Artificial Intelligence II

2013/2014 - Prof: Daniele Nardi, Joachim Hertzberg

Exercitation 6 - Roberto Capobianco

Contract Net Protocol, Distributed Constraint Optimization

Contract Net Protocol (Recap)

- *Manager* and *Suppliers* communicate via *offers*;
- Single-manager or multi-manager;
- Contract specification language:
  - Requirements, format and deadline for presentation;

Contract Net Protocol (Recap)

- *Manager* and *Suppliers* communicate via *offers*;
- Single-manager or multi-manager;
- Contract specification language:
  - Requirements, format and deadline for presentation;
- Simple and easy to implement;
- Dynamic and easily adaptable:
  - Bilateral contract between the manager(s) and the supplier;

Contract Net Protocol (Recap)

- *Manager* and *Suppliers* communicate via *offers*;
- Single-manager or multi-manager;
- Contract specification language:
  - Requirements, format and deadline for presentation;
- Simple and easy to implement;
- Dynamic and easily adaptable:
  - Bilateral contract between the manager(s) and the supplier;
- Many messages;
- Synchronization problems;
- Agent behaviour:
  - Cautious, brave or moderate agents.

CNP: Single Manager (Recap)

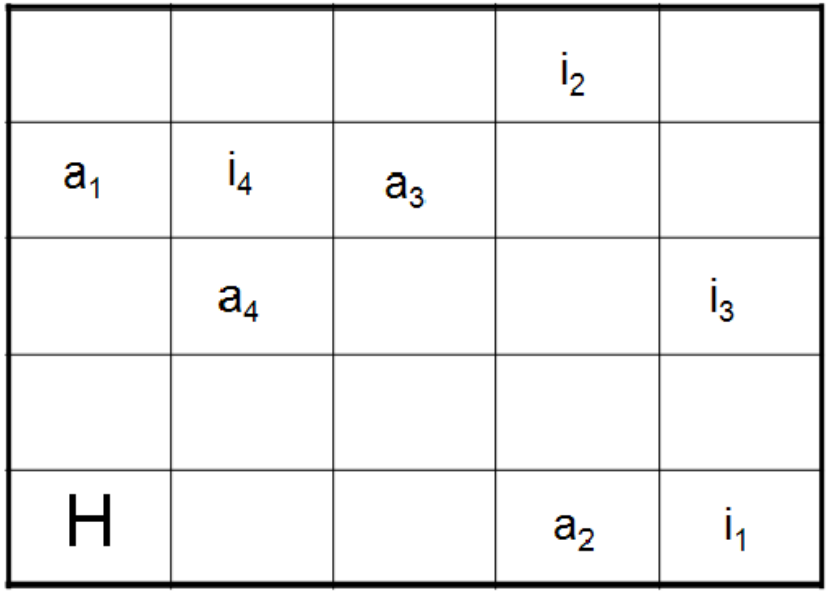
- How to get a contract:
  - *Announcement*: manager sends the task description to all the possible suppliers;
  - *Bidding*: suppliers evaluate the offer and send a proposal to the manager;
  - *Awarding*: manager evaluates the proposals and allocates the contract to the best supplier;
  - *Expediting*: the chosen supplier accepts or rejects the assignment.

CNP: Example (1)

- Consider the problem of N robotic agents willing to cook a pie:
  - They have to collect the ingredients, then meet in the home location H to prepare the pie;
  - The agents start cooperating at time 1, and have to meet in the home location at time T;
- Each agent is initially given the assignment to buy a certain ingredient;
- Each agent must make sure to complete its tasks and arrive to the home location before time T;

CNP: Example (2)

- Each agent uses Manhattan distances (in a world represented by a grid) to know the time needed to reach each destination;
- The time needed by the agents to exchange messages for coordination is irrelevant;



CNP: Example (3)

- Describe a simple coordination protocol to let the agents reach the goal of preparing the pie (having all the ingredients in the home location on time), based on CNP;
  - It is not requested that the protocol reaches always the solution if it exists, but it always terminates, either with an assignment, or with failure. If possible, show the pseudo-code executed by the agents;
- Provide a possible execution of the coordination protocol, and the final assignment of agents and ingredients, assuming that:
  - There are N=4 agents and the meeting time is at time T=10;
  - Agents and ingredients are located as shown in the figure;
  - Agent a_1 is initially in charge of buying ingredient i_1, agent a_2 of buying ingredient i_2, etc.

