Optimization

Quelques formalismes

- Dynamic CSP
- maxCSP
- Soft constraints
- Cost function networks

Dynamic CSP

- A set C_H of physical (hard) constraints and a set C_P of preferences (soft constraints)

- A sequence N_0, N_1, …, N_i, … , where $N_i = (X, D, C_i)$ with $C_i = C_{i-1} \pm \{c\}$, $c \in C_P$ and $C_0 = C_H$

maxCSP

- *Instance*: A constraint network $N=(X,D,C)$
- *Question:* Find an assignment on X that satisfies the **maximum number** of constraints
- Does not discriminate between hard and soft constraints

Soft CSPs / COPs

- Tackle optimization with standard CP solvers
- $c_j(X_1,..,X_k)$ ➜ *soft-*$c_j(X_1,..,X_k,Y_j)$, where Y_j *is the cost for* c_j *of the assignment on* $X_1,..,X_k$

Soft CSPs / COPs

- Tackle optimization with standard CP solvers
- $c_j(X_1,..,X_k)$ ➜ *soft-*$c_j(X_1,..,X_k,Y_j)$, where Y_j *is the cost for* c_j *of the assignment on* $X_1,..,X_k$

- Example 1: penalty for late delivery:
  - X_1 : delivery date;
    X_2 : due date;
    Y_j : penalty
  - $X_1 < X_2$ ➜
    $Y_j = max(0, X_1-X_2)$

Soft CSPs / COPs

- Tackle optimization with standard CP solvers
- $c_j(X_1,..,X_k) \rightarrow soft\text{-}c_j(X_1,..,X_k, Y_j)$, where Y_j *is the cost for* c_j *of the assignment on* $X_1,..,X_k$

- Example 1: penalty for late delivery:
  - X_1 : delivery date;
    X_2 : due date;
    Y_j : penalty
  - $X_1 < X_2 \rightarrow$
    $Y_j = max(0, X_1\text{-}X_2)$

- Example 2: alldifferent:
  - Y_j = number of values already taken

X_1	X_2	X_3	Y_i
1	2	3	0
1	1	2	1
2	2	2	2
3	1	3	1

Soft CSPs / COPs

- Tackle optimization with standard CP solvers
- $c_j(X_1,..,X_k)$ ➜ *soft-*$c_j(X_1,..,X_k,Y_j)$, where Y_j *is the cost for* c_j *of the assignment on* $X_1,..,X_k$

- Example 1: penalty for late delivery:
  - X_1 : delivery date;
    X_2 : due date;
    Y_j : penalty
  - $X_1 < X_2$ ➜
    $Y_j = max(0, X_1\text{-}X_2)$

