AAAII-20 Tutorial on:

Multi-Agent Distributed Constrained Optimization

Ferdinando Fioretto
Syracuse University

William Yeoh
Washington University in St. Louis
Schedule

• 2:00pm: Preliminaries
• 2:20pm: DCOP Algorithms
• 3:00pm: DCOP Extensions
• 3:20pm: Applications
• 3:35pm: Challenges and Open Questions
• 3:45pm: Coffee!!
Lil’ Bit of Shameless Promotion :) 

- Tutorial materials are based on our recent JAIR survey paper:  
  
  Ferdinando Fioretto, Enrico Pontelli, and William Yeoh.  
  Distributed Constraint Optimization Problems and Applications: A Survey.  

- Includes more models, algorithms, and applications.
Preliminaries

AAAI-20 Tutorial on
Multi-Agent Distributed Constrained Optimization
MOTIVATING DOMAIN: SENSOR NETWORK
MOTIVATING DOMAIN: SENSOR NETWORK
MOTIVATING DOMAIN: SENSOR NETWORK
MOTIVATING DOMAIN: SENSOR NETWORK
MOTIVATING DOMAIN: SENSOR NETWORK

X1

X2

X3

X4

X5

X6
MOTIVATING DOMAIN: SENSOR NETWORK

Model the problem as a CSP

<table>
<thead>
<tr>
<th>x₁</th>
<th>x₃</th>
<th>x₅</th>
<th>Sat?</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>X</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>E</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>X</td>
</tr>
<tr>
<td>S</td>
<td>W</td>
<td>N</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>X</td>
</tr>
<tr>
<td>W</td>
<td>W</td>
<td>W</td>
<td>X</td>
</tr>
</tbody>
</table>
MOTIVATING DOMAIN: SENSOR NETWORK

Model the problem as a CSP
CSP
CONSTRAINT SATISFACTION

• Variables \( X = \{x_1, \ldots, x_n\} \)
• Domains \( D = \{D_1, \ldots, D_n\} \)
• Constraints \( C = \{c_1, \ldots, c_m\} \)
  where a constraint \( c_i \subseteq D_{i_1} \times D_{i_2} \times \ldots \times D_{i_n} \)
denotes the possible valid joint assignments for the variables \( x_{i_1}, x_{i_2}, \ldots, x_{i_n} \) it involves

• **GOAL:** Find an assignment to all variables that satisfies all the constraints
CSP
CONSTRAINT SATISFACTION

Model the problem as a CSP

<table>
<thead>
<tr>
<th>x1</th>
<th>x3</th>
<th>x5</th>
<th>Sat?</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>X</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>E</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>S</td>
<td>W</td>
<td>N</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Max-CSP
MAX CONSTRAINT SATISFACTION

Model the problem as a Max-CSP

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_3$</th>
<th>$x_5$</th>
<th>Sat?</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>X</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>E</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>W</td>
<td>N</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>W</td>
<td>W</td>
<td>X</td>
</tr>
</tbody>
</table>
Max-CSP

MAX CONSTRAINT SATISFACTION

• Variables $X = \{x_1, \ldots, x_n\}$
• Domains $D = \{D_1, \ldots, D_n\}$
• Constraints $C = \{c_1, \ldots, c_m\}$
  where a constraint $c_i \subseteq D_{i_1} \times D_{i_2} \times \ldots \times D_{i_n}$
  denotes the possible valid joint assignments for the variables $x_{i_1}, x_{i_2}, \ldots, x_{i_n}$ it involves

• **GOAL**: Find an assignment to all variables that satisfies a maximum number of constraints
Max-CSP
MAX CONSTRAINT SATISFACTION

Model the problem as a Max-CSP

<table>
<thead>
<tr>
<th>x_1</th>
<th>x_3</th>
<th>x_5</th>
<th>Sat?</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>X</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>E</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>S</td>
<td>W</td>
<td>N</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>W</td>
<td>W</td>
<td>W</td>
<td>X</td>
</tr>
</tbody>
</table>
Model the problem as a COP
WCSP (COP)  
CONSTRAINT OPTIMIZATION

- Variables  \( X = \{x_1, \ldots, x_n\} \)
- Domains  \( D = \{D_1, \ldots, D_n\} \)
- Constraints  \( C = \{c_1, \ldots, c_m\} \)
where a constraint  \( c_i : D_{i_1} \times \ldots \times D_{i_n} \rightarrow \mathbb{R}_+ \cup \{\infty\} \)
expresses the degree of constraint violation

- **GOAL**: Find an assignment that minimizes the sum of the costs of all the constraints
WCSP (COP) CONSTRAINT OPTIMIZATION

- Objective: maximize number of constraints satisfied
WCSP (COP)
CONSTRAINT OPTIMIZATION

- **CSP**
  - Hard constraints to
  - Soft constraints

- **Max-CSP**
  - Objective: maximize #constraints satisfied

- **COP**
  - Objective: minimize cost
Imagine that each sensor is an autonomous agent.

