LECTURE 9

Decentralised Coordination

Contents

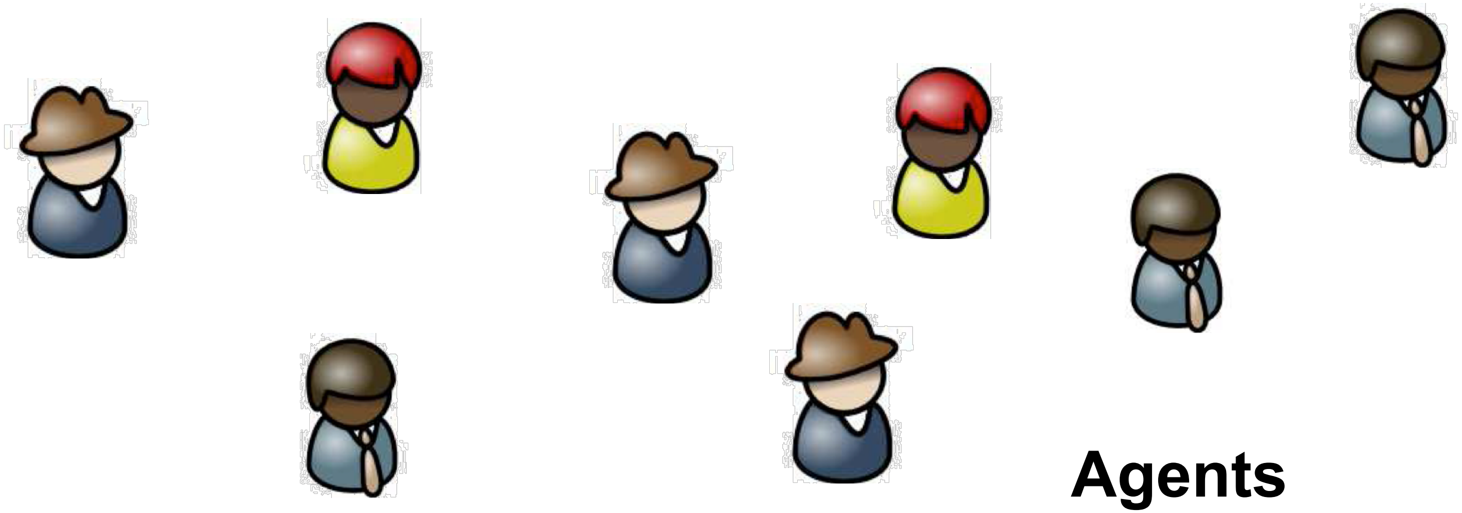
- Working Together
- Decentralised Coordination
- Distributed Constraint Optimisation Problems
  - Max-sum algorithm

Working Together

- In many settings agents must work together to achieve their goals
  - When filtering options, the choices of other agents must be considered
    - In Robocup rescue the time to rescue a civilian depends on how many agents are working together
  - Depends on number of agents to be considered
    - Many agents – model collectively as the environment
    - Few agents – interesting difficult problem
    - No agents – same case as before