- Example 2: alldifferent:
  - Y_j = number of values already taken

X_1	X_2	X_3	Y_i
1	2	3	0
1	1	2	1
2	2	2	2
3	1	3	1

- Example 3: some components have a cost

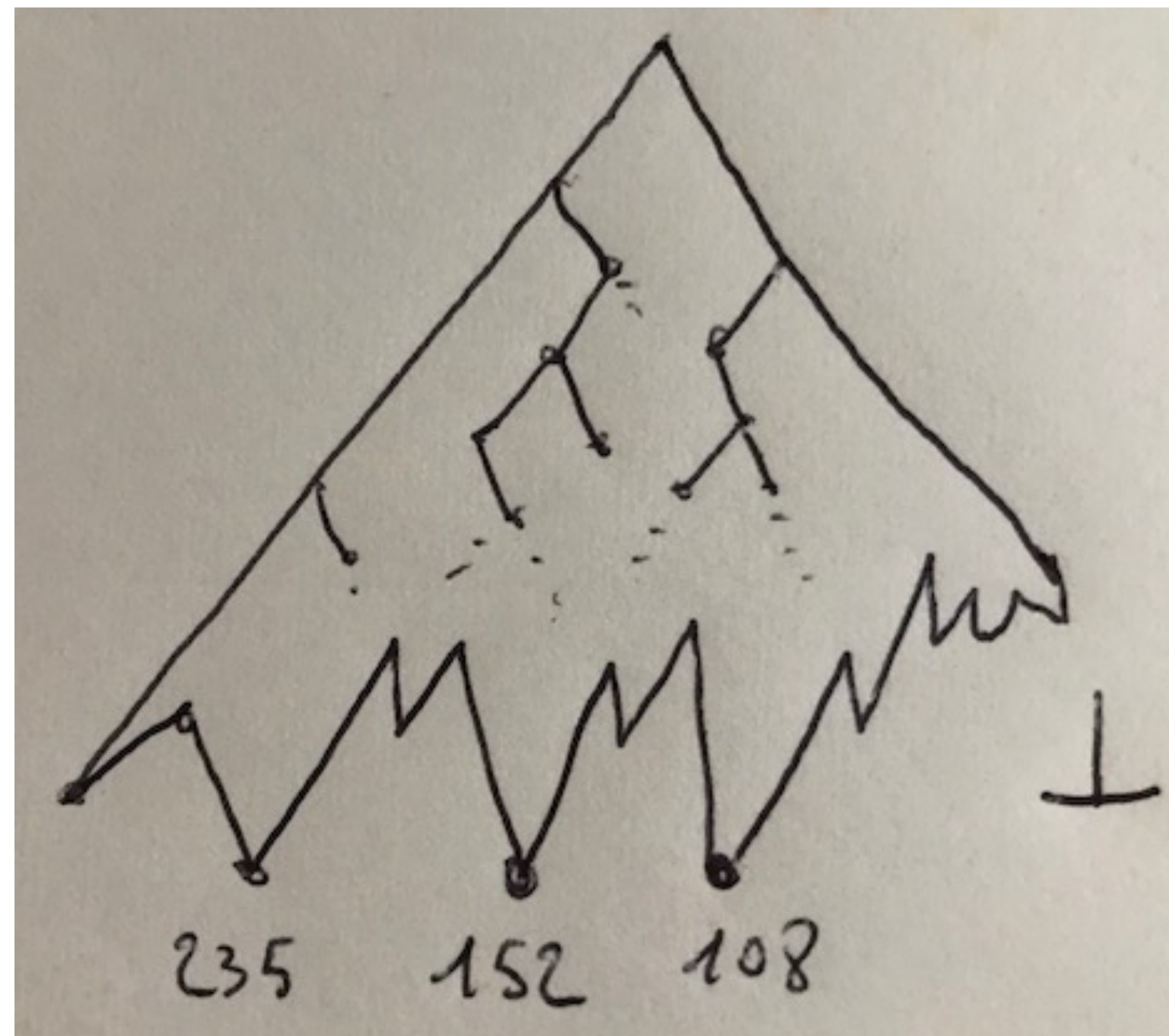
X_i	Y_i
1	5€
2	2€
3	7€
4	1€
5	9€

Soft CSPs / COPs

- The solver is called with the extra constraint

$$\sum_j y_j < UB$$

where UB is the cost of the previously best solution found

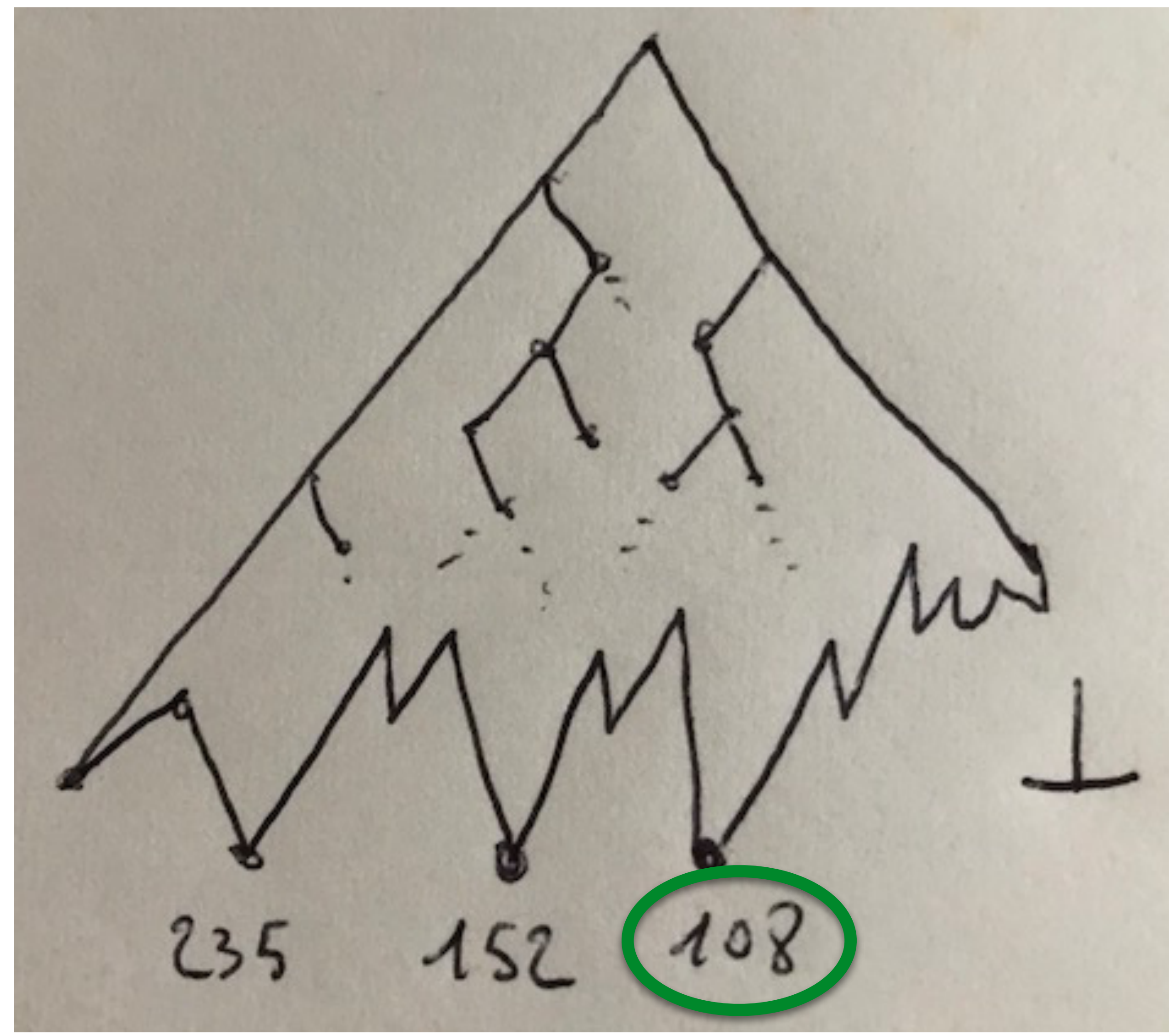


Soft CSPs / COPs

- The solver is called with the extra constraint

$$\sum_j y_j < UB$$

where UB is the cost of the previously best solution found



Cost Function Networks

- A cost function network is a network (X, D, F, k), where every $f \in F$ is a function from $X(f)$ to $0..k$.

X_1	X_2	$\mathbf{f}(X_1,X_2)$
0	0	0
0	1	0
0	2	0
1	0	1
1	1	0
1	2	0
2	0	2
2	1	1
2	2	0

$f(X_1,X_2)$ is the cost (penalty) of the assignment on X_1,X_2

Cost Function Networks

- *Instance*: A cost function network *N=(X, D, F, k)*
- *Question:* Find an assignment *I* on *X* such that

