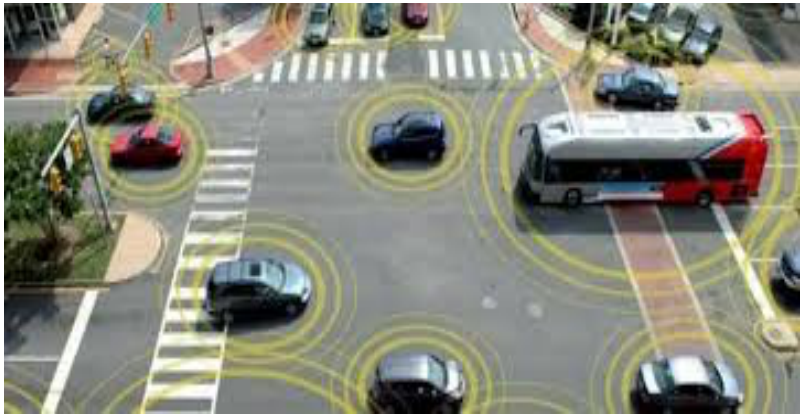
Multiagent Constraint Optimization on the Constraint Composite Graph

Ferdinando Fioretto Hong Xu Sven Koenig T. K. Satish Kumar

University of Michigan University of Southern California

OptMAS 2018




CAM B2
DOMICOPTER_TESTDELIVERY_#00032

Content

- Distributed Constraint Optimization Problems
- The Constraint Composite Graph (CCG)
- CCG-MaxSum
- Experimental Evaluation
- Conclusions

Distributed Constraint Optimization

$<\mathcal{X}, \mathcal{D}, \mathcal{F}, \mathcal{A}, \alpha>$:

- $\mathcal{X}$: Set of variables.
- $\mathcal{D}$: Set of finite domains for each variable.
- $\mathcal{F}$: Set of constraints between variables.
- $\mathcal{A}$: Set of agents, controlling the variables in $\mathcal{X}$.
- α: Mapping of variables to agents.

x_1	x_2	f_1
0	0	0.5
0	1	0.6
1	0	0.7
1	1	0.3

- GOAL: Find a cost minimal assignment.

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \mathbf{F}(\mathbf{x})$$

$$= \operatorname*{argmin}_{\mathbf{x}} \sum_{f \in \mathcal{F}} f(\mathbf{x}|_{\text{scope}(f)})$$

Distributed Constraint Optimization

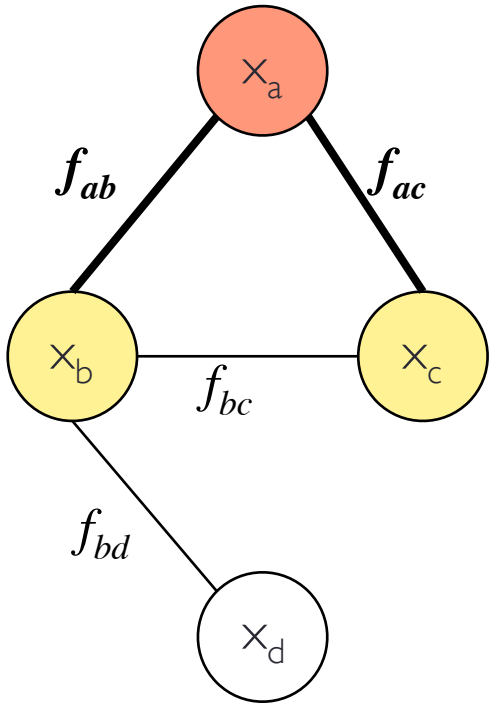
- Agents coordinate an assignment for their variables.
- Agents operate distributedly.

Communication:

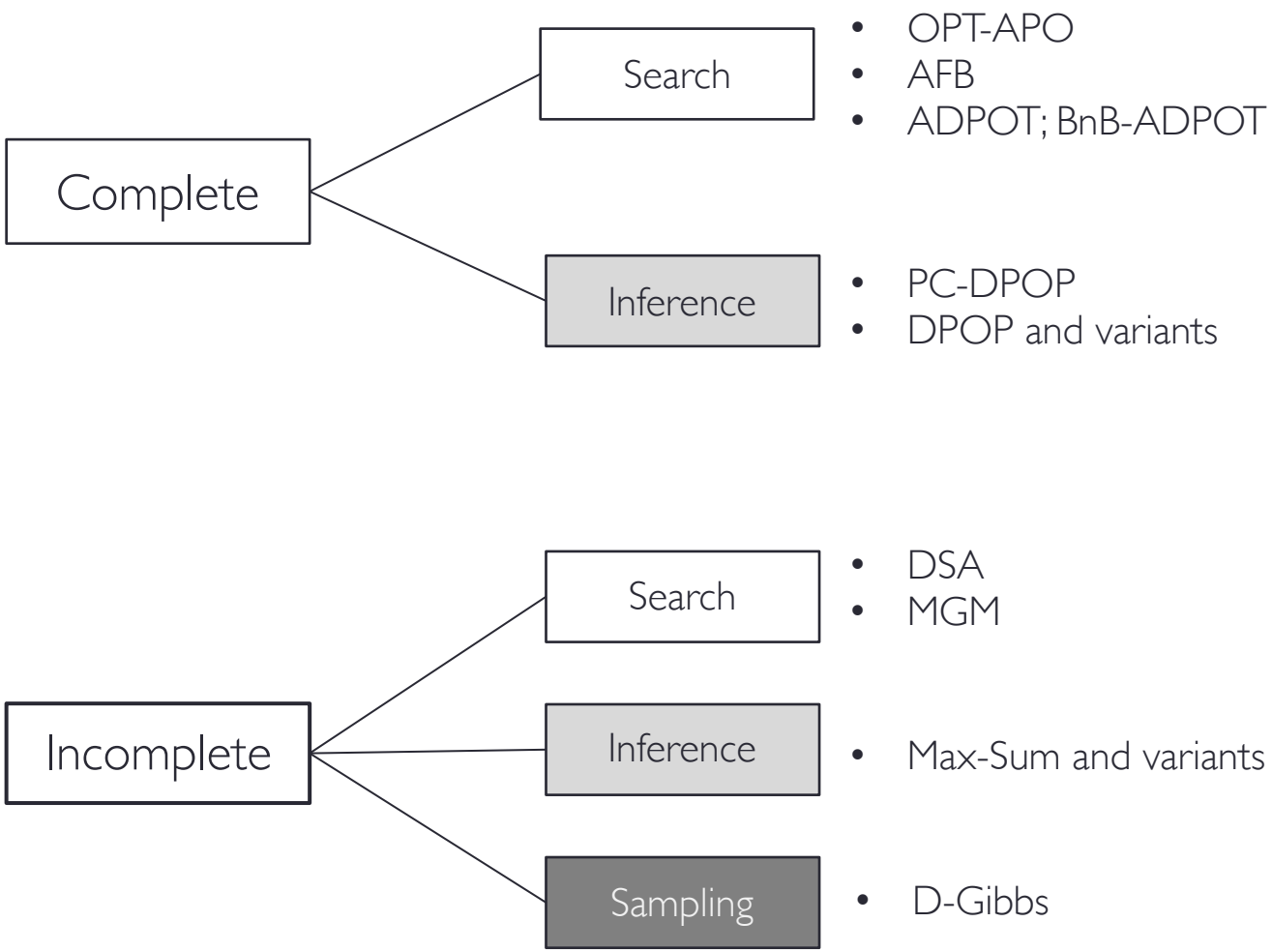
- By exchanging messages.
- Restricted to agent's local neighbors.

