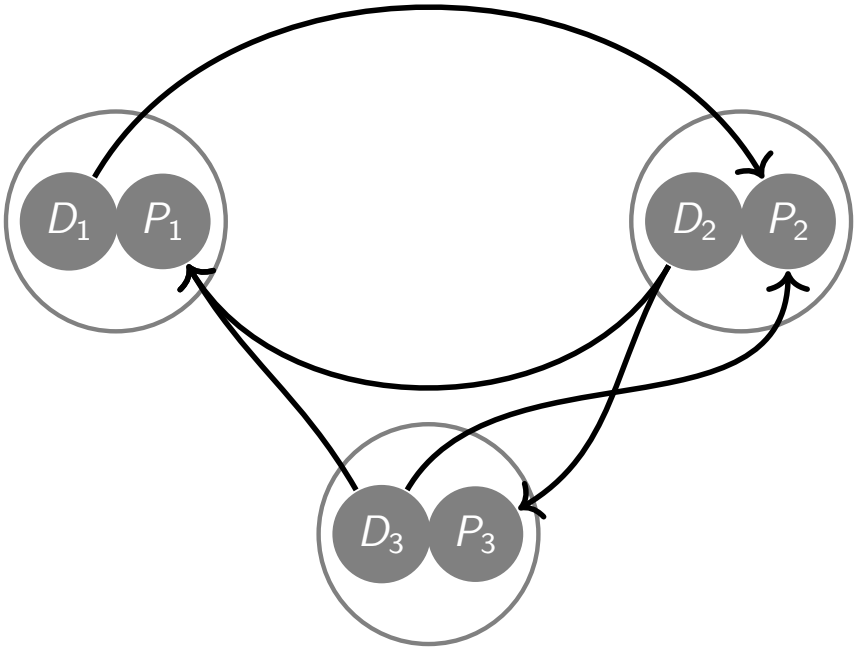


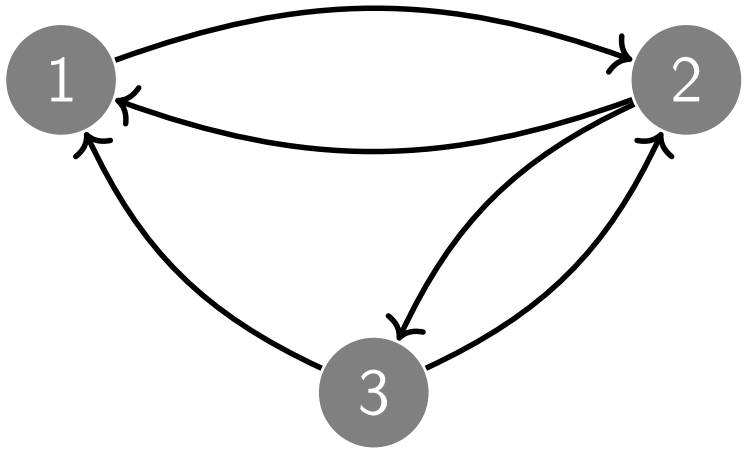
Introduction: background



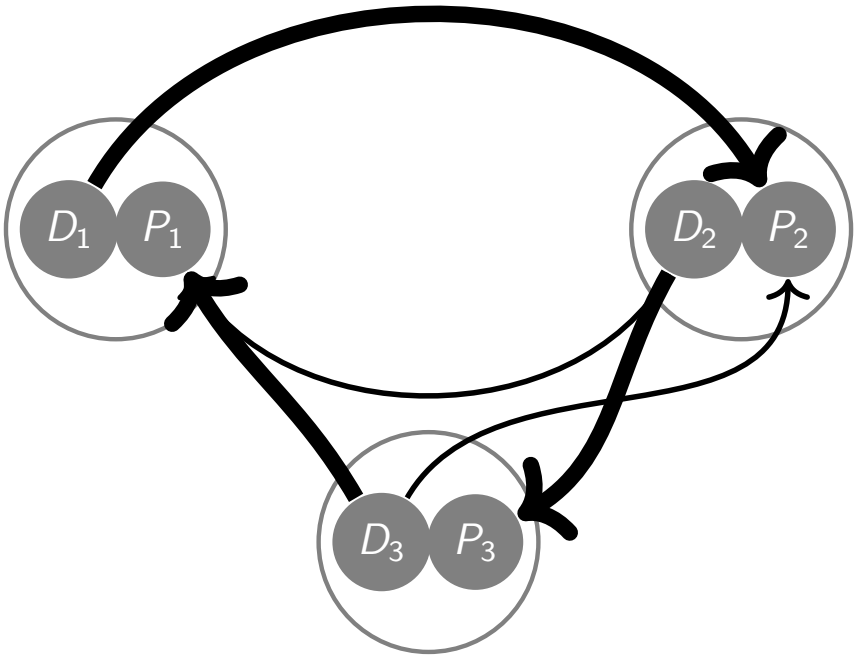


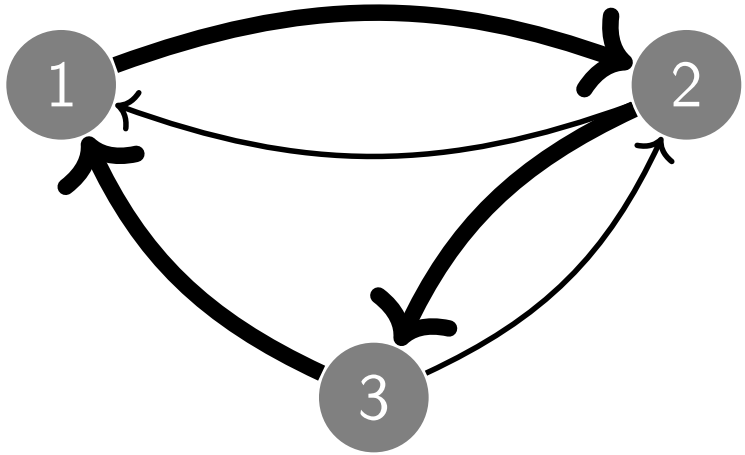
Introduction: background



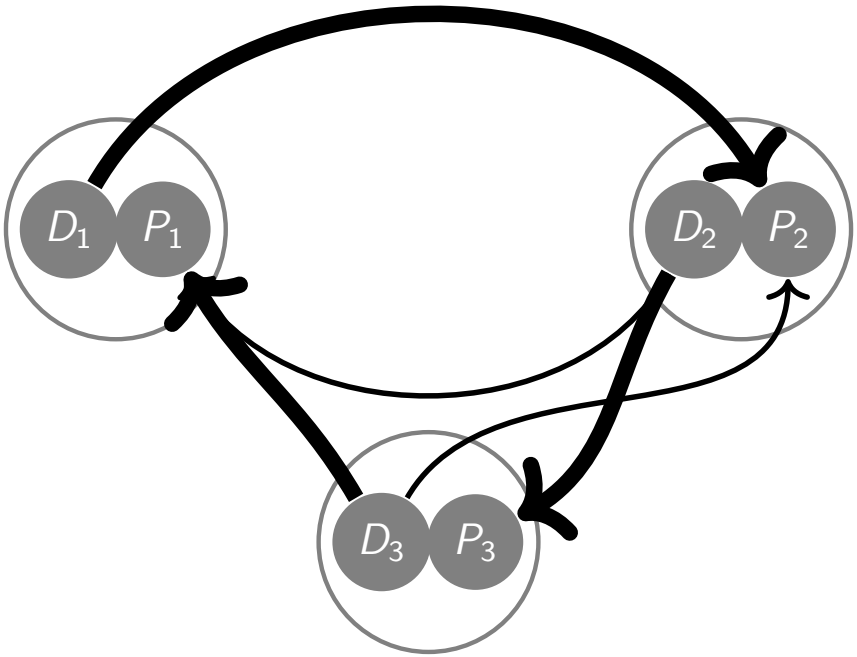


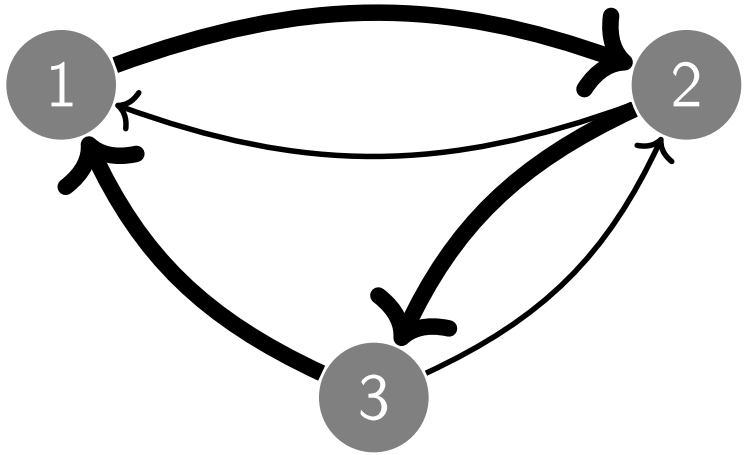
Introduction: background





Introduction: background





- Exchange between incompatible pairs in a directed graph
  - thin arrows $\rightarrow$ preliminary assessment compatibility
  - thick arrows $\rightarrow$ matching

Background: kidney exchange programs

- in many countries, recent legislation allows patients needing a kidney transplant to receive it from a living donor
- what to do when the transplant from that donor is not possible?
  - blood type
  - other incompatibilities
- patient-donor pair may enter a kidney exchange program (KEP)

