Discrete Optimization

Constraint Programming: Part VIII

Goals of the Lecture

- Illustrate how to implement global constraints
  - knapsack
  - alldifferent
- Significant area of research
  - over 100 global constraints proposed so far

The Gold Standard for Pruning

- After pruning
  - if value v is in the domain of variable x, then there exists a solution to the constraint with value v assigned to variable x
- Many historical names
  - arc consistency, domain consistency
- Optimal pruning
  - cannot prune more if only domains are considered
- Complexity
  - may not be enforced in polynomial time

Binary Knapsack

- The constraint

$$l \leq \sum_{k \in R} w_k x_k \leq u$$
$$x_k \in \{0, 1\}$$

- Example

$$10 \leq 2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 12$$

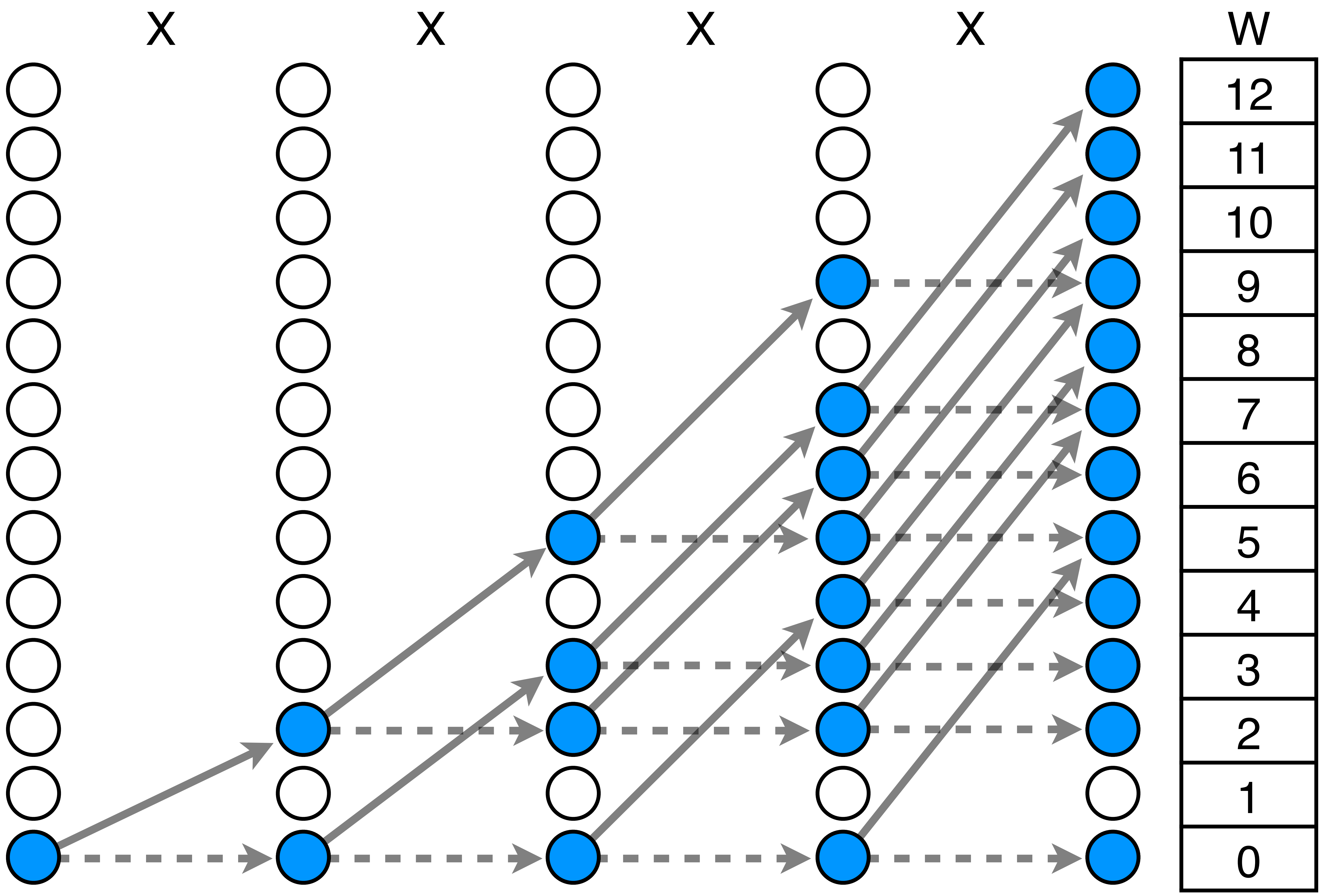
  - feasibility
    - can we find a solution to the constraint?
  - pruning
    - can we eliminate values from the domains?

Binary Knapsack

- Feasibility
  - use dynamic programming for feasibility
  - pseudo-polynomial
- Pruning
  - exploit the dynamic programming table
  - forward phase
    - keep dependency links
  - backward phase
    - update dependency links to keep only feasible values
  - combine feasibility with pruning

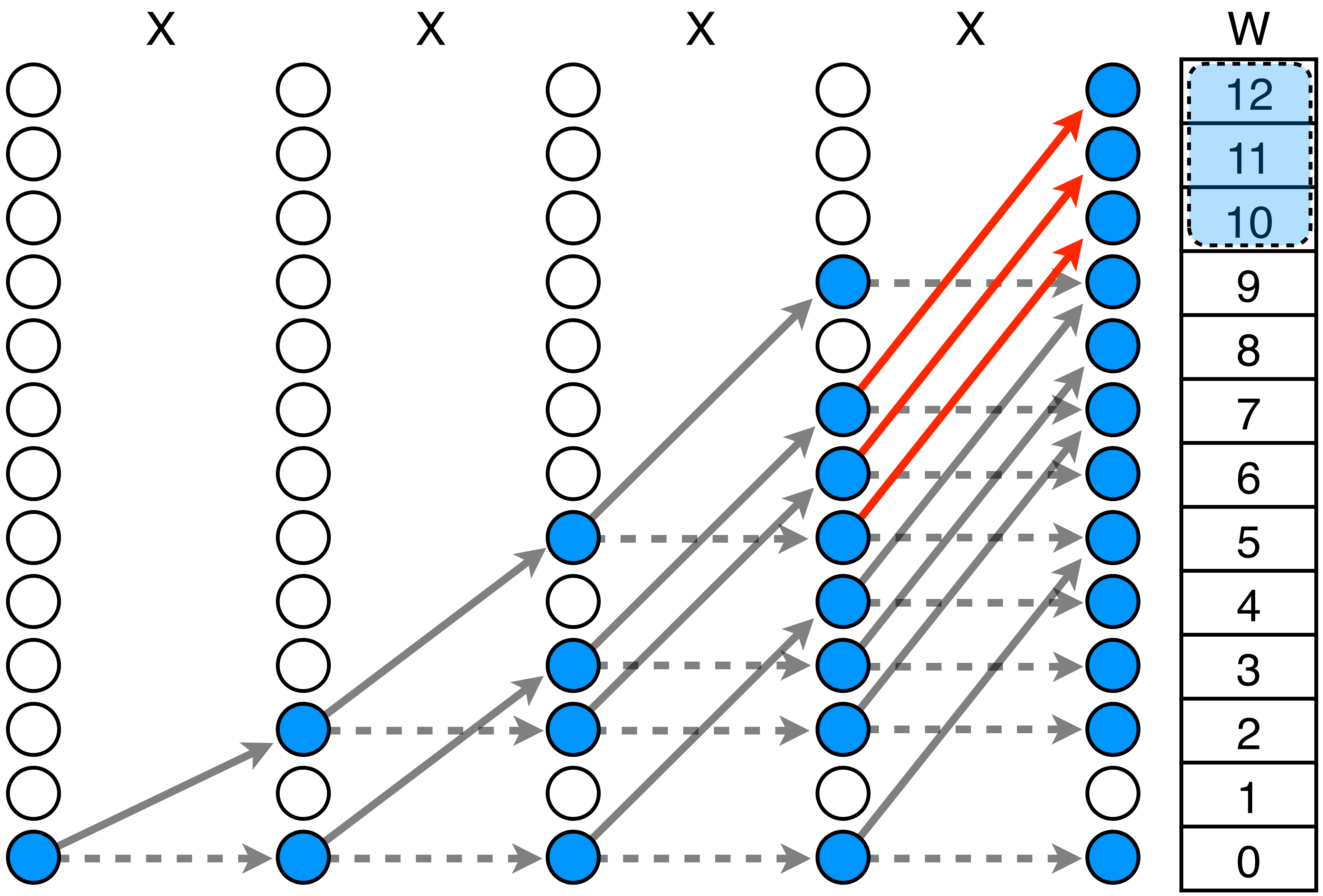
Forward Phase

$$10 \leq 2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 12$$



Backward Phase

$$10 \leq 2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 12 \qquad \boxed{x_4 = 1}$$



Alldifferent Constraint

- The constraint
  - alldifferent(x_1,...,x_n)
- Feasibility
  - can we find values in the domains of the variables so that each two variables are assigned a different value?
- Pruning
  - are there values in the domain of a variable that the variable cannot take, i.e., if the variable takes that value, then there is no solution.

