Distributed Constraint Reasoning

Makoto Yokoo
Kyushu University, Japan
yokoo@inf.kyushu-u.ac.jp
http://agent.inf.kyushu-u.ac.jp/~yokoo/
Outline

• Constraint Satisfaction Problem (CSP)
  – Formalization
  – Algorithms
• Distributed Constraint Satisfaction Problem (Dis-CSP)
  – Formalization
  – Algorithms
• Distributed Constraint Optimization Problem (DCOP)
  – Formalization
  – Algorithms
• Advanced Topics
  – Coalition Structure Generation based on DCOP
Example: Constraint Satisfaction Problem (8-queens)

Goal: place eight chess queens on a chess board so that these queens do not threaten each other
Example: Constraint Satisfaction Problem (map coloring)

- **Goal:** color these regions using three colors (green, red, yellow), so that adjoining regions have different colors.
Characteristic of these Problems

• Find a combinatorial structure that satisfies given conditions (constraints)
  – position of queen1, position of queen2, ...
  – color of region1, color of region 2, ...

We need to solve such problems in various application areas (e.g., resource allocation, diagnosis/interpretation, design, planning).
Formal Definition of Constraint Satisfaction Problem (CSP)

Definition:
- a set of variables $x_1, x_2, ..., x_n$ which take their values from finite, discrete domains $D_1, D_2, ..., D_n$
- a set of constraints; a constraint is represented as a predicate (e.g., $p_{ij}(x_i, x_j)$)

Goal:
- to find the value assignment of the variables that satisfies all constraints (NP-complete)
Example: alphametic

• How should we choose variables, domains, and constraints?
• How big is the search space?

SEND
+MORE
MONEY
What people might say on CSPs

• *Oh, please no more puzzles, I’m fed up with AI toy problems!*  
  – CSP techniques scale up well. They can solve fairly large real-life problems.

• **The formalization is too simple, my problem has non Boolean constrains, continuous domains, etc.**  
  – There is a trade-off between the expressive power of the formalization and the efficiency of the algorithms.
What does CSP Provide for MAS?

General framework and general algorithms
Assume that variables are distributed among agents...
• Finding a value assignment that satisfies inter-agent constraints can be viewed as achieving coherence/consistency among agents.
  – infrastructure of cooperation
• Various application problems in MAS can be formalized as (distributed) CSPs by extracting an essential part of the problem.
• There exists a variety of ready-made algorithms for solving CSPs.
Algorithms for solving CSPs

- backtracking
- iterative improvement
- hybrid algorithms (weak-commitment, etc.)
- consistency algorithm
Backtracking

Basic ideas

• When the number of variables is $n$, the size of the domain of each variable is $d$, if we check all possible combinations, we need to check $d^n$ possibilities --- impossible.

• Try to find a solution of a small part of the problem (partial solution), then expand it to find a solution of the original problem.

• If one combination is not a solution to the partial solution, it cannot be a solution of the original problem.
Backtracking

• A partial solution is expanded by adding new variables one by one.
• When for one variable, no value satisfies the constraints between the partial solution, the value of most recently added variable is changed (backtracking).

\[ \times \times \times \times \]
Exercise: 7-queens

• Find a solution by backtracking, from top to bottom (variables) and from left to right (values).
Heuristics for Backtracking

Backtracking is a simple depth-first tree search algorithm, but many things must be considered to improve efficiency.

- variable/value ordering
- reducing redundant computation
- early detection of dead-ends
- smart (intelligent) backtracking
Min-conflict Backtracking (Minton, et al. AAAI92)

• Each variable has a tentative initial value.
• The *tentative initial value is revised* when the variable is added to the partial solution such that the new value satisfies:
  – all of the constraints between the partial solution
  – as many constraints between tentative initial values as possible (*min-conflict heuristic*).

*complete, but slow when a bad partial solution is constructed*
6-queens

• Find a solution by min-conflict backtracking.
Completeness of the Algorithm

• The algorithm is guaranteed to find a solution if one exists.
• If there exists no solution, the algorithm finds out the fact and terminates.