Kidney exchange programs

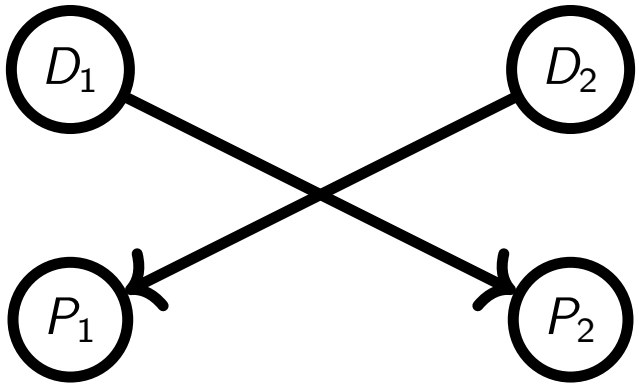
- KEPs were first proposed by (Rapaport, 1986)
- First transplants within a KEP were done in South Korea, 1991
- Many countries have now KEPs (USA, Switzerland, Turkey, Romania, Netherlands, UK, Canada, Australia, New Zealand, Spain)
- A KEP started in Portugal in 2011; presently, transplants are routinely performed

Kidney exchange programs

- Usually involves a large number or pairs
- Maximizing the number of transplants
  - problem know to be computationally difficult
  - not an issue in practice, using best-known formulations

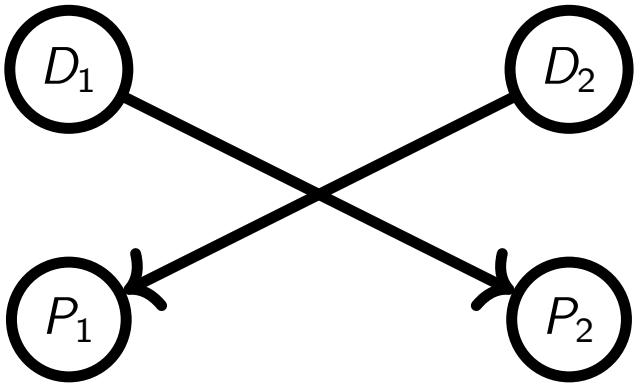
Kidney exchanges

- Suppose there are two patient-donor pairs (D_1, P_1) and $(D_2, P2)$
- Donor D_1 is willing to give kidney to patient P_1 but they are incompatible
- The same for pair D_2, P_2
- D_1 is compatible with P_2 and D_2 is compatible with P_1
- Then, D_1 can give a kidney to P_2 and D_2 can give a kidney to P_1



Kidney 2-exchanges

- allow two patients in incompatible pairs to exchange their donors
- each patient receives a compatible kidney from the donor of the other pair



Incompatible pairs $P_1 - D_1$ and $P_2 - D_2$ exchange donors

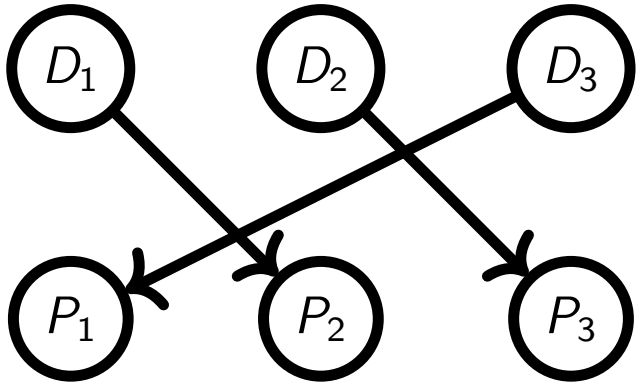
- P_1 receives a transplant from D_2 and vice versa

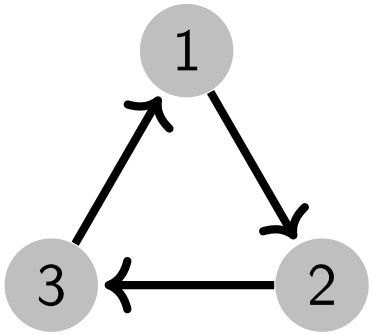
Graph representation:

- vertices are patient-donor pairs
- arcs link a donor to compatible patients

Kidney 3-exchanges

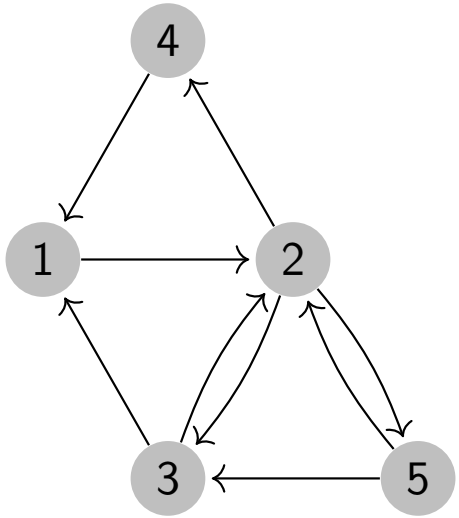
- The idea can be easily extended to 3 or more pairs:





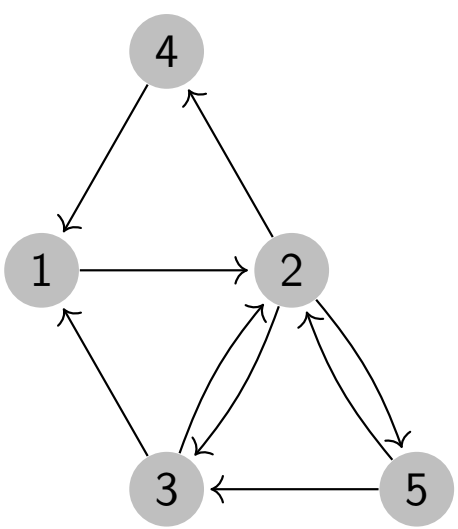
- Representation with a directed exchange graph:
  - each incompatible pair (D_i, P_i) corresponds to a node i
  - there exists an arc between i and j if donor D_i can give a kidney to patient P_j
  - a cycle with k nodes in this graph corresponds to a k-exchange

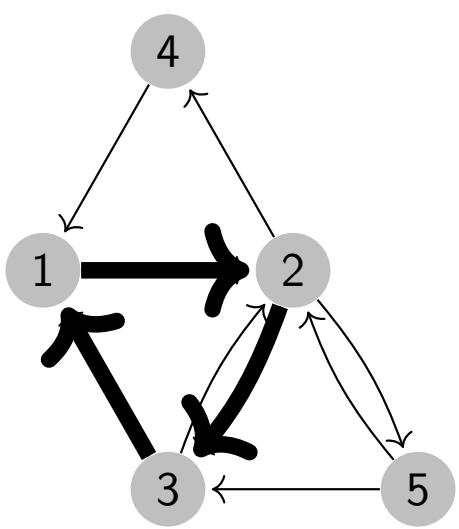
Kidney exchanges: example

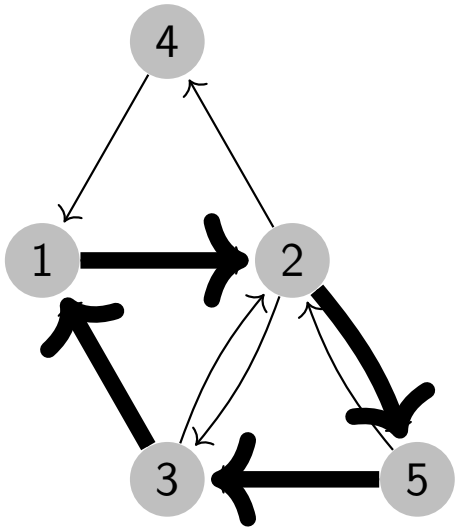


- instance with five pairs
- what is the maximum number of transplants?
- what if the allowed number of simultaneous transplants is limited?

