Artificial Intelligence II
Multi-Agent Systems

Working together:
Distributed Constraint Optimization
(Complete approaches)

Alessandro Farinelli, Luca Iocchi, Daniele Nardi

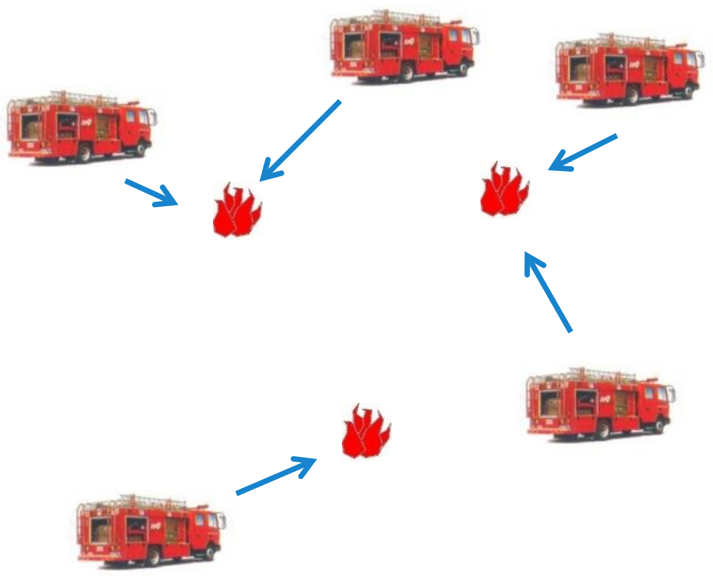
Outline

- Introduction
  - DCOP for MAS
  - how to model problems in the DCOP framework
- Solution Techniques for DCOPs
  - Exact algorithms (DCSP, DCOP)
    - DPOP
  - Approximate Algorithms (without/with quality guarantees)
    - DSA, MGM, Max-Sum, k-optimality, bounded max-sum

Working together

Coordination problem:

Choose agent's individual actions so to maximise a system-wide objective (benevolent agents)



Task allocation:

individual actions: which fire to tackle

system-wide objective: minimise total extinguish time

solution: a joint action

Decentralised Coordination

- Decentralised coordination: Local decision with local information
- Why Decentralised coordination ?
  - In general no benefit for computation or solution quality
  - Robustness
    - avoid single point of failure
  - Scalability
    - Not enough bandwidth to communicate/process all information
  - Leads to problem decomposition
    - Each agent cares only of local neighbours

Decentralized Coordination: example

Issues:

No central controller:
  distributed knowledge

Information sharing:
  limited communication

Complex system:
  difficult to design/analyse

Decentralized Coordination: issues

- No central controller: distributed knowledge
- Information sharing: limited communication
- Complex system: difficult to design/analyse

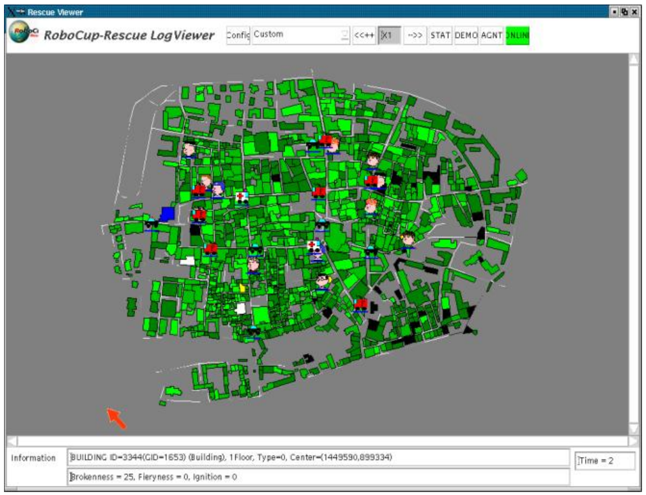
DCOPs for Decentralized Coordination

Why DCOPs for decentralized coordination ?

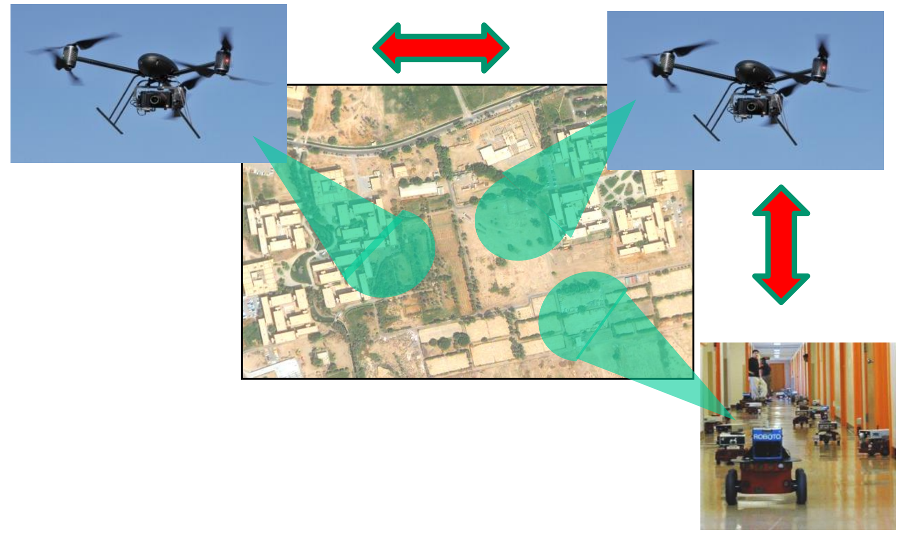
- Well defined problem
  - Clear mathematical formulation that captures most important aspects
  - Many solution techniques
    - Optimal: ABT, ADOPT, DPOP, ...
    - Approximate: DSA, MGM, Max-Sum, ...
- Solution techniques can handle large problems
  - compared for example to CNP and other sequential dec. making (MDP, POMDP)