Alldifferent Representation

$X_1 \in \{1, 2\}$

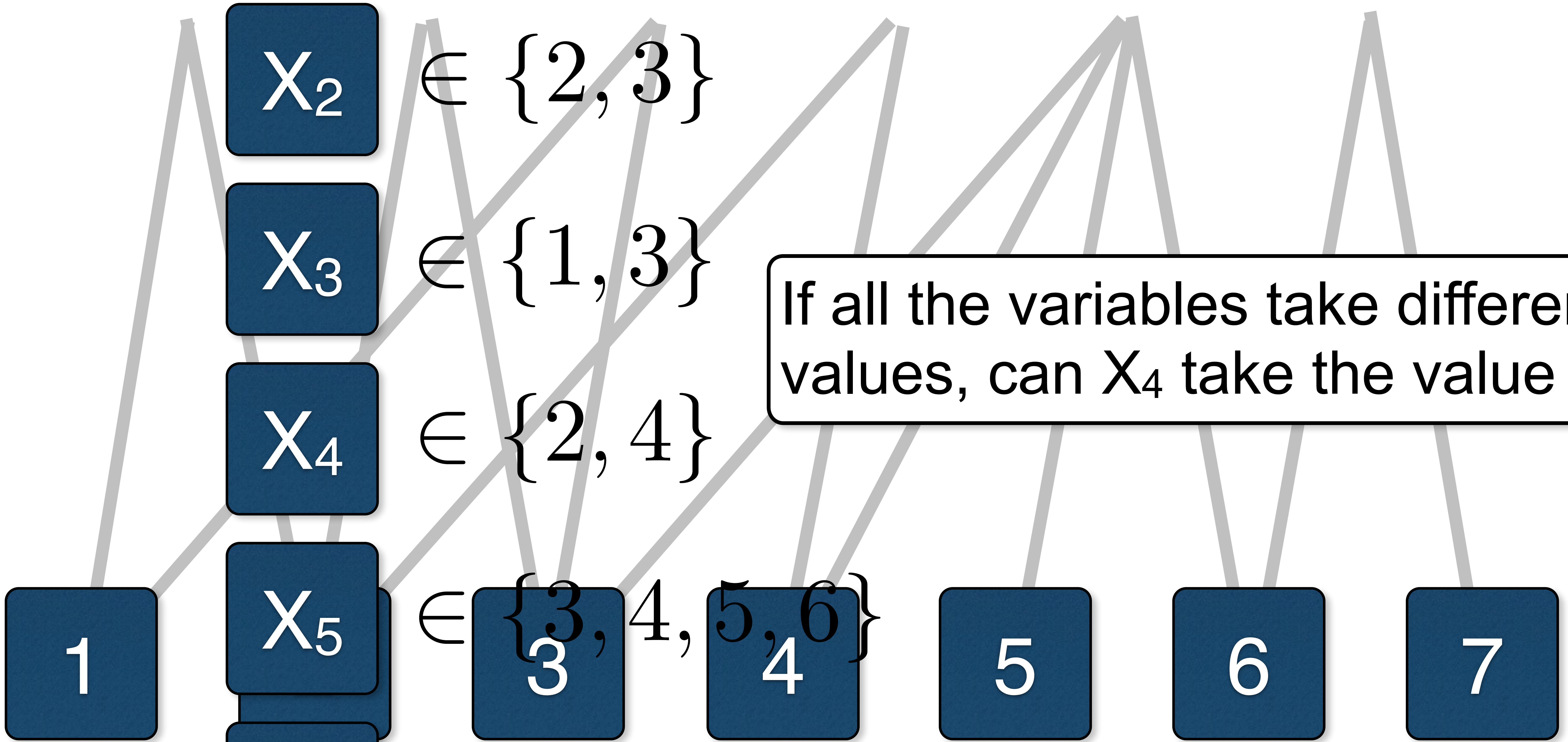
$X_2 \in \{2, 3\}$

$X_3 \in \{1, 3\}$

$X_4 \in \{2, 4\}$

$X_5 \in \{3, 4, 5, 6\}$

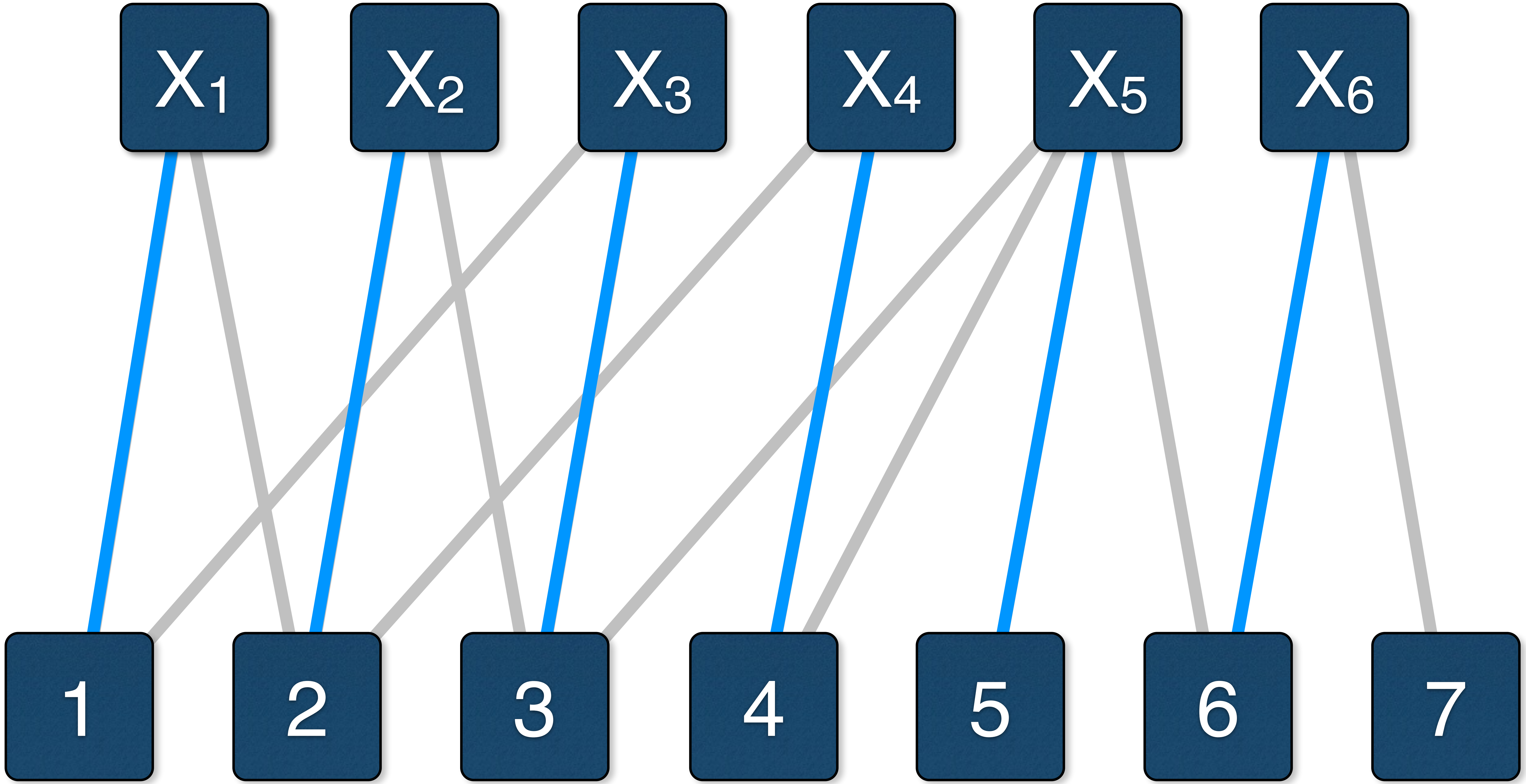
X_6



If all the variables take different values, can X_4 take the value 2?

Can X_4 take the value 2?