Knowledge:

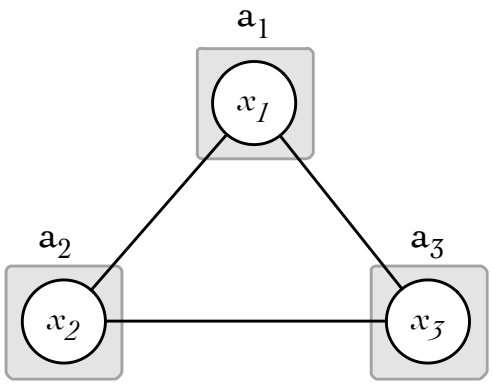
- Restricted to agent's sub-problem.



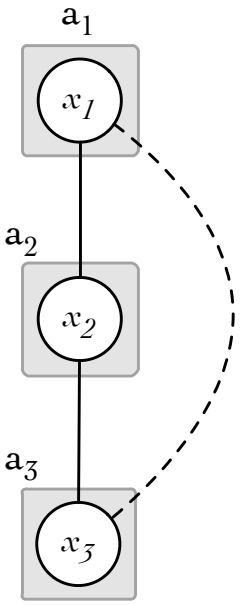
DCOP: Algorithms



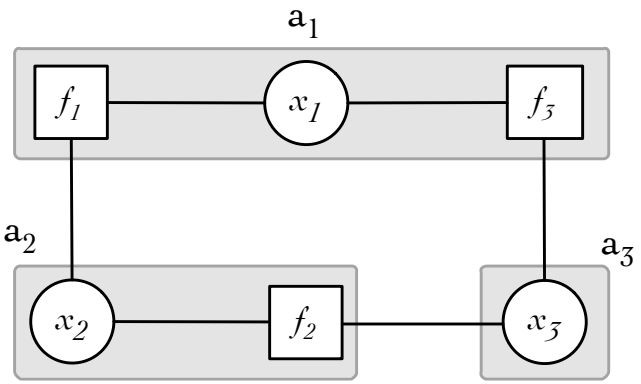
DCOP: Representation



Constraint Graph

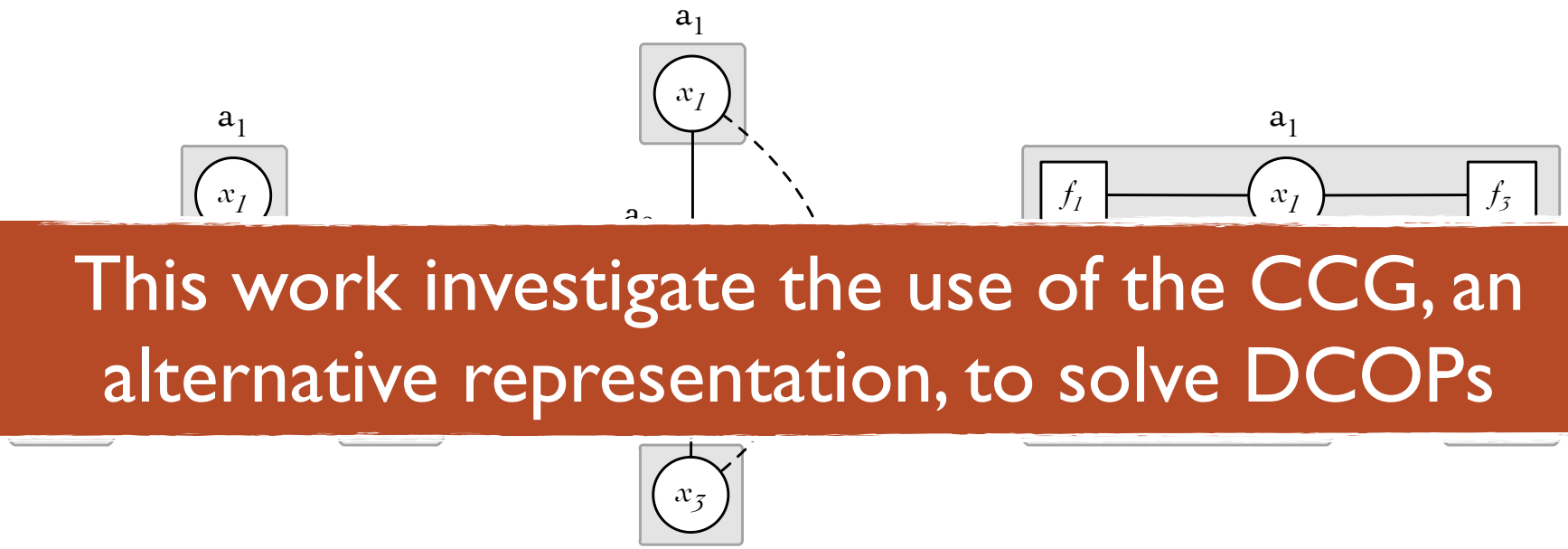


Pseudo-Tree



Factor Graph

DCOP: Representation



This work investigate the use of the CCG, an alternative representation, to solve DCOPs