Decentralised Coordination

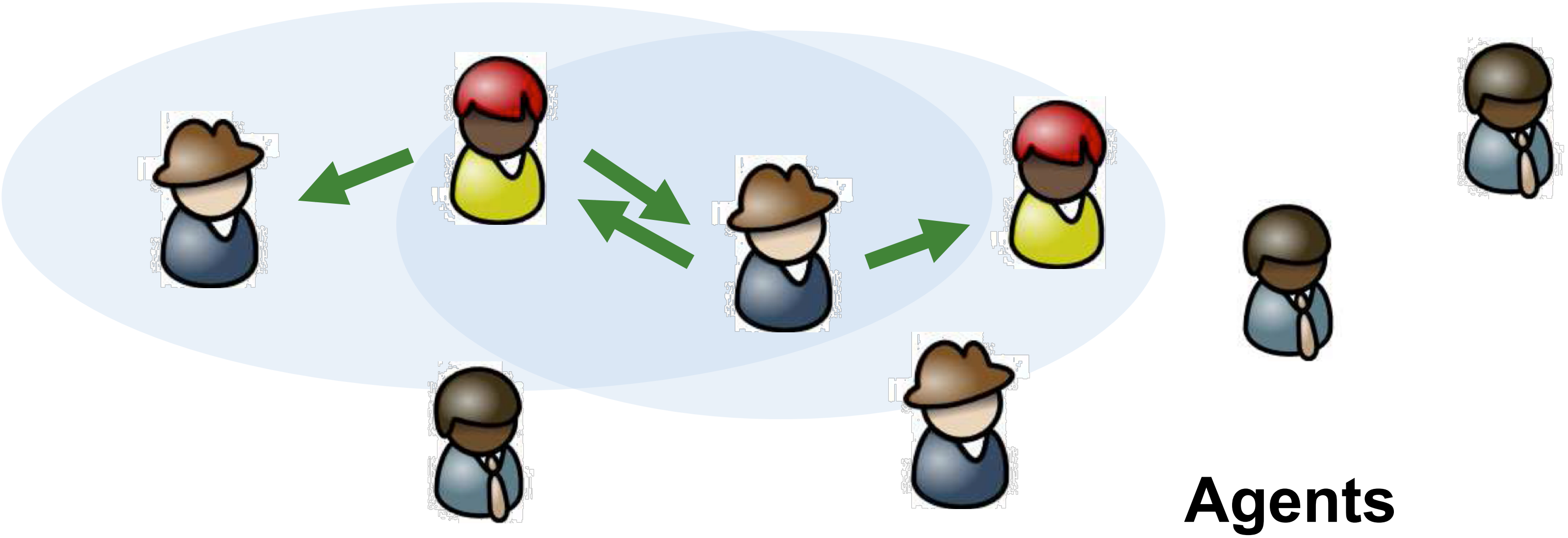
Discrete set of possible actions



Agents

Decentralised Coordination

Discrete set of possible actions
Some locality of interaction



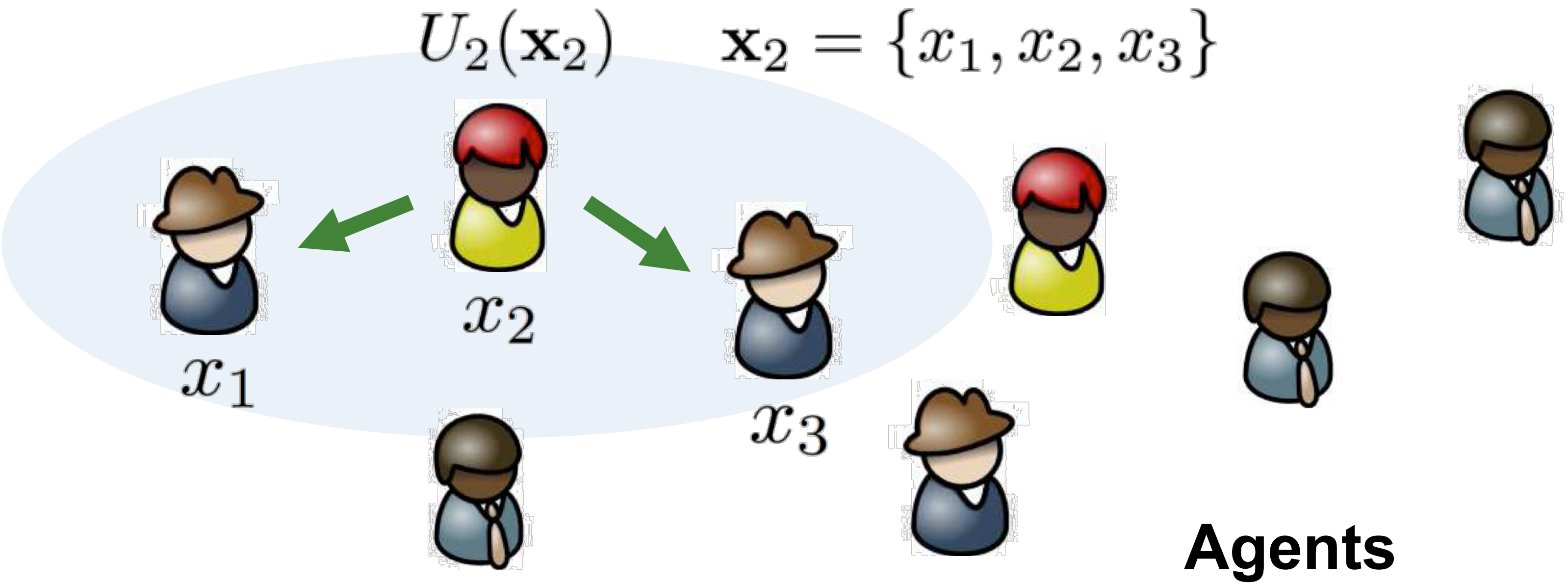
Decentralised Coordination

Maximise Social Welfare:

$$\arg\max_{\mathbf{x}} \sum_{m=1}^{M} U_m(\mathbf{x}_m)$$

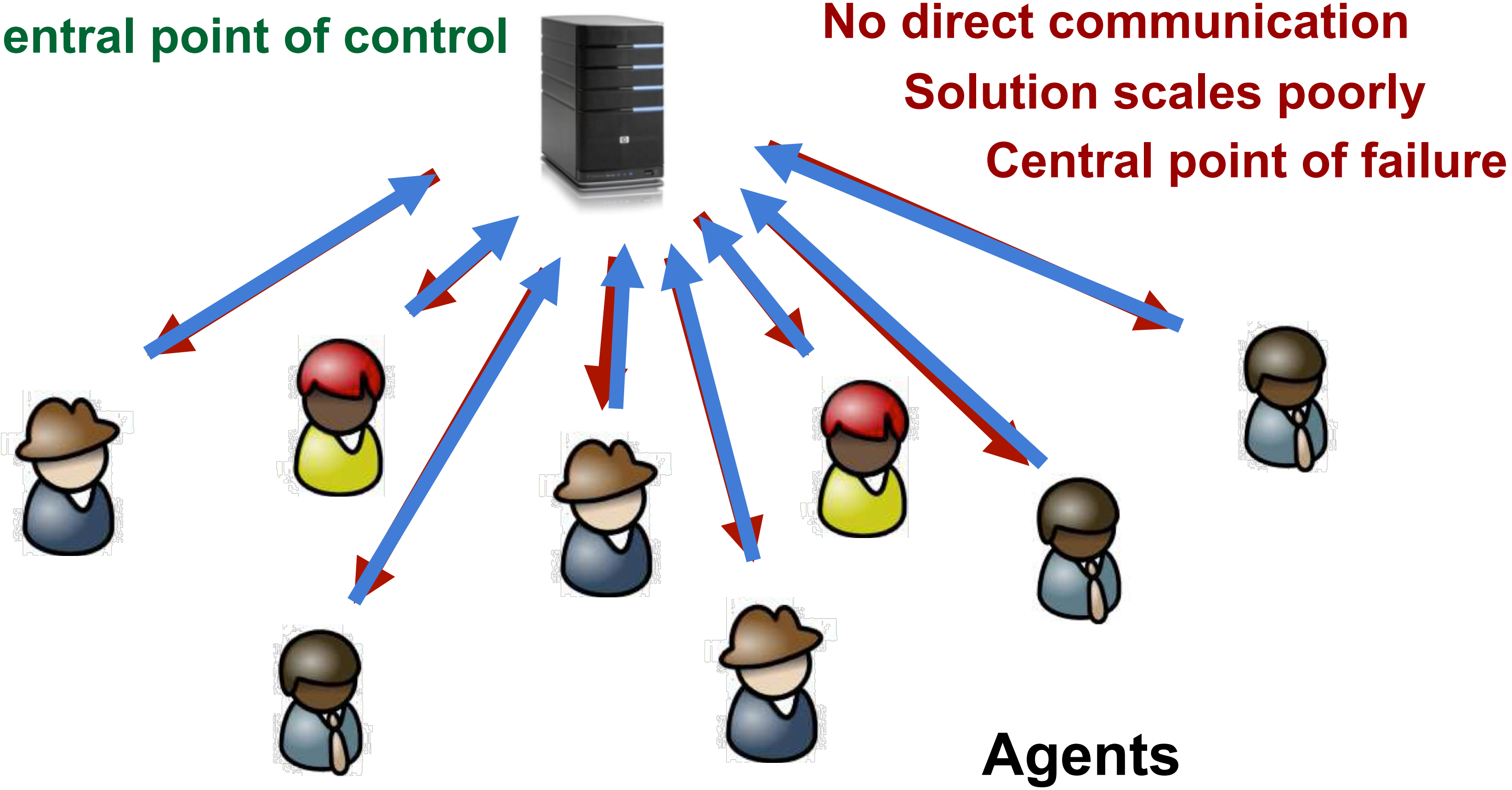
Discrete set of possible actions

Some locality of interaction



$$U_2(\mathbf{x}_2) \qquad \mathbf{x}_2 = \{x_1, x_2, x_3\}$$

Decentralised Coordination



Central point of control

No direct communication