Reference Applications

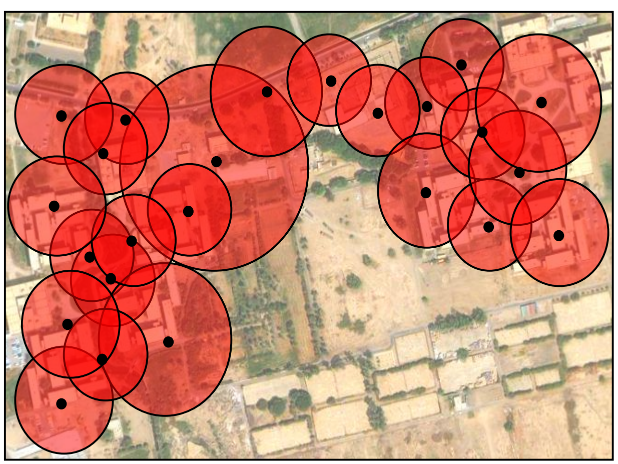
Incident Management



Cooperative Exploration



Environment monitoring

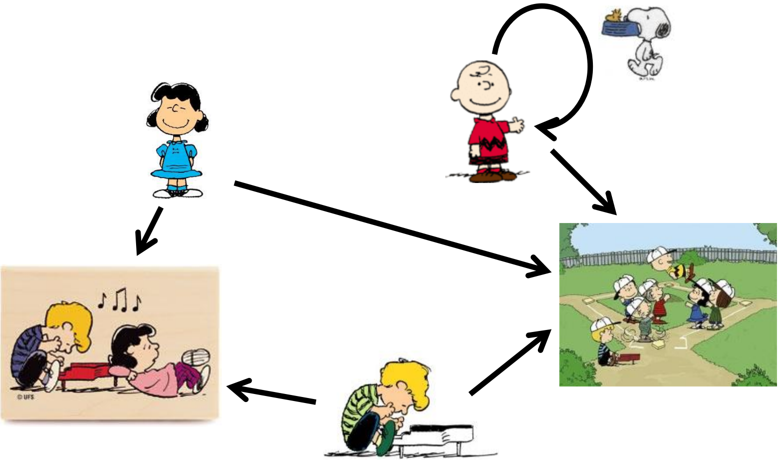


Energy management

Modeling Problems as DCOP

- Surveillance

- Meeting Scheduling

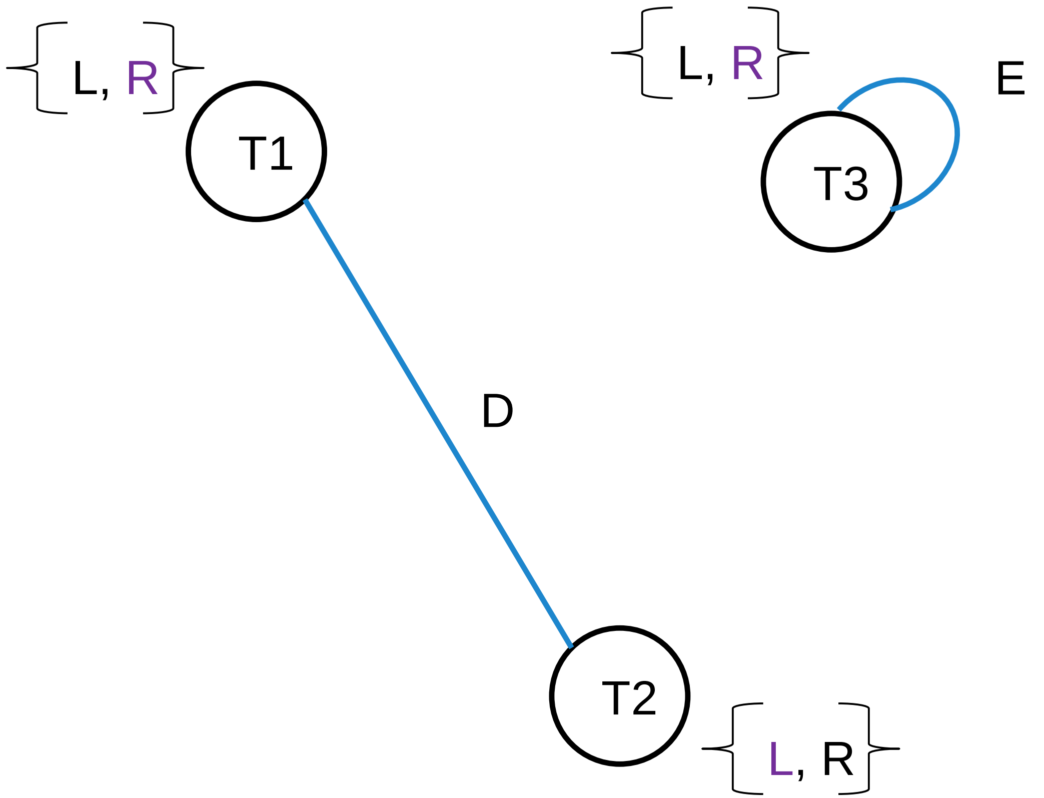


Target Tracking

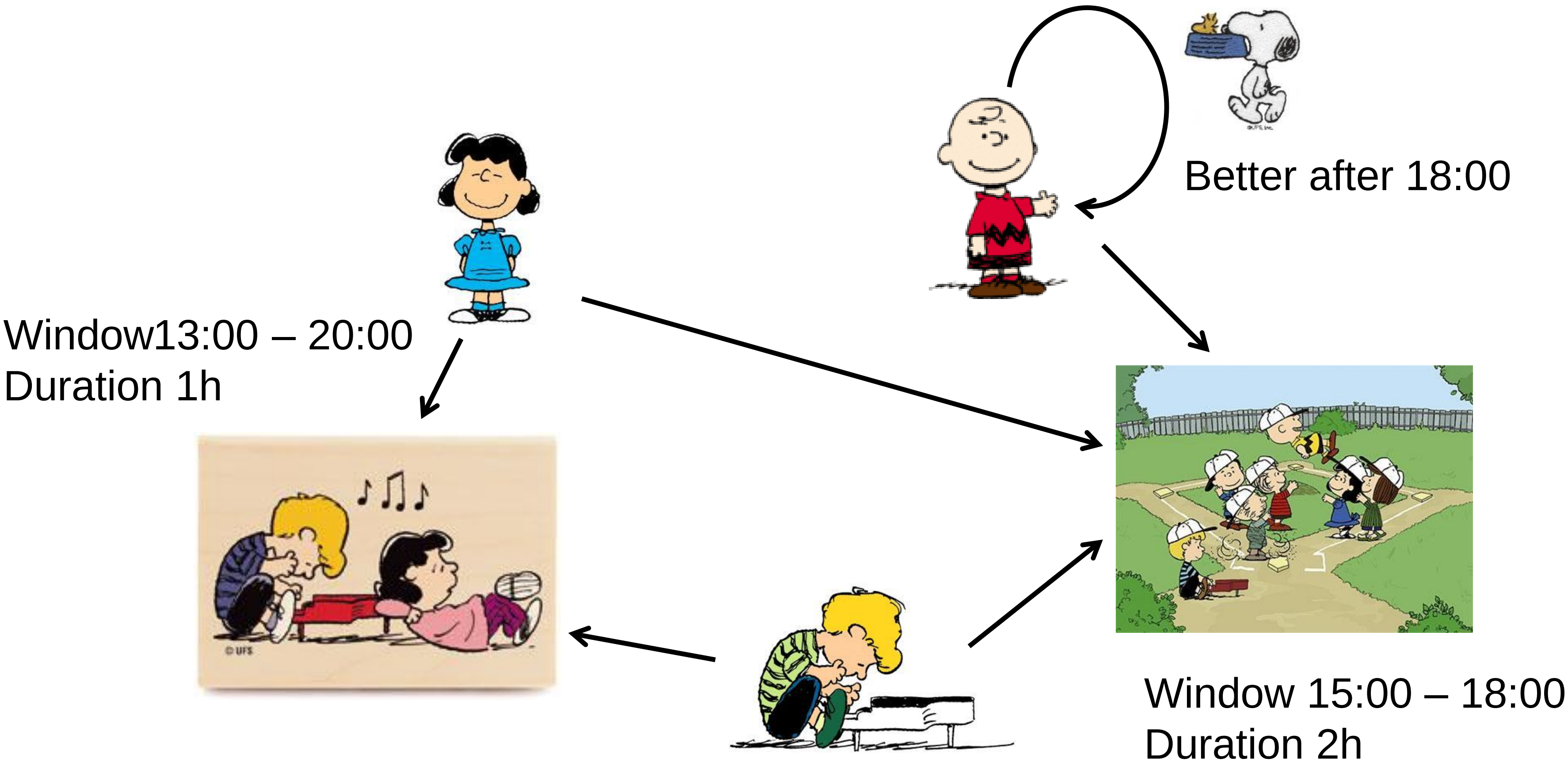
- Why decentralize
  - Robustness to failure and message loss

Target Tracking - DCOP

- Variables -> Cameras
- Domains -> Camera actions
  - look left, look right
- Constraints
  - One constraint per target
  - Better assessment with more cameras
- Maximise “observation”