How should this problem be modeled and solved in a decentralized manner?
Multi-Agent Systems

- **Agent**: An entity that behaves autonomously in the pursuit of goals
- **Multi-agent system**: A system of multiple interacting agents

An agent is:
- **Autonomous**: Is of full control of itself
- **Interactive**: May communicate with other agents
- **Reactive**: Responds to changes in the environment or requests by other agents
- **Proactive**: Takes initiatives to achieve its goals
Imagine that each sensor is an autonomous agent.

How should this problem be modeled and solved in a decentralized manner?
DCOP
DISTRIBUTED CONSTRAINT OPTIMIZATION

\[ a_1 \rightarrow x_1 \]
\[ a_2 \rightarrow x_2 \]
\[ a_3 \rightarrow x_3 \]
\[ a_4 \rightarrow x_4 \]
\[ a_5 \rightarrow x_5 \]
\[ a_6 \rightarrow x_6 \]
DCOP
DISTRIBUTED CONSTRAINT OPTIMIZATION
DCOP
DISTRIBUTED CONSTRAINT OPTIMIZATION

• Agents \( A = \{a_i, \ldots, a_n\} \)
• Variables \( X = \{x_1, \ldots, x_n\} \)
• Domains \( D = \{D_1, \ldots, D_n\} \)
• Constraints \( C = \{c_1, \ldots, c_m\} \)
• Mapping of variables to agents

• **GOAL**: Find an assignment that minimizes the sum of the costs of all the constraints
DCOP
DISTRIBUTED CONSTRAINT OPTIMIZATION

- Hard constraints to Soft constraints
- Objective: minimize cost
- Objective: maximize #constraints satisfied
DCOP
DISTRIBUTED CONSTRAINT OPTIMIZATION

- Variables are controlled by agents
- Communication model
- Local agents’ knowledge
DCOP
DISTRIBUTED CONSTRAINT OPTIMIZATION

• Why distributed models?
  • Natural mapping for multi-agent systems
  • Potentially faster by exploiting parallelism
  • Potentially more robust: no single point of failure, no single network bottleneck
  • Maintains more private information
  • ...
DCOP Algorithms

AAAI-20 Tutorial on
Multi-Agent Distributed Constrained Optimization
DCOP Algorithms

Complete
- Partially Decentralized
  - Synchronous
    - Inference
    - Search
  - Asynchronous
    - Inference
    - Search

Incomplete
- Fully Decentralized
  - Synchronous
    - Sampling
    - Search
  - Asynchronous
    - Inference
DCOP Algorithms

- **Important Metrics:**
  - Agent complexity
  - Network loads
  - Message size

- Complete
  - Partially Decentralized
    - Synchronous
      - Search
      - Inference
    - Asynchronous
      - Search
  - Fully Decentralized
    - Synchronous
      - Inference
    - Asynchronous
      - Search

- Incomplete
  - Fully Decentralized
    - Synchronous
      - Sampling
    - Asynchronous
      - Search
      - Inference
DCOP Algorithms

Important Metrics:

- Agent complexity
- Network loads
- Message size
- Anytime
- Quality guarantees
- Execution time vs. solution quality
• Systematic process, divided in steps.
• Each agent waits for particular messages before acting.
• Consistent view of the search process.
• Typically, increases idle-time.
DCOP Algorithms

- Decision based on agents’ local state
- Agents’ actions do not depend on sequence of received messages
- Minimizes idle-time
- No guarantees on validity of local views
DCOP Algorithms

- **Complete**
  - Partially Decentralized
    - Synchronous
      - Search
      - Inference
  - Fully Decentralized
    - Synchronous
      - Search
      - Inference
    - Asynchronous
      - Search
      - Inference

- **Incomplete**
  - Fully Decentralized
    - Synchronous
      - Sampling
    - Asynchronous
      - Search
      - Inference