CNP: Example Solution (1)

- These can be the settings to reach an agreement:
  - Agents who are not able to reach home on time, due to their current assignments, start the CNP, on the most inconvenient task, in order to let it be performed by other agents (they are the managers);
  - The policy to make proposals is the Manhattan distance between the initial position and the destination;
  - An agent is not interested if that task would not let him reach home in time, and if it would reach an assignment already reached in the past;
  - Agents do not schedule their tasks when proposing (brave agents);
  - If an agent is not able to complete its tasks in time, neither to give it to someone else through CNP, declares the failure of the protocol.

CNP: Example Solution (2)

- A possible execution is:
  - a_1 will start CNP for i_1, receiving proposals <-,1,5,5> and assigning the task to a_2;
  - a_2 will start CNP for i_2, receiving proposals <-,-,2,-> and assigning the task to a_3;
  - Since a_3 is assigned with i_3 and i_2, it will start CNP for i_3; it receives proposals <-,3,3,3>, and assigns the task, for example, to a_2;
  - Since a_2 is now assigned with i_1 and i_3, it will start CNP for i_3 again. It receives proposals <-,-,-,3>, and assigns the task, to a_4;
  - Since a_4 is now assigned with i_3 and i_4, it will start CNP for i_3, receiving no answers;
  - It starts CNP for i_4 and receives proposals <1,9,1,->, and assigns the task, for example to a_1;
  - The final assignment will be $<a_1, i_4>$, $<a_2, i_1>$, $<a_3, i_2>$, $<a_4, i_1>$.

CNP: Example Solution (3)

- In terms of the retrieved solution, the reached assignment state is not optimal in general:
  - If the initial assignment was able to let each agent reach the home destination in time, but with inconvenient paths, none would start the coordination;
  - In terms of messages, each iteration of CNP will require (2+N) messages;
  - The CNP will be executed at most N time for each task, to assign different winners every time;
  - The cost in terms of number of messages will be quadratic in the number of agents;
  - An existing assignment will not always be reached by the protocol.

CNP: Exercise (1)

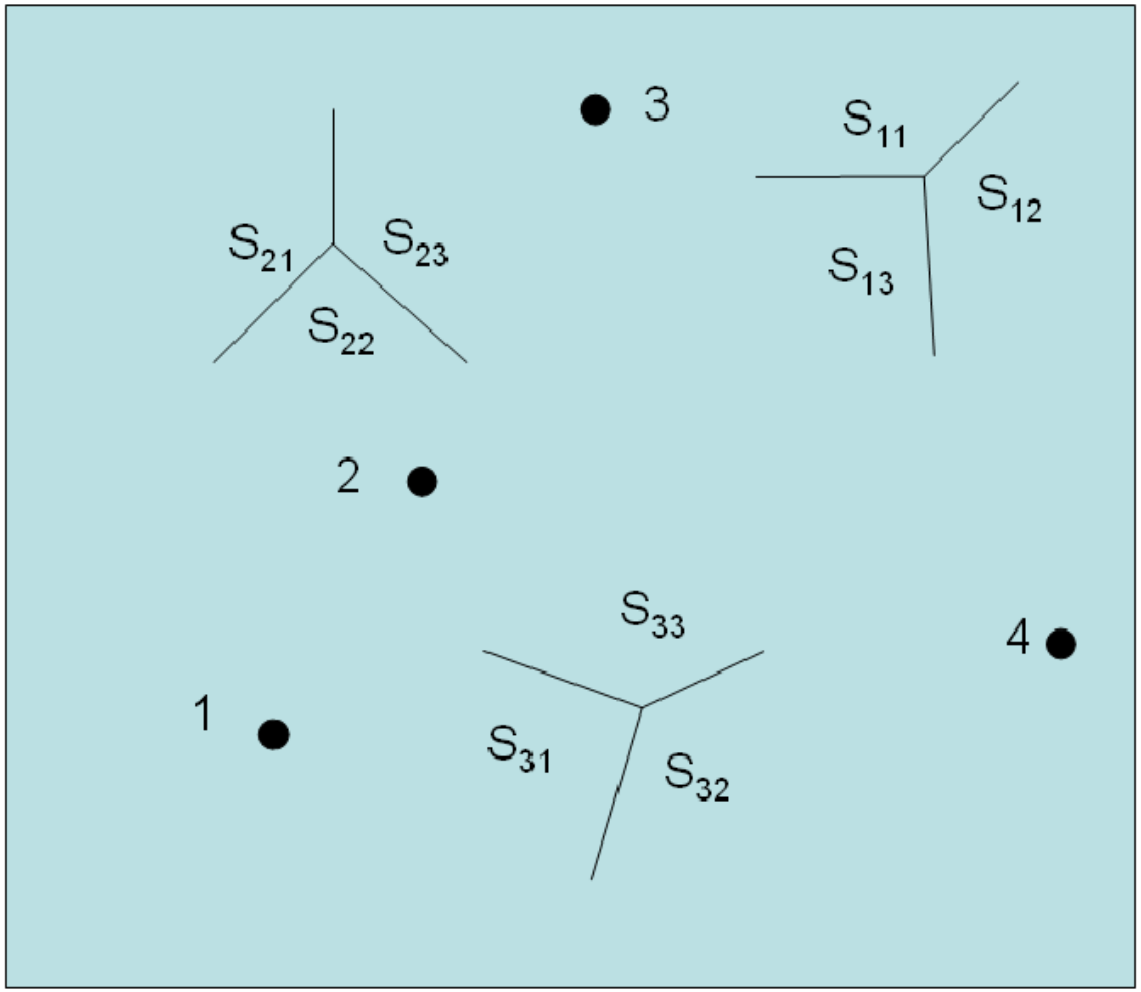
- Agents and domain: each agent controls one sensor;
- S_i may turn-on only one slot at each time instance;
- $<S_{ij}, t_k>$: the slot j of sensor i is turned-on at time t_k;
- In order to detect a vehicle, two sensors at least have to read the vehicle in max X time instances since the first detection of the vehicle in a position;
- False detections are possible: the second sensor does not detect the vehicle when it was first perceived;