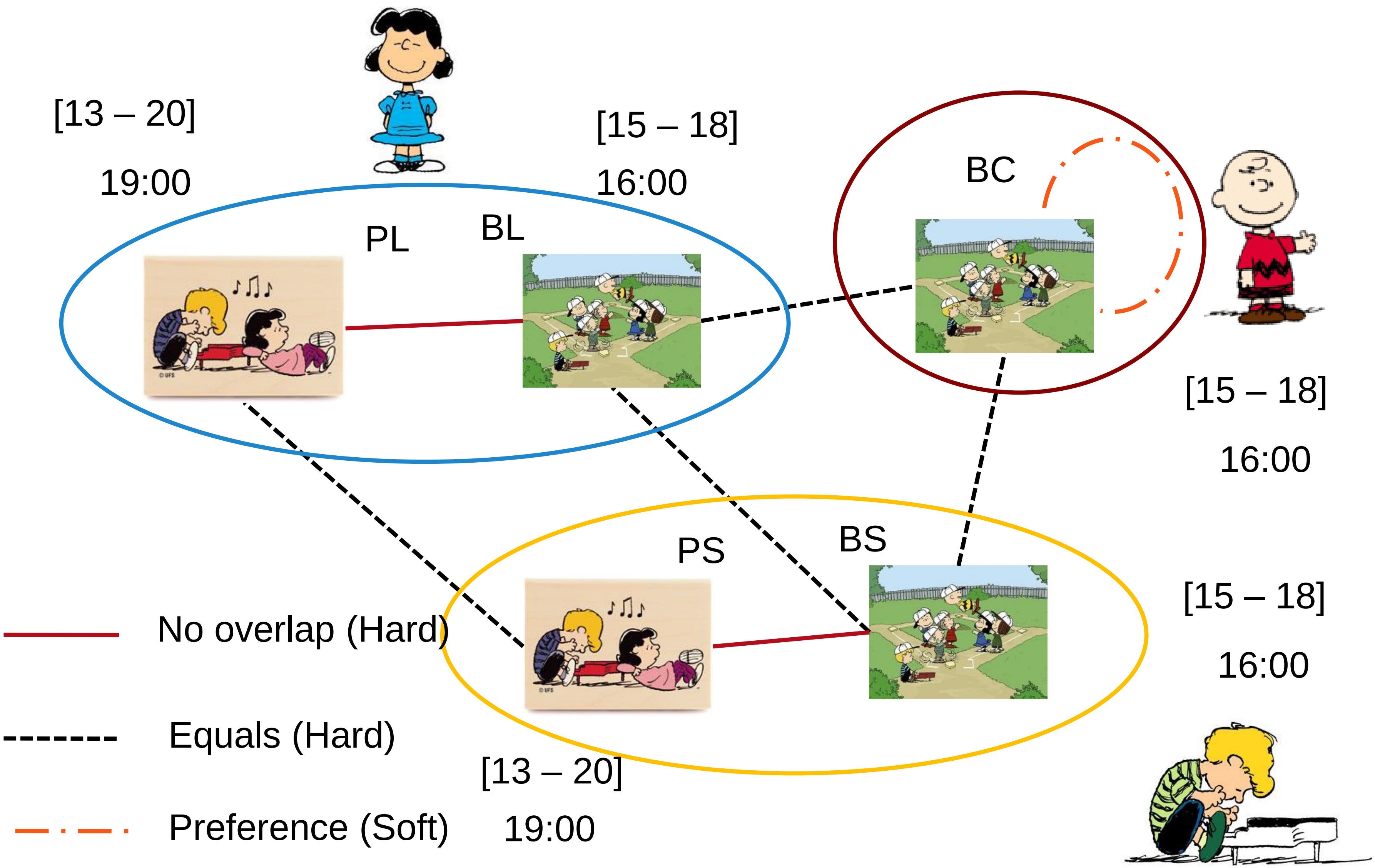


Meeting Scheduling



- Why decentralize
  - Privacy

Meeting Scheduling - DCOP



Constraint Networks

$X = \{X_1, \ldots, X_n\}$ a set of variables (e.g. meetings)

$D = \{D_1, \ldots, D_n\}$ a set of discrete variable domains (e.g. time slots)

$C = \{C_1, \ldots, C_m\}$ a set of constraints (e.g., equality, non overlap,)

$S_i \subseteq X$ Scope of constraint C_i

Hard constraints

R_i	x_j	x_k
	0	1
	1	0

Soft constraint

F_i	x_j	x_k
2	0	0
0	0	1
0	1	0
1	1	1

Objectives for constraint networks

- Constraint Satisfaction Problem (**CSP**)
  - Objective: find an assignment for all the variables in the network that satisfies all constraints
- Constraint Optimization Problems (**COP**)
  - Objective: find an assignment for all the variables in the network that satisfies all constraints and optimizes a global objective function

$$X^* = \arg\max_X \left(\sum_i F_i(X_i) \right)$$

Global function: an aggregation (i.e., sum) of local functions $F_i(X_i)$

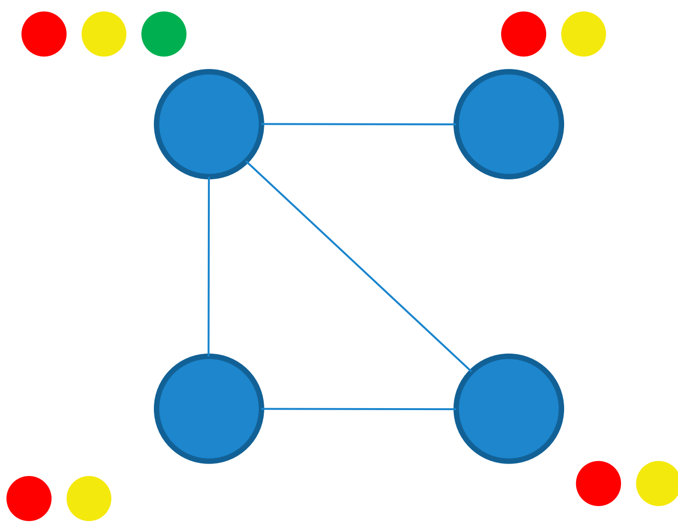
Benchmarking problems

- Motivations
  - Analysis of complexity and optimality is not enough
  - Need to empirically evaluate algorithms on the same problem
- Graph coloring
  - Simple to formalise very hard to solve
  - Well known parameters that influence complexity
    - Number of nodes, number of colors, density (number of link/number of nodes)
  - Many versions of the problem
    - CSP, MaxCSP, COP

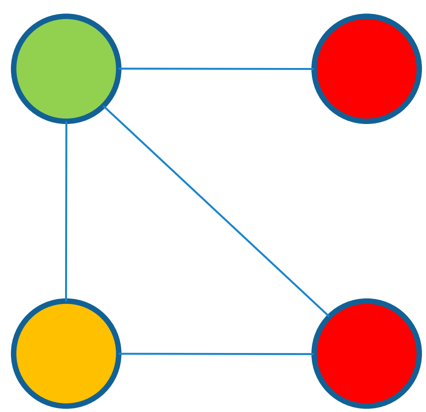
Graph Coloring

- Network of nodes
- Nodes can take on various colors
- Adjacent nodes should not have the same color
  - If it happens this is a conflict