To guarantee the completeness, we need to search systematically, or need to have an exponential size of history.
Iterative Improvement

Basic Idea

• give up completeness
• do not construct a consistent partial solution
• strolling in $m^n$ search space
• change a variable value one by one so that the number of constraint violations is decreased
Iterative Improvement

• Chance to be trapped in a local minimum
  – The number of constraint violations cannot be decreased by changing any single variable value.

• Method for escaping form local minima:
  – give up and restart from a new initial value assignments
  – add noise
  – change the weight of constraints (breakout)

• A mistake can be revised without an exhaustive search.

*efficient, but not complete*
Exercise: 6-queens

• Find a solution by iterative improvement algorithm.
Hybrid Algorithm: Weak-commitment Search (Yokoo, AAAI94)

- Each variable has a tentative initial value.
- A consistent partial solution is constructed.
- When no consistent value exists for a variable with the partial solution, the whole partial solution is abandoned.
- The search process is restarted using the current assignment of variable values as new tentative initial values.
- The abandoned partial solution is recorded as a new nogood.

complete, can be efficient
Example of Algorithm Execution (weak-commitment)

- Black dot: in the partial solution
- Green dot: not part of the partial solution, satisfies all constraints
- Red dot: not part of the partial solution, violates some constraint
Consistency Algorithm

• A preprocessing procedure executed before search to reduce wasteful backtracking
  - Removes the value that cannot be a part of a final solution

2-consistency: tries to guarantee that for each value $v_i$ of variable $x_i$, for another variable $x_j$, there exists at least one value $x_j = v_j$, such that it is consistent with $x_i = v_i \Rightarrow$ removes $v_i$ if this is not true.

• After achieving 2-consistency:
  - Case i: a variable has an empty domain $\Rightarrow$ no consistent solution exists.
  - Case ii: the domain of each variable has exactly one value: their combination is a solution.
  - Case iii: we are not sure whether there exists a consistent solution or not.
procedure revise \((i, j)\);
For all \(v_i \in D_i\) do

\(\text{If there is no value } v_j \in D_j \text{ such that } v_j \text{ is consistent with } v_i,\)

then delete \(v_i\) from \(D_i\); end;
Example: crossword puzzle

- A set of candidate words are given.
- Find an assignment of words for each column or row.
- A word can be used at most once.

word candidates:

AFT, ALE, LEE, EEL, TIE
LINE, HEEL, HIKE, KEEL, KNOT
HOSES, LASER, SAILS, SHEET, STEER
Formalization as CSP

- Each variable corresponds to each row/column (8 variables).
- The domain of a variable is a set of candidate words in which the number of letters matches.
  1ACROSS: HOSES, LASER, SAILS, SHEET, STEER
  2DOWN: HOSES, LASER, SAILS, SHEET, STEER
  3DOWN: HOSES, LASER, SAILS, SHEET, STEER
  4ACROSS: LINE, HEEL, HIKE, KEEL, KNOT
  5DOWN: LINE, HEEL, HIKE, KEEL, KNOT
  6DOWN: AFT, ALE, LEE, EEL, TIE
  7ACROSS: AFT, ALE, LEE, EEL, TIE
  8ACROSS: HOSES, LASER, SAILS, SHEET, STEER
- The size of the search space is $5^8 = 390625$
Exercise: solve crossword puzzle by 2-consistency

variables:
1ACROSS: HOSES, LASER, SAILS, SHEET, STEER
2DOWN: HOSES, LASER, SAILS, SHEET, STEER
3DOWN: HOSES, LASER, SAILS, SHEET, STEER
4ACROSS: LINE, HEEL, HIKE, KEEL, KNOT
5DOWN: LINE, HEEL, HIKE, KEEL, KNOT
6DOWN: AFT, ALE, LEE, EEL, TIE
7ACROSS: AFT, ALE, LEE, EEL, TIE
8ACROSS: HOSES, LASER, SAILS, SHEET, STEER

constraints:
1ACROSS[3] = 2DOWN[1]
1ACROSS[5] = 3DOWN[1]
Exercise: solve crossword puzzle by 2-consistency

1ACROSS: HOSES, LASER, SAILS, SHEET, STEER
2DOWN: HOSES, LASER, SAILS, SHEET, STEER

2-consistent!