Alldifferent Feasibility

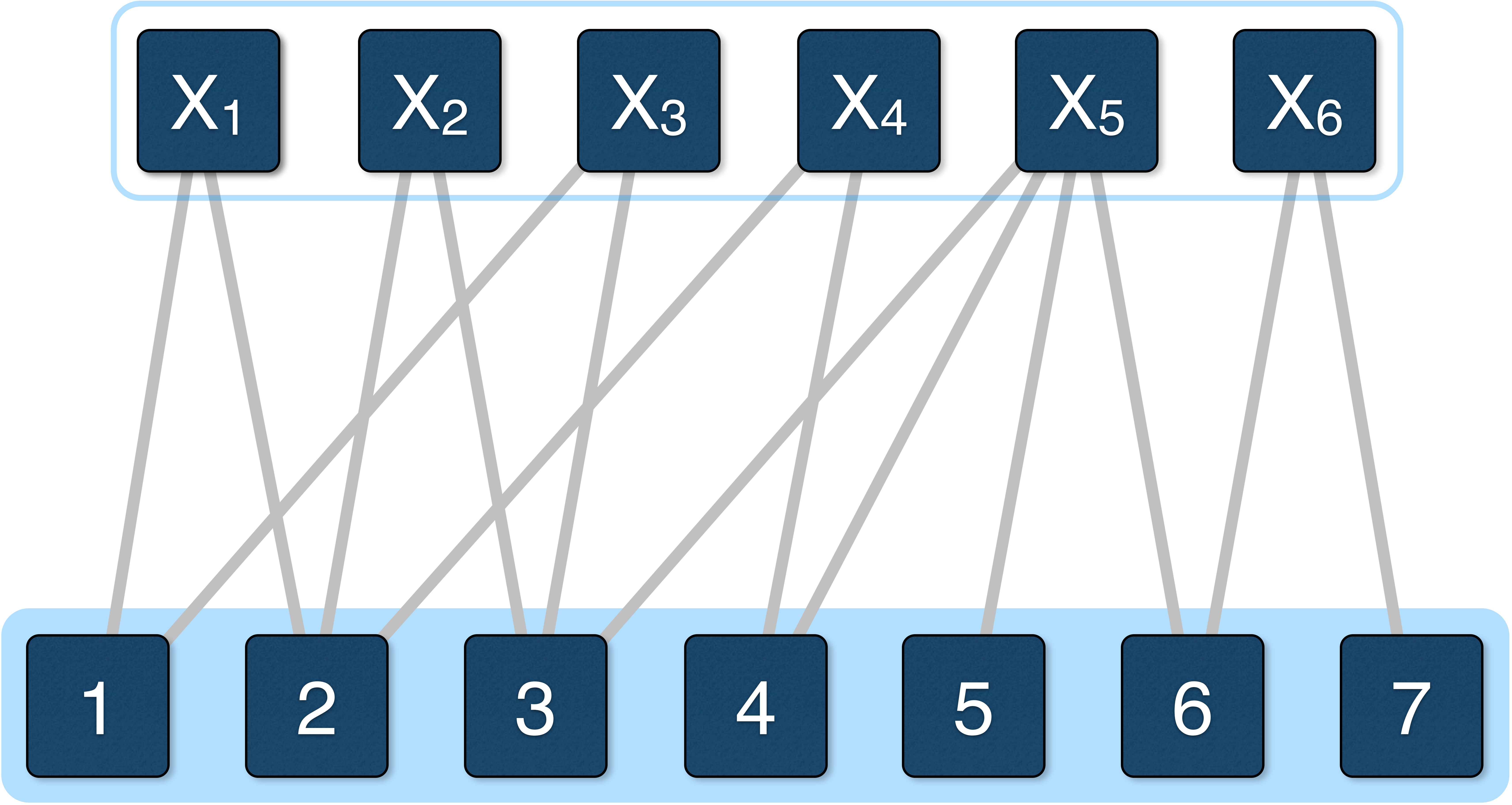


This is a feasible solution!

Alldifferent Constraint

- Bipartite graphs
  - a graph with two types of vertices
  - edges only between vertices of different types
- Alldifferent constraint
  - vertices for the variables
  - vertices for the values
  - edges between variables and values

Alldifferent Feasibility

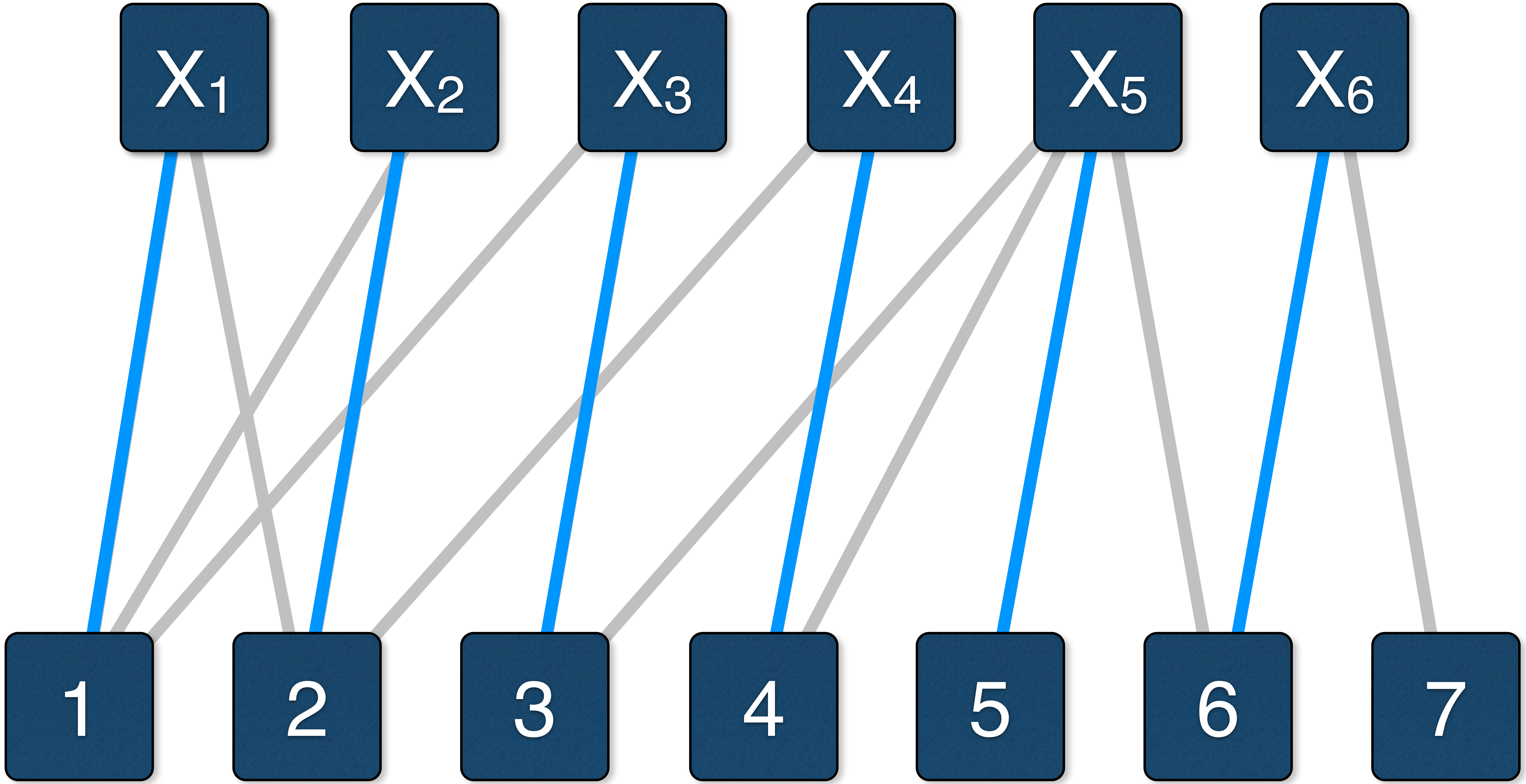


Matching and Alldifferent

- A **matching** for a graph $G=(V,E)$ is a set of edges in E such that no two edges in E share a vertex.
- A **maximum matching** M for a graph G is a matching with the largest number of edges
- Feasibility
  - finding a maximum matching in a bipartite graph