- **Goal**: *verify* the maximum possible number of vehicles;
  - One agent can not accomplish the goal alone: the agents must cooperate to turn on their slots and verify the vehicle.

CNP: Exercise (2)

- Use the CNP multi-manager protocol to solve this problem, letting the agent take meeting times for vehicle verifications;
- Show the execution of the protocol starting from the initial situation $\mathrm{on}(S_{13})$, $\mathrm{on}(S_{23})$, $\mathrm{on}(S_{31})$, $t=1$;
- Study the performances of the protocol, in terms of optimality, waiting times, and number of messages needed.

CNP: Exercise – Initial Situation



CNP: Exercise – Solution (1)

- The agent who perceives the vehicle in position <x, y> becomes manager and broadcasts request for actions for undetected <x, y>;
- It will accept the earliest proposal;
- Only interested agents answer: each agent who is able to schedule a meeting time, offers the first free time slot;
- Agents who proposed a meeting time during the CNP and wins, can not refuse its proposal.

CNP: Exercise – Solution (2)

checkEndBids()
for all endBidTime(o_k) **do**
  (a_i, t_j) = selectWinner(); schedule($s(o_k), t_j$); send_accept(o_k, a_i);
  for all $a_l \neq a_i$ **do**
    send_refusal(o_k, a_l)
  end for
end for

checkMsgBox()
for all msgReceived m **do**
  if m=req($o_k, a_i, t_{bid}, [t_{min}, t_{max}]$) **then**
    send_propose($o_k, a_i, firstFree([t_{min}, t_{max}])$);
    else if m=accept(o_k, a_i, t_{task})
    schedule(s(o_k),t_{task});
  end if
end for

checkNewObjects()
s_i=chooseBestSector();
for all $o_k \in o(s_i)$ **do**
  $t_m = firstFree(t_{cur} + 2, \infty)$;
  broadcast_req($o_k, t_{bid} = t_{cur} + 1, [t_m, t_m + 3)$)
end for
if scheduled(t_{cur}) **then**
  turnOn($scheduled\_sector(t_{cur})$)
end if