Synchronous Branch and Bound (SBB)
Katsutoshi Hirayama, Makoto Yokoo: Distributed Partial Constraint Satisfaction Problem. CP 1997: 222-236
How do we solve this distributedly?
• Agents operate on a complete ordering
• Agents exchange CPA messages containing partial assignments.
• When a solution is found, its solution cost as an UB is broadcasted to all agents.
• The UB is used for branch pruning.
SBB

UB = infinity
SBB

UB = infinity

A

B

5

C

D
SBB

UB = infinity
SBB

A

B

C

D

UB = 18
SBB

UB = 18
SBB

UB = 18
SBB

UB = 18

A

B

C

D

18 23
15 19
5
0
0

AAAII-20 Tutorials
UB = 18
SBB

UB = 18
SBB

UB = 18
SBB

UB = 18
SBB

UB = 18
<table>
<thead>
<tr>
<th>SBB</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Correct</strong></td>
<td>Yes</td>
</tr>
<tr>
<td>the solution it finds is optimal</td>
<td></td>
</tr>
<tr>
<td><strong>Complete</strong></td>
<td>Yes</td>
</tr>
<tr>
<td>it terminates</td>
<td></td>
</tr>
<tr>
<td><strong>Message Complexity</strong></td>
<td>$O(d)$</td>
</tr>
<tr>
<td>max size of a message</td>
<td></td>
</tr>
<tr>
<td><strong>Network Load</strong></td>
<td>$O(b^d)$</td>
</tr>
<tr>
<td>max number of messages</td>
<td></td>
</tr>
<tr>
<td><strong>Runtime</strong></td>
<td>$O(b^d)$</td>
</tr>
</tbody>
</table>
Can we speed this up by parallelizing some computations?

*Hint: Are there independent or conditionally independent subproblems?*
SBB
These computations are the same; independent of C!
Definition: A spanning tree of the constraint graph such that no two nodes in sibling subtrees share a constraint in the constraint graph.
DCOP Algorithms

Distributed Pseudotree Optimization Procedure (DPOP)
DPOP

- Extension of the Bucket Elimination (BE)
- Agents operate on a pseudo-tree ordering
- UTIL phase: Leaves to root
- VALUE phase: Root to leaves

DPOP

Pseudo-tree Ordering

<table>
<thead>
<tr>
<th>B</th>
<th>D</th>
<th>(B,D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>3</td>
</tr>
<tr>
<td>r</td>
<td>g</td>
<td>8</td>
</tr>
<tr>
<td>g</td>
<td>r</td>
<td>10</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>3</td>
</tr>
</tbody>
</table>

UTILITY
DPOP

Pseudo-tree Ordering

MSG to B

<table>
<thead>
<tr>
<th>B</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>3</td>
</tr>
<tr>
<td>g</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>D</th>
<th>(B,D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>3</td>
</tr>
<tr>
<td>r</td>
<td>g</td>
<td>8</td>
</tr>
<tr>
<td>g</td>
<td>r</td>
<td>10</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>3</td>
</tr>
</tbody>
</table>

\[
\min\{3, 8\} = 3 \\
\min\{10, 3\} = 3
\]
DPOP

Pseudo-tree Ordering

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>(B,C)</th>
<th>(A,C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>g</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>r</td>
<td>g</td>
<td>r</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>r</td>
<td>g</td>
<td>g</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>g</td>
<td>r</td>
<td>r</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>g</td>
<td>r</td>
<td>g</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>r</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>g</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
DPOP

Pseudo-tree Ordering

MSG to B

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>(B,C)</th>
<th>(A,C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>g</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>r</td>
<td>g</td>
<td>r</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>r</td>
<td>g</td>
<td>g</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>g</td>
<td>r</td>
<td>r</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>g</td>
<td>r</td>
<td>g</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>r</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>g</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

A | B | UTIL |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>10</td>
</tr>
<tr>
<td>r</td>
<td>g</td>
<td>8</td>
</tr>
<tr>
<td>g</td>
<td>r</td>
<td>7</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>6</td>
</tr>
</tbody>
</table>
DPOP

Pseudo-tree Ordering

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>(A,B)</th>
<th>Util C</th>
<th>Util D</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>5</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>r</td>
<td>g</td>
<td>8</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>g</td>
<td>r</td>
<td>20</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>
**DPOP**