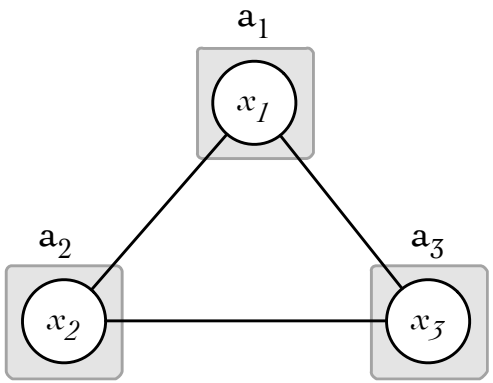
Constraint Graph | Pseudo-Tree | Factor Graph

Assumption: The focus of this talk is restricted to Boolean DCOPs

Constraint Composite Graph

- DCOP structure:
  - Graphical Structure: Interaction of cost functions and joint assignments
  - Numerical Structure: Values associated to cost functions
- How can we exploit both these structures during problem solving?

Graphical Structure



Numerical Structure

x_1	x_2	f_1
0	0	0.5
0	1	0.6
1	0	0.7
1	1	0.3

Constraint Composite Graph

- The Constraint Composite Graph (CCG) [Kumar:08] is a graph

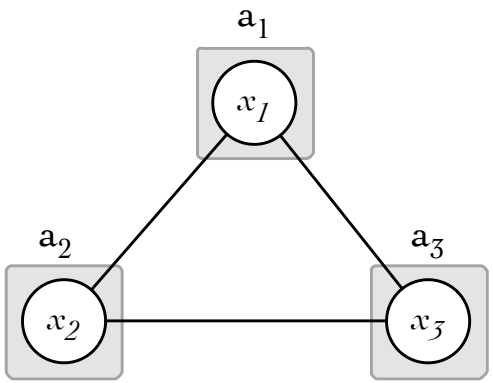
$$G = (X \cup Y \cup Z, E, w)$$

DCOP variables — auxiliary variables — weights

Represents explicitly both structures

- Can be constructed in polytime

Graphical Structure



Numerical Structure

x_1	x_2	f_1
0	0	0.5
0	1	0.6
1	0	0.7
1	1	0.3

Constraint Composite Graph

- The Constraint Composite Graph (CCG) [Kumar:08] is a graph

$$G = (X \cup Y \cup Z, E, w)$$

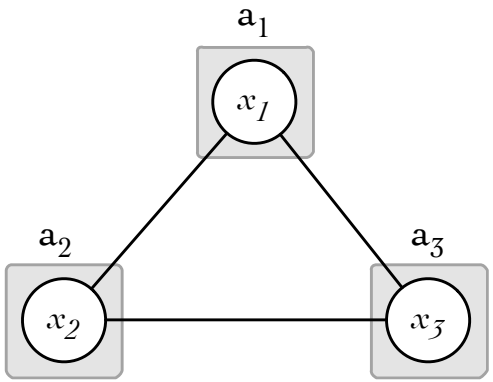
DCOP variables — auxiliary variables — weights

Represents explicitly both structures

- Can be constructed in polytime

- Solving a DCOP can be reformatted as solving a Minimum Weighted Vertex Cover on its associated CCG [extended result from Kumar:16]

Graphical Structure



Numerical Structure

x_1	x_2	f_1
0	0	0.5
0	1	0.6
1	0	0.7
1	1	0.3

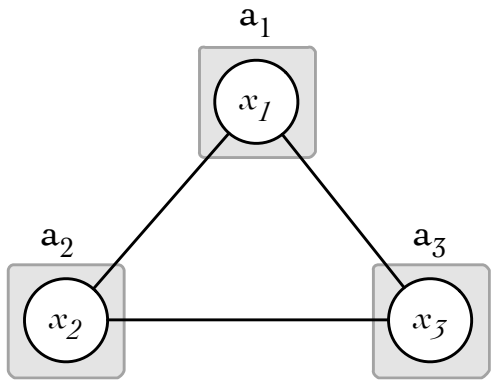
The Nemhauser-Trotter (NT) Reduction

- Polytime kernelization technique used to reduce the size of the MWVC
- Minimum Weighted Vertex Cover

$$\text{Minimize} \quad \sum_{i=1}^{|V|} w_i Z_i$$

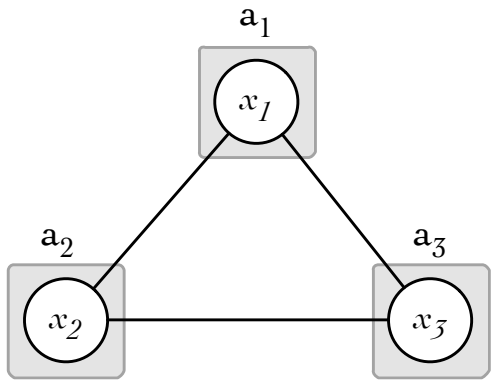
$$\forall v_i \in V : \quad Z_i \in \{0, 1\}$$

$$\forall (v_i, v_j) \in E : \quad Z_i + Z_j \geq 1$$



The Nemhauser-Trotter (NT) Reduction

- Polytime kernelization technique used to reduce the size of the MWVC
- Minimum Weighted Vertex Cover



$$\text{Minimize} \quad \sum_{i=1}^{|V|} w_i Z_i$$

$$\forall v_i \in V : \quad Z_i \in [0,1] \subseteq \mathbb{R}$$

$$\forall (v_i, v_j) \in E : \quad Z_i + Z_j \geq 1$$

- Relax LP is half integral $Z \in \{0, \frac{1}{2}, 1\}$
- NT: There is a MWVC that includes v_i if $Z_i=1$ and exclude v_i if $Z_i=0$

CCG Construction

#1 Construct Polynomial

- Each agent, expresses its cost function as a polynomial

x_1	x_2	f_1
0	0	0.5
0	1	0.6
1	0	0.7
1	1	0.3

$$p_1(x_1, x_2) = c_{00} + c_{01}x_1 + c_{10}x_2 + c_{11}x_1x_2$$

Its coefficients can be computed by standard Gaussian Elimination so that:

$$p_1(0,0) = 0.5 \qquad p_1(0,1) = 0.6$$
$$p_1(1,0) = 0.7 \qquad p_1(1,1) = 0.3$$

$c_{00} = 0.5$, $c_{01} = 0.2$, $c_{10} = 0.1$, $c_{11} = -0.5$.