CNP: Exercise – Execution

$on(S_{13}), on(S_{23}), on(S_{31}), t = 1$
A_1 $broadcast\_req(o_1, A_1, t_2, [t_3, t_6])$; $broadcast\_req(o_2, A_1, t_2, [t_3, t_6]))$;
A_2 $receives\ m = req(o_1, A_1, t_2, [t_3, t_6])) \rightarrow send\_propose(o_1, A_1, t_3)$; $receives\ m = req(o_2, A_1, t_2, [t_3, t_6])) \rightarrow send\_propose(o_2, A_1, t_3)$;
$broadcast\_req(o_3, A_2, t_2, [t_4, t_7]))$; $broadcast\_req(o_4, A_2, t_2, [t_4, t_7]))$;
A_3 $receives\ m = req(o_1, A_1, t_2, [t_3, t_6])) \rightarrow send\_propose(o_1, A_1, t_3)$;
$receives\ m = req(o_2, A_1, t_2, [t_3, t_6])) \rightarrow send\_propose(o_2, A_1, t_4)$;
$receives\ m = req(o_3, A_2, t_2, [t_4, t_7])) \rightarrow send\_propose(o_3, A_2, t_4)$;
$receives\ m = req(o_4, A_2, t_2, [t_4, t_7])) \rightarrow send\_propose(o_4, A_2, t_5)$;
$t = 2$
A_1 $endBidTime(o_1) \rightarrow$
$send\_accept(o_1, A_2, t_3)$; $send\_refusal(o_1, A_3)$; $schedule(S_{13}, t_3)$
$endBidTime(o_2) \rightarrow$
$send\_accept(o_2, A_2, t_3)$; $send\_refusal(o_2, A_3)$
$receives\ m = req(o_3, A_2, t_2, [t_4, t_7])) \rightarrow send\_propose(o_3, A_2, t_4)$;
$receives\ m = req(o_4, A_2, t_2, [t_4, t_7])) \rightarrow send\_propose(o_4, A_2, t_5)$;
A_2 $endBidTime(o_3) \rightarrow$
$send\_accept(o_3, A_3, t_4)$; $send\_refusal(o_3, A_1)$; $schedule(S_{23}, t_4)$
$endBidTime(o_4) \rightarrow$
$send\_accept(o_4, A_3, t_5)$; $send\_refusal(o_4, A_1)$; $schedule(S_{23}, t_5)$
$receives\ m = accept(o_1, A_2, t_3)\ and\ m = accept(o_2, A_2, t_3) \rightarrow schedule(S_{22}, t_3)$
A_3 $receives\ m = accept(o_3, A_3, t_4) \rightarrow schedule(S_{33}, t_4)$
$receives\ m = accept(o_4, A_3, t_5) \rightarrow schedule(S_{32}, t_5)$

	A_1	A_2	A_3
t=1			
t=2			
t=3	S_{13}	S_{22}	
t=4		S_{23}	S_{33}
t=5		S_{23}	S_{32}

CNP: Exercise - Analysis

$$num_{msg} \leq num_{task} \times (3 \times num_{ag}) = o(num_{task} \times num_{ag})$$
$$T_{tot} = (bidtime + 2) \times T_{msg} + T_{execTask}$$

- The retrieved solution is not optimal (trivial);

Distributed Constraint Optimization (Recap)

- DCOP = Constraint Network + Agents
- Each agent:
  - Controls a subset of the variables (tipically just one);
  - Is only aware of constraints that involve the variables it controls;
  - Communicate only with neighbours;
- Solution quality (the higher the better):
  - Optimality not always achievable,
- Coordination Overhead (the lower the better):
  - Computation effort (time complexity);
  - Number and size of messages (network load);
- Desirable properties (hard to quantify)
  - Robustness to failures, parallelism, flexibility, privacy maintenance, etc.

DCOP Solution Techniques (Recap)

- Complete approaches:
  - Guarantee optimal solution;
  - Exponential coordination overhead;
  - ADOPT, DPOP, OptAPO;
- Heuristics
  - Low coordination overhead;
  - No guarantees on optimality;
  - DSA, MGM, Max-Sum;
- Approximate approaches
  - Low coordination overhead;
  - Optimality guarantees;
  - Bounded max-sum, k-optimality.