**Pseudo-tree Ordering**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Util C</th>
<th>Util D</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>r</td>
<td>g</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>19</td>
</tr>
<tr>
<td>g</td>
<td>r</td>
<td>20</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>12</td>
</tr>
</tbody>
</table>

**MSG to A**

<table>
<thead>
<tr>
<th>A</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>18</td>
</tr>
<tr>
<td>g</td>
<td>12</td>
</tr>
</tbody>
</table>

**UTIL**
DPOP

Pseudo-tree Ordering

<table>
<thead>
<tr>
<th>A</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>18</td>
</tr>
<tr>
<td>g</td>
<td>12</td>
</tr>
</tbody>
</table>

optimal cost = 12
DPOP

Pseudo-tree Ordering

<table>
<thead>
<tr>
<th>A</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>18</td>
</tr>
<tr>
<td>g</td>
<td>12</td>
</tr>
</tbody>
</table>

- Select value for A = ‘g’
- Send MSG A = ‘g’ to agents B and C
DPOP

Pseudo-tree Ordering

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>(A,B)</th>
<th>Util C</th>
<th>Util D</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>5</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>r</td>
<td>g</td>
<td>8</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>g</td>
<td>r</td>
<td>20</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

• Select value for B = ‘g’
• Send MSG B = ‘g’ to agents C and D
DPOP

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>(B,C)</th>
<th>(A,C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>g</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>r</td>
<td>g</td>
<td>r</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>r</td>
<td>g</td>
<td>g</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>g</td>
<td>r</td>
<td>r</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>g</td>
<td>r</td>
<td>g</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>r</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>g</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

- Select value for C = ‘g’
DPOP

Pseudo-tree Ordering

<table>
<thead>
<tr>
<th>B</th>
<th>D</th>
<th>(B,D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>3</td>
</tr>
<tr>
<td>r</td>
<td>g</td>
<td>8</td>
</tr>
<tr>
<td>g</td>
<td>r</td>
<td>10</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>3</td>
</tr>
</tbody>
</table>

• Select value for $D = 'g'$
# DPOP

<table>
<thead>
<tr>
<th></th>
<th>SBB</th>
<th>DPOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>the solution it finds is optimal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complete</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>it terminates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message Complexity</td>
<td>$O(d)$</td>
<td>$O(b^d)$</td>
</tr>
<tr>
<td>max size of a message</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network Load</td>
<td>$O(b^d)$</td>
<td>$O(d)$</td>
</tr>
<tr>
<td>max number of messages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Runtime</td>
<td>$O(b^d)$</td>
<td>$O(b^d)$</td>
</tr>
</tbody>
</table>

branching factor $= b$
num variables $= d$
Critical Overview

Search Algorithms
- increasing memory
- polynomial
- exponential

Inference Algorithms
- decreasing network load
- exponential
- polynomial
DCOP Algorithms

- Complete
  - Partially Decentralized
    - Synchronous
      - Search
      - Inference
  - Fully Decentralized
    - Synchronous
      - Search
      - Inference
    - Asynchronous
      - Search
- Incomplete
  - Fully Decentralized
    - Synchronous
      - Sampling
    - Asynchronous
      - Search
      - Inference

Distributed Local Search
Local Search Algorithms


Local Search Algorithms

- DSA: Distributed Stochastic Algorithm
- MGM: Maximum Gain Messages Algorithm
- Every agent individually decides whether to change its value or not
- Decision involves:
  - knowing neighbors’ values
  - calculation of utility gain by changing values
  - probabilities

DSA Algorithm

• All agents execute the following
  • Randomly choose a value
  • while (termination is not met)
    • if (a new value is assigned)
      • send the new value to neighbors
    • collect neighbors’ new values if any
  • select and assign the next value based on assignment rule

DSA Algorithm

\[ \text{Utility}(A,B) = 5 \]
\[ \text{Utility}(B,C) = 5 \]

\[ \{ \} \]

\[ \{ \} \]

\[ \{ \} \]

\[ \begin{array}{ccc|cc}
 x_i & x_j & \text{Utility} (A,B) & \text{Utility} (B,C) \\
 \hline
 5 & 5 & 5 & 5 \\
 0 & 0 & 0 & 0 \\
 8 & 8 & 8 & 8 \\
\end{array} \]
**DSA Algorithm**

- $U=10, \Delta=10$
- $U=0, \Delta=0$

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$x_j$</th>
<th>Utility $(A,B)$</th>
<th>Utility $(B,C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