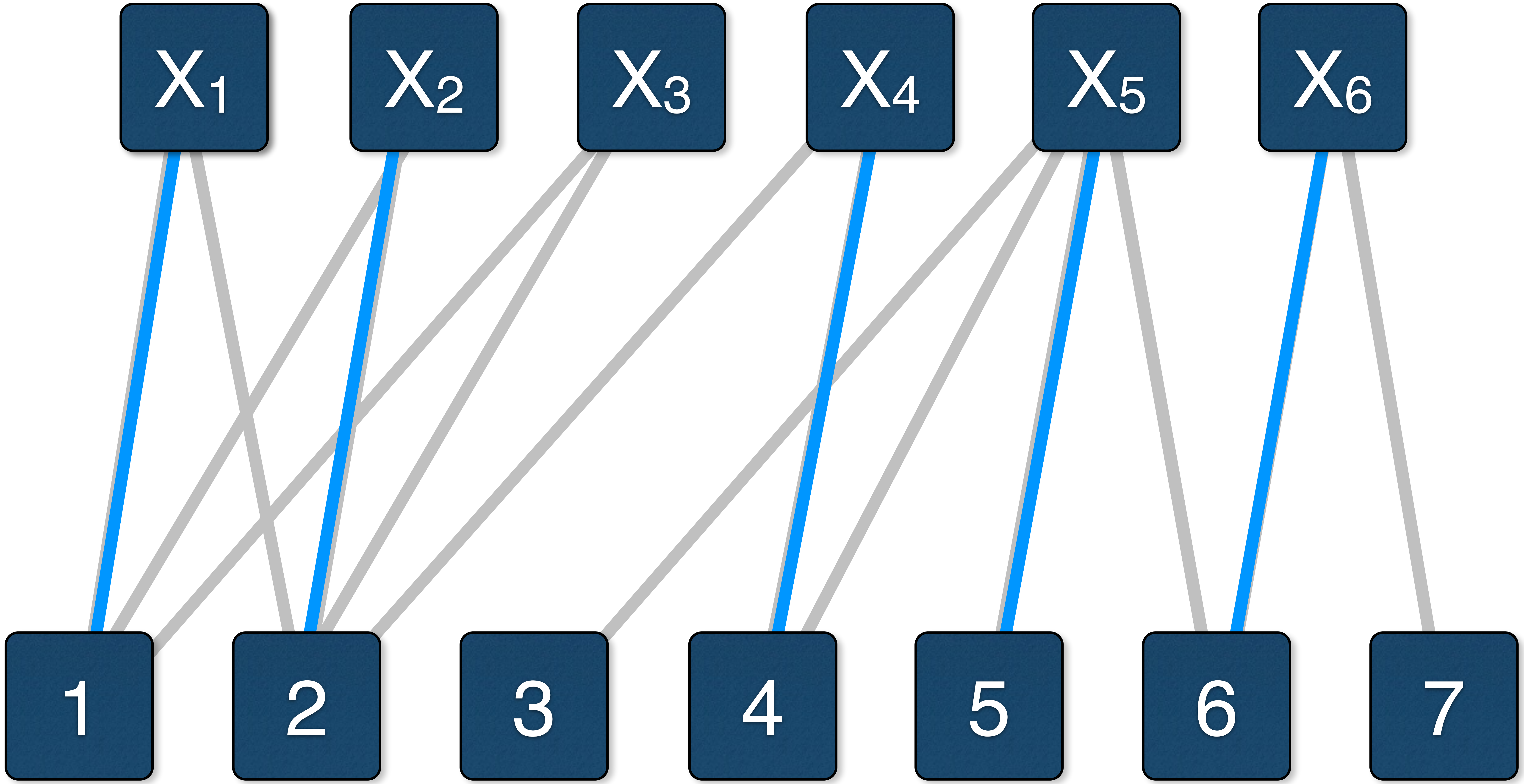
if the maximum matching has a size equal to the number of variables, then the constraint is feasible; otherwise, it is not feasible

Alldifferent Feasibility



The constraint is feasible

Alldifferent Feasibility

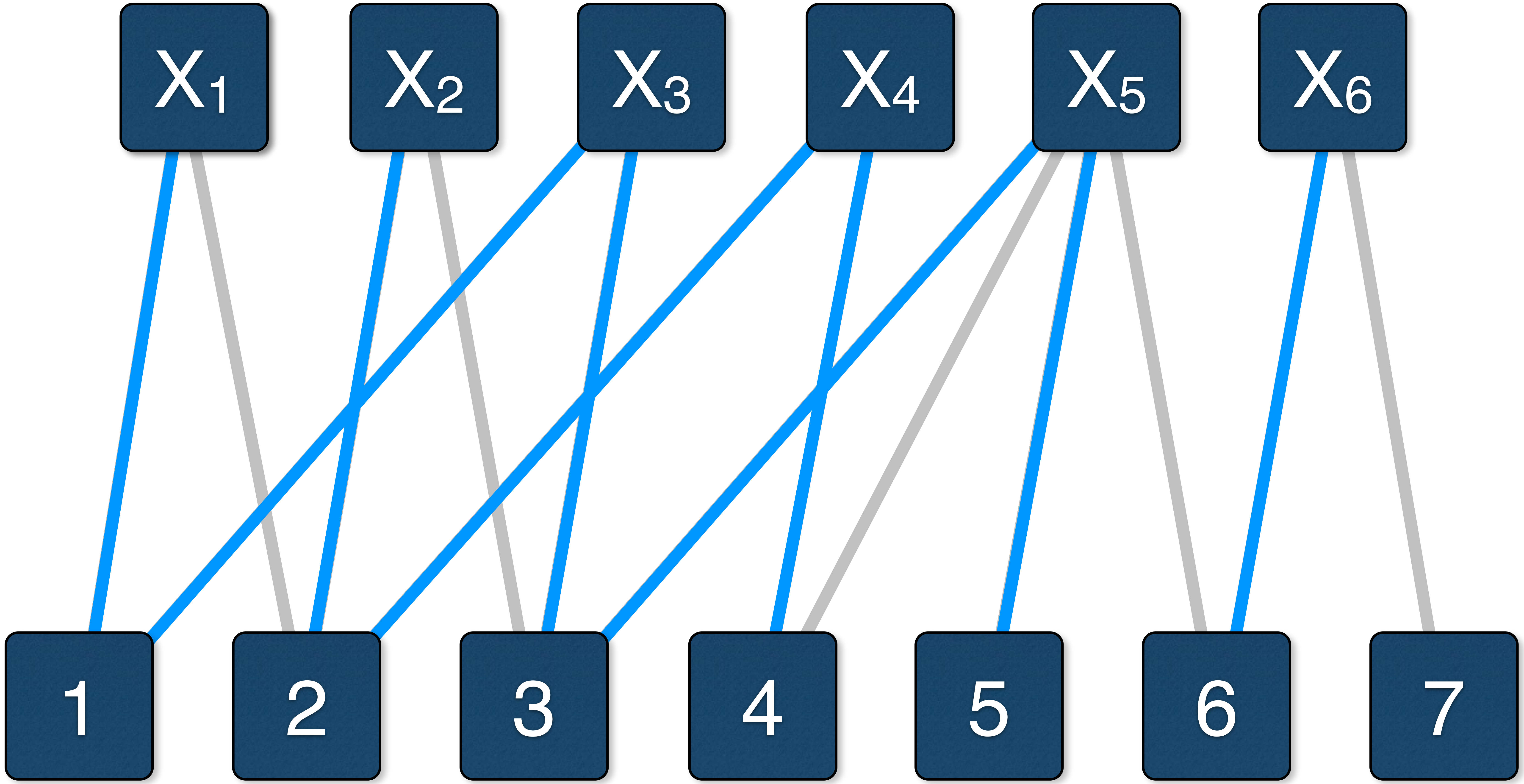


The constraint is not feasible

How to Find a Maximum Matching?

- How to find a maximum matching?
  - start with any matching
  - improve the matching
- When no improvement is possible
  - we have a maximum matching

How to Improve a Matching?