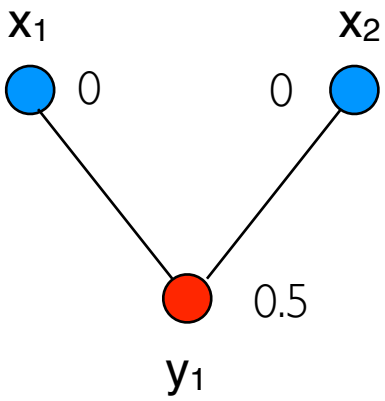
CCG Construction

#2 Construct Gadget for each polynomial

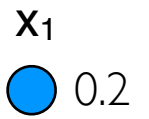
- Construct a "lifted representation" or a gadget graph for each term of a polynomial of each cost function

x_1	x_2	f_1
0	0	0.5
0	1	0.6
1	0	0.7
1	1	0.3

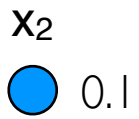
$$p_1(x_1, x_2) = c_{00} + c_{01}x_1 + c_{10}x_2 + c_{11}x_1x_2$$



$$c_{00} + c_{11}x_1x_2$$



$$c_{01}x_1$$



$$c_{10}x_2$$

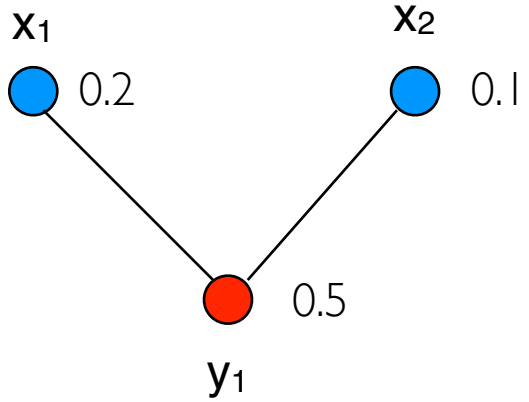
CCG Construction

#2 Construct Gadget for each polynomial

- Construct a "lifted representation" or a gadget graph for each term of a polynomial of each cost function

x_1	x_2	f_1
0	0	0.5
0	1	0.6
1	0	0.7
1	1	0.3

$$p_1(x_1, x_2) = c_{00} + c_{01}x_1 + c_{10}x_2 + c_{11}x_1x_2$$



$c_{00} = 0.5$, $c_{01} = 0.2$, $c_{10} = 0.1$, $c_{11} = -0.5$.

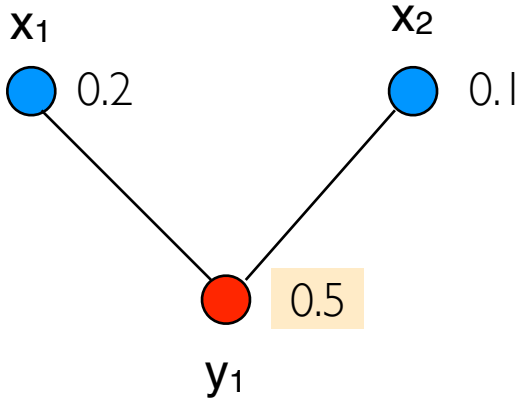
CCG Construction

#2 Construct Gadget for each polynomial

- Construct a "lifted representation" or a gadget graph for each term of a polynomial of each cost function

x_1	x_2	f_1
0	0	0.5
0	1	0.6
1	0	0.7
1	1	0.3

$$p_1(x_1, x_2) = c_{00} + c_{01}x_1 + c_{10}x_2 + c_{11}x_1x_2$$



$c_{00} = 0.5$, $c_{01} = 0.2$, $c_{10} = 0.1$, $c_{11} = -0.5$.

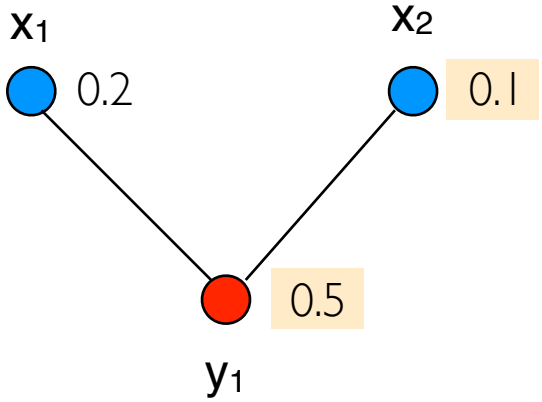
CCG Construction

#2 Construct Gadget for each polynomial

- Construct a "lifted representation" or a gadget graph for each term of a polynomial of each cost function

x_1	x_2	f_1
0	0	0.5
0	1	0.6
1	0	0.7
1	1	0.3

$$p_1(x_1, x_2) = c_{00} + c_{01}x_1 + c_{10}x_2 + c_{11}x_1x_2$$



$c_{00} = 0.5$, $c_{01} = 0.2$, $c_{10} = 0.1$, $c_{11} = -0.5$.