- $U=0, \Delta=8$
- $U=8, \Delta=8$
**DSA Algorithm**

- $U=10, \Delta=2$
- $U=8, \Delta=0$
- $U=0, \Delta=-8$
- $U=0, \Delta=0$
- $U=8, \Delta=8$

### Table

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$x_j$</th>
<th>Utility $(A,B)$</th>
<th>Utility $(B,C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
### DSA Algorithm

**Utility Table**

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$x_j$</th>
<th>Utility $(A,B)$</th>
<th>Utility $(B,C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

**Node Values**

- $U=0, \Delta=0$
- $U=5, \Delta=5$
- $U=16, \Delta=16$
- $U=0, \Delta=0$
### DSA Algorithm

**Utility Table**

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$x_j$</th>
<th>Utility $(A,B)$</th>
<th>Utility $(B,C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
One possible execution trace

### DSA Algorithm

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$x_j$</th>
<th>Utility $(A,B)$</th>
<th>Utility $(B,C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

### Utility Function

$$U(A,B) = 5$$

$$U(B,C) = 5$$
MGM Algorithm

• All agents execute the following
  • Randomly choose a value
  • while (termination is not met)
    • if (a new value is assigned)
      • send the new value to neighbors
    • collect neighbors’ new values if any
    • calculate gain and send it to neighbors
    • collect neighbors’ gains
    • if (it has the highest gain among all neighbors)
      • change value to the value that maximizes gain
MGM Algorithm

- All agents execute the following
  - Randomly choose a value
  - while (termination is not met)
    - if (a new value is more acceptable)
      - send the value to the neighboring agents
    - collect new values from neighbors
    - calculate gain and send it to neighbors
    - if (it has the highest gain among all neighbors)
      - change value to the value that maximizes gain

Great if you need an anytime algorithm

DCOP Extensions

AAAI-20 Tutorial on
Multi-Agent Distributed Constrained Optimization
Dynamic DCOP

• Why dynamic DCOPs?
  • MAS commonly exhibit dynamic environments
  • The capture scenarios with:
    • Moving agents, change of constraints, change of preferences
    • Additional information become available during problem solving
  • Application domains: Sensor networks, cloud computing, smart home automation, …
Dynamic DCOP

- A Dynamic DCOP is sequence $P_1, P_2, \ldots, P_k$ of $k$ DCOPs
- The agent knowledge about the environment is confined within each time step
- Each DCOP is solved sequentially
Dynamic DCOP

• Current proposals on Dynamic DCOPs have focused on balancing reactiveness vs. proactiveness
  • Reactive algorithms: React to changes of the environment as soon as they are observed
  • Proactive algorithms: Use predictions about future events to better react to the changes in the environment
Continuous DCOPs

- Solves problems in which variables domains are continuous
- Arbitrary constraints (e.g., non-convex)

Continuous DCOP algorithms

- Continuous Maxsum (CMS)
  Solve problems where constraints are linear piecewise functions
- Hybrid Continuous Maxsum (HCMS)
  No restriction on form of the functions
  Local optimization approach to improve quality
- C-DPOP (similar to DPOP but uses piecewise functions)
Distributed Convex Optimization

- Decentralized data and constraints
- Continuous variables
- Can describe many problems in optimization and learning
- Consider a (centralized) problem of the form

\[
\min_{x \in \mathbb{R}^n} f(x)
\]

s.t. \(g_i(x) \leq 0\) \(i = 1, \ldots, m\)

\(h_i(x) = 0\) \(i = 1, \ldots, p\)
Dual Problem

- The Lagrangian function \( L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \rightarrow \mathbb{R} \) is:

\[
L(x, \nu, \lambda) = f(x) + \sum_{i=1}^{m} \nu_i g_i(x) + \sum_{i=1}^{p} \lambda_i h_i(x)
\]

- Weighted sum of the constraint functions

- The Lagrangian dual function \( LD : \mathbb{R}^m \times \mathbb{R}^p \rightarrow \mathbb{R}_{\geq 0} \) is

\[
LD(\nu, \lambda) = \inf_{x \in \mathbb{R}^n} L(x, \nu, \lambda)
\]

- Main advantage: it is concave

Lagrangian Multipliers
Dual Problem

• Note that, for any feasible value \( \tilde{x} \) we have

\[
\sum_{i=1}^{m} \nu g_i(\tilde{x}) + \sum_{i=1}^{p} \lambda_i h_i(\tilde{x}) \leq 0
\]

• That is

\[
LD(\nu^*, \lambda^*) \leq L(\tilde{x}, \nu, \lambda) \leq f(\tilde{x})
\]

for any feasible \( \tilde{x} \)

Each minimum value of \( L(x, \nu) \) is less than or equal to \( f(x) \)
Dual Problem

• How to find the best lower bound?

