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
Content

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

\(<X, D, F, A, \alpha>:\)

- **\(X\):** Set of variables.
- **\(D\):** Set of finite domains for each variable.
- **\(F\):** Set of constraints between variables.
- **\(A\):** Set of agents, controlling the variables in \(X\).
- **\(\alpha\):** Mapping of variables to agents.

**GOAL:** Find a cost minimal assignment.

\[
x^* = \arg\min_x \ F(x)
= \arg\min_x \sum_{f \in F} f(x|_{\text{scope}(f)})
\]
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

- **Complete**
  - Search
  - Inference
    - OPT-APO
    - AFB
    - ADPOT; BnB-ADPOT

- **Incomplete**
  - Search
  - Inference
    - PC-DPOP
    - DPOP and variants
  - Sampling
    - DSA
    - MGM
    - Max-Sum and variants
    - D-Gibbs
DCOP: Representation

Constraint Graph

Pseudo-Tree

Factor Graph
This work investigates the use of the CCG, an alternative representation, to solve DCOPs.

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

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

Represents explicitly both structures

- Can be constructed in polytime
The Constraint Composite Graph (CCG) [Kumar:08] is a graph

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

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]
The Nemhauser-Trotter (NT) Reduction

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

Minimize \( \sum_{i=1}^{|V|} w_i Z_i \)

\( \forall v_i \in V : Z_i \in \{0, 1\} \)

\( \forall (v_i, v_j) \in E : Z_i + Z_j \geq 1 \)
The Nemhauser-Trotter (NT) Reduction

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

\[
\text{Minimize } \sum_{i=1}^{\vert V \vert} w_i Z_i
\]

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

\forall (v_i, v_j) \in E : \ 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

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

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

\[ p_1(0, 0) = 0.5 \quad p_1(0, 1) = 0.6 \]
\[ p_1(1, 0) = 0.7 \quad p_1(1, 1) = 0.3 \]

\[ c_{00} = 0.5, \ c_{01} = 0.2, \ c_{10} = 0.1, \ c_{11} = -0.5. \]
# 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

### Example Polynomials

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$f_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

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

- $c_00 + c_{11}x_1x_2$
- $c_{01}x_1$
- $c_{10}x_2$
#2 Construct Gadget for each polynomial

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

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

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(f_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

\(c_{00} = 0.5, \ c_{01} = 0.2, \ c_{10} = 0.1, \ c_{11} = -0.5\).
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

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( f_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

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

\[
c_{00} = 0.5, \ c_{01} = 0.2, \ c_{10} = 0.1, \ c_{11} = -0.5.
\]
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

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(f_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

\(c_{00} = 0.5, \ c_{01} = 0.2, \ c_{10} = 0.1, \ c_{11} = -0.5\).
#2 Construct Gadget for each polynomial

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

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>0 1</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>1 0</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td>0.3</td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

- Construct a gadget graph for each term of a polynomial of each cost function.
#2 Construct Gadget for each polynomial

- Construct a “lifted representation” or a gadget graph for each term of a polynomial of each cost function.

\[
\begin{array}{ccc}
x_1 & x_2 & f_1 \\
0 & 0 & 0.5 \\
0 & 1 & 0.6 \\
1 & 0 & 0.7 \\
1 & 1 & 0.3 \\
\end{array}
\]

\[
\begin{array}{ccc}
x_1 & y_1 & \\
0 & 0 & \infty \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{ccc}
x_1 & x_2 & \\
0 & 0 & 0.2 \\
1 & 0 & 0.1 \\
\end{array}
\]

\[
\begin{array}{ccc}
x_1 & y_1 & \\
0 & 0 & 0.5 \\
\end{array}
\]

\[
\begin{array}{ccc}
0 & 0 \\
1 & 0 \\
\end{array}
\]
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

\[
\mu_{u \rightarrow v}^i = \max \left\{ w_u - \sum_{t \in N(u) \setminus \{v\}} \mu_{t \rightarrow u}^{i-1}, 0 \right\}
\]
When the algorithm terminates, for a node $v$, if

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

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 ($\mathcal{X}$) 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 $a_i$ fetches from server $a_j$ if a user in $a_i$ follows a user in $a_j$.
- Fetching strategies:
  - **freq-fetch**: fetches frequently and caches less (high bandwidth costs, low storage costs). $x_i = 1$
  - **more-cache**, fetches less frequently and caches more (low bandwidth costs, high storage costs). $x_i = 0$

\[
\begin{align*}
\alpha_{ij}(c_i^b + c_j^b), & \quad \text{if } x_i = x_j = 1 \\
\alpha_{ij} c_i^s, & \quad \text{otherwise}
\end{align*}
\]

\[
\begin{align*}
(\alpha_{ij} + \alpha_{ji})(c_i^b + c_j^b), & \quad \text{if } x_i = x_j = 1 \\
\alpha_{ij} c_i^s + \alpha_{ji} c_j^s, & \quad \text{otherwise}
\end{align*}
\]

- $c_i^b, c_i^s =$ bandwidth and storage costs
- $\alpha_{ij} =$ 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

Random DCOPs

We now discuss the solution cost of the algorithms on random minimization Boolean DCOPs. The costs of each assignment to the variables involved in a constraint are generated by sampling from the discrete uniform distribution $U(1, 100)$. For grid networks, we generate two-dimensional 10x10 grids and connect each node with its four nearest neighbors. For scale-free networks, we create an n-node network based on the Barabasi-Albert model [1]. Starting from a connected 2-node network, we repeatedly add a new node, randomly connecting it to two existing nodes. These two nodes are selected with probabilities that are proportional to the numbers of their incident edges. Finally, for random networks, we create an n-node network whose density $p_1$ produces $b_n(n-1)p_1$ edges. We report experiments on low-density problems ($p_1=0.2$) and high density problems ($p_1=0.6$) and fix the maximum constraint arity to 4. Constraints of arity 4 and 3, respectively, are generated by merging first all cliques of size 4 and then those of size 3. The other edges are used to generate binary constraints. In each configuration, we verify that the resulting constraint graph is connected and set the number of agents to 100.

The results are similar to the ones on FSN problems: The algorithms in order of their solution costs (from highest to lowest) tend to be: Max-Sum, DSA, and both CCG-Max-Sum variants, except on high-density random networks.
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!