Kidney exchanges: example



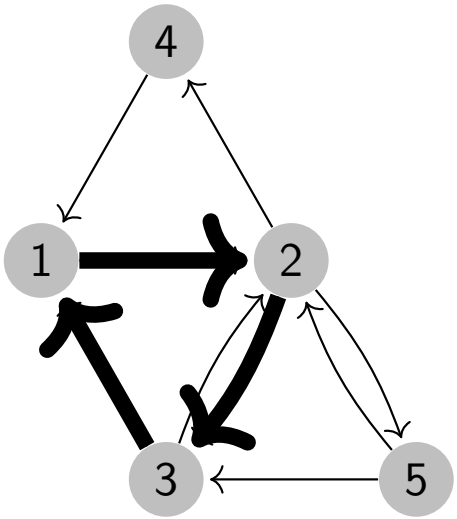


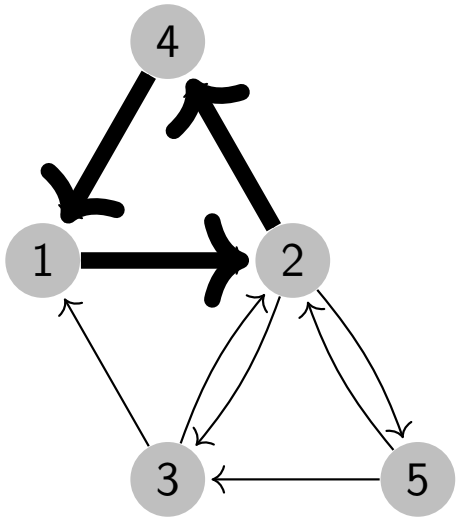


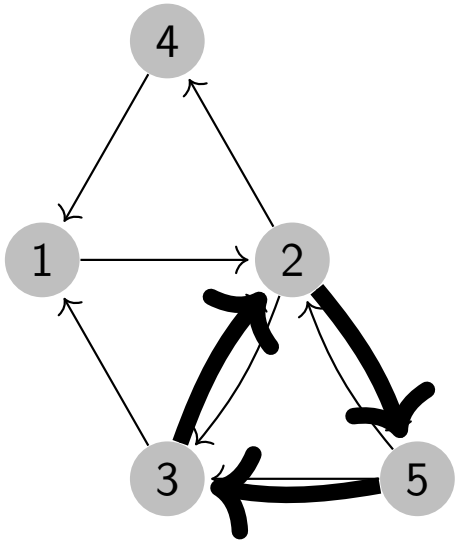
- feasible exchange: a set of vertex-disjoint cycles (*e.g.*, $1-2-3-1$)
- size of an exchange: sum of the lengths of its cycles
- maximum exchange in this example: 4 (cycle $1-2-5-3-1$)

Kidney exchanges: maximum cycle size

- In many situations the length of each cycle is limited
- If maximum cycle size is $K = 3$, several solutions are possible.





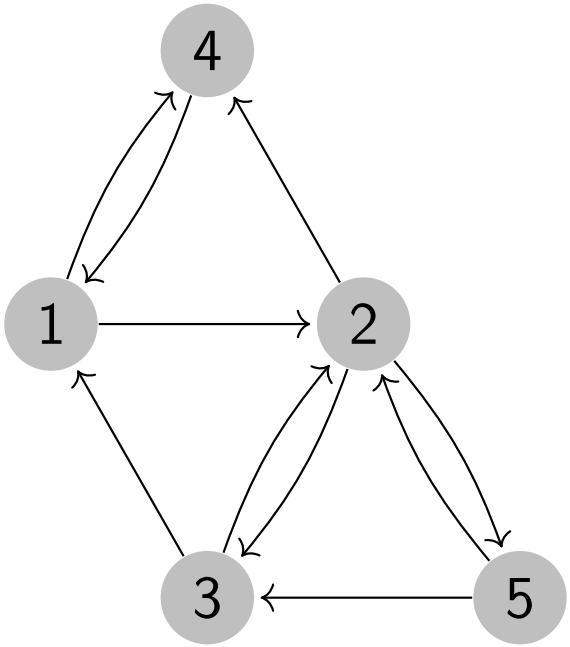


...

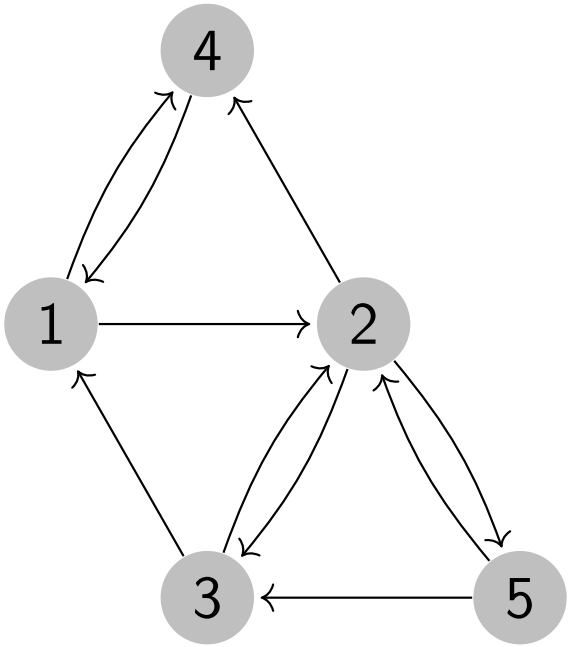
Kidney exchanges: why limiting size

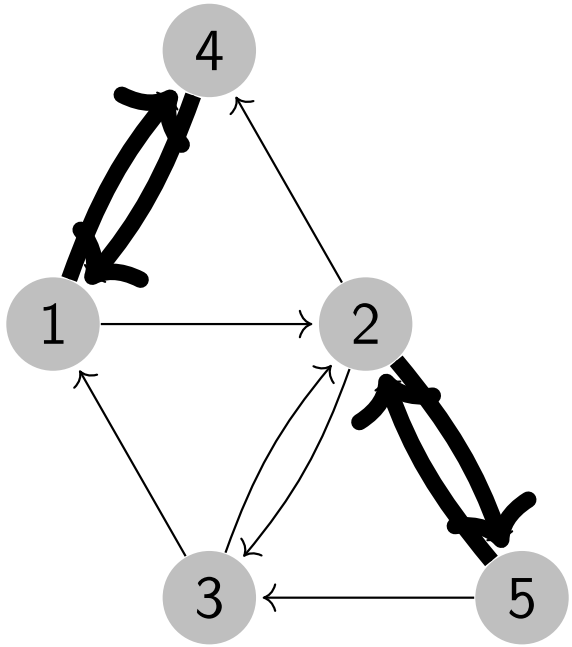
- Two main reasons:
  - usually, all transplants in a cycle should be done at same time
    - someone could withdraw from the program
  - last-minute incompatibility test (crossmatch, just before transplantation)
    - if positive, no transplantation can be done for any pair in this cycle
    - (rearrangements may change the previous limitation)
- However, optimum number of transplants increases with maximum size allowed
- Most programs have $k = 2$ or $k = 3$