Solution scales poorly

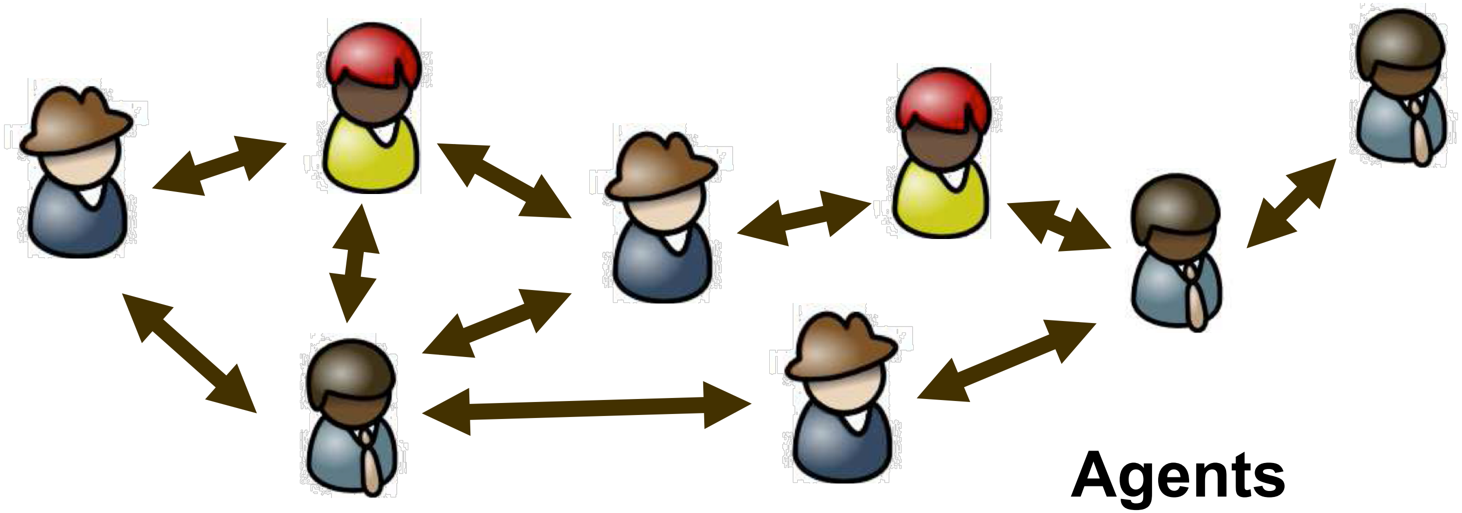
Central point of failure

Agents

Decentralised Coordination

Decentralised control and coordination through local computation and message passing.

- **Speed of convergence, guarantees of optimality, communication overhead, computability**