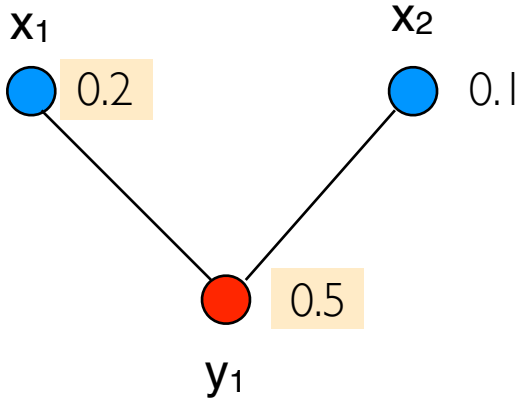
CCG Construction

#2 Construct Gadget for each polynomial

- Construct a "lifted representation" or a gadget graph for each term of a polynomial of each cost function

x_1	x_2	f_1
0	0	0.5
0	1	0.6
1	0	0.7
1	1	0.3

$$p_1(x_1, x_2) = c_{00} + c_{01}x_1 + c_{10}x_2 + c_{11}x_1x_2$$



$c_{00} = 0.5$, $c_{01} = 0.2$, $c_{10} = 0.1$, $c_{11} = -0.5$.

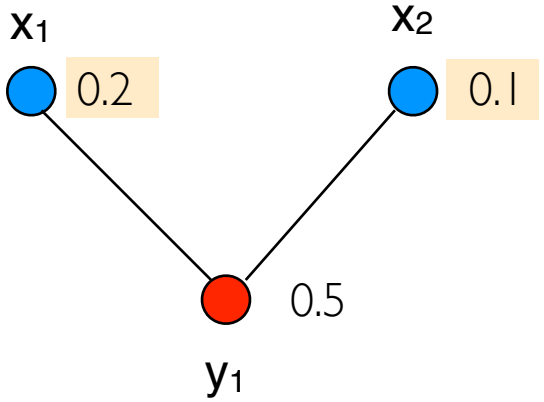
CCG Construction

#2 Construct Gadget for each polynomial

- Construct a "lifted representation" or a gadget graph for each term of a polynomial of each cost function

x_1	x_2	f_1
0	0	0.5
0	1	0.6
1	0	0.7
1	1	0.3

$$p_1(x_1, x_2) = c_{00} + c_{01}x_1 + c_{10}x_2 + c_{11}x_1x_2$$



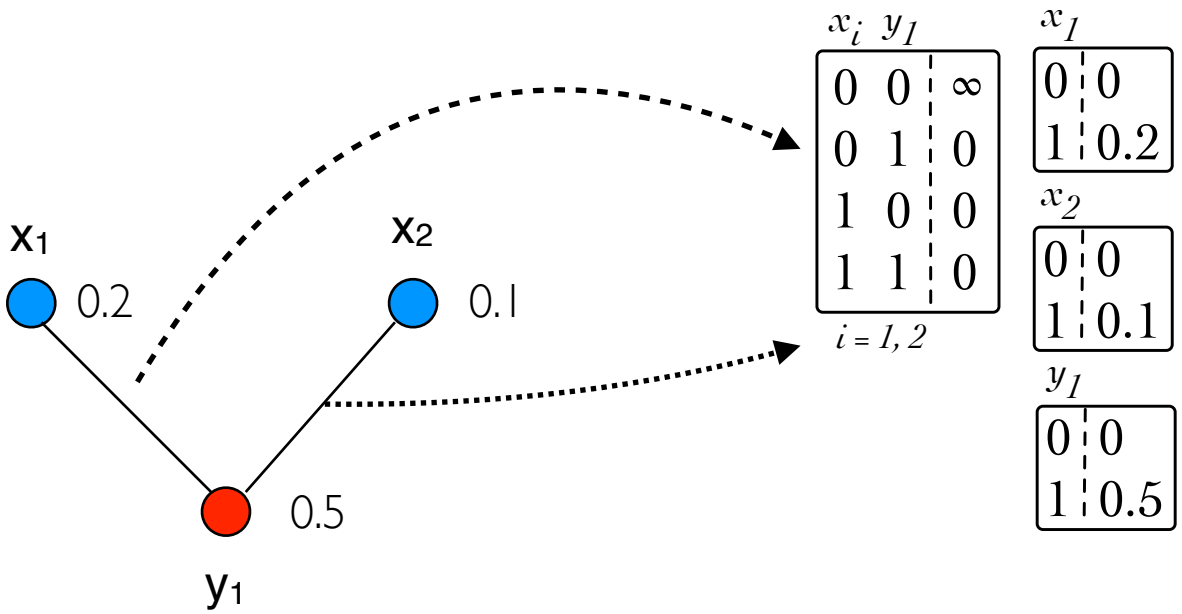
$c_{00} = 0.5$, $c_{01} = 0.2$, $c_{10} = 0.1$, $c_{11} = -0.5$.

CCG Construction

#2 Construct Gadget for each polynomial

- Construct a "lifted representation" or a gadget graph for each term of a polynomial of each cost function