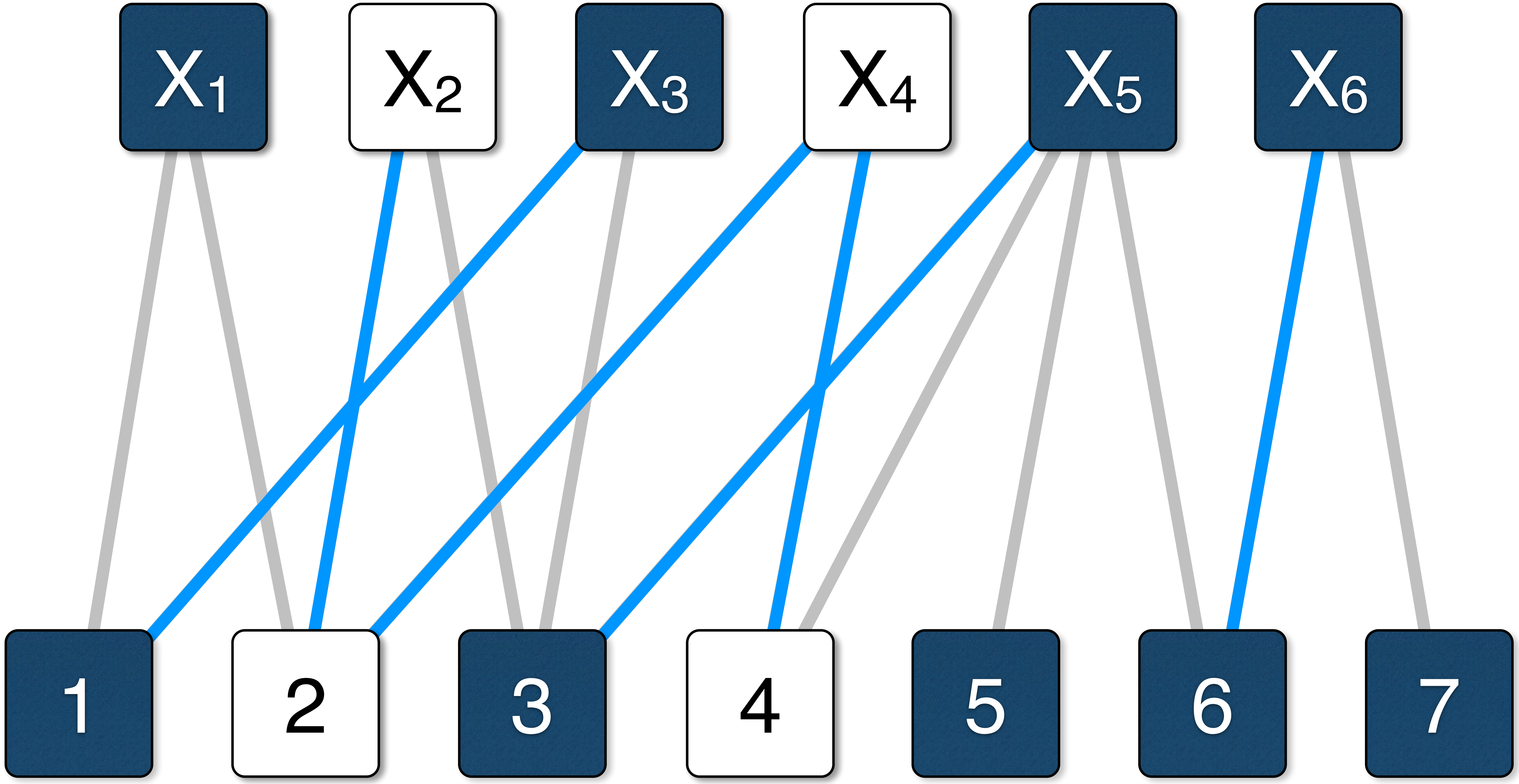


Replace $(X_3,1)$ and $(X_5,3)$ by $(X_1,1)$, $(X_3,3)$, $(X_5,5)$

How to Find a Maximum Matching?

- How to find a maximum matching?
  - start with any matching
  - improve the matching
- How to find an improvement?
  1. start from a free vertex x
  2. if there is an edge (x,v) where v is not matched, then insert (x,v) in the matching
  3. otherwise, take a vertex v matched to y. remove (y,v) and add (x,v) from the matching and restart at step 2 with y instead of x

How to Improve a Matching?



Add $(X_2,2)$,

Match X_4 and 4

onsider X_4

Alternating Paths

- An alternating path P for a matching M is a path from a vertex x in X to a vertex v in V (both of which are free) such that the edges in the path are alternatively in $E \backslash M$ and M.

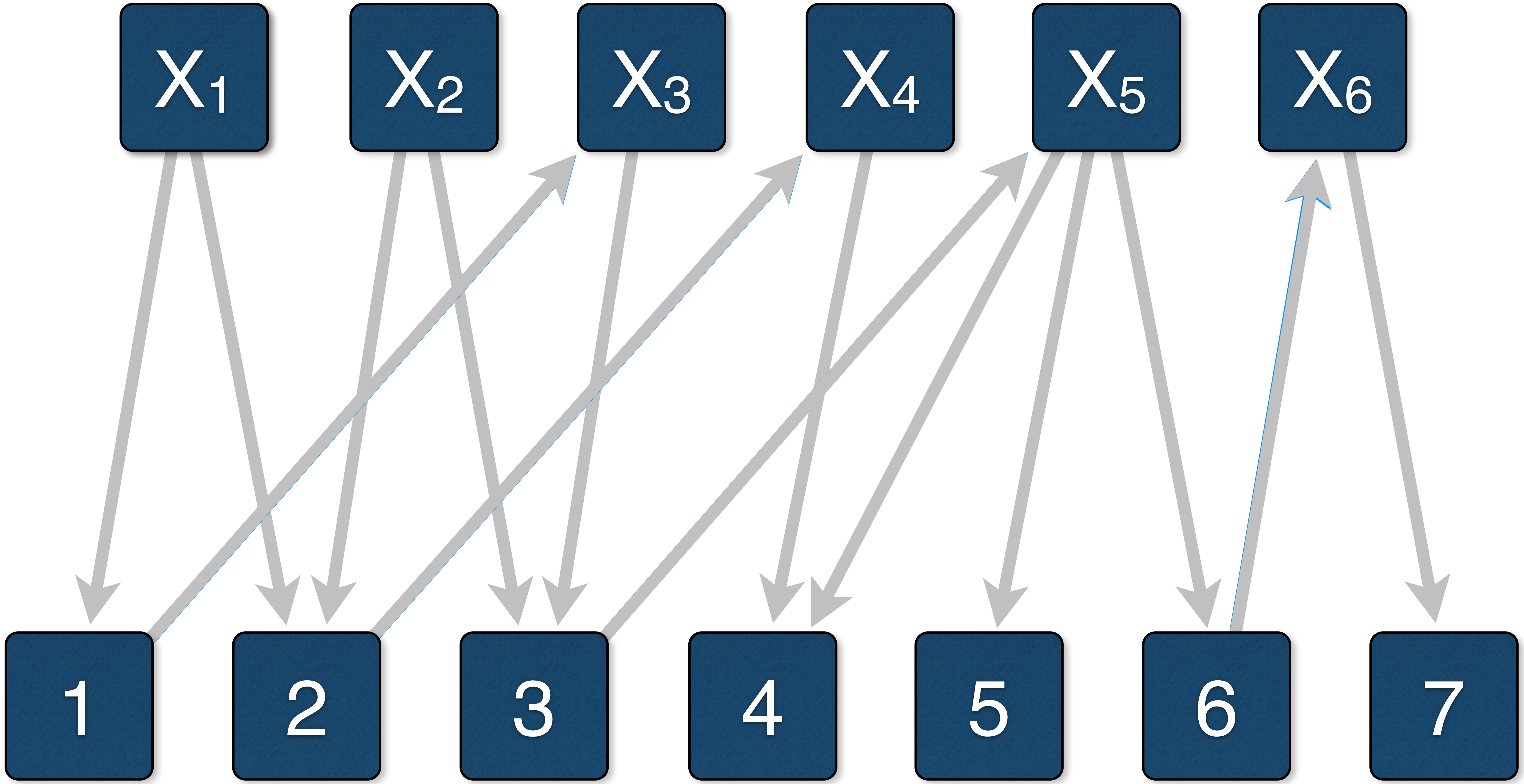
- Odd number of edges
  - why?

- We improve the matching!

How to Find Alternative Paths

- Create a directed graph.
- Given a matching
  - edges in the matching are oriented from bottom to top
  - edges not in the matching are oriented from top to bottom

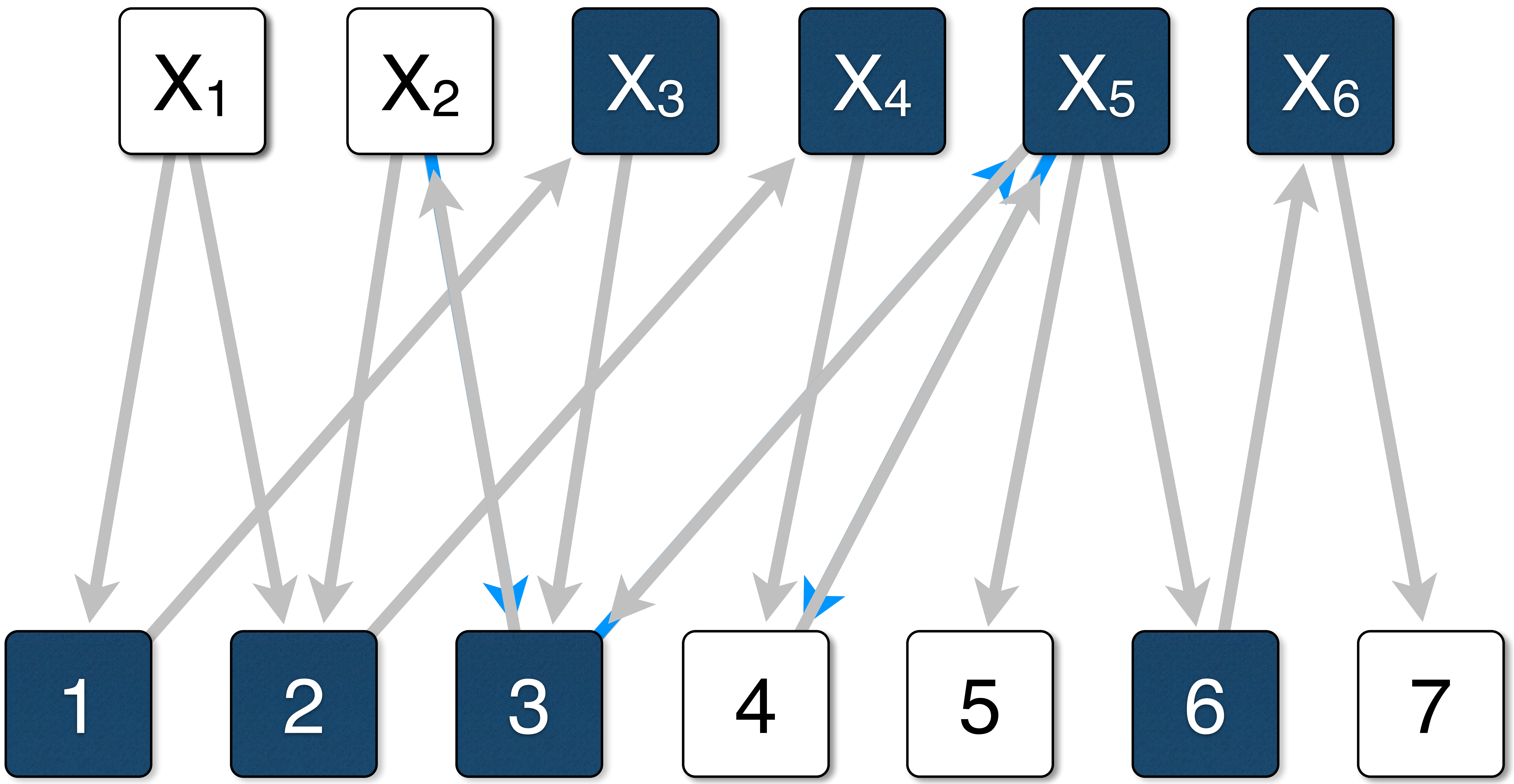
How to Improve a Matching?