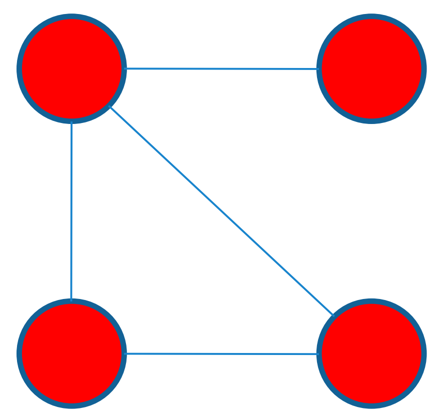
CSP



Yes

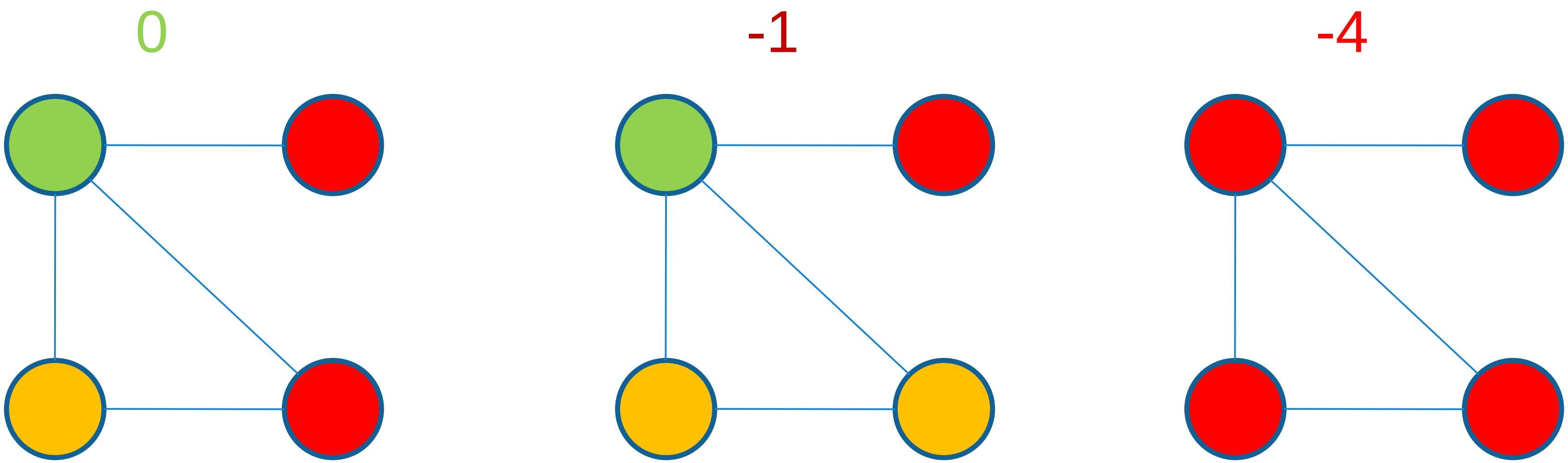


No



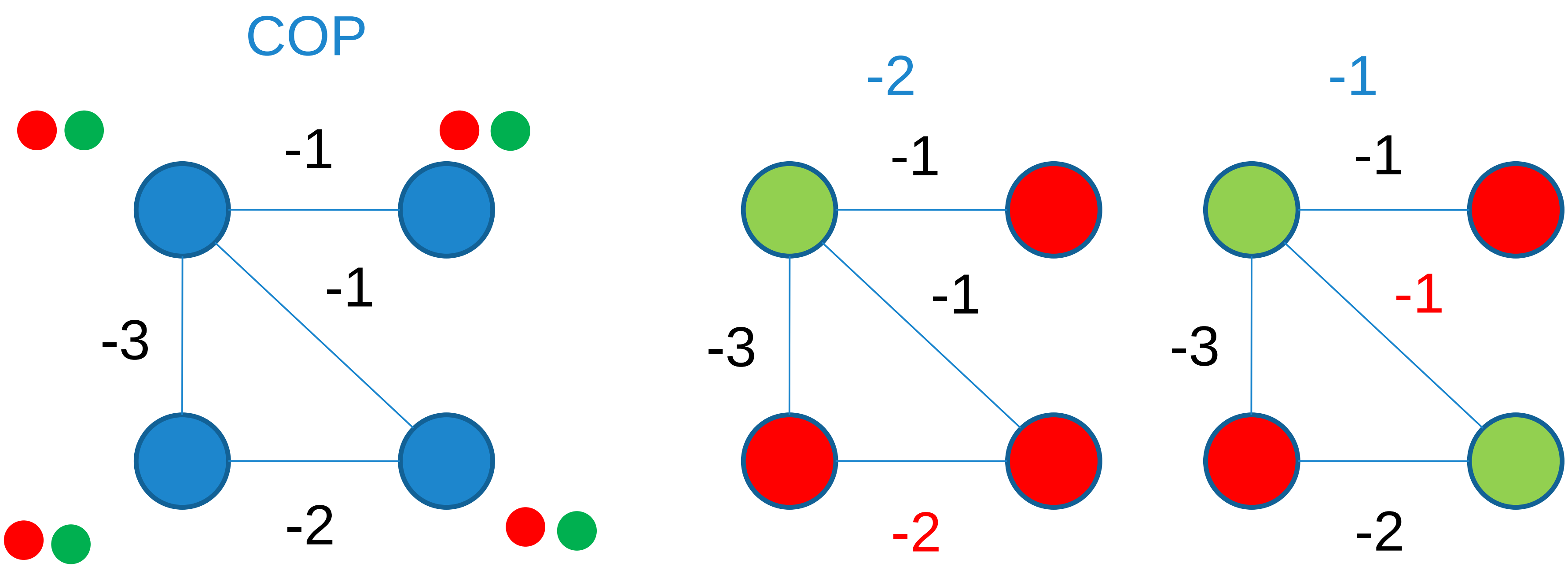
Graph Coloring - MaxCSP

- Optimization Problem
- Natural extension of CSP
- Minimise number of conflicts



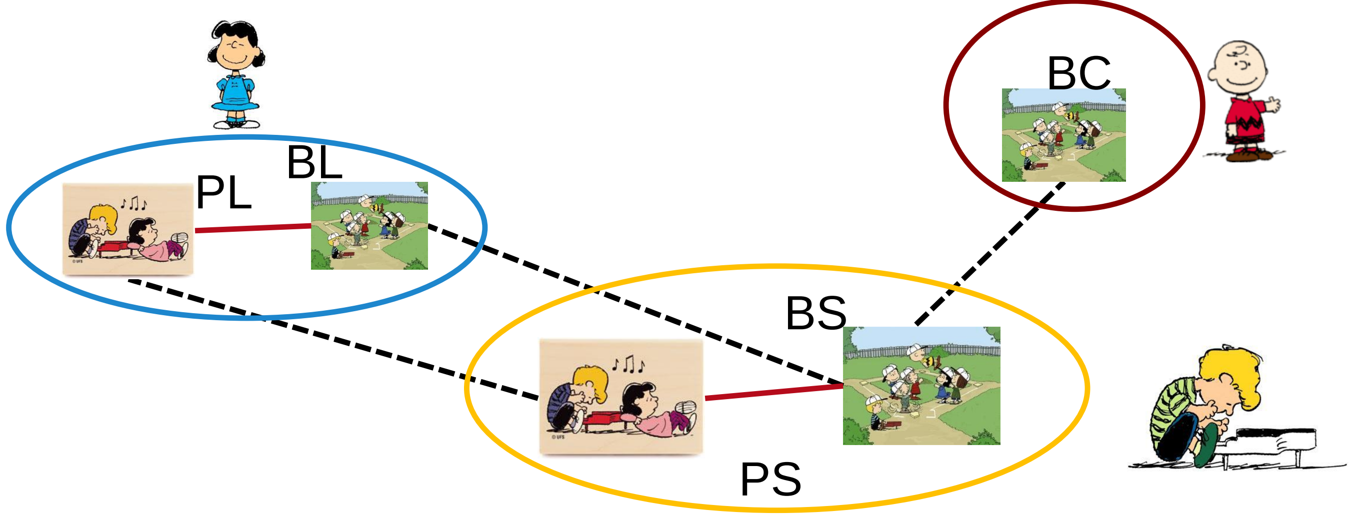
Weighted Graph Coloring - COP

- Optimization Problem
- Conflicts have a weight
- Maximise the sum of weights of violated constraints