x_1	x_2	f_1
0	0	0.5
0	1	0.6
1	0	0.7
1	1	0.3

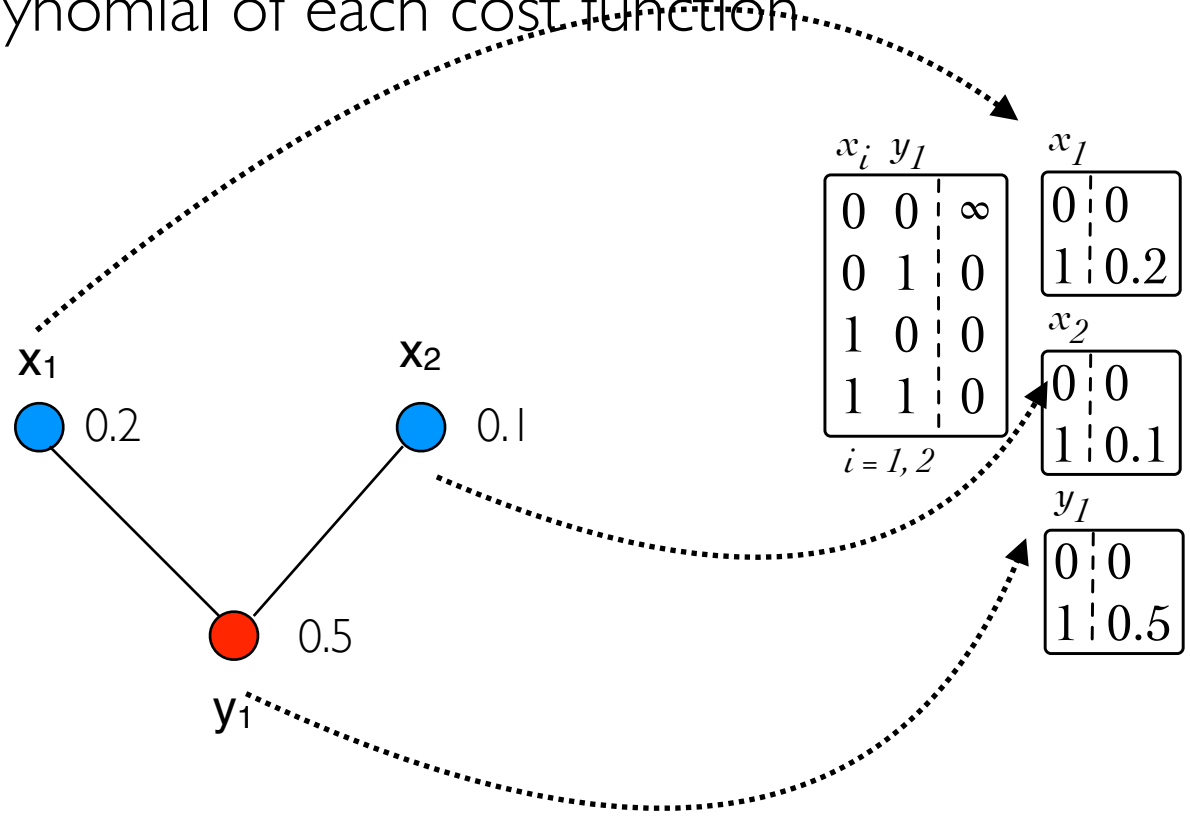


CCG Construction

#2 Construct Gadget for each polynomial

- Construct a "lifted representation" or a gadget graph for each term of a polynomial of each cost function

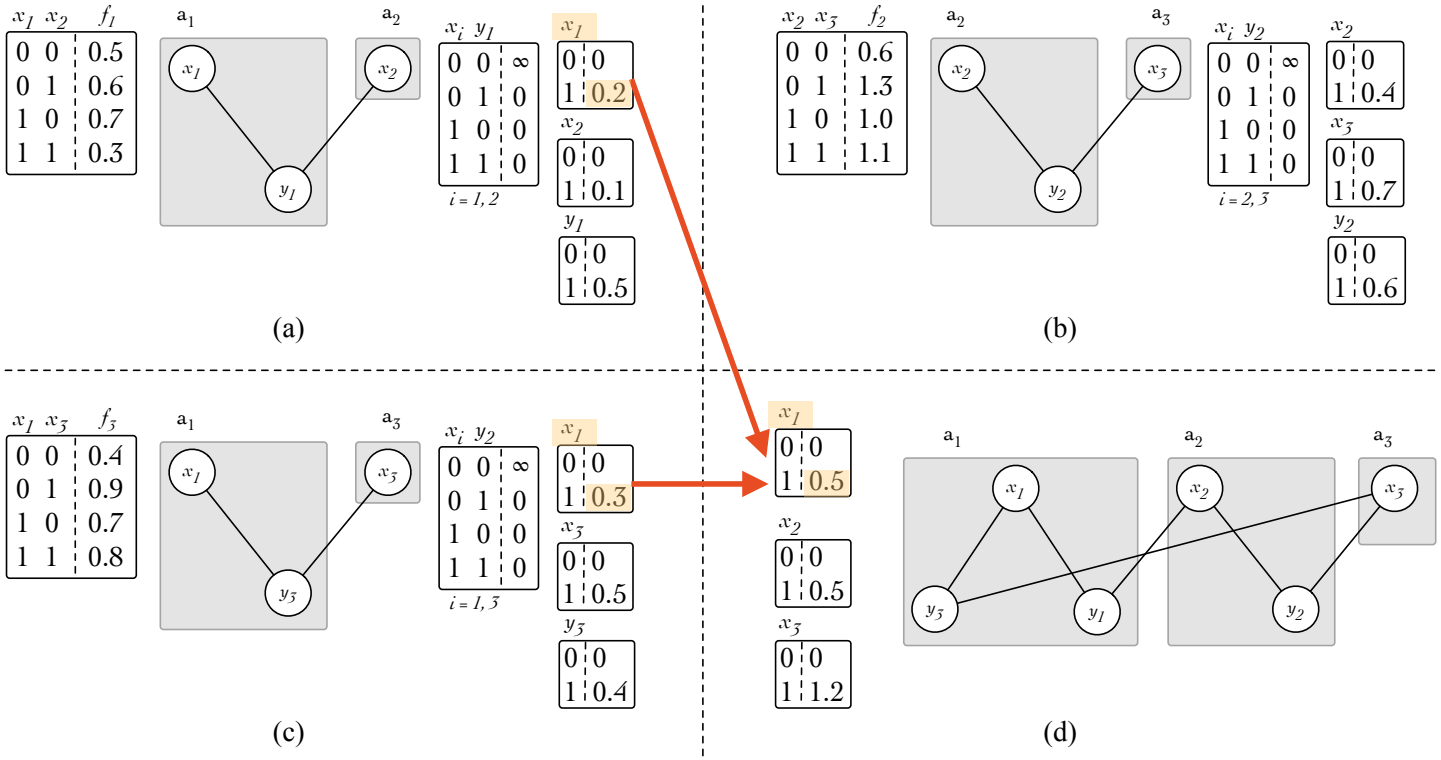
x_1	x_2	f_1
0	0	0.5
0	1	0.6
1	0	0.7
1	1	0.3