DPOP (Recap)

- Partial order based on a DFS search;
- Linear number of messages;
- Exponential message size (in width of DFS search tree);
- DFS-tree width tipically much less than number of agents;
- Token passing;
- Utility propagation:
  - Compile information to compute optimal value;
  - Util messages from leaves to root;
- Value Propagation:
  - Root chooses optimal value and propagate decision;
  - Value messages from root to leaves;

DFS-Tree Building (Recap)

- Traverse the graph using a recursive procedure;
- Each time we reach X_i from X_j we mark X_i as visited and state that X_j is the father of X_i (and X_i is a children of X_j);
- When a node X_i has a visited neighbour that is not its parent we state that X_j is a pseudo-parent of X_i (and X_i is a pseudochildren of X_j);
- Can be done with a distributed procedure:
  - Each node need only to communicate with neighbours;
  - Token passing to propagate information (e.g., visited nodes);

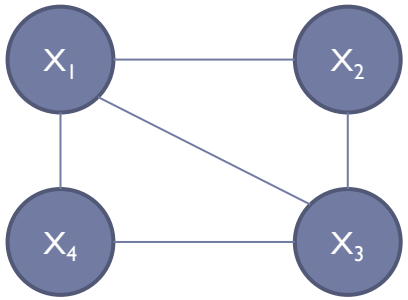
- Maximum Connected node heuristic:
  - Choose node with max number of neighbours as root;
  - Select the neighbour with the highest number of neighbours;
  - Brake ties arbitrarily;

DPOP: Example

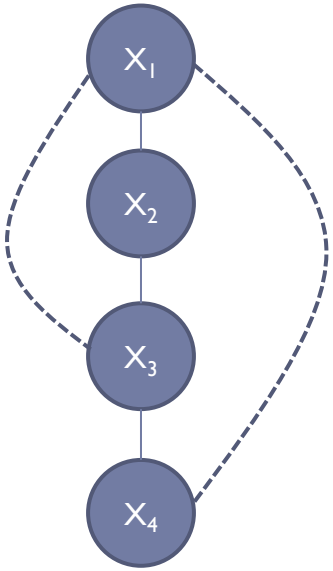
- Consider the following binary constraint network:
  - $X = \{x_1; x_2; x_3; x_4\}$;
  - $D = \{d_1 = d_2 = d_3 = d_4 = \{Red; Blue\}\}$;
  - $C = \{<x_1; x_2>; <x_1; x_3>; <x_1; x_4>; <x_2; x_3>; <x_3; x_4>\}$;
- Assume that every node in the graph is controlled by one agent: answer the following questions:
  - Compute the size of the biggest message for DPOP with the following pseudo-tree ordering: $o_1 = <x_1; x_2; x_3; x_4>$ (measure the size as the number of values sent in each message);
  - Show the pseudo-tree resulting from a DFS search using the Maximum Connected node heuristic, starting from node x_1;
  - Compute the size of the biggest message for DPOP with this pseudo-tree ordering;
  - Compute the total number of messages that DPOP would use to solve this problem: in general, is this number dependent on the pseudo-tree used?

DPOP: Example – Solution (1)

Constraint Graph



DFS-Tree



Separator(X1) = {}

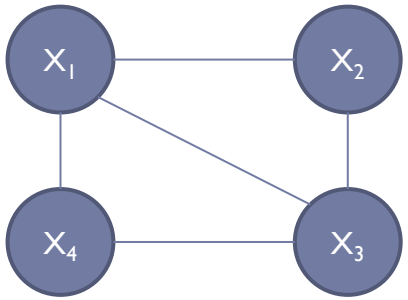
Separator(X2) = {X1}

Separator(X3) = {X1, X2}

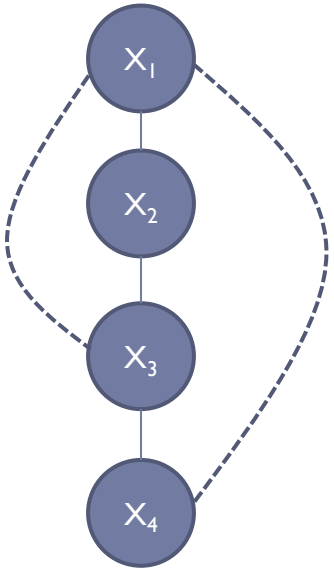
Separator(X4) = {X1, X3}

DPOP: Example – Solution (2)