Distributed COP

- We focus on optimization
- DCOP = Constraint Network + Agents $A = \{A_1, \ldots, A_k\}$
- Where each agent:
  - Controls a subset of the variables (typically just one)
  - Is only aware of constraints that involve the variables it controls
  - Communicates only with neighbors



Performance measures

- Solution quality (the higher the better)
  - Optimality not always achievable,
  - Optimality Guarantees
- Coordination Overhead (the lower the better)
  - Computation: computation effort (time complexity)
  - Communication: number and **size** of messages (network load)
- Desirable properties (hard to quantify)
  - Robustness to failures, parallelism, flexibility, privacy maintenance, etc.

DCOP Solution techniques

- 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

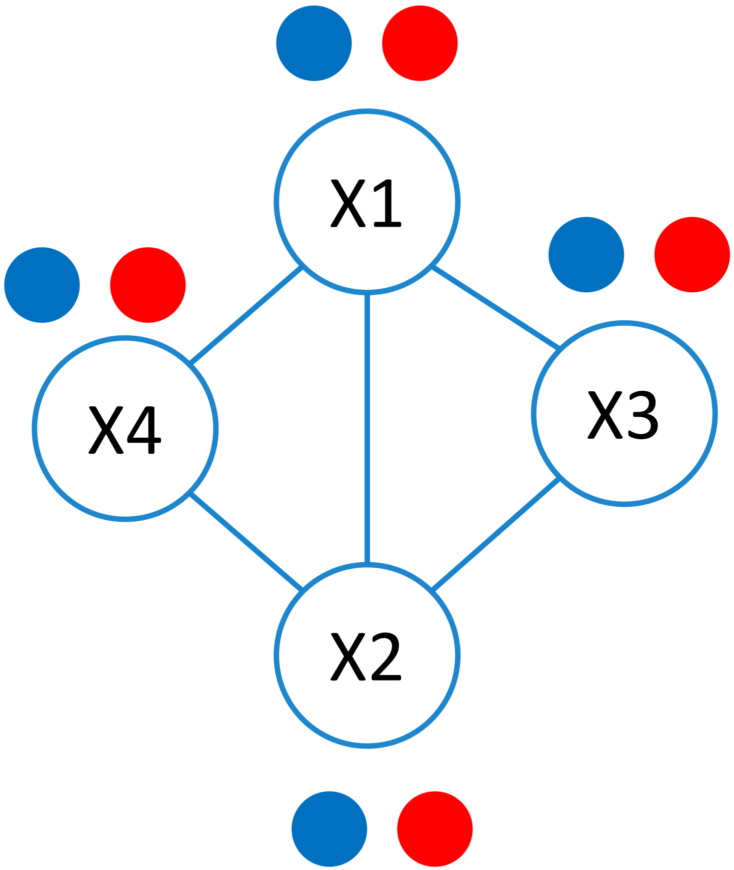
Complete Approaches

- ADOPT (Search based) [Modi et al 05]
  - Distributed branch and bound
  - Partial order based on a DFS search
  - Asynchronous (high parallelism, flexible)
  - Number of messages exponential in number of agents
- DPOP (Dynamic programming) [Petcu and Faltings 07]
  - Distributed Bucket Elimination
  - Partial order based on a DFS search
  - Linear number of messages
  - Exponential message size (in width of DFS search tree)
  - DFS-tree width typically much less than number of agents

Dynamic Programming Optimization Protocol

- DFS-tree building (special case of Pseudo tree)
  - Constraint graph → DFS-Tree
  - 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

DPOP complete example

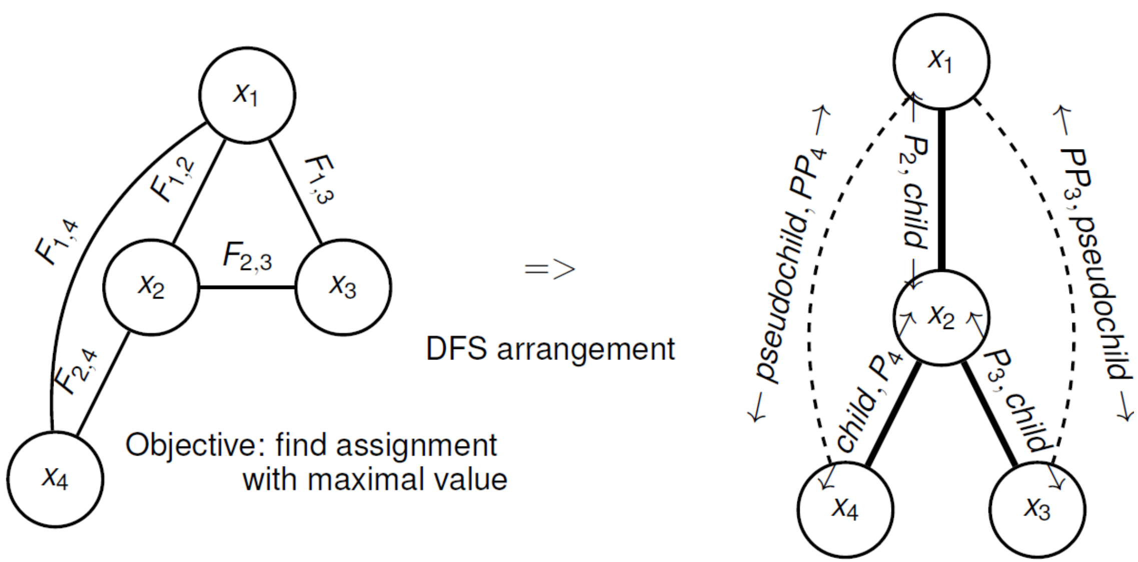


Value of Each constraint:
same color -1
different colors 0

Solve this constraint network with DPOP. Assume one agent per variable

Pseudotrees: basic concepts

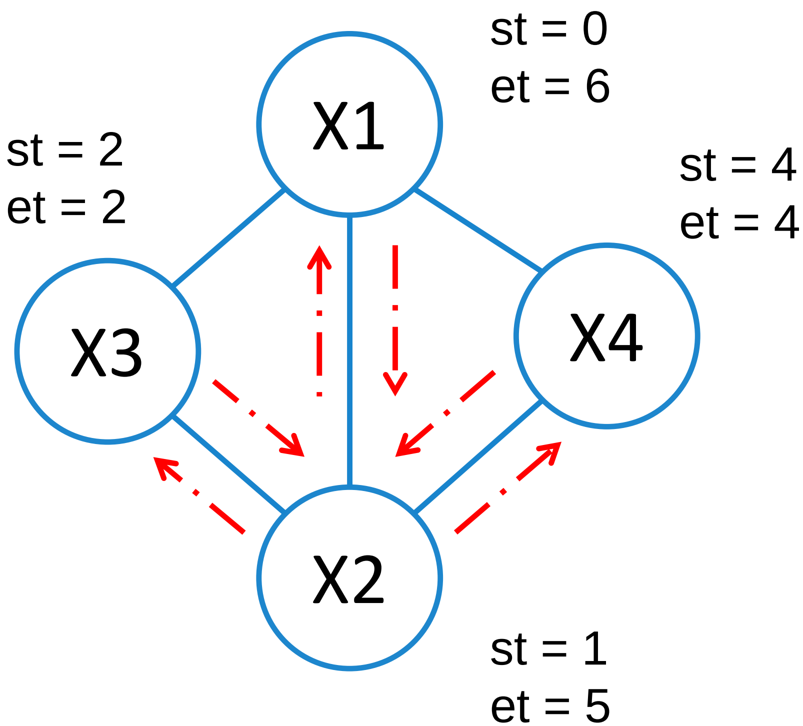
- Pseudotree arrangement of a graph G
  - A rooted tree with the same nodes as G
  - Adjacent nodes in G fall in the same branch of the tree
    - Nodes in different branches do not share direct constraints
  - A DFS visit of a graph induces a Pseudotree
    - Not every pseudotree can be obtained with a DFS visit