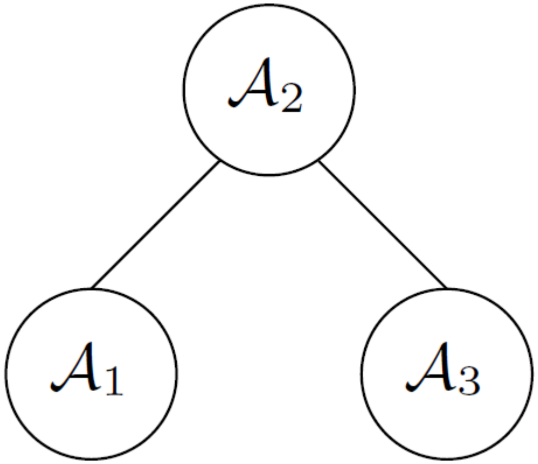
Agents

Contents

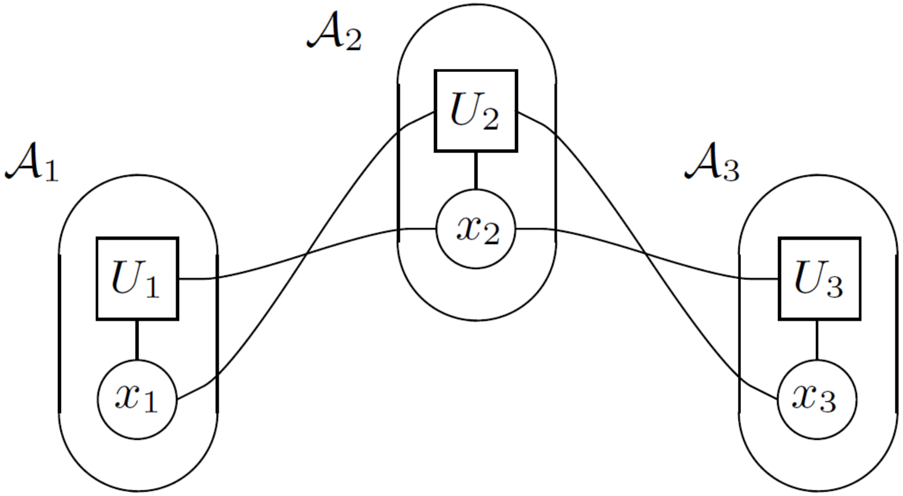
- Working Together
- Decentralised Coordination
- Distributed Constraint Optimisation Problems
    - Max-sum algorithm

Factor Graphs

- Can represent utility relationships as bi-partite factor graphs



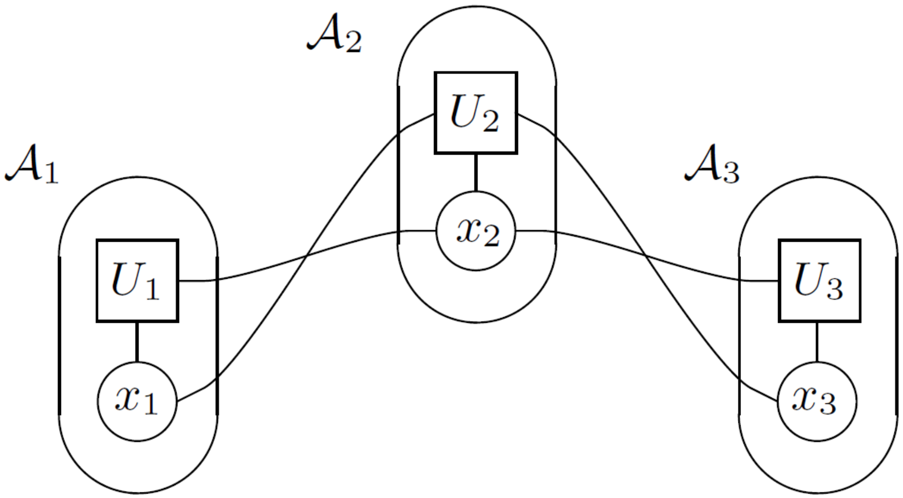
Agent graph



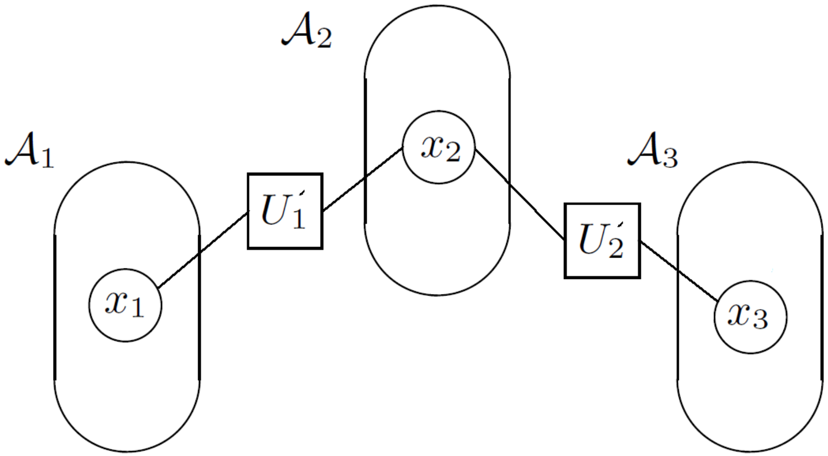
Factor graph

Constraint Graphs

- Often we can rearrange the factor graph to yield a constraint graph



Factor graph



Constraint graph

$$U_1(x_1, x_2) + U_2(x_1, x_2, x_3) + U_3(x_2, x_3) = U'_1(x_1, x_2) + U'_2(x_2, x_3)$$