Constraint Graph

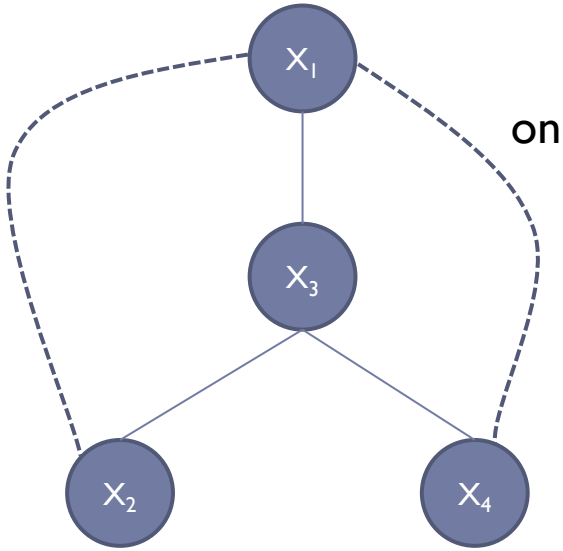


DFS-Tree



- Biggest message d^s, where d is the size of the domain and s is the size of the biggest separator;
- In this case: d=2, s=2 → biggest message size = 4.

DPOP: Example – Solution (3)



DFS-Tree with MCN;
one possible order: $<x_1;\ x_3;\ x_2;\ x_4>$

Separator(X1) = {}
Separator(X2) = {X1, X3}
Separator(X3) = {X1}
Separator(X4) = {X1, X3}

Biggest message: same as before.

DPOP: Example – Solution (4)

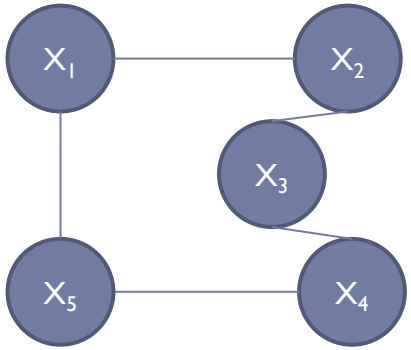
- The total number of messages is 2L, where L is the number of tree edges;
- In this case: 2L = 2*3 = 6;

DPOP: Exercise

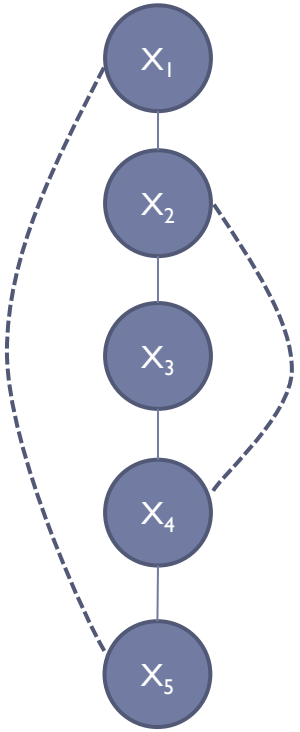
- Consider the following binary constraint network
  - $X = \{x_1; x_2; x_3; x_4; x_5\}$;
  - $C = \{<x_1; x_2>; <x_1; x_5>; <x_2; x_3>; <x_2; x_4>; <x_3; x_4>; <x_4; x_5>\}$;
- Assume that every node in the graph is controlled by one agent: answer the following questions:
  - Compute the separators of every node with the following pseudo-tree ordering: $o_1 = <x_1;\ x_2;\ x_3\ ;\ x_4\ ;\ x_5>$;
  - Is it possible to find a DFS-tree ordering for this network so that the size of the largest separator is smaller?
  - If so give such a DFS-tree and compute the separators for each node. Otherwise motivate your answer.

DPOP: Exercise – Solution (Part of)

Constraint Graph



DFS-Tree



Separator(X1) = {}
Separator(X2) = {X1}
Separator(X3) = {X1, X2}
Separator(X4) = {X1, X2, X3}
Separator(X5) = {X1, X4}

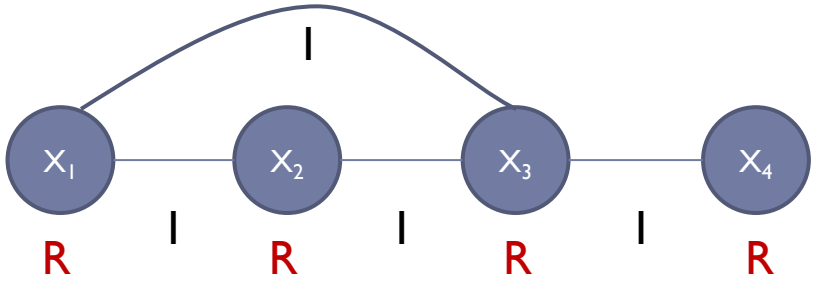
Max separator size: 3

K-optimality: Example (1)