$$\bigoplus_{f \in F} f(I[X(f)])$$

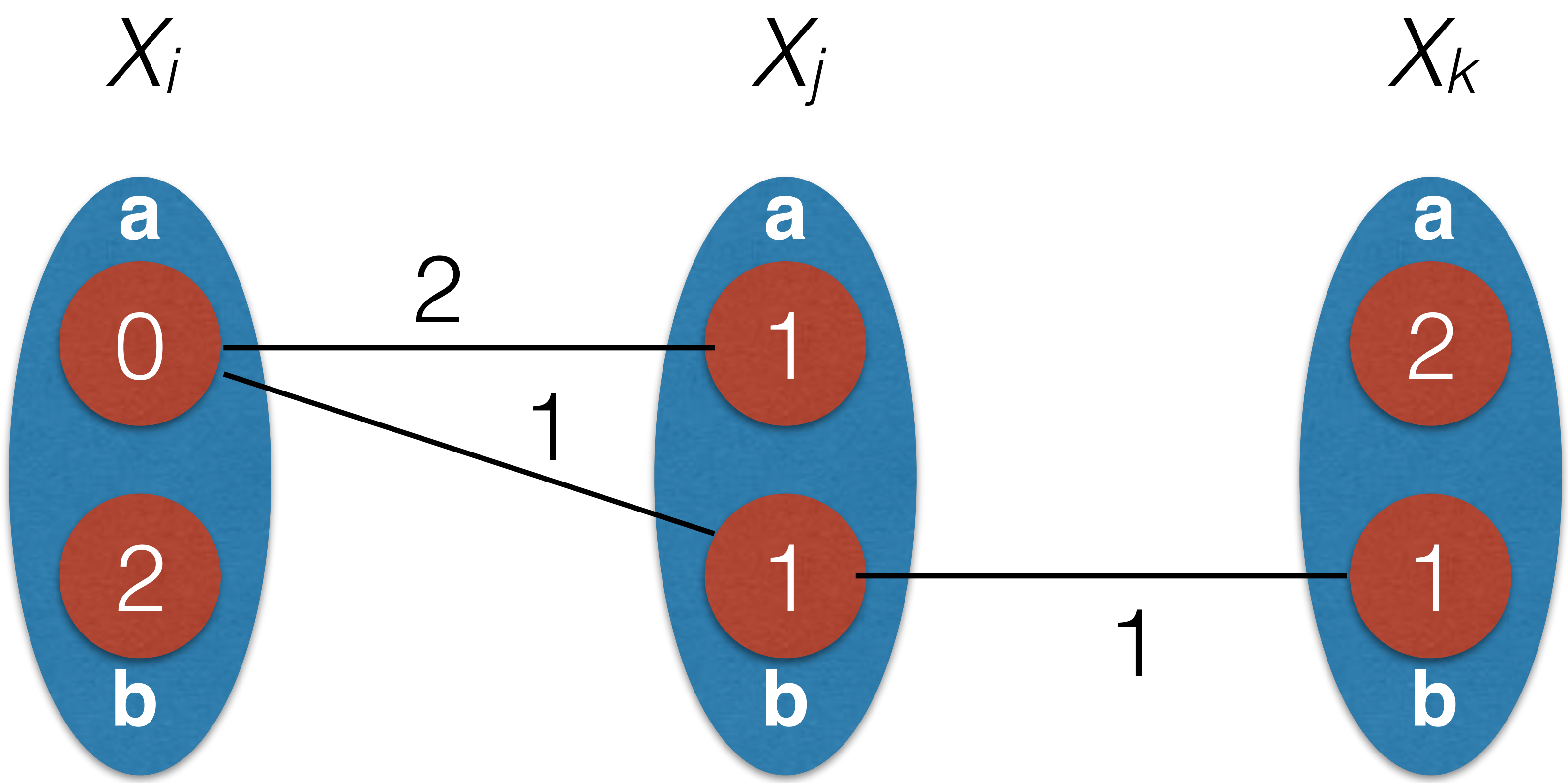
is **minimal** and strictly smaller than *k.*

$$\boxed{a \oplus b = min(a+b, k).}$$

Propagate?

- Arc consistency = **extend/project** transformations instead of standard propagation