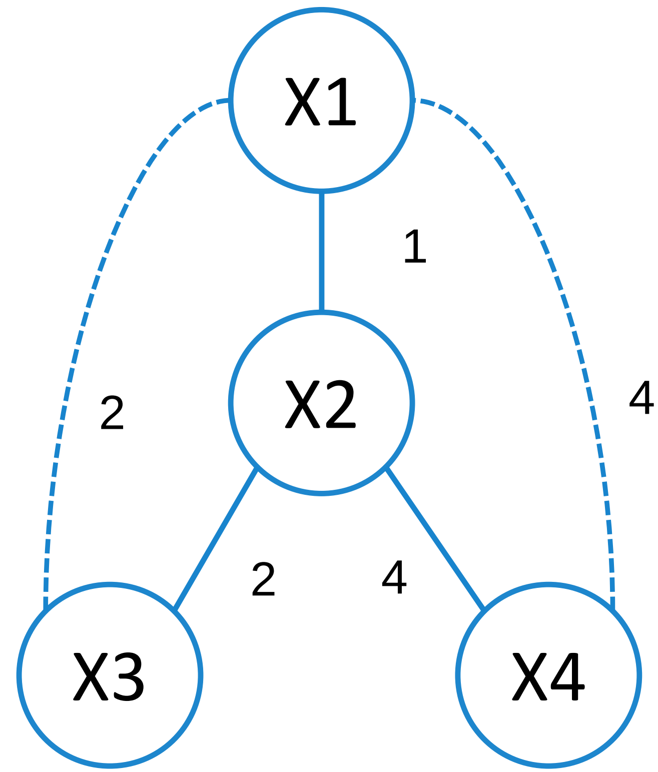


Building a DFS tree

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

Building a DFS-tree: example





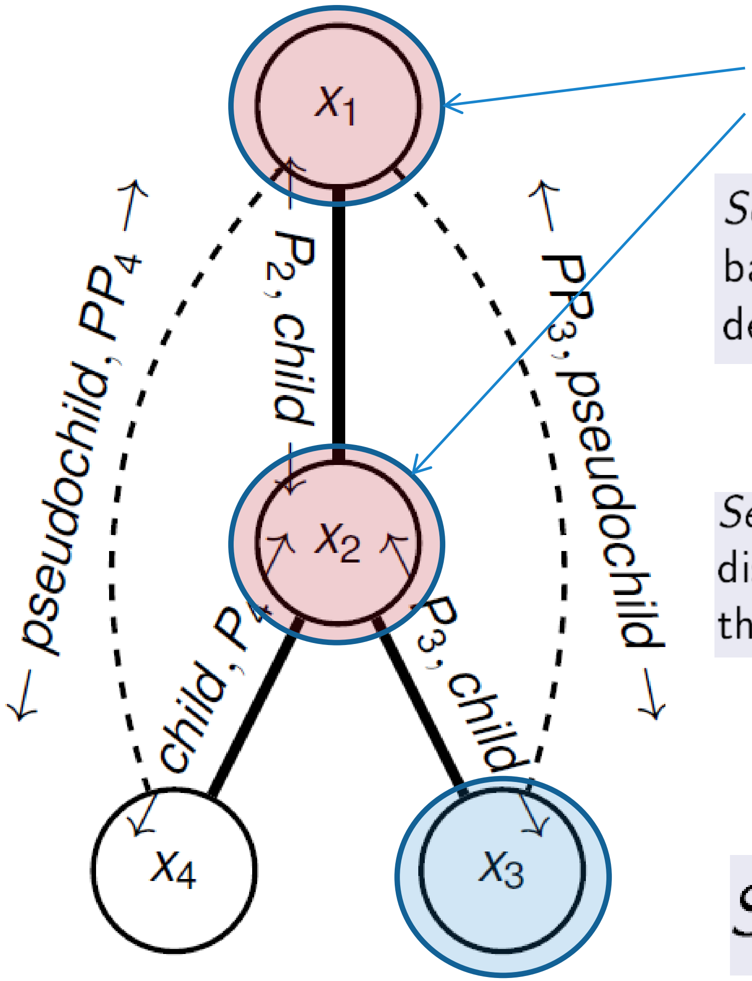
token movement

st: first time node received the token

et: last time node sent the token

time++ = each time token moves

Pseudotrees and Separator



Definition

Sep_i separator of node X_i: all ancestors (though tree and back edges) which are connected with X_i and with any descendant of X_i

Basic Property

Sep_i minimal set of ancestors that, if removed, completely disconnects the subtree rooted at node X_i from the rest of the problem

Operative Definition

$$Sep_i = \cup_{X_j \in C_i} Sep_j \cup P_i \cup PP_i \setminus X_i$$

Ci: children of node i

Util Propagation

Aim: build a value function so that root agent can make optimal decision.
Dynamic programming: provide only key information

Each agent computes messages for its parent based on messages received from children and relevant constraints.

Each message projects out X_i (by maximisation) and aggregates (by summation) functions received from children and all constraints with ancestors (parents and pseudoparents)