• Lagrangian dual problem: \[ \max_{\nu, \lambda} LD(\nu, \lambda) \]
  \[
  \text{s.t. } \nu \geq 0
  \]

• Optimal Lagrangian multipliers: \((\nu^*, \lambda^*)\)

• Assuming strong duality: \[ x^* = \arg \min_x L(x, \nu^*, \lambda^*) \]
Dual Ascent

• (Consider only equality constraints for ease of notation)
• We solve the dual problem using gradient ascent
• Assuming LD is differentiable, the gradient $\nabla LD(\lambda)$ can be evaluated as:
  1. Find $x^+ = \arg\min_x L(x, \lambda)$
  2. Compute $\nabla LD(\lambda) = h(x^+) = Ax^+ - b$

Dual Ascent:

$$x^{k+1} = \arg\min_x L(x, \lambda^k)$$

$$\lambda^{k+1} = \lambda^k + s^k (Ax^{k+1} - b)$$

Step size $> 0$
Dual Ascent

- (Consider only equality constraints for ease of notation)
- We solve the dual problem using gradient ascent
- Assuming LD is differentiable, the gradient $\nabla LD(\lambda)$ can be evaluated as:

1. Find $x^+ = \arg \min_x L(x, \lambda)$
2. Compute $\nabla LD(\lambda) = h(x^+) = Ax^+ - b$

**Dual Ascent:**

$x^{k+1} = \arg \min_x L(x, \lambda^k)$

$\lambda^{k+1} = \lambda^k + s^k (Ax^{k+1} - b)$
Dual Ascent

• (Consider only equality constraints for ease of notation)
• We solve the dual problem using gradient ascent
• Assuming LD is differentiable, the gradient $\nabla LD(\lambda)$ can be evaluated as:
  1. Find $x^+ = \arg\min_x L(x, \lambda)$
  2. Compute $\nabla LD(\lambda) = h(x^+) = Ax^+ - b$

Dual Ascent:

$x^{k+1} = \arg\min_x L(x, \lambda^k)$

$\lambda^{k+1} = \lambda^k + s^k (A x^{k+1} - b)$

Dual variable update
Dual Decomposition

• Dual ascent can lead to a decentralized algorithm when $f$ is separable (as in DCOPs)

$$\sum_{i=1}^{n} f_i(x_i) \text{ s.t. } A_i x_i - b = 0 \quad (i \in [n])$$

• The Lagrangian can be written as:

$$L(x, \lambda) = \sum_{i=1}^{n} L_i(x_i, \lambda) = \sum_{i=1}^{n} f_i(x_i) + \lambda^T (A_i x_i^* b)$$

Dual Ascent:

$$x^{k+1} = \arg \min_x L(x, \lambda^k)$$

$$\lambda^{k+1} = \lambda^k + s^k (Ax^{k+1} - b)$$
Dual Decomposition

• Dual ascent can lead to a decentralized algorithm when $f$ is separable (as in DCOPs)

$$\sum_{i=1}^{n} f_i(x_i) \text{ s.t. } A_i x_i - b = 0 \ (i \in [n])$$

• The Lagrangian can be written as:

$$L(x, \lambda) = \sum_{i=1}^{n} L_i(x_i, \lambda) = \sum_{i=1}^{n} f_i(x_i) + \lambda^T(A_i x_i - b)$$

Dual Decomposition:

$$x^{k+1} = \arg \min_{x_i} L_i(x_i, \lambda^k), \ i \in [n]$$

$$\lambda^{k+1} = \lambda^k + s^k \left( \sum_{i=1}^{n} A_i x_i^{k+1} - b \right)$$

Decentralized x-minimization step (in parallel)
Dual Decomposition

• Dual ascent can lead to a decentralized algorithm when $f$ is separable (as in DCOPs)

$$\sum_{i=1}^{n} f_i(x_i) \text{ s.t. } A_i x_i - b = 0 \ (i \in [n])$$

• The Lagrangian can be written as:

$$L(x, \lambda) = \sum_{i=1}^{n} L_i(x_i, \lambda) = \sum_{i=1}^{n} f_i(x_i) + \lambda^T(A_i x_i - b)$$