Another example

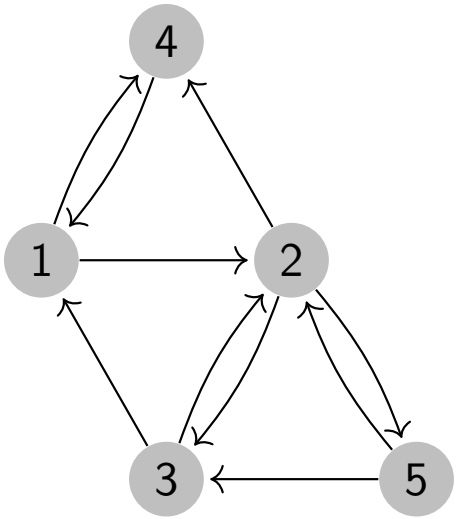


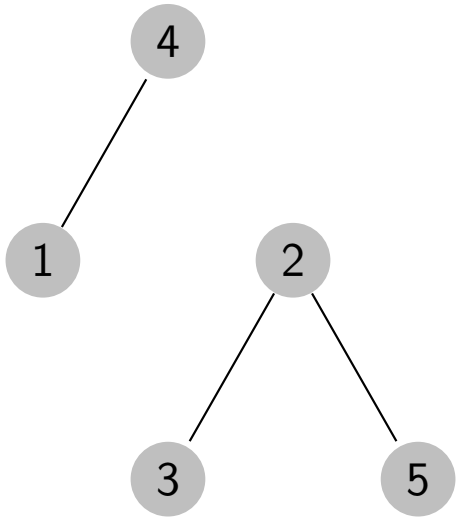
Another example

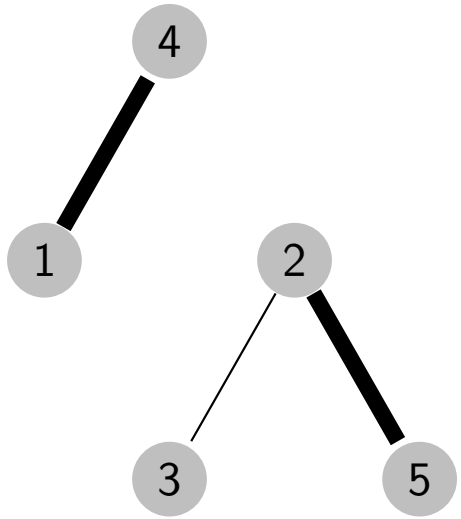




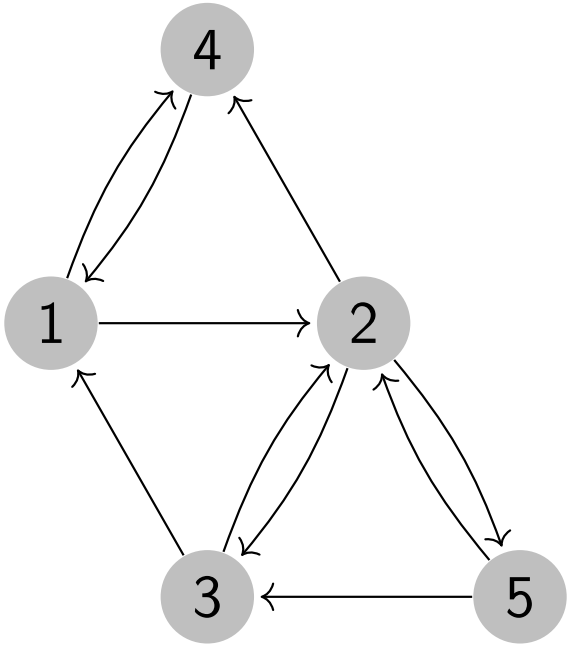
Another example: 2 exchanges



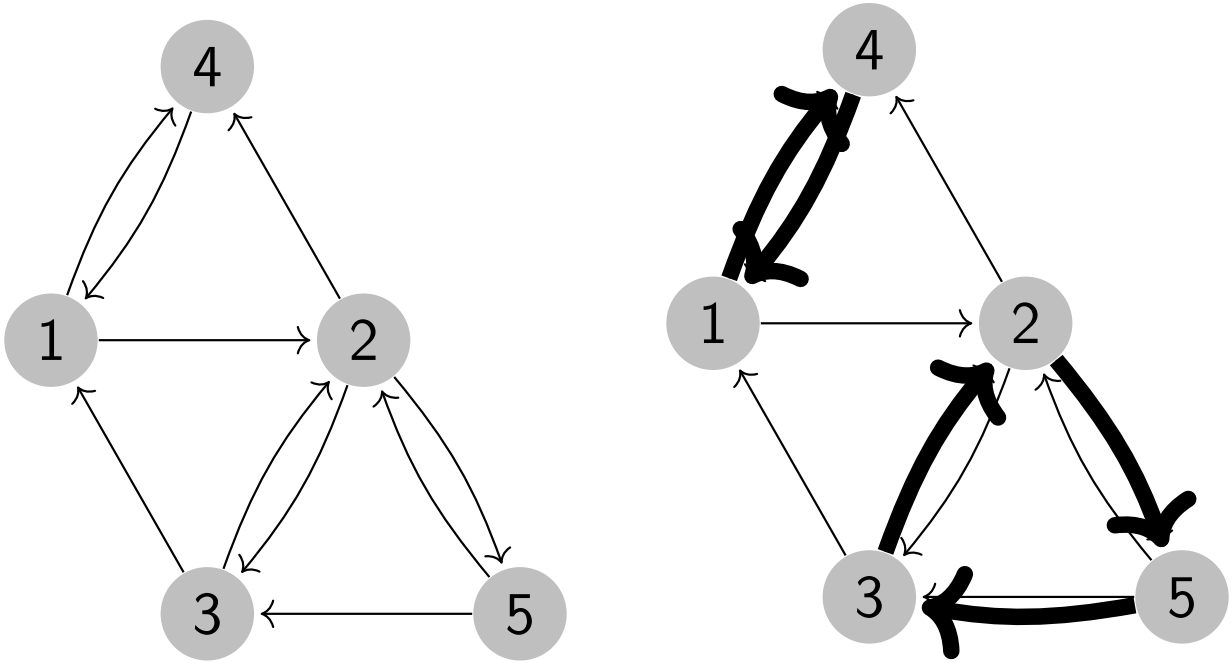




Another example: 3 exchanges