1ACROSS[3] = 2DOWN[1]
Solution

The contents of this slide is removed for the handout...
Forward Checking

• Apply partial 2-consistency during the execution of backtracking.
Forward Checking

• For each variable (which are not in the partial solution yet), we maintain the list of values that are consistent with the current partial solution.
• If for one variable, the list becomes empty, the current partial solution cannot be a final solution.
• If for one variable, the list contains only one value, we can determine the value of the variable immediately (unit variable).
• If there exists no unit variable, we should choose a variable whose list is shortest (first-fail principle).
First-fail Principle

• Intuitively, when solving a CSP, we should determine the value of a variable that is most constrained.

• Assume we are going to Tokyo from here via Beijing Airport.
  – We should determine the schedule of a flight to Tokyo first.
  – It is wasteful to consider how to reach the airport or how to reach the bus stop before we fix the flight schedule.
6-queens

• Solve by Forward-checking.
6-queens
6-queens
6-queens
6-queens
6-queens
6-queens
Example: Sudoku

- Place numbers from 1 to 9 in each cell.
- For each row/column/group, one number occurs only once.
- How to represent this problem as a CSP?

```
<table>
<thead>
<tr>
<th></th>
<th>8</th>
<th>2</th>
<th></th>
<th>7</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>1</td>
<td></td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td></td>
<td>3</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>6</td>
<td>5</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>9</td>
<td>8</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
```
Formalization

• Assume there exists a variable for each empty cell, whose domain is \{1, 2, 3, \ldots, 9\}.

• There exists a non-equality constraint between any pair of variables in the same row/column/group.
Example: Sudoku

- We can determine the value of a unit variable immediately.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
### Advanced Constraint Propagation

- Exactly one of A, B, C, D, E should be 1.
- However, B, C, D, E cannot be 1.
- Thus, A should be 1.

<table>
<thead>
<tr>
<th>A</th>
<th>8 2</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>B C</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>D E</td>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 8</td>
<td>7</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>6</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```plaintext
<table>
<thead>
<tr>
<th>A</th>
<th>8 2</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>B C</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>D E</td>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 8</td>
<td>7</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>6</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Exercise: Sudoku

• Let’s solve!
Solution

The contents of this slide is removed for the handout...
Example: Sudoku (super-hard)

• Cannot be solved by well-known techniques.
Outline

• Constraint Satisfaction Problem (CSP)
  – Formalization
  – Algorithms
• Distributed Constraint Satisfaction Problem (Dis-CSP)
  – Formalization
  – Algorithms
• Distributed Constraint Optimization Problem (DCOP)
  – Formalization
  – Algorithms
• Advanced Topics
  – Coalition Structure Generation based on DCOP
Distributed Constraint Satisfaction Problem (DisCSP)

Definition:

- There exist a set of agents 1, 2, ..., n
- Each agent has one or multiple variables.
- There exist intra/inter-agent constraints.

Assumptions:

- Communication between agents is done by sending messages.
- The delay is finite, though random.
- Each agent has only partial knowledge of the problem.
Distributed CSP≠Parallel Processing

• In parallel processing, we are concerned with efficiency.
  – *We can choose any parallel architecture to solve the problem efficiently.*

• In a Distributed CSP, a situation in which the problem is distributed among automated agents already exists.
  – *We have to solve the problem in this given situation.*
Applications of Distributed CSP (and Distributed Constraint Optimization Problem)

• Resource allocation
  – Resource allocation in communication network
  – Distributed Sensor network

• Scheduling
  – Nurse Time-tabling
  – Meeting Scheduling

• Planning/controlling
  – Evacuation planning
  – Surveillance
  – Smart grid
Resource Allocation in a Distributed Communication Network

• [Conry, et al. IEEE SMC91]
• Each region is controlled by an agent.
• The agents assign communication links cooperatively.