Max-sum Algorithm

- Applies to tree-based constraint graphs
- Algorithm proceeds as follows:
  - Leaf nodes send messages
  - If a node has *k* neighbours, it waits until it has received all *k-1* messages, before sending message to each particular neighbour
  - Algorithm ends when each node has received *k* messages

Max-Sum

- Messages flow between function and variable nodes of the factor graph

From variable to factor:

$$Q_{n\to m}(x_n) = \sum_{m' \in M(n) \setminus m} R_{m' \to n}(x_n)$$

From factor to variable:

$$R_{m\to n}(x_n) = \max_{\mathbf{x}_m \setminus n} \left(U_m(\mathbf{x}_m) + \sum_{n' \in N(m) \setminus n} Q_{n' \to m}(x_{n'}) \right)$$

Max-Sum

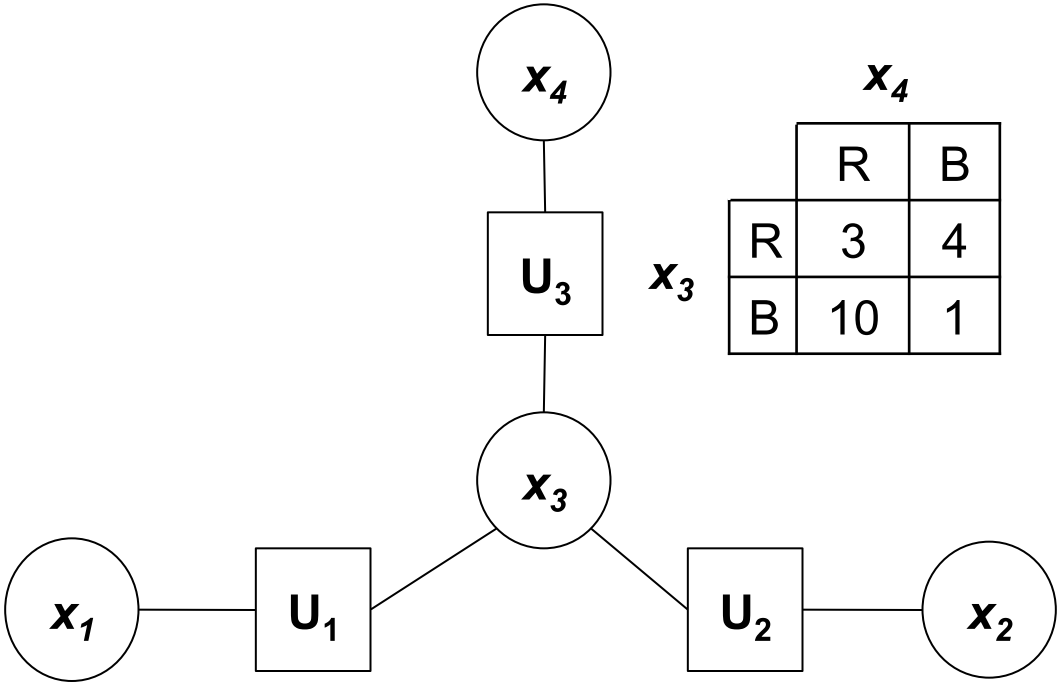
- At completion

$$Z_n(x_n) = \sum_{m \in M(n)} R_{r \to n}(x_n)$$

- Solution given by

$$x_n^* = \arg\max_{x_n} Z_n(x_n)$$

Example



		x_4	
		R	B
x_3	R	3	4
	B	10	1

		x_3	
		R	B
x_1	R	1	3
	B	2	1

		x_3	
		R	B
x_2	R	1	5
	B	4	2

🟥 🟦

Example

- Step 1

$Q_{1\to1}(x_1)=\{0,0\}$ $Q_{2\to2}(x_2)=\{0,0\}$ $Q_{4\to3}(x_4)=\{0,0\}$

- Step 2

$R_{1\to3}(x_3)=\{2,3\}$ $R_{2\to3}(x_3)=\{4,5\}$ $R_{3\to3}(x_4)=\{4,10\}$

- Step 3

$Q_{3\to1}(x_3)=\{8,15\}$ $Q_{3\to2}(x_3)=\{6,13\}$ $Q_{3\to3}(x_3)=\{6,8\}$

Example

- Step 4

$R_{1\rightarrow 1}(x_1)=\{18,16\}$ $R_{2\rightarrow 2}(x_2)=\{18,15\}$ $R_{3\rightarrow 4}(x_4)=\{18,10\}$