Message Computation

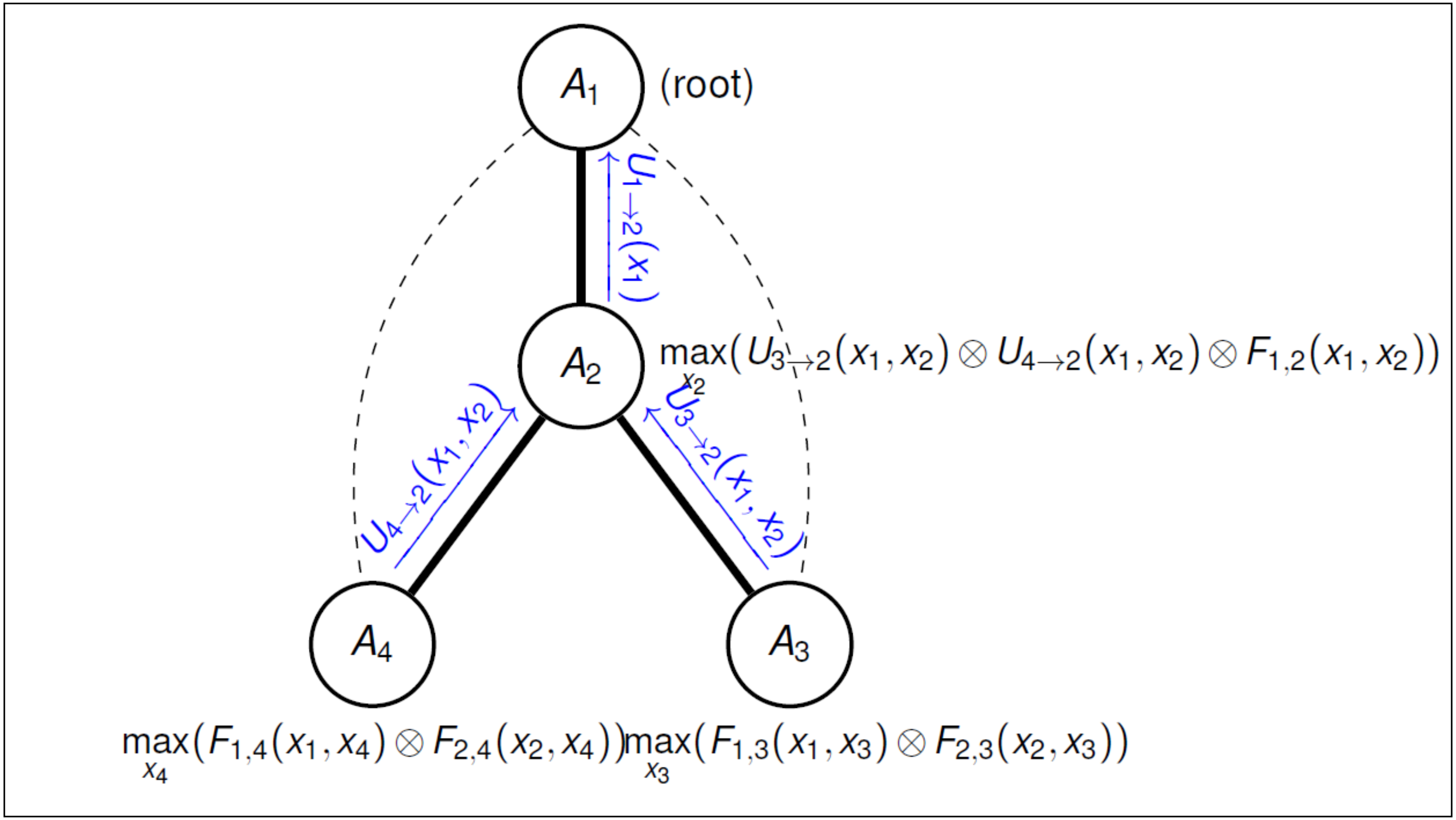
Functions → tables (variable are all discrete)
Aggregation → join operator (relational algebra)
Maximization → projection (keeping most valuable tuples)

The *Util* message $U_{i \to j}$ that agent A_i sends to its parent A_j can be computed as:

$$U_{i \to j}(Sep_i) = \max_{x_i} \left(\bigotimes_{A_k \in C_i} U_{k \to i} \otimes \bigotimes_{A_p \in P_i \cup PP_i} F_{i,p} \right)$$

The $\otimes$ operator is a join operator that sums up functions with different but overlapping scores consistently.

Util Propagation: messages



Util Propagation: message comp

Consider the computation of the node x_3:

$$U_{3\to 2}(x_1, x_2) = max_{x_3}(F_{1,3}(x_1, x_3) \oplus F_{2,3}(x_2, x_3))$$

1. Create $F_{1,3}(x_1, x_3)$ and $F_{2,3}(x_2, x_3)$ by checking the value of the constraints
2. Build the table for all the 8 combinations of the variables x_1, x_2, x_3 with the values of $F_{1,3}$ and $F_{2,3}$
3. Sum the values of $F_{1,3}$ and $F_{2,3}$, (call it V)
4. Maximize wrt , x_3, i.e. for each pair of x_1, x_2, select the one with the highest value of V

DPOP complete example

<u>Value of Each constraint</u>:
same color -1
different colors 0

X1	X3	F13
0	0	-1
0	1	0
1	0	0
1	1	-1

X2	X3	F23
0	0	-1
0	1	0
1	0	0
1	1	-1

X1	X2	X3	F13	F23	F13+F23
0	0	0	-1	-1	-2
0	0	1	0	0	0
0	1	0	-1	0	-1
0	1	1	0	-1	-1
1	0	0	0	-1	-1
1	0	1	-1	0	-1
1	1	0	0	0	0
1	1	1	-1	-1	-2

X1	X2	U32
0	0	0
0	1	-1
1	0	-1
1	1	0

$$U_{3\to 2}(x_1,x_2) = max_{x_3}(F_{1,3}(x_1,x_3) \oplus F_{2,3}(x_2,x_3))$$

Value util example

X1	X2	F12	U32	U42	F12+U32+F42
0	0	-1	0	0	-1
0	1	0	-1	-1	-2
1	0	0	-1	-1	-2
1	1	-1	0	0	-1

X1	U12
0	-1
1	-1

$$U_{2\to 1}(x_1) = max_{x_2}(U_{3\to 2}(x_1, x_2) \oplus U_{4\to 2}(x_1, x_2) \oplus F_{1,2}(x_1, x_2))$$

When propagated to the root a decision can be taken on *x1*

Value Propagation

Aim: inform all agents about decision from above so that they can choose best values for their variables

Root agent A_r computes x_r^* which is the argument that maximises the sum of messages received by all children

It sends a message $V_{r \to c} = \{X_r = x_r^*\}$ containing this value to all children C_r

The generic agent A_i sends a message to each child A_j $V_{i \to j} = \{X_s = x_s^*\} \cup X_i = x_i^*$, where $X_s \in Sep_i \cap Sep_j$

Value Computation

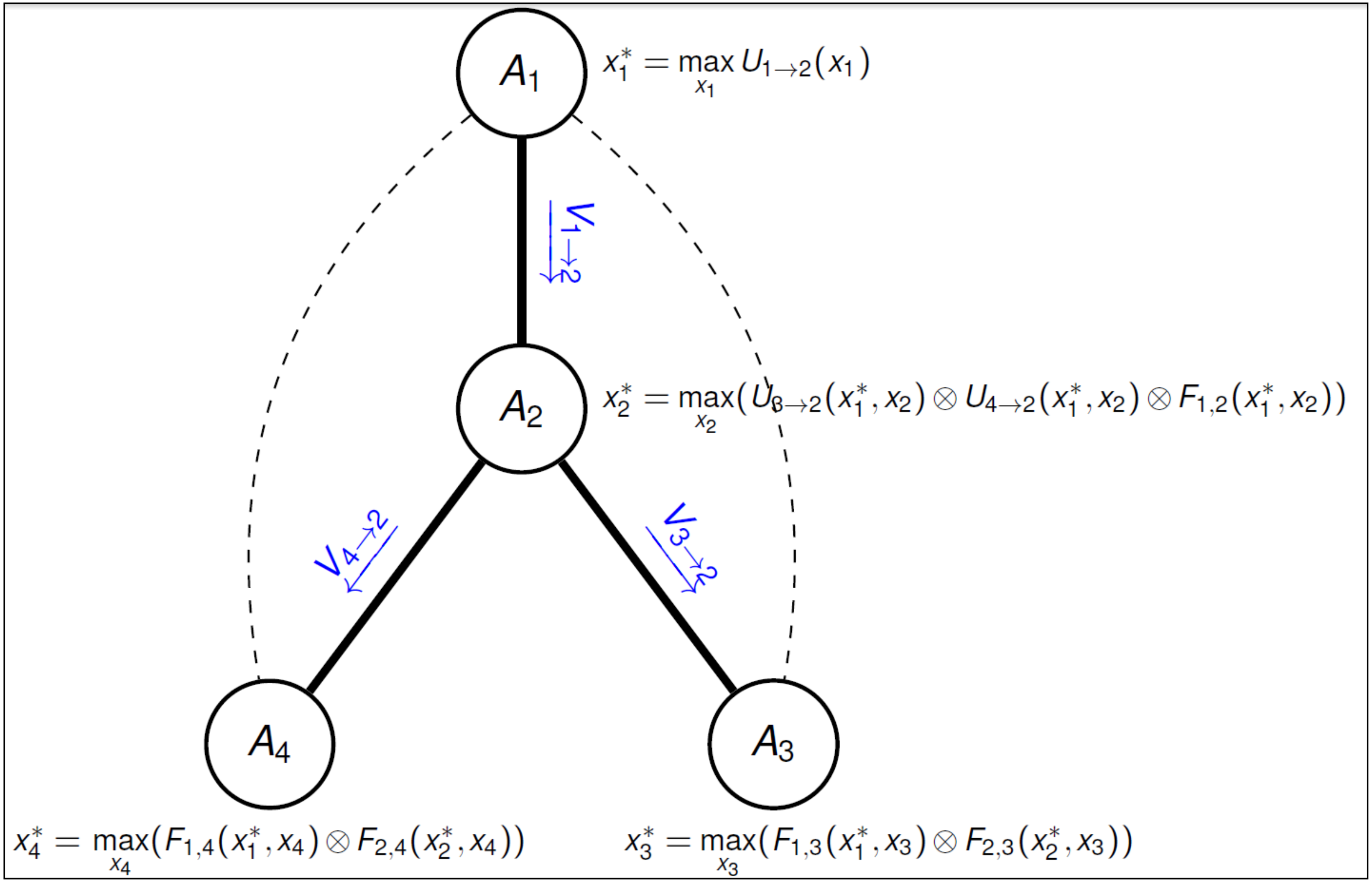
After receiving a value message from its parent, each agent computes the optimal value for its variable as:

$$x_i^* = arg\max_{x_i}\left(\sum_{A_j \in C_i} U_{j \rightarrow i}(x_p^*) + \sum_{A_j \in P_i \cup PP_i} F_{i,j}(x_i, x_j^*) \right)$$

where $x_p^* = \bigcup_{A_j \in P_i \cup PP_i} \{x_j^*\}$ is the set of optimal values for A_i's parent and pseudoparents received from A_i's parent.

Can reuse stored tables for computing util messages

Value Propagation: messages



Value propagation

x1 <- 0

X1	X2	F12	U32	U42	F12+U32+F42
0	0	-1	0	0	-1
0	1	0	-1	-1	-2
1	0	0	-1	-1	-2
1	1	-1	0	0	-1

x2 <- 0

Value propagation

x1 <- 0

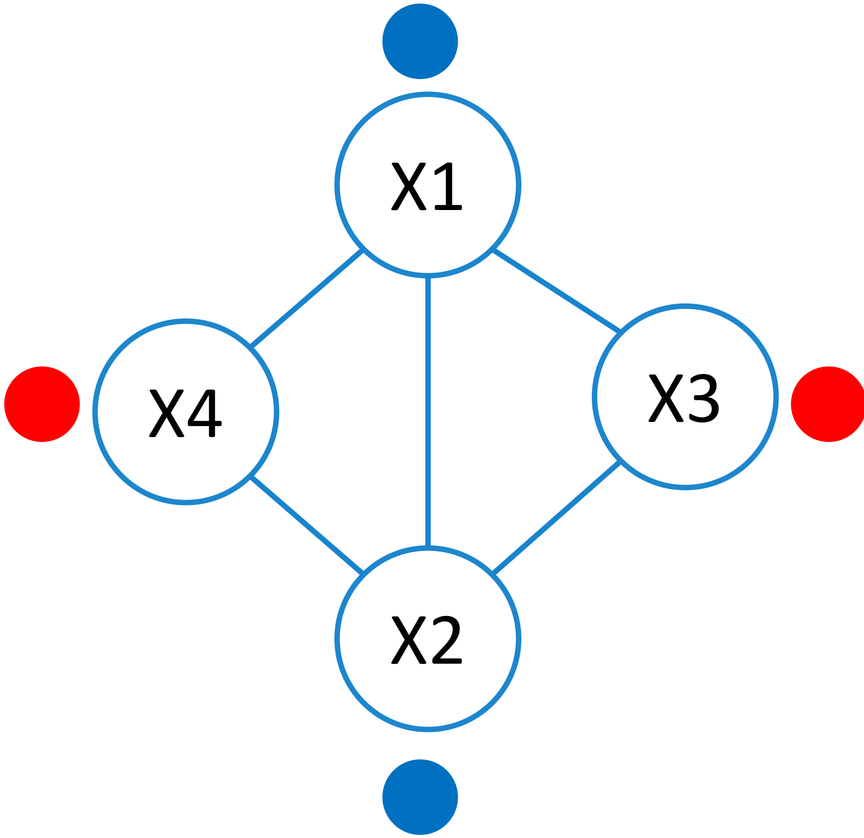
x2 <- 0

X1	X2	X3	F13	F23	F13+F23
0	0	0	-1	-1	-2
0	0	1	0	0	0
0	1	0	-1	0	-1
0	1	1	0	-1	-1
1	0	0	0	-1	-1
1	0	1	-1	0	-1
1	1	0	0	0	0
1	1	1	-1	-1	-2

x3 <- 1

x4 <- 1

DPOP complete example



Only 1 constraint is violated

DPOP analysis

- Synchronous algorithm
- Linear number of messages but exponentially large
- Messages (and computation) is exponential in separators size
- Separator size → graph induced width with DFS ordering

DFS tree and efficiency

- Depth first order is crucial for DPOP efficiency
- Finding optimal order is hard
  - Optimal → minimize separator size
- Good heuristics:
  - Maximum Connected Node (MCN)
  - Maximum Cardinality Set (MCS) for DFS

DFS tree Pseudotrees

- DPOP would work on any pseudotree arrangement of primal graph
- <u>But</u> DFS induces only a specific set of orderings:
  - Not all pseudotres are DFS trees
- We might <u>loose good orderings</u> to keep <u>computation local</u>
  - Trade-off depends on applications

Summary

- DCOPs, general framework to address Multi-Agent coordination
  - Many solution techniques for (relatively) large scale systems
- Complete approaches
  - Suffer from exponential element (DCOPs are hard problems)
  - ADOPT:
    - search based, asynchronous
    - Small messages but exponentially many
  - DPOP:
    - Dynamic programming based, synchronous
    - Few messages but exponentially large
    - Typically much more efficient than ADOPT

References

Multiagent Systems edited by *G. Weiss* MIT Press, 2013, 2nd edition
http://www.the-mas-book.info/index.html

Chapter 12 **Distributed Constraint Handling and Optimization**

A. Farinelli, M. Vinyals, A. Rogers, and N.R. Jenning

FURTHER READING

Constraint Network

- **Constraint Processing**, *R. Dechter, Morgan Kaufmann*

ADOPT

- [Modi et al., 2005] P. J. Modi, W. Shen, M. Tambe, and M.Yokoo. **ADOPT: Asynchronous distributed constraint optimization with quality guarantees**. Artificial Intelligence Journal, (161):149-180, 2005.

DPOP

- [Petcu 2007] A. Petcu. **A Class of Algorithms for Distributed Constraint Optimization.** PhD. Thesis No. 3942, Swiss Federal Institute of Technology (EPFL), Lausanne (Switzerland), 2007. (Chapters 2, 3 and 4)