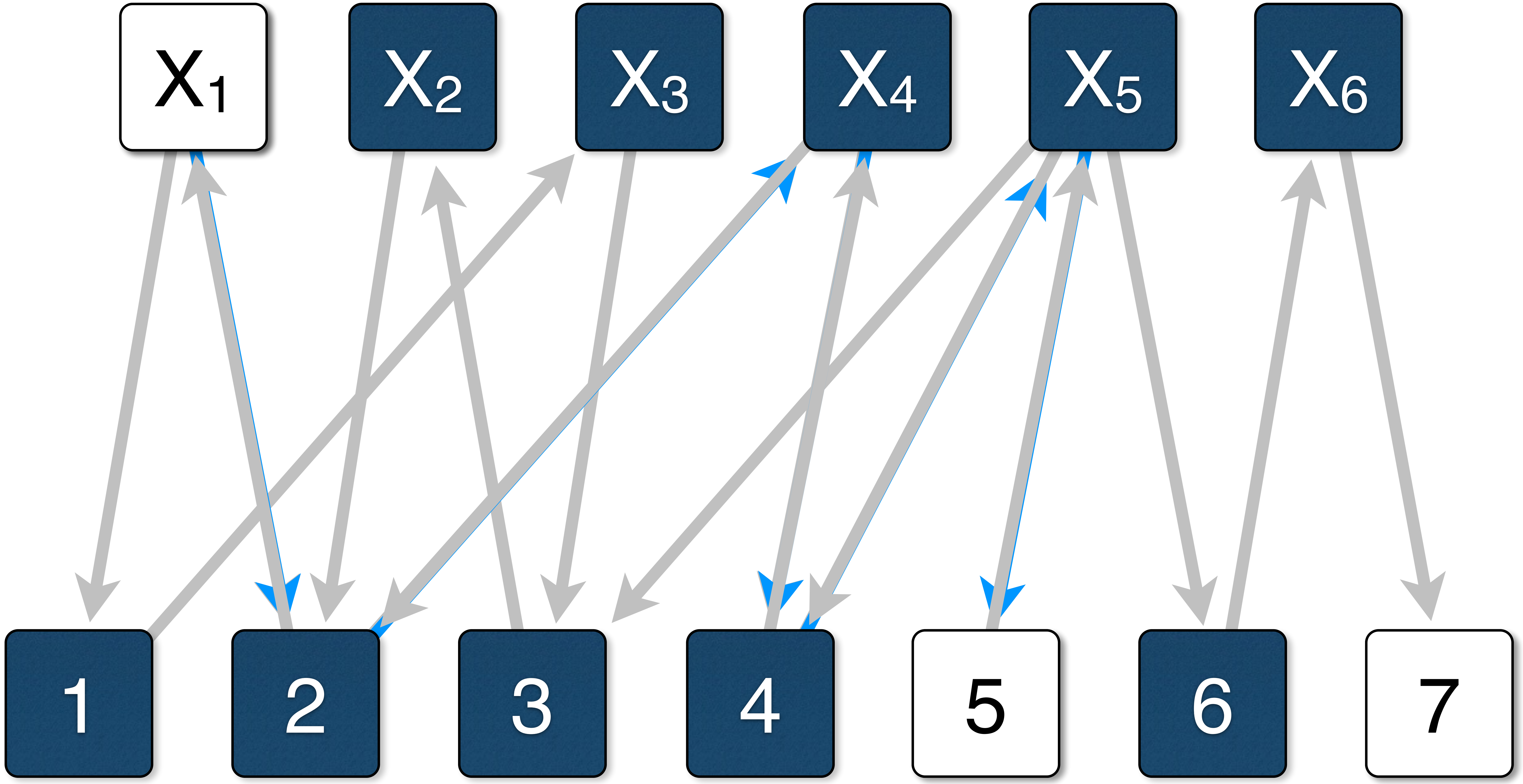
How to Find Alternative Paths

- Create a directed graph.
- Given a matching
  - edges in the matching are oriented from bottom to top
  - edges not in the matching are oriented from top to bottom
- An alternating path is thus a path starting from a free vertex x and ending in another free vertex v

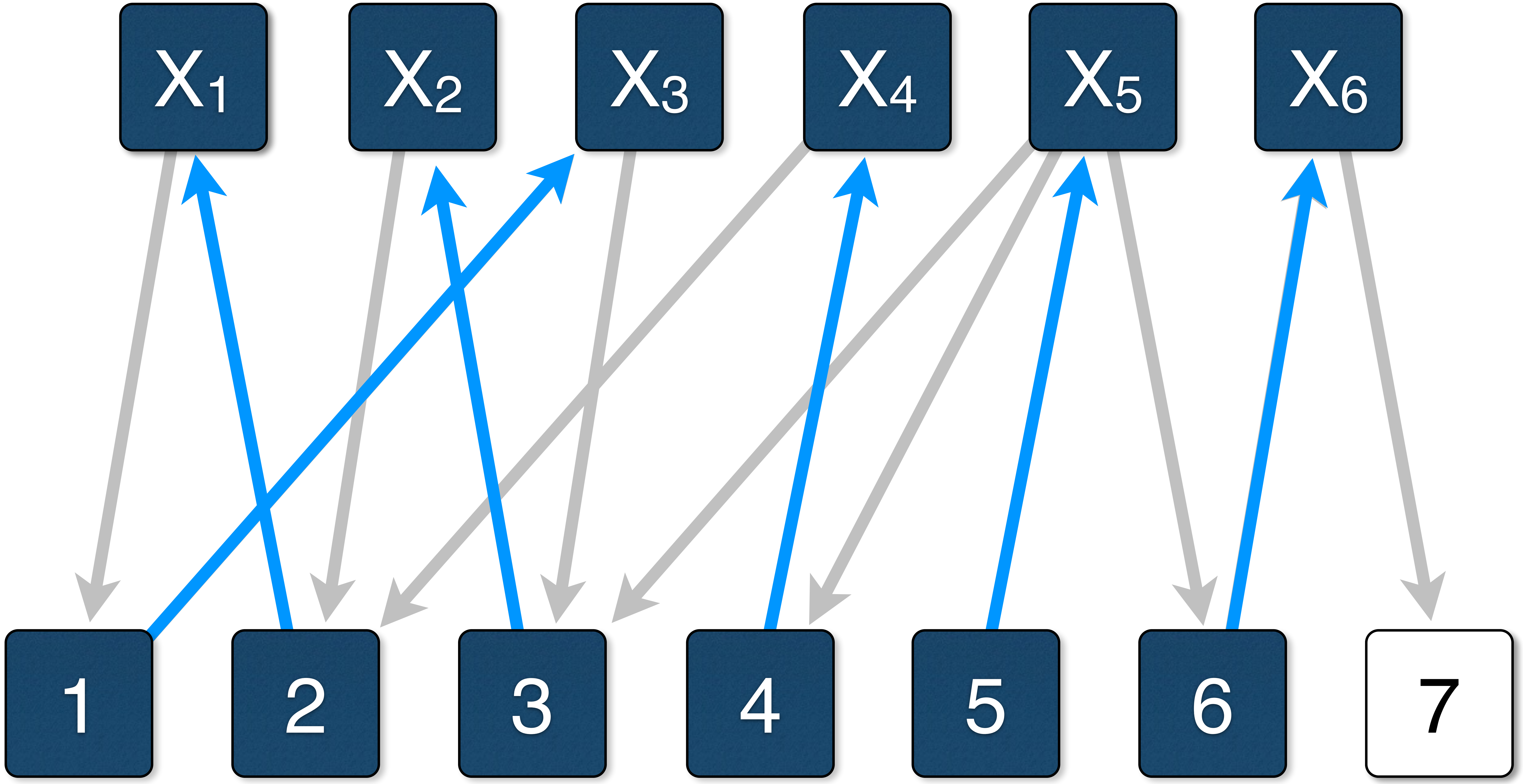
How to Find Alternative Paths



How to Find Alternative Paths



How to Find Alternative Paths



How to Find Alternative Paths

- Given a matching
  - edges in the matching are oriented from bottom to top
  - edges not in the matching are oriented from top to bottom
- An alternating path is thus a path starting from a free vertex *x* and ending in another free vertex *v*
- How to find an alternative path?
  - use depth-first or best-first search
  - $O(|X| + |E|)$ where X is the set of vertices and E is the set of edges

Feasibility of the Alldifferent constraint

- Use a bipartite graph
  - vertex set for the variables
  - vertex set for the values
  - edge *(x,v)* if *v* is in *D(x)*
- Feasibility
  - alldifferent is feasible iff the size of the maximum matching is the number of variables
- Finding a maximum matching
  - improve a matching using alternating paths in the directed graph obtained by a proper orientation of the edges

How to Prune?