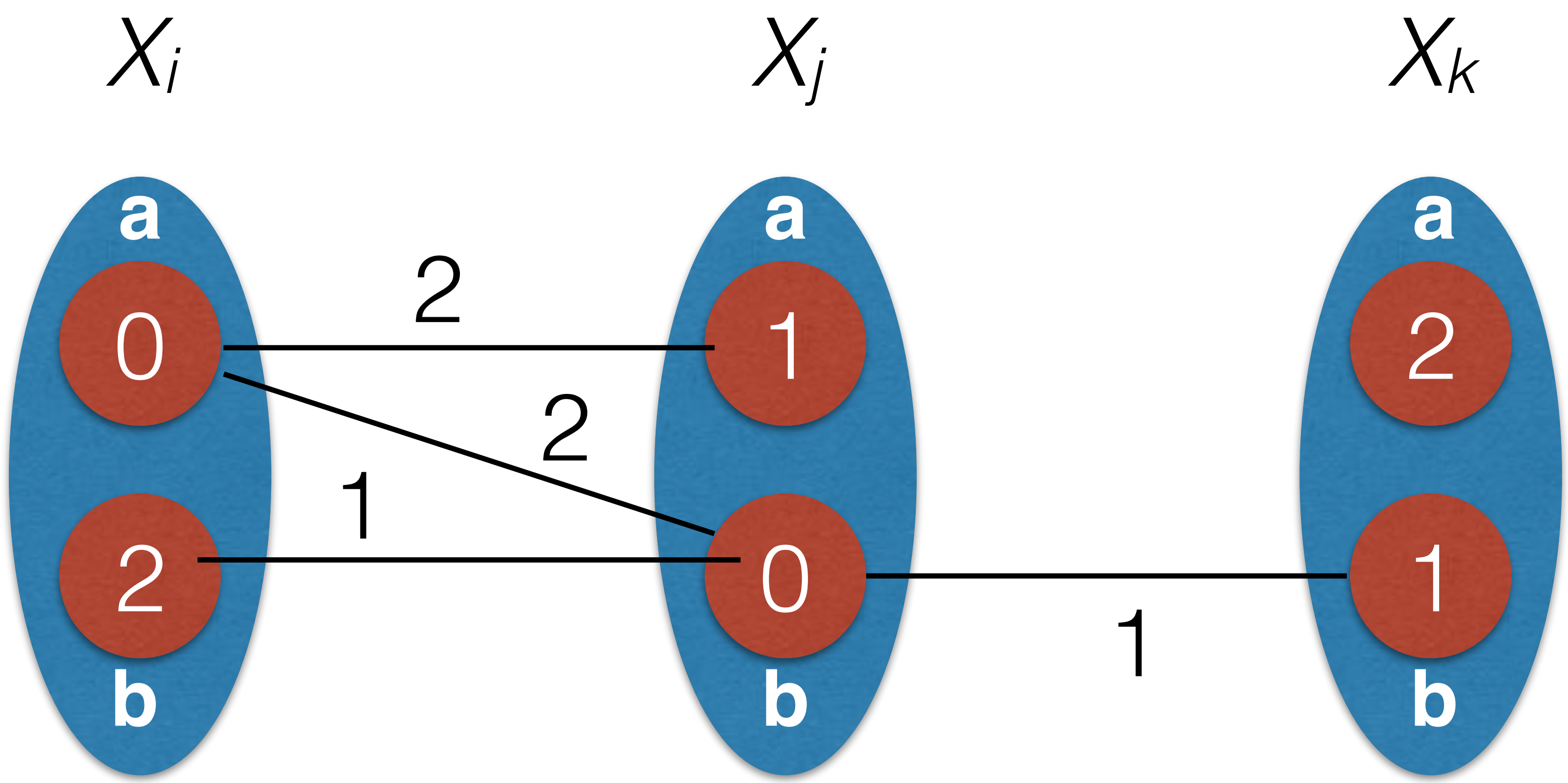
Extend/Project



k=4

f_0=0

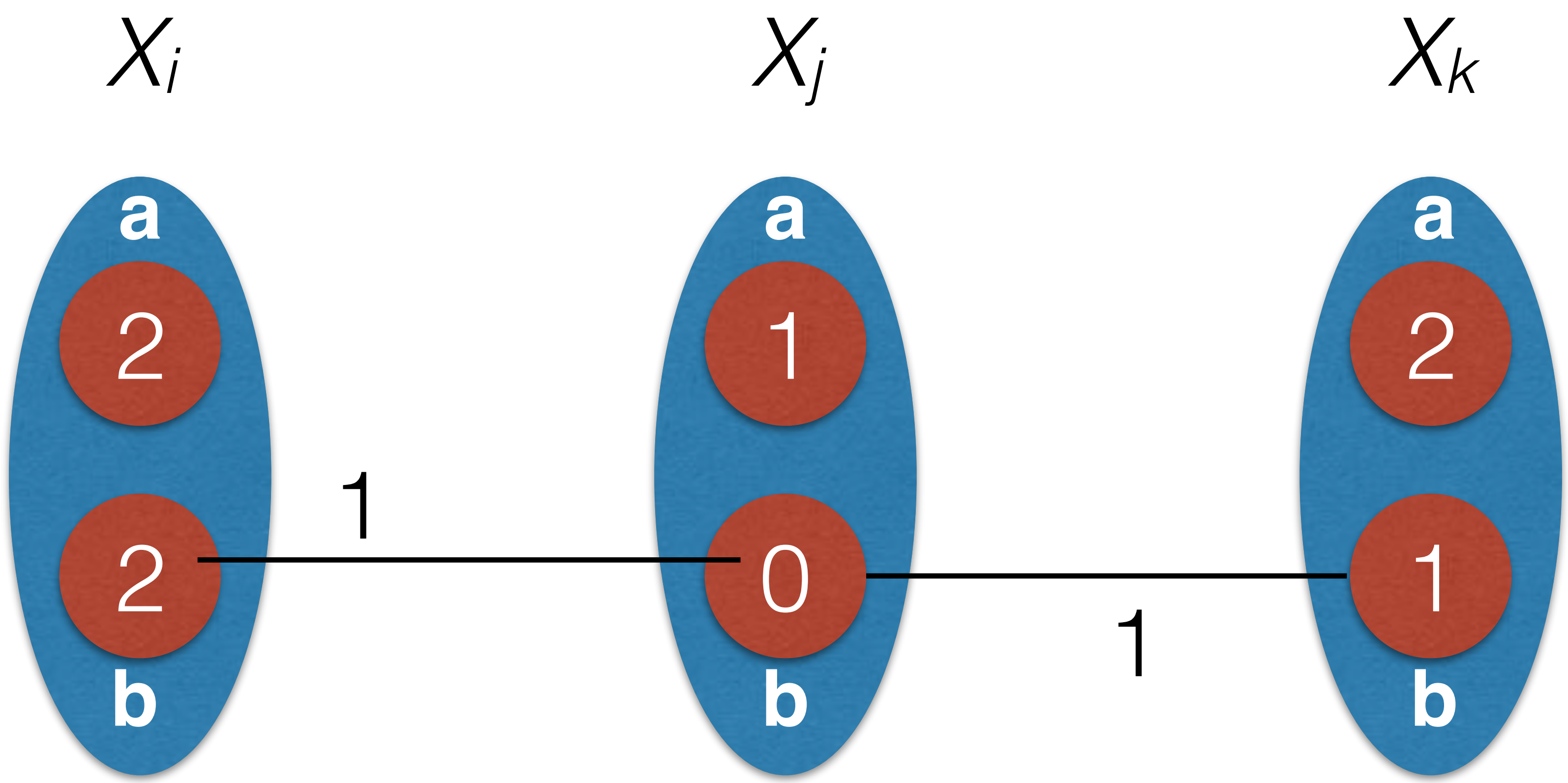
Extend/Project



k=4

f_0=0

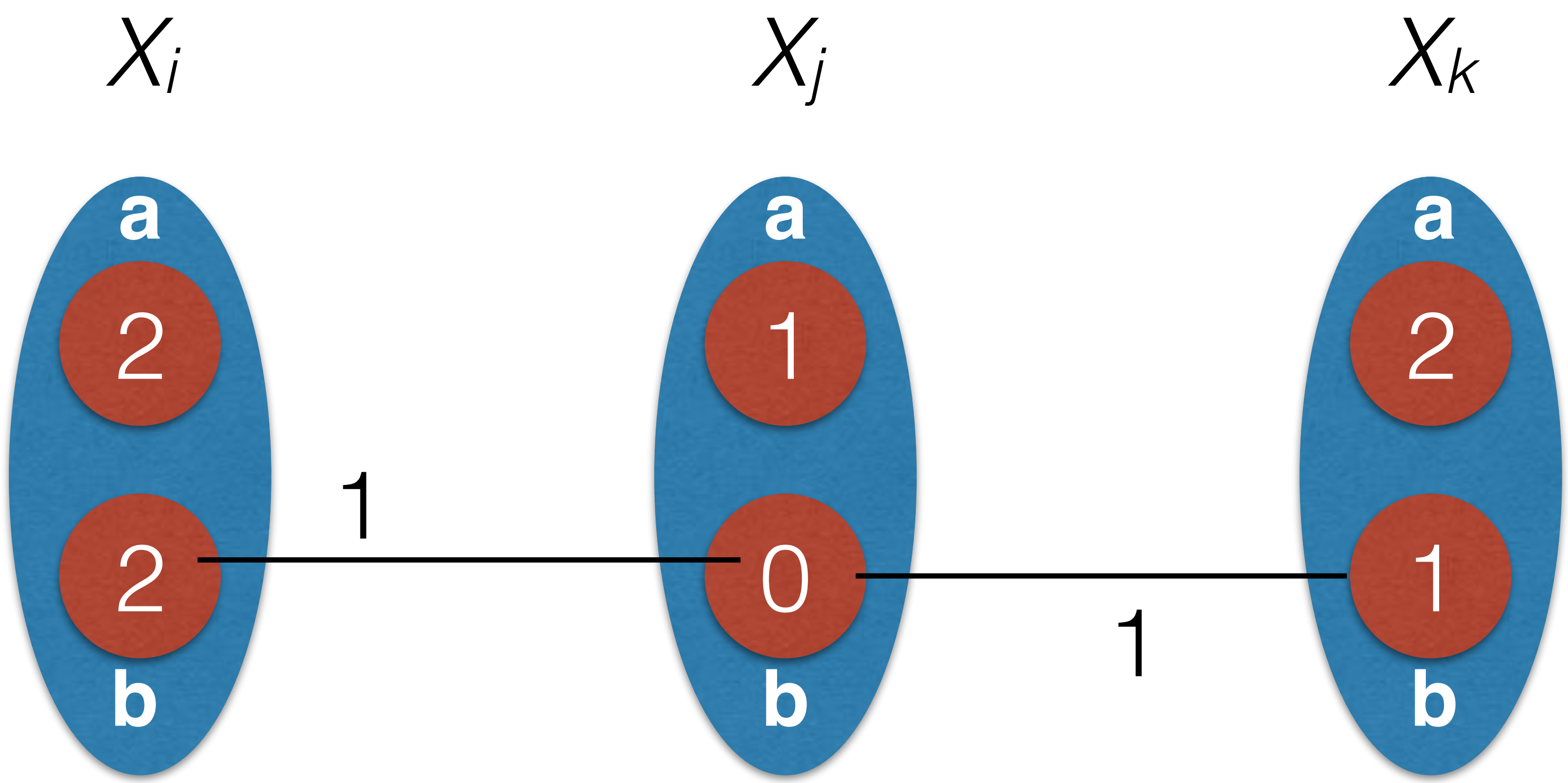
Extend/Project



k=4

f_0=0

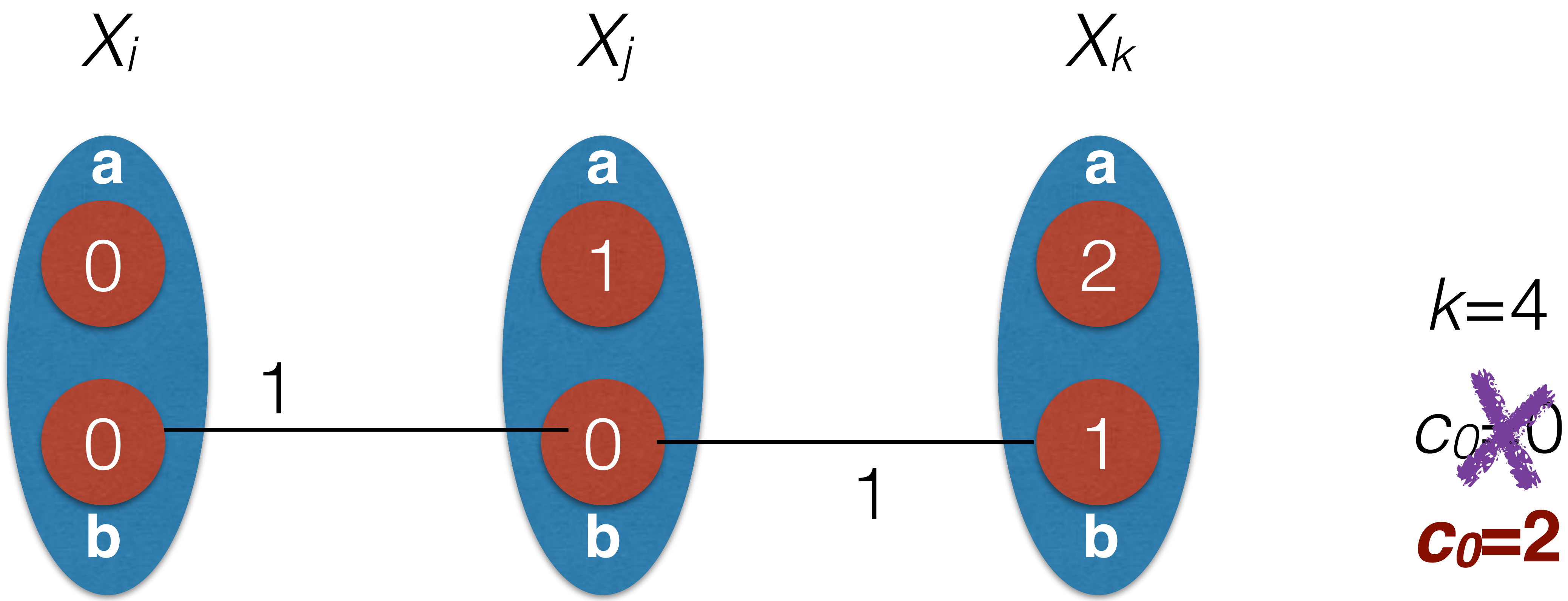
Extend/Project



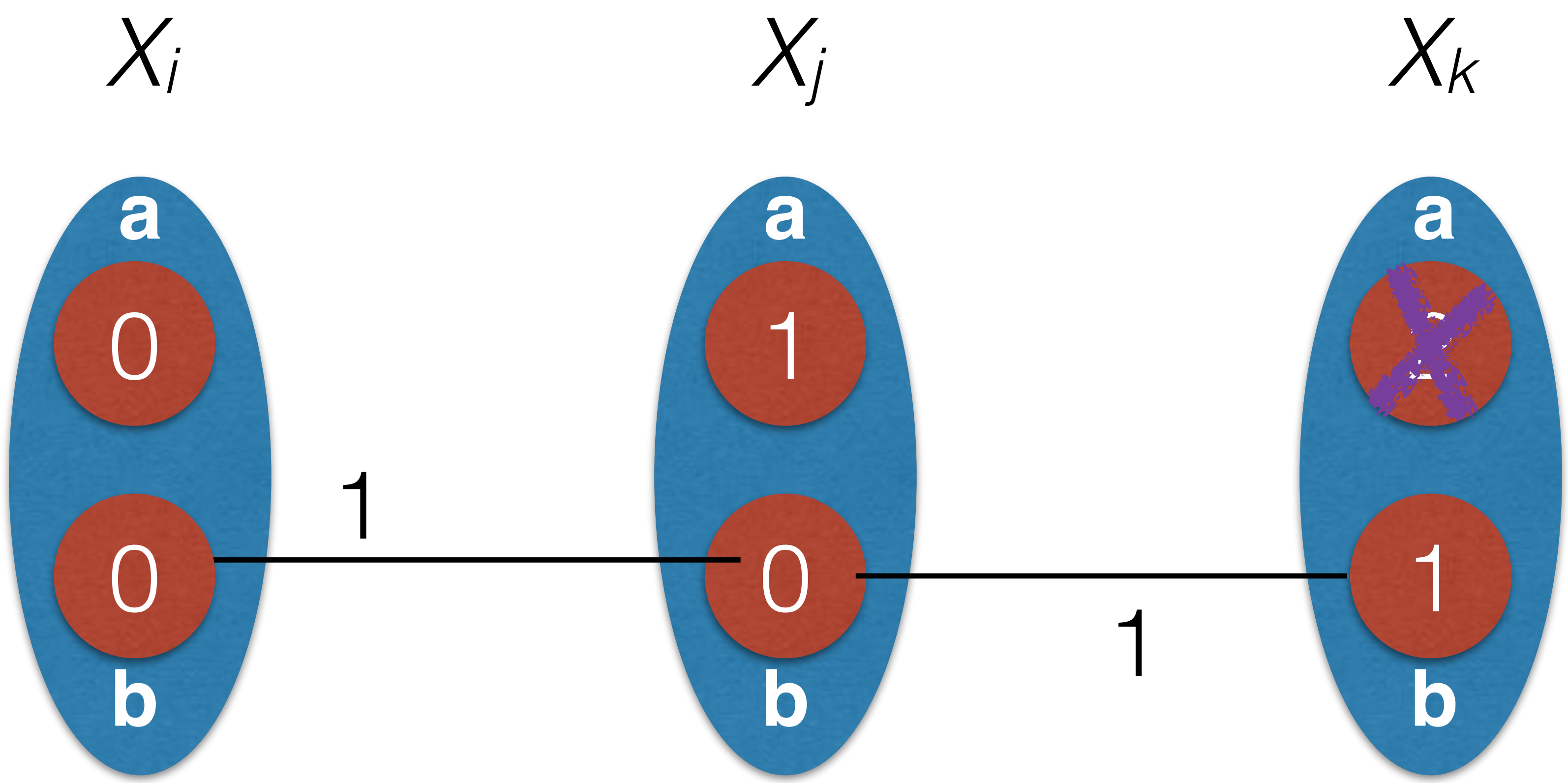
k=4

c_0=0

Extend/Project



Extend/Project

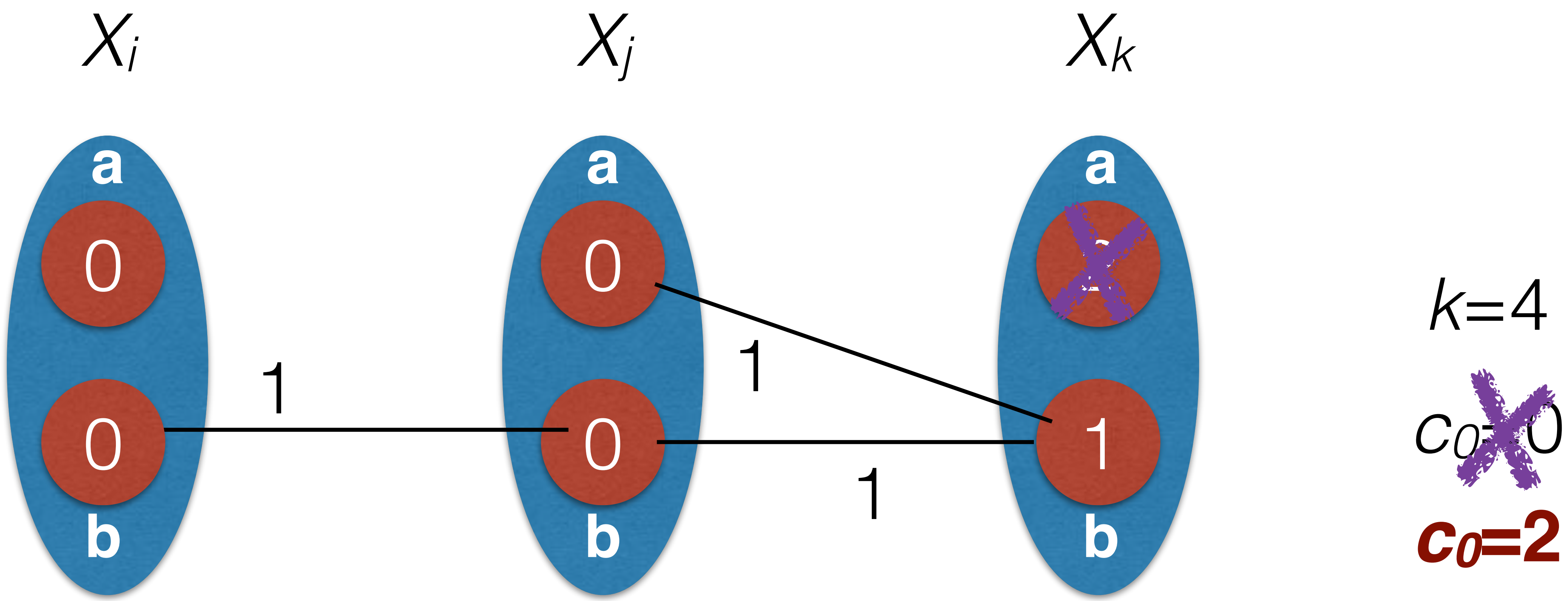


$k=4$

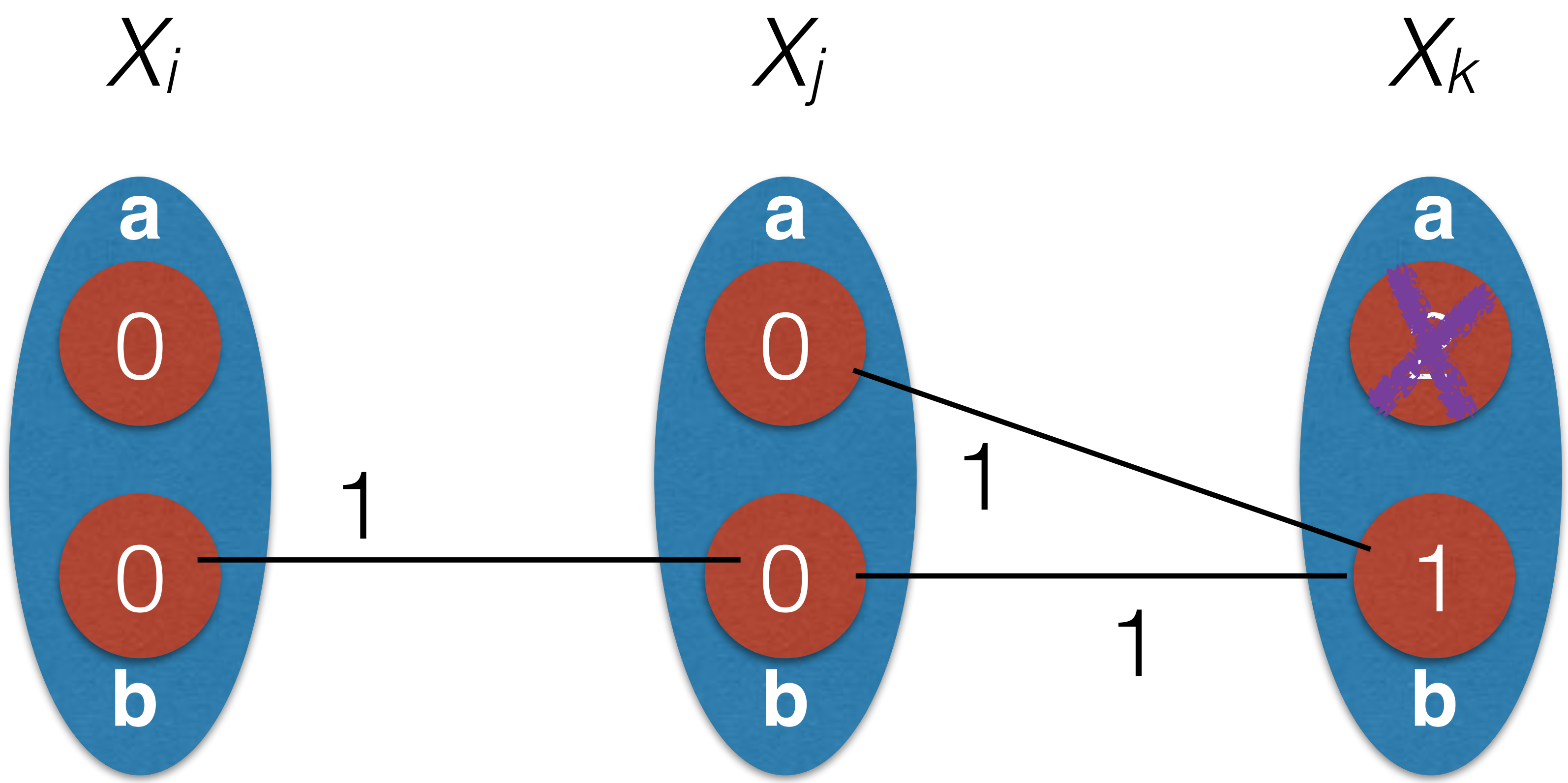