CCG Construction

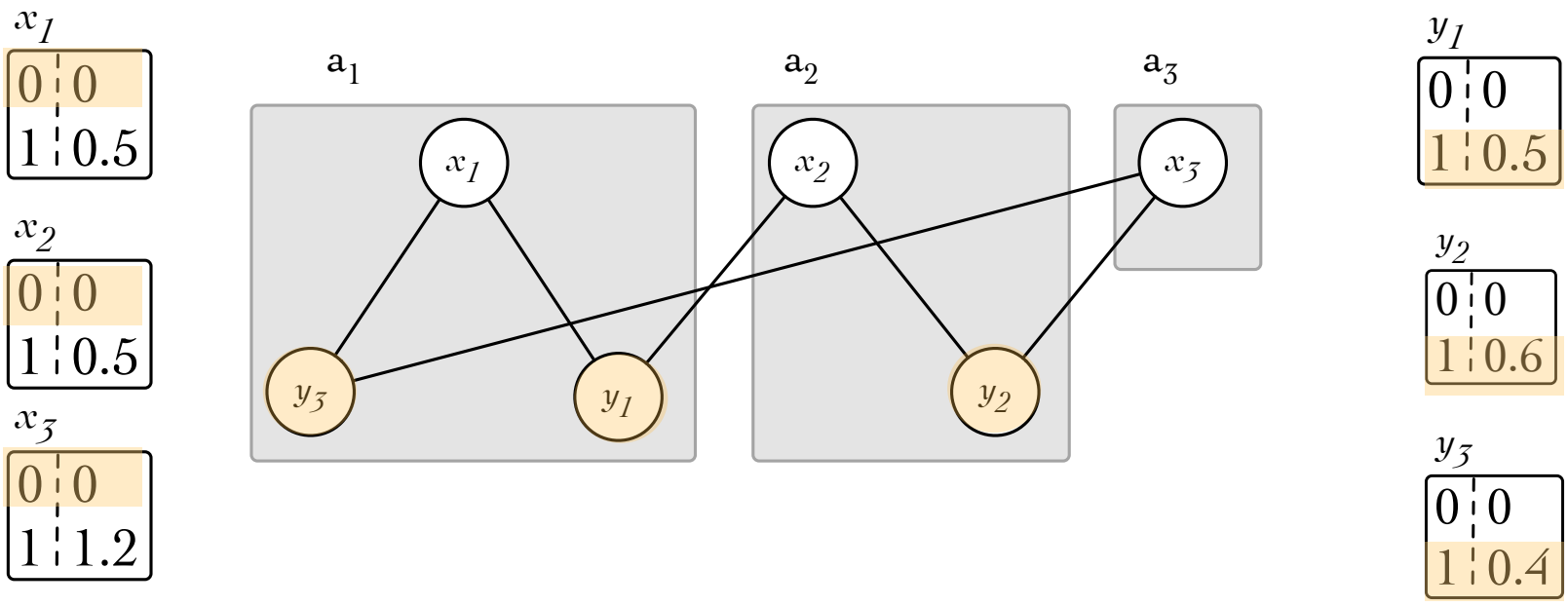
#3 Merge Gadget Graphs into a CCG

- Each agent merge their constructed gadget graphs into a CCG



CCG-MaxSum

- An MWVC on the CCG encodes an optimal solution of the Original DCOP



CCG-MaxSum

#4 Message Passing Phase

- Use an instance of the MaxSum algorithm to solve the MWVC encoding on the CCG
- Messages are passed between adjacent vertices

$$\mu^{i}_{u \to v} = \max \left\{ w_u - \sum_{t \in N(u) \setminus \{v\}} \mu^{i-1}_{t \to u}, 0 \right\}$$

CCG-MaxSum

#4 Message Passing Phase

- When the algorithm terminates, for a node v, if

$$w_v < \sum_{u \in N(v)} \mu_{u \to v}$$

  then v is included into the MWVC.

- Variables are assigned values 1 if they are in the MWVC and 0 otherwise.
- Extracting the values associated to the decision variables (X) gives a solution to the original DCOP

Evaluation

Algorithms

- CCG-MaxSum
- CCG-MaxSum with NT reduction
- MaxSum — with damping (0.7)
- DSA

Benchmarks

- Federated Social Network
- Random Benchmarks (grid, scale-free, random)

Evaluation: Federated Social Networks

- Multiple servers (agents) are used to store information of a social network
- Each server ai fetches from server a_j if a user in a_i follows a user in a_j
- Fetching strategies:
  - freq-fetch: fatches frequently and caches less (high bandwidth costs, low storage costs). xi = 1
  - more-cache, fetches less frequently and caches more (low bandwidth costs, high storage costs). xi = 0

$$\begin{cases} \alpha_{ij}(c_i^b + c_j^b), & \text{if } x_i = x_j = 1 \\ \alpha_{ij} c_i^s, & \text{otherwise} \end{cases} \qquad \begin{cases} (\alpha_{ij} + \alpha_{ji})(c_i^b + c_j^b), & \text{if } x_i = x_j = 1 \\ \alpha_{ij} c_i^s + \alpha_{ji} c_j^s, & \text{otherwise} \end{cases}$$

- c_i^b, c_i^s = bandwidth and storage costs
- α_{ij} = amount of information a_i need to fetch from a_j

Evaluation: Federated Social Networks

- Input: Twitter network data (456k nodes and 15M edges).
- 100, 500, 1000 servers