- Pruning is equivalent to
  - v must be removed from the domain of x if the edge (x,v) appears in no maximum matching
- Only need to look at the edges not present in the maximum matching
- Naive approach,
  - force the edge (x,v) in the matching, i.e., remove all other edges (x,w).
  - search for a maximum matching.
  - If it is smaller than the number of variables, v must be removed from $D(x)$

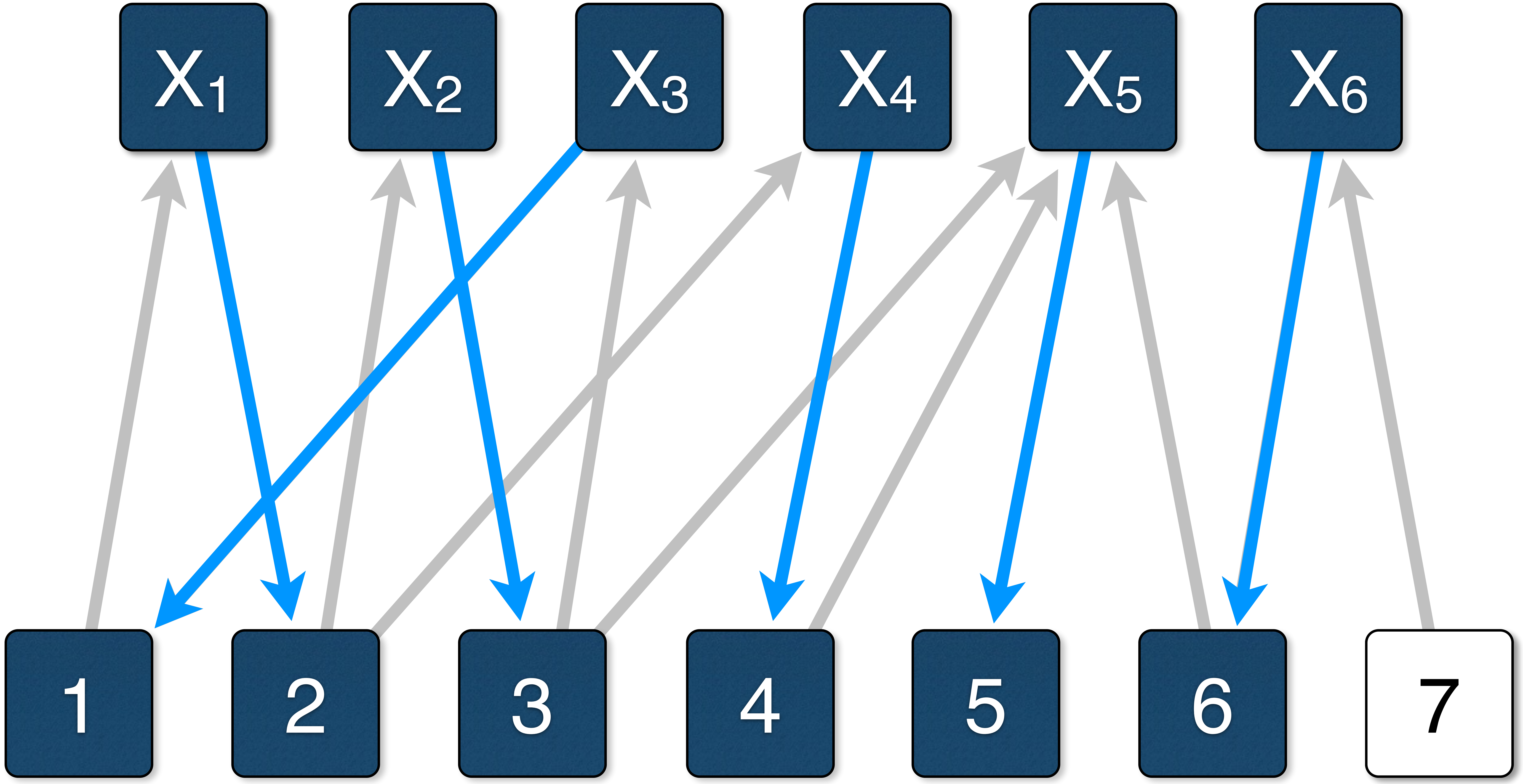
How to Prune?

- Basic property (Berge, 1970)
  - an edge belongs to some but not all maximum matchings iff, given a maximum matching M, it belongs to either
    - an even alternating path starting at a free vertex
    - an even alternating cycle
- Note that
  - the edges not in the maximum matching do not belong to all maximum matchings
  - the above property tells us whether they belong to at least one maximum matching
  - the free vertices are ?

Pruning

- Create a directed graph like before but reserve the direction of the edges
- Given a matching
  - edges in the matching are oriented from top to bottom
  - edges not in the matching are oriented from bottom to top

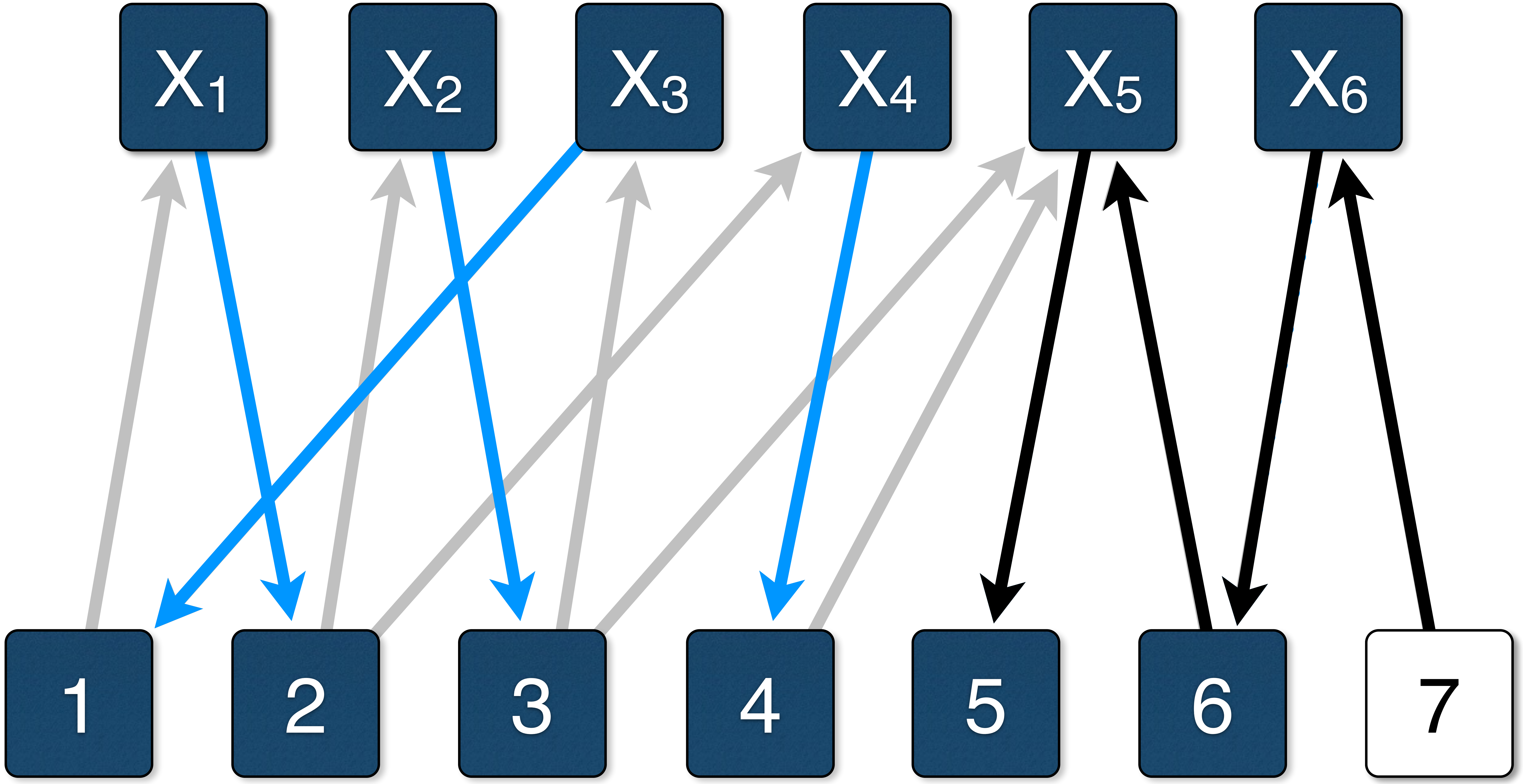
How to Find Alternative Paths



Pruning

- Given a matching M, create a directed graph like before but reserve the direction of the edges
- Search for even alternating path starting from a free vertex: P