Can be formalized as a distributed CSP

– An agent has variables which represent requests.
– The domain of a variable is possible plans for satisfying a request.
– Goal: find a value assignment that satisfies resource constraints.
Distributed Sensor Network

• Multiple geographically distributed sensors are tracking a moving target.

• To identify the position of the target, these sensors must coordinate their activities.
Nurse Time-tabling Task

- [Solotorevsky & Gudes CP-96]
- Assign nurses to shifts of each department
- The time-table of each department is basically independent
- Inter-agent constraint: transportation
- A real problem, 10 departments, 20 nurses for each department, 100 weekly assignments, was solved.

Department A
morning: nurse1, nurse3, ..
afternoon: ...
night: ...

Department B
morning: nurse2, nurse4, ..
afternoon: ...
night: ..
Meeting Scheduling

Window 13:00 – 20:00
Duration 1h

Better after 18:00

Window 15:00 – 18:00
Duration 2h

• Why decentralize
  – Privacy
Evacuation Planning

• [Las, et al. AAMAS-08]
• Assign people to shelters under various constraints.
Surveillance

- [Rogers, et al. SOAR-09]
- Event Detection
  - Vehicles passing on a road
- Energy Constraints
  - Sense/Sleep modes
  - Recharge when sleeping
- Coordination
  - Activity can be detected by single sensor
  - Roads have different traffic loads
- Aim
  - Focus on road with more traffic load
Smart Grid

• [Kumar, et al. AAMAS-09]
• Distributed multiple generators coordinate their activities to satisfy various constraints.
Algorithms for Solving Distributed CSP

- synchronous backtracking
- asynchronous backtracking
- asynchronous weak-commitment search
- distributed breakout

Assumptions for Simplicity
- All constraints are binary.
- Each agent has exactly one variable.
Synchronous Backtracking

- Simulate centralized backtracking by sending messages
- If the constraint network has a tree like shape, agents in different branches can act concurrently (Collin, et al., IJCAI91)

**Drawback:** agents must act in a predefined sequential order; global knowledge is required
Asynchronous Backtracking
(Y, Durfee, Ishida, Kuwabara, ICDCS-92, IEEE TDKE-98)

Characteristics:
– Each agent acts asynchronously and concurrently without any global control.
  • Each agent communicates the tentative value assignment to related agents, then negotiates if constraint violations exist.

Merit:
– no communication/processing bottleneck, parallelism, privacy/security
Research Issues

• If agents act concurrently and asynchronously, guaranteeing the completeness is rather difficult.
  – If a solution exists, agents will find it.
  – If there is no solution, agents eventually find this out and terminate.

• To guarantee completeness, we must make sure that agents do not:
  • fall into an infinite processing loop,
  • stack in dead-ends.
Avoiding Infinite Processing Loops

Cause of infinite processing loops:
- cycle in the constraint network
- If there exists no cycle, an infinite processing loop never occurs.

Remedy:
- directing links without creating cycles
- use priority ordering among agents

![Diagram showing the cycle and its removal with priority ordering]
Escaping from Dead-Ends

When there exists no value that satisfies constraints: derive/communicate a new constraint (nogood)
– other agents try to satisfy the new constraint; thus the nogood sending agent can escape from the dead-end
– can be done concurrently and asynchronously

\[ x_2 \neq \{1, 2\} \]

\[ \text{(nogood, } \{(x_1, 1), (x_2, 2)\}) \]
Asynchronous Weak-commitment Search (Yokoo, CP95)

**Main cause of inefficiency of asynchronous backtracking:**
- Convergence to a solution becomes slow when the decisions of higher priority agents are poor; the decisions cannot be revised without an exhaustive search.