Another example: 3 exchanges



Kidney Exchange Model

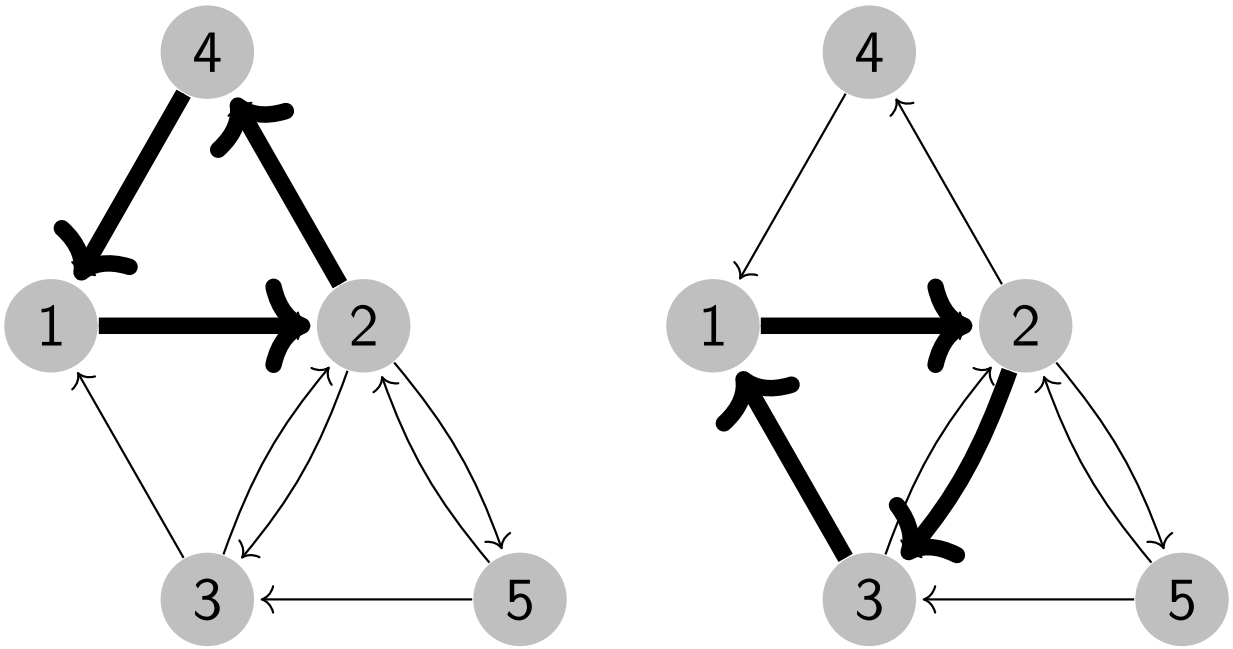
- Given:
  - a pool of n incompatible donor-patient pairs
  - the compatibility between all donors and all patients
- find the maximum number of kidney exchanges with cycles of size at most k