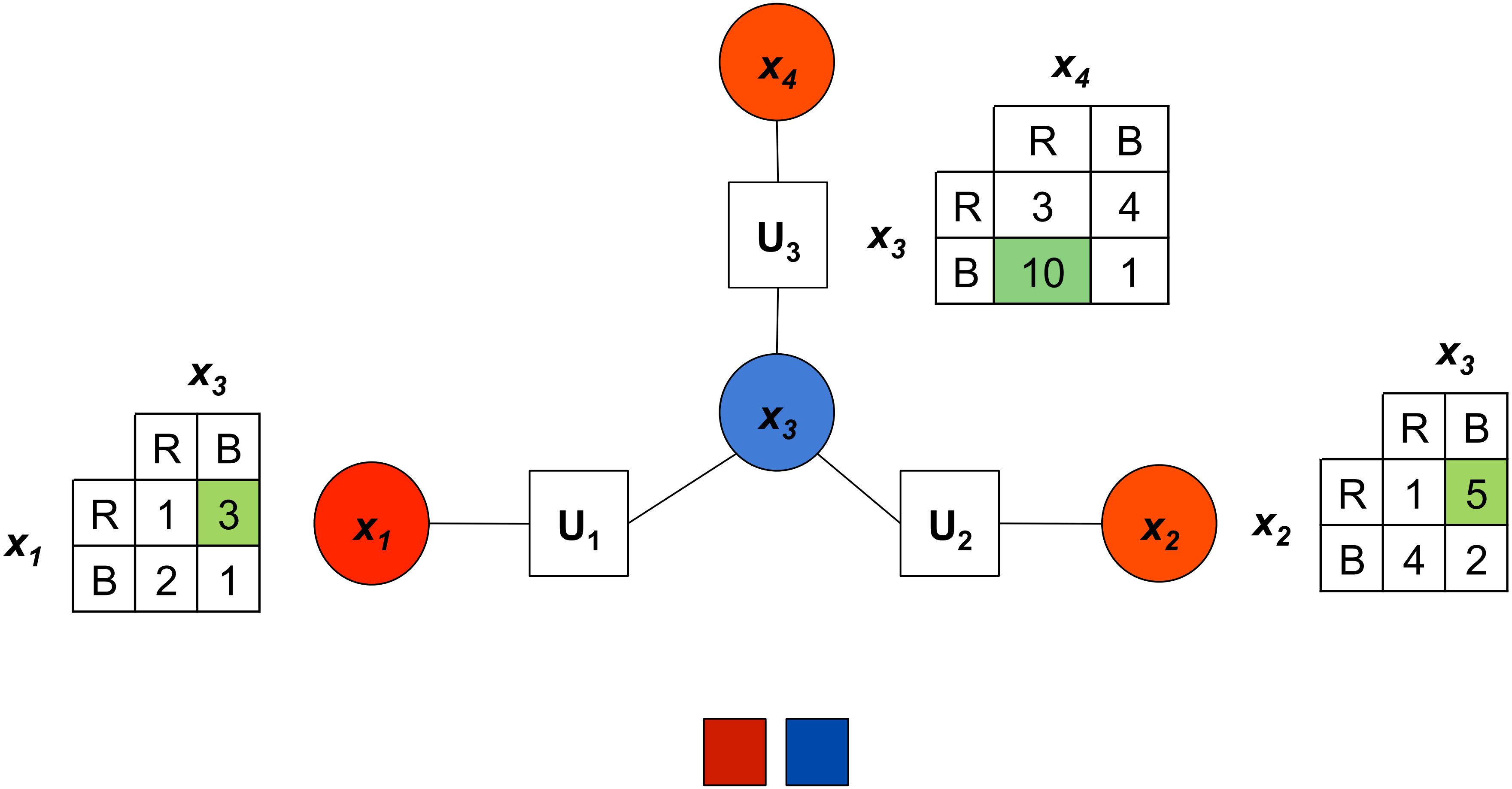
- Step 5

$Z_1(x_1)=\{18,16\}$ $Z_2(x_2)=\{18,15\}$ $Z_3(x_3)=\{10,18\}$ $Z_4(x_4)=\{18,10\}$