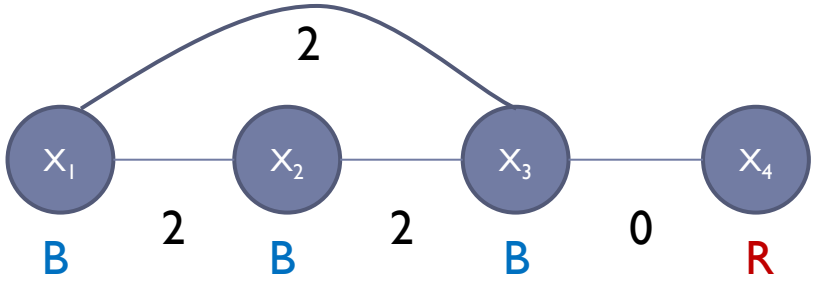
- Consider the following constraint network:
  - $X = \{x_1; x_2; x_3; x_4\}$;
  - $D = \{d_1 = d_2 = d_3 = d_4 = \{Red; Blue\}\}$;
  - $C = \{f(x_1; x_2); f(x_1; x_3); f(x_2; x_3); f(x_3; x_4)\}$;

  where $f(x_i; x_j) = \{(<R; R>; 1); (<R; B>; 0); (<B; R>; 0); (<B; B>; 2)\}$;
- We are dealing with a maximization task: answer the following questions:
  - Give a 2-Optimal solution which is not 3-optimal. Motivate your solution, showing why it is 2-optimal but not 3-optimal;
  - Can you find a 1-Optimal solution which is not 2-Optimal for this network?

K-Optimality: Example – Solution (1)