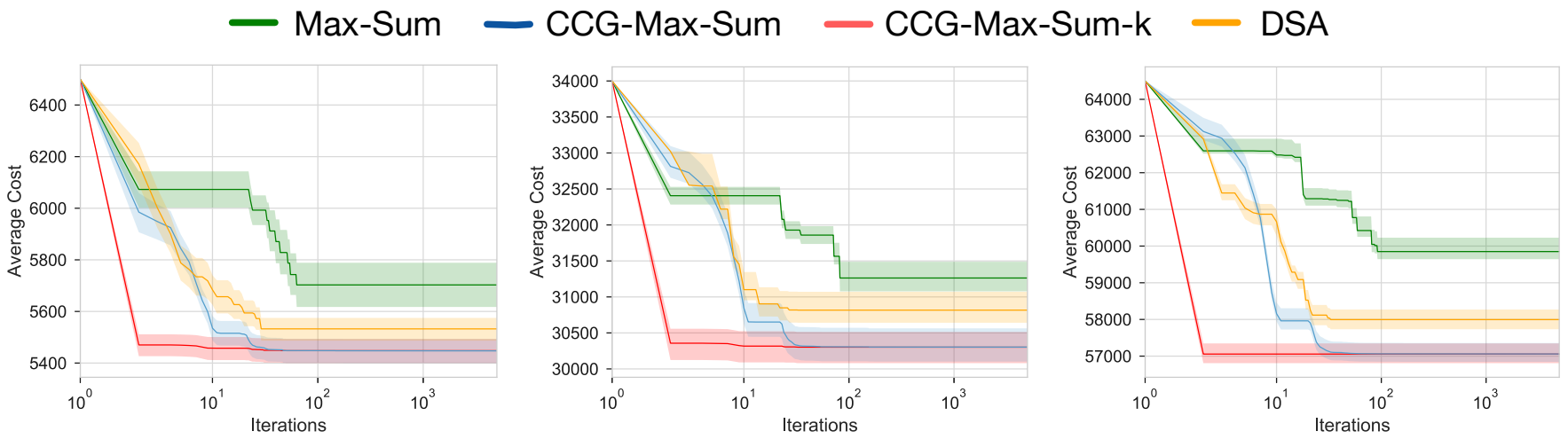
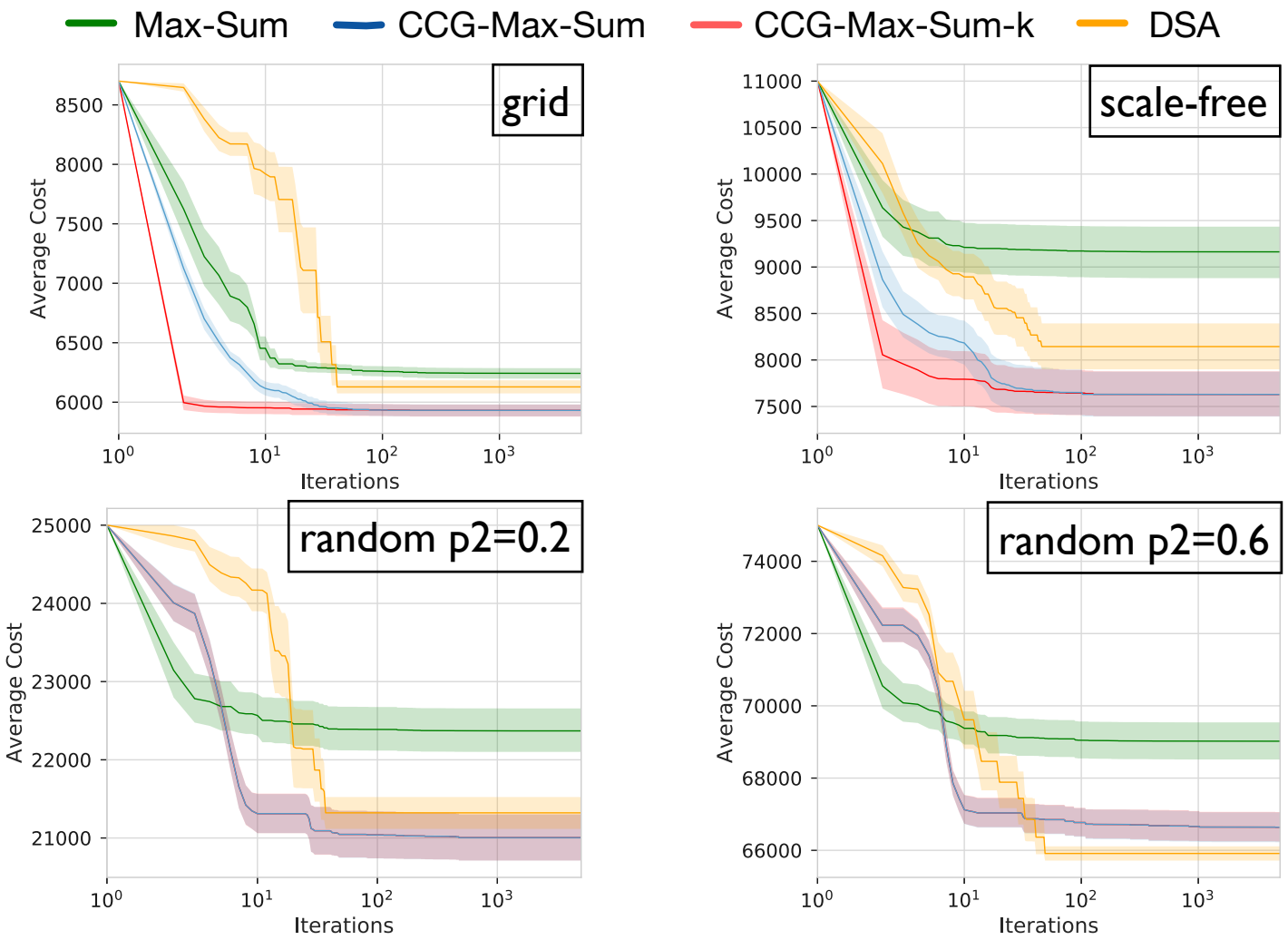


Fig. 5: FSN based on Twitter network data: 100 agents (left), 500 agents (center), and 1000 agents (right).

Evaluation: Random Benchmarks



Conclusions and Future Work

- Motivated by the exciting results obtained in the centralized setting
- Adapted the CCG for encoding DCOPs, a novel representation for multi-agent reasoning.
- Proposed CCG-MaxSum, to solve DCOPs on the CCG
- Results:
  - CCG MaxSum finds solutions of better qualities and within fewer iteration on several benchmarks.
- On-going/Future Work:
  - We believe this encoding can also be exploited with other classes of DCOP algorithms.
  - Extending this approach to non-Boolean DCOPs

Conclusions and Future Work

- Motivated by the exciting results obtained in the centralized setting
- Adapted the CCG for encoding DCOPs, a novel representation for multi-agent reasoning.
- Proposed CCG-MaxSum, to solve DCOPs on the CCG
- Results:
  - CCG MaxSum finds solutions of better qualities and within fewer iteration on several benchmarks.
- On-going/Future Work:
  - We believe this encoding can also be exploited with other classes of DCOP algorithms.
  - Extending this approach to non-Boolean DCOPs

Thank You!