Complexity

- Is this problem easy to solve?
  - YES, if $k = 2$ or no limit is imposed on the size of the cycles
  - NO, if $k = 3, 4, 5, \ldots$
- If $k = 2$ the problem reduces to finding a maximum matching in a undirected graph, which can be solved efficiently (Edmonds 1965)
- If no limit is imposed on the size of the cycles the problem can be formulated as an assignment problem (can be solved efficiently by hungarian algorithm)
- The problem is NP-hard for $k = 3, 4, 5, \ldots$ (hence, no polynomial algorithms are known to solve it)

Mathematical programming formulations

- There are several possibilities for modeling the problem in mathematical programming
- One of the most successful is the cycle formulation:
  - enumerate all cycles in the graph with length at most K
  - for each cycle c, let variable x_c be 1 if c is chosen, 0 otherwise
  - every feasible solution corresponds to a set of vertex-disjoint cycles



Cycle formulation

$$\text{maximize} \quad \sum_{c} w_c x_c \tag{1}$$

$$\text{subject to} \quad \sum_{c: i \in c} x_c \leq 1 \quad \forall i \tag{2}$$

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

- case of $0-1$ weights: $w_c = |c|$, (length of cycle c)
- objective: maximize the weight of the exchange
- constraints: every vertex is at most in one cycle (*i.e.*, donate/receive at most one kidney)
- difficulty: number of variables

Cycle formulation

- Exponential number of variables
- Not all are needed for solving the problem
- Use only those necessary $\longrightarrow$ column generation

Implementation

Implementation 1: AMPL model

```
set V;    # vertices in the original graph
param m; # number of cycles
set C := 0..m-1;
set CYCLE{C};    # vertices in each cycle

var x{C} binary;

maximize z: sum {c in C} card(CYCLE[c]) * x[c];

Packing {i in V}: sum {c in C: i in CYCLE[c]} x[c] <= 1;
```

Implementation 2: graph as adjacency list

Example:

```
edges = [(1,2), (1,5), (1,8),
         (2,3), (2,4), (2,5), (2,6), (2,8),
         (3,1), (3,2),
         (4,1), (4,8),
         (5,2), (5,3), (5,7),
         (6,2), (6,7), (6,8),
         (7,5), (7,6),
         (8,2), (8,6),
         ]

adj = {}
for (i,j) in edges:
    if i in adj:
        adj[i].append(j)
    else:
        adj[i] = [j]
```

Implementation 3: calling AMPL model

```
def kep_cycle(adj,cycles):
    ampl = AMPL()
    ampl.option['solver'] = 'gurobi'
    ampl.read("kep.mod")
    ampl.set['V'] = list(adj)    # vertices are keys in 'adj'
    ampl.param['m'] = len(cycles)
    for i,c in enumerate(cycles):
        ampl.set['CYCLE'][i] = list(c)

    ampl.solve()
    z = ampl.obj['z']
    print("Optimum number of transplants:", z.value())

    print("Selected cycles:")
    x = ampl.var['x']
    sol = []
    for i,c in enumerate(cycles):
        if x[i].value() > 0.5:
            sol.append(c)

    return sol
```

Implementation 4: cycle enumeration

Prepare call to recursive function

```
def get_all_cycles(adj,K):
    tovisit = set(adj.keys())
    visited = set()
    cycles = set()
    for i in tovisit:
        tmpvisit = set(tovisit)
        tmpvisit.remove(i)
        first = i
        all_cycles(cycles,[],first,tmpvisit,adj,K)
    return cycles
```

Implementation 5: cycle enumeration (recursion)

Recursive call + cycle relabeling (to start with lowest vertex)

```
def normalize(cycle):
    cmin = min(cycle)
    while cycle[0] != cmin:
        v = cycle.pop(0)
        cycle.append(v)


def all_cycles(cycles, path, node, tovisit, adj, K):
    for i in adj[node]:
        if i in path:
            j = path.index(i)
            cycle = path[j:]+[node]
            normalize(cycle)
            cycles.add(tuple(cycle))
        if i in tovisit:
            if K-1 > 0:
                all_cycles(cycles,path+[node],i,tovisit-set([i]),adj,K-1)
    return cycles
```

Implementation 6: main program

```
adj = ...
cycles = get_all_cycles(adj,3)
sol = kep_cycle(adj,cycles)
print("solution:", sol)
```

This lesson

- matching problems
- kidney exchange problems

Next lessons:

- kidney exchange problems
- machine learning $\rightarrow$ data-driven components
- (if time allows, more optimization at the end: nonlinear)