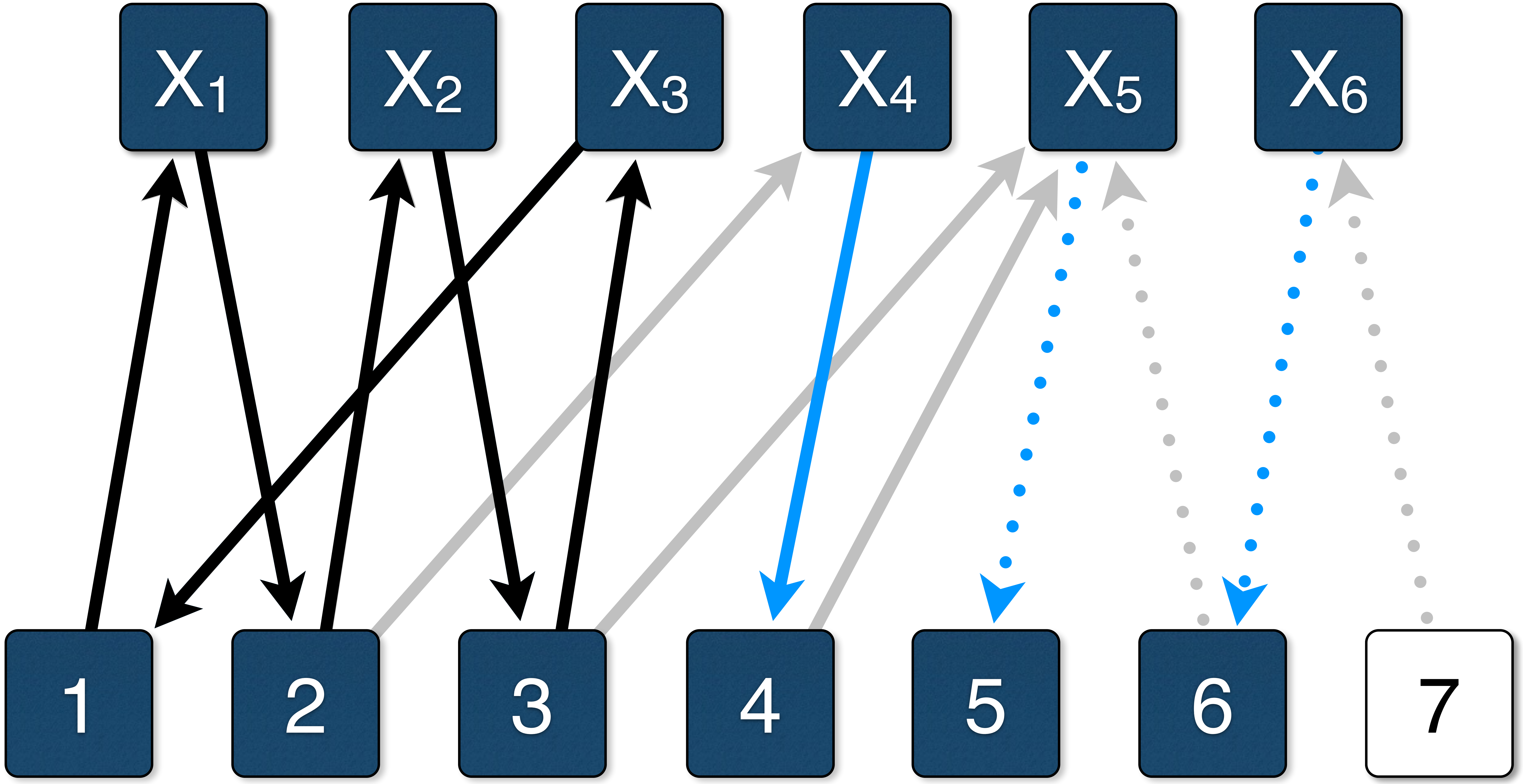
Even Alternating Path from a Free Vertex



Pruning

- Given a matching M, create a directed graph like before but reserve the direction of the edges
- Search for even alternating path starting from a free vertex: P
- Search for all strongly connected components and collect all the edges belonging to them: C

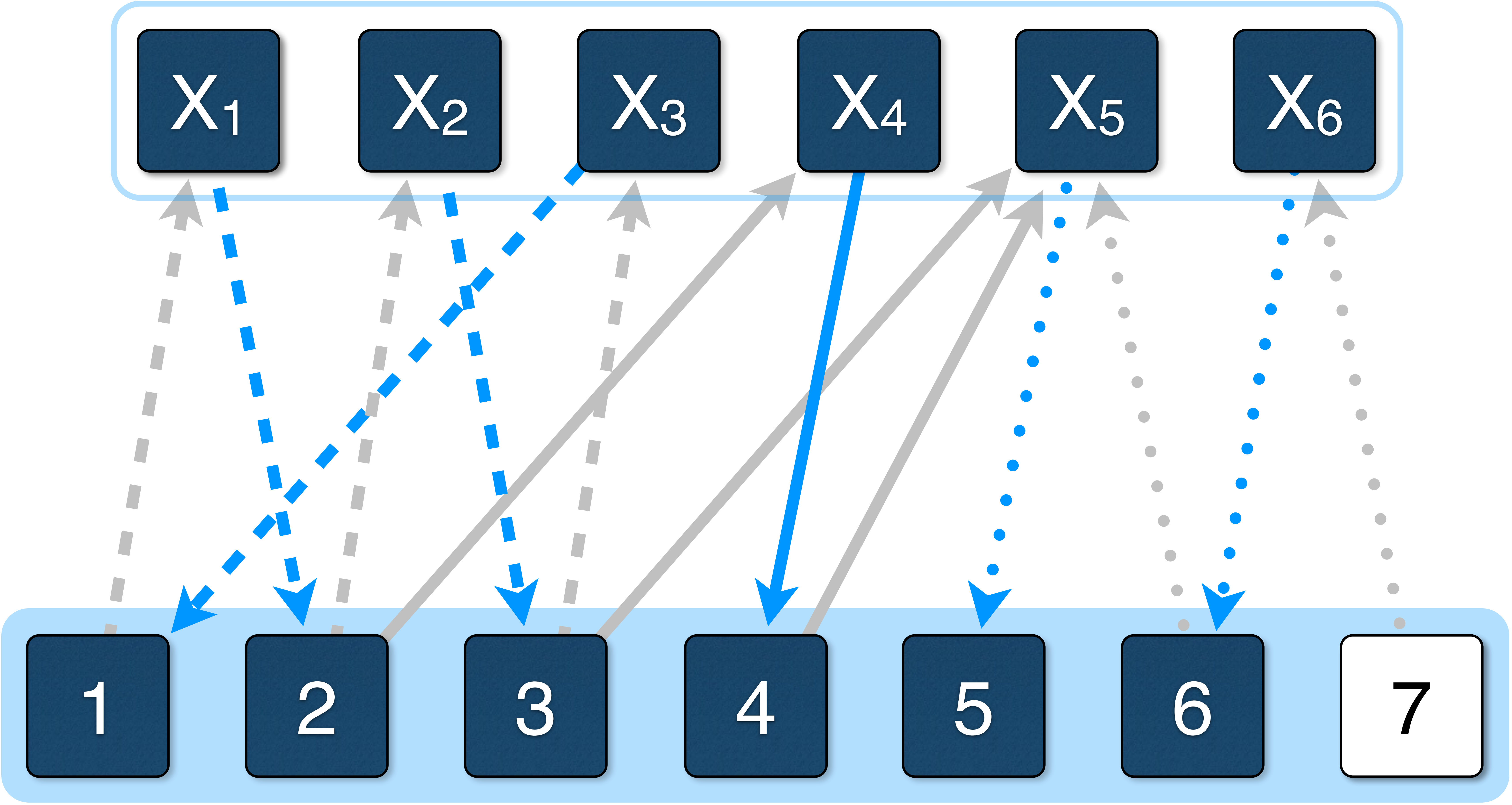
Even Alternating Cycles



Pruning

- Given a matching M, create a directed graph like before but reserve the direction of the edges
- Search for even alternating path starting from a free vertex: P
- Search for all strongly connected components and collect all the edges belonging to them: C
- Remove all edges not in M, P, or C

The Pruned Domains



Pruning

- Given a matching M, create a directed graph like before but reserve the direction of the edges
- Search for even alternating path starting from a free vertex: P
- Search for all strongly connected components and collect all the edges belonging to them: C
- Remove all edges not in M, P, or C
- *Complexity:* $O((|X| + |V|) * |E|)$