**Dual Decomposition:**

$$x^{k+1} = \arg \min_{x_i} L_i(x_i, \lambda^k), \ i \in [n]$$

$$\lambda^{k+1} = \lambda^k + s^k (\sum_{i=1}^{n} A_i x_i^{k+1} - b)$$

Requires a “gather” step update and distribute the global $\lambda$ variable
Dual Decomposition

• If the step size is well chosen and other assumptions hold, then $x^k$ converges to an optimal point and $\lambda^k$ to an optimal dual point.

• However, these assumptions do not hold in many applications.
Method of the Multipliers

- Used to robustly dual ascent
- Uses the *Augmented Lagrangian*.

\[
\min_{x \in \mathbb{R}^n} f(x)
\]

\[
\text{s.t. } Ax - b = 0
\]

\[
L_\rho(x, \lambda) = f(x) + \lambda^T (Ax - b) + \frac{\rho}{2} \|Ax - b\|^2
\]

- Problem unchanged: has same local minima
- **Advantage:** The associated Lagrangian dual \( \text{LD}_\rho(\lambda) = \inf_x L_\rho(x, \lambda) \) can be shown to be differentiable.
Method of the Multipliers

• **Advantage:** The associated Lagrangian dual $\mathcal{L}_D(\lambda) = \inf_x L_\rho(x, \lambda)$ can be shown to be differentiable.

\[
x^{k+1} = \arg\min_x L_\rho(x, \lambda^k)
\]
\[
y^{k+1} = \lambda^k + \rho(Ax^{k+1} - b)
\]

Specific step size

• Compared to decomposition, converges under milder assumptions (f can be non-differentiable, take on $\pm \infty$ values)

• However, the quadratic penalty destroys the splitting of the x-update, so cannot be decomposed
Alternating Method of the Multipliers (ADMM)

• Support decomposition.

• Consider a problem of the form \((f, g, \text{convex})\)

\[
\begin{align*}
\text{minimize} & \quad f(x) + g(z) \\
\text{subject to} & \quad Ax + Bz = c
\end{align*}
\]

• Two sets of variables with separate objective

\[
L_p(x, z, \lambda) = f(x) + g(z) + \lambda^T (Ax + Bz - c) + \frac{\rho}{2} \| Ax + Bz - c \|^2_2
\]
Alternating Method of the Multipliers (ADMM)

• Support decomposition.

• Consider a problem of the form \((f, g, \text{convex})\)

\[
\min f(x) + g(z) \\
\text{subject to} \quad Ax + Bz = c
\]

• Two sets of variables with separate objective

\[
L_p(x, z, \lambda) = f(x) + g(z) + \lambda^T (Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|_2^2
\]

ADMM

\[
x^{k+1} = \arg\min_x L_\rho(x, z^k, \lambda^k) \quad \text{x-minimization}
\]

\[
z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, \lambda^k) \quad \text{z-minimization}
\]

\[
\lambda^{k+1} = \lambda^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \quad \text{dual update}
\]
Alternating Method of the Multipliers (ADMM)

- If we minimized over $x$ and $z$ jointly, reduces to method of the multipliers.
- We can decompose because we minimize over $x$ with fixed $z$, and vice-versa.

\[
\begin{align*}
    x^{k+1} &= \arg\min_x L_\rho(x, z^k, \lambda^k) & \text{x-minimization} \\
    z^{k+1} &= \arg\min_z L_\rho(x^{k+1}, z, \lambda^k) & \text{z-minimization} \\
    \lambda^{k+1} &= \lambda^k + \rho(Ax^{k+1} + Bz^{k+1} - c) & \text{dual update}
\end{align*}
\]

ADMM
Convergence
(ADMM)

• Mild assumptions
  • $f, g$, convex, closed, proper

• Then ADMM converges:
  • Iterates approach feasibility $Ax^k + Bz^k - c \to 0$
  • Objective approaches optimal value: $f(x^k) + g(z^k) \to \text{OPT}$
Applications

AAAI-20 Tutorial on
Multi-Agent Distributed Constrained Optimization
DCOP APPLICATIONS

• Scheduling Problems
  • Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Multi-Event Scheduling. AAMAS 2004