$\cancel{c_0=0}$

$\boldsymbol{c_0=2}$

Extend/Project



Extend/Project

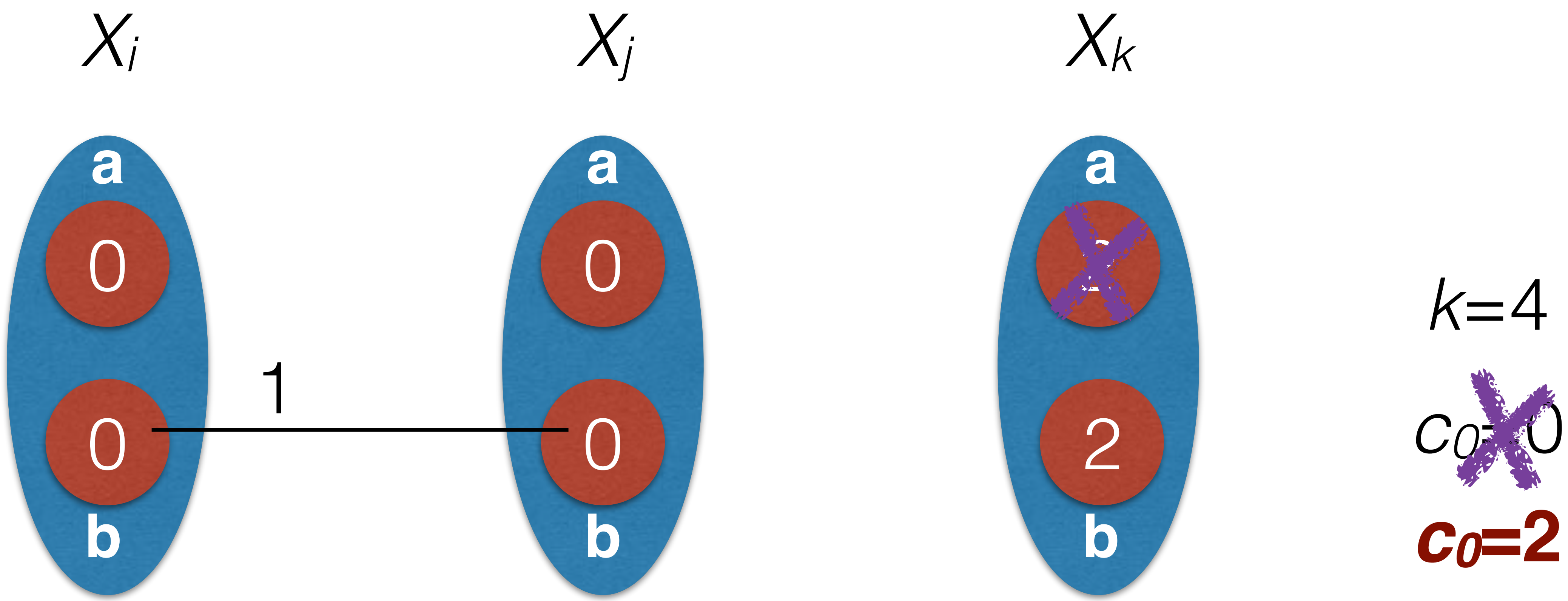


k=4

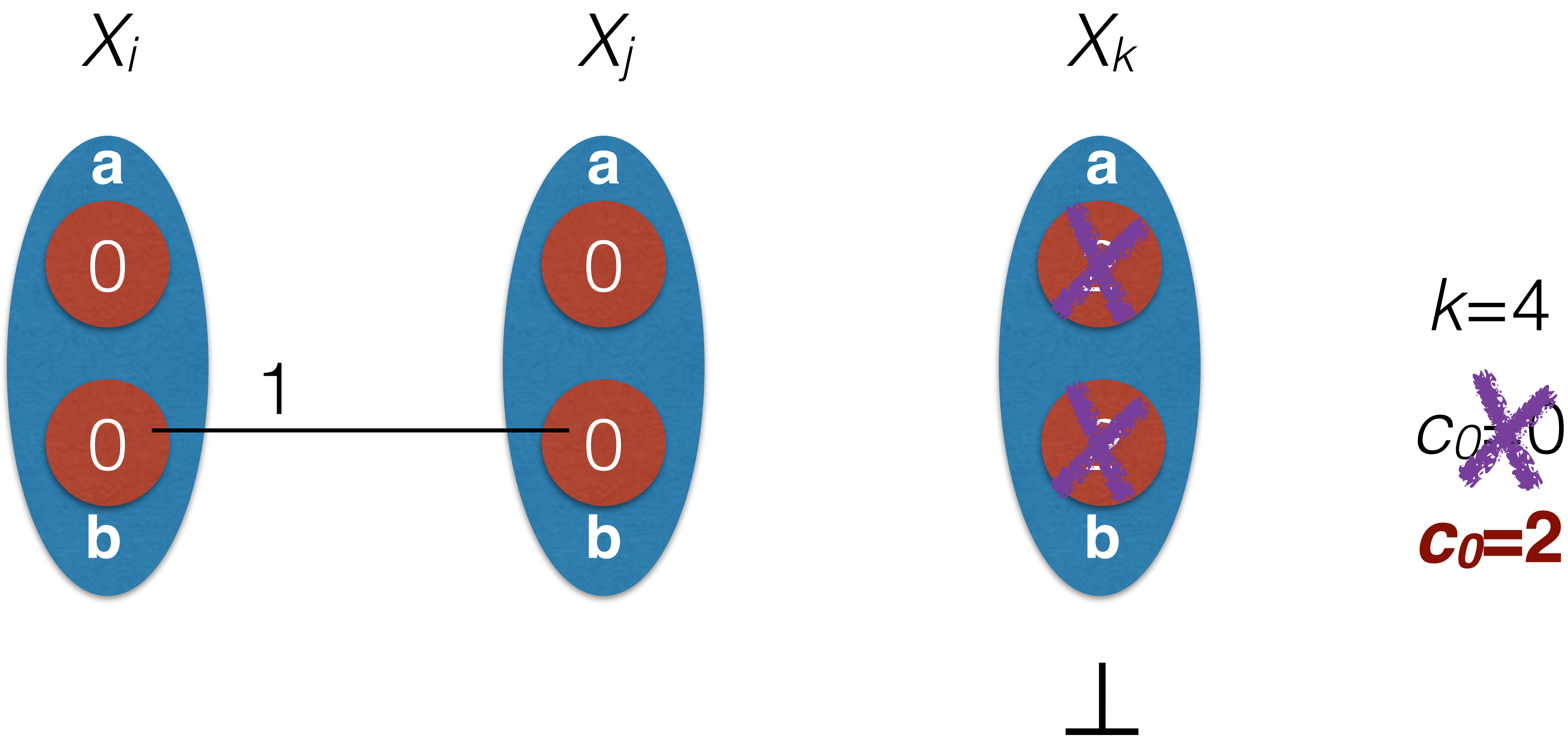
$\cancel{c_0=0}$

c_0=2

Extend/Project



Extend/Project



Un peu de modeling

Dual models

Project assignment

- A set S of students and a set P of projects. Each student must specify the projects she would like to do
- Each project will be assigned to a single student