- Step 7

$x_1=R$ $x_2=R$ $x_3=B$ $x_4=R$

Example



x_4

x_3

	R	B
R	3	4
B	10	1

x_3

x_1

	R	B
R	1	3
B	2	1

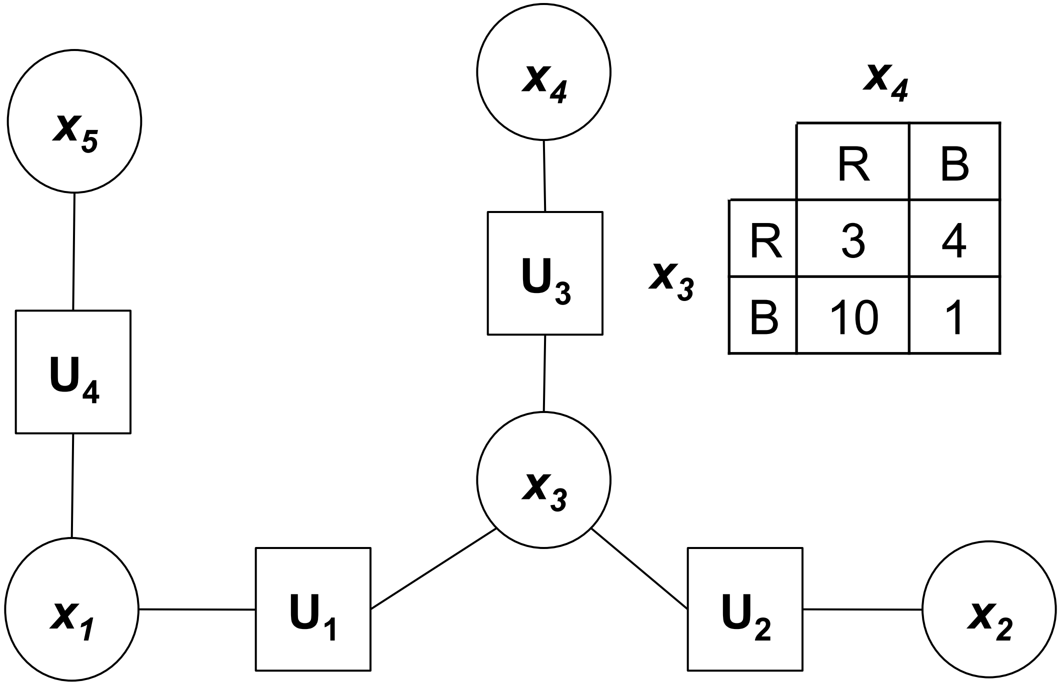
x_3

x_2

	R	B
R	1	5
B	4	2

🟥 🟦

Exercise



x_5

		R	B
x_1	R	2	4
	B	8	3

x_4

		R	B
x_3	R	3	4
	B	10	1

x_3

		R	B
x_2	R	1	5
	B	4	2

x_3

		R	B
x_1	R	1	3
	B	2	1

🟥 🟦

Max-Sum Properties

- Optimal algorithm
  - When constraint graph is a tree
- Solution converges after *2 x (n-1)* messages
- Can propagate messages iteratively and asynchronously
  - Will still converge to the optimal solution

DPOP Algorithm

- Extends max-sum to constraint graphs with cycles or loops
  - Create pseudo-tree
    - Extend variables to combinations of variables
    - This is like a junction tree in graphical models
  - Optimal algorithm
  - Exponentially sized messages!

Other Approaches

- Bounded max-sum
  - Prune cyclic graph back to a tree
    - Optimally solve the new problem
    - Calculate maximum bound from solution of the original problem.
- Loopy max-sum
  - Use max-sum as is even though the graph has cycles
    - Normalise messages to stop them exploding
    - Works surprisingly well

Reading

- Weiss: Multiagent Systems
  - Chapter 12