• Radio Frequency Allocation Problems
  • Improving DPOP with Branch Consistency for Solving Distributed Constraint Optimization Problems. CP 2014

• Sensor Networks
  • Preprocessing techniques for accelerating the DCOP algorithm ADOPT. AAMAS 2005

• Home Automation
  • A Multiagent System Approach to Scheduling Devices in Smart Homes. AAMAS 2017, IJCAI 2016

• Traffic Light Synchronization
  • Evaluating the performance of DCOP algorithms in a real world, dynamic problem. AAMAS 2008

• Disaster Evacuation
  • Disaster Evacuation Support. AAAI 2007; JAIR 2017

• Combinatorial Auction Winner Determination
  • H-DPOP: Using Hard Constraints for Search Space Pruning in DCOP. AAAI 2008
Meeting Scheduling

• Values: time slots to hold the meetings

• All agents participating in a meeting must meet at the same time

• All meetings of an agent must occur at different times

Edge Computing

Edge Computing

Khoi D. Hoang, Christabel Wayllace, William Yeoh, Jacob Beal, Soura Dasgupta, Yuanqiu Mo, Aaron Paulos, Jon Schewe:
New Distributed Constraint Reasoning Algorithms for Load Balancing in Edge Computing, PRIMA 2019: 69-86
Edge Computing

- Agents: $a_v$ for each vertex $v \in V$
- Variables: $v(v, s, c)$
  - Denotes amount of load to serve for service $s$ by client $c$ on vertex $v$
  - Each controlled by agent $a_v$
- Domain: $0 \leq D(v, s, c) \leq \text{cap}(v)$
- Constraints:
  - $\sum_{(s \in S, c \in C)} v(v, s, c) \leq \text{cap}(v)$
  - $\sum_{(v \in V \ s \in S, c \in C)} v(v, s, c) \geq \sum_{(v \in V \ s \in S, c \in C)} \text{load}(v,s,c)$
- Maximize: $\sum_{(v \in V \ s \in S, c \in C)} v(v, s, c) / \text{dist}(v,c)$
demands are served by an energy provider. We assume that 

We now provide a description of the Smart Homes Device Scheduling problem. Let us consider a building with a set of devices, each characterized by a cost function and a schedule. The goal is to find a schedule for these devices that minimizes the total cost, subject to certain constraints.

Definition 1: The Smart Building Devices Scheduling (SBDS) problem is the problem of scheduling the devices of each building in the neighborhood in a coordinated fashion so as to minimize the monetary costs and, at the same time, ensure user comfort.

\[ \min_{x} \sum_{i} F_i(x) \]

subject to

\[ \sum_{j} E_{ij}(x) = 1 \quad \forall i \]

where

- \( F_i(x) \) represents the cost of device \( i \) at time step \( t \)
- \( E_{ij}(x) \) is the interaction cost between device \( i \) and device \( j \)

The SBDS problem is the problem of scheduling the devices of each building in the neighborhood in a coordinated fashion so as to minimize the monetary costs and, at the same time, ensure user comfort.
A **smart home** has:

- **Smart devices** (roomba, HVAC) that it can control
- **Sensors** (cleanliness, temperature)
- A set of locations

Smart Homes Device Scheduling

Challenges and Open Questions

AAAI-20 Tutorial on
Multi-Agent Distributed Constrained Optimization
MAS Coordination

• Decentralized coordination in a MAS is expensive


MAS Coordination

• Decentralized coordination in a MAS is expensive
  • Can we study various tradeoff (solution quality vs. runtime vs. communication time) to improve coordination?
  • Can we use sampling methods to develop new, efficient, anytime, DCOP incomplete algorithms?

Dynamic Environment

• Interaction in a dynamic environment is required to be robust to several changes

Dynamic Environment

• Interaction in a dynamic environment is required to be robust to several changes
  • How do agents respond to dynamic changes?
  • Can we study adaptive algorithms so that the MAS interaction is resilient and adaptive to changes in the communication layer, the underlying constraint graph, etc.?

Agent Preferences

• How to model, learn, and update agent preferences?
Agent Preferences

• How to model, learn, and update agent preferences?
  • Agent’s preferences are assumed to be available. This is not always feasible. How to efficiently elicit agents’ preferences?
  • When full elicitation is not possible, how to adaptively learn the preference of an agent?

Thank You!
Ferdinando Fioretto & William Yeoh