**Remedy:**
- Introduce dynamic change of the priority order, so that agents can revise poor decisions without an exhaustive search:
  - If a agent becomes a dead-end situation, the priority of the dead-end agent becomes higher.
Dynamically Changing Priority Order

• Define a non-negative integer value (priority value) representing the priority order of a variable/agent.
  – A variable/agent with a larger priority value has higher priority.
• Ties are broken using alphabetical order.
• Initial priority values are 0.
• The priority value of a dead-end agent is changed to $m+1$, where $m$ is the largest priority value of related agents.
Distributed Breakout
(Y. & Hirayama, *ICMAS-96, AIJ-05*)

**Key ideas**

- **mutual exclusion among neighbors**
  - If two neighboring agents move simultaneously, agents cannot guarantee that the number of constraint violations is reduced.
  - If only one agent can change its value at a time, agents cannot take advantage of parallelism.

- **weight change at quasi-local-minimum**
  - To detect the fact that agents as a whole are in a local-minimum, the agents have to globally exchange information among themselves.
Mutual Exclusion using the Degree of Improvements

• Neighboring agents exchange values of possible improvements

• Only the agent that can maximally improve the value within the neighbors is given the right to change its value (ties are broken using the agent identifiers)

*Non-neighbors can change their values simultaneously.*
Quasi-local-minimum

**weaker (locally detectable) condition than real local-minimum**

A state is a quasi-local-minimum from $x_i$’s viewpoint iff:

– $x_i$ is violating some constraint, and the possible improvements of $x_i$ and all of its neighbors are 0.
Example of Algorithm Execution

(a) $x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6$

(b) $x_1 \ x_2 \ x_5 \ x_6 \ x_3 \ x_4$

(c) $x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6$

(d) $x_2 \ x_5 \ x_6 \ x_3 \ x_4 \ x_1$
Outline

• Constraint Satisfaction Problem (CSP)
  – Formalization
  – Algorithms
• Distributed Constraint Satisfaction Problem (Dis-CSP)
  – Formalization
  – Algorithms
• Distributed Constraint Optimization Problem (DCOP)
  – Formalization
  – Algorithms
• Advanced Topics
  – Coalition Structure Generation based on DCOP
Distributed Constraint Optimization Problem (DCOP)

• In a standard CSP, each constraint and nogood is Boolean (satisfied or not satisfied).

• We generalize the notion of a constraint so that a cost is associated with it:
  – e.g., choosing \( x_1 = x_2 = \text{red} \) is cost 10, while choosing \( x_1 = x_2 = \text{blue} \) is cost 15.

• The goal is to find a solution with a minimal total cost.

• A standard (Dis) CSP is a special case where the cost is either 0 or infinity.
DCOP Algorithms

• Complete algorithms
  – ADOPT [Modi, et al., 2003]
  – DPOP [Petcu and Faltings, 2005]

• Incomplete algorithms
  – p-optimality algorithm
    [Okimoto, et al., 2011]
Depth-first Search (DFS) tree (pseudo-tree)

- Defined based on the identifiers of agents.
- “1” becomes the root node.
- Connected agents who have smaller identifiers are ancestors.
- The closest one is the parent.
- An edge between a non-parent ancestor is called back-edge.
- No edges between different branches.
Basic Terms (2/2)

- **Induced width (tree width):** the maximum number of back-edges + 1 (ancestors must be connected)
  - Parameter how close a given graph is to a tree.
  - Induced width of a graph is one, it is a tree.
  - Induced width of a complete graph with $n$ variables is $n-1$.

- **Example**

```
  1
 / \
1   2
 |   |
3   4
 |   |
5   2
```

Induced width = 3

Connect 5’s ancestors
ADOPT (Asynchronous Distributed OPTimization) Algorithm  
(Modi, Shen, Tambe, & Y. AIJ-05)

Characteristics:

– Fully asynchronous; each agent acts asynchronously and concurrently.
– Can guarantee to find an optimal solution
– Require only polynomial memory space

First algorithm that satisfies these characteristics