- Each project is proposed by a company
- A company cannot receive more than k students
- A set R of couples *(i,j)* of students who must be sent to companies in the same city

Model 1

- A variable X_i for each student *i* in S
- $D(X_i)$={projects *i* would like to do}
- $X_i \neq X_j$ for all *i,j*
- For each *(i,j)* in R, a constraint $c(X_i, X_j)$ ensures *i* and *j* will do projects in the same city
- How to encode the cardinality on companies??

Model 2

- A variable Y_j for each project in j in P
- D(Y_j)={0} U {students who selected project j}
- ($Y_j = 0$) or ($Y_j \neq Y_{j'}$) for all j,j'
- *atmost*[|P|-|S|][0]($Y_1,\ldots,Y_{|P|}$)
- For each set T of projects from a company, a constraint *atleast*[|T|-k][0](Y[T])
- How to encode the constraint on cities??

Dual model

- A variable X_i for each student in S **and** a variable Y_j for each project in P

- For each pair (i, j) in S x P, a ***channelling*** constraint:

  $$X_i = j \longleftrightarrow Y_j = i$$

- For each *(i,j)* in R, a constraint c(X_i, X_j) ensures i and j will do projects in the same city

- For each set T of projects from a company, a constraint *atleast*[|T|-k][0](Y[T])

Planning as satisfiability

[Kautz & Selman 1992]

- Planning is hard (Pspace-complete)

    —> bound the horizon

    —> becomes NP

    —> use SAT (or CSPs)!