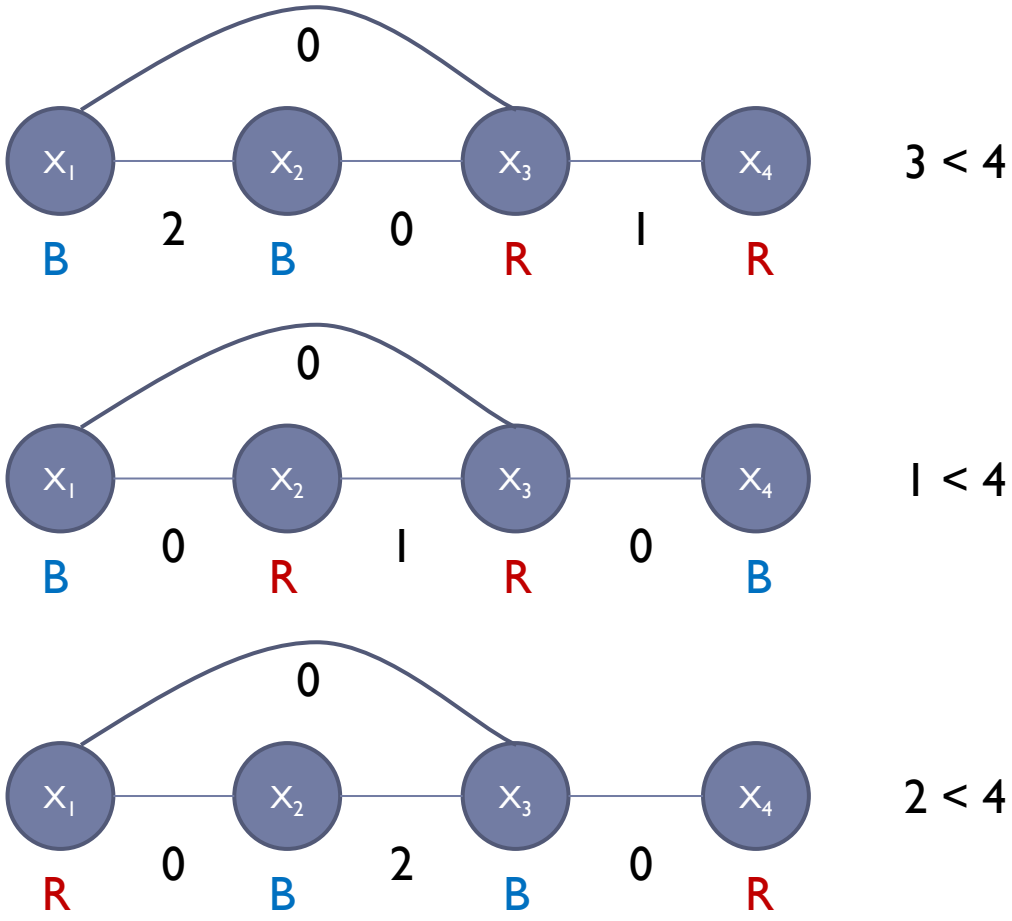


- Value of the objective function: 4;
- This is not 3-optimal: 6 > 4;



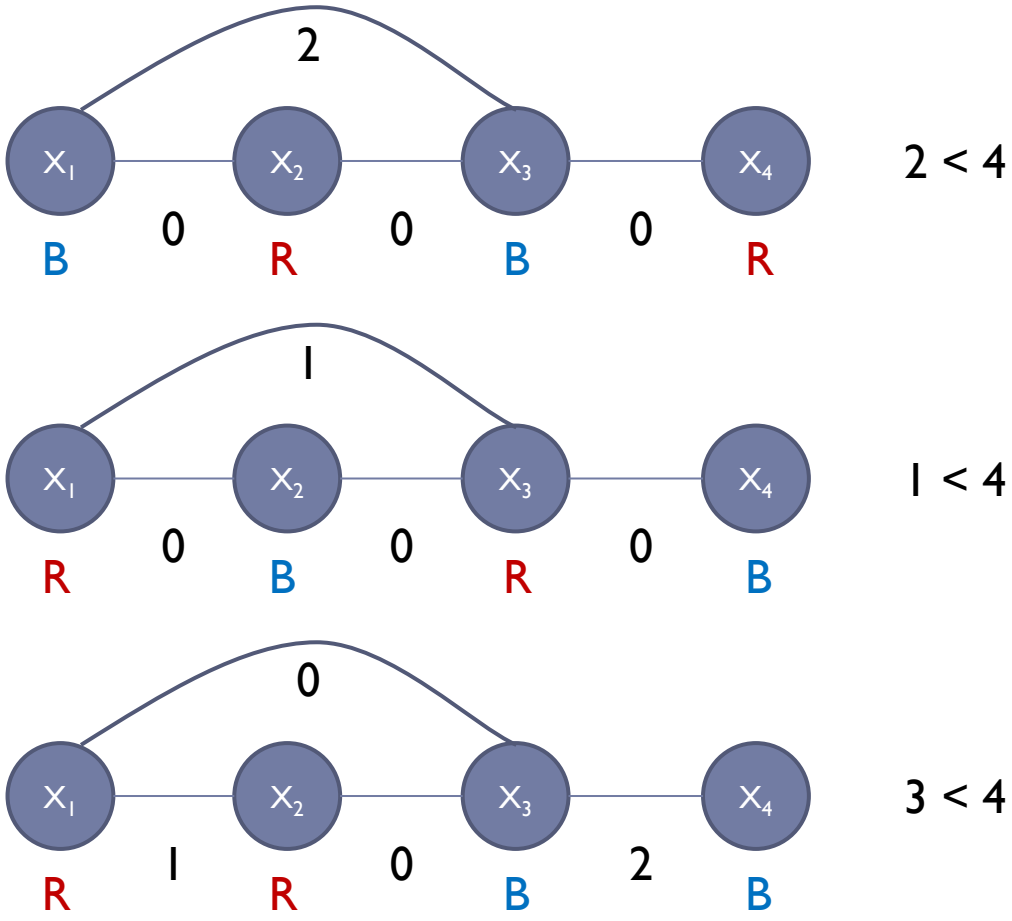
K-Optimality: Example – Solution (2)

- This is 2-optimal:



K-Optimality: Example – Solution (3)

- This is 2-optimal:



K-Optimality: Example – Solution (4)

- It is not possible to have a 1-optimal solution which is not 2-optimal. In fact, the only 1-optimal solution is RRRR, that is also 2-optimal;
- RRRR is the only 1-optimal solution because, setting x_1, x_2 or x_3 to B, it is better to switch also another one to B. Setting x_4 to B, instead, it is always better to switch it to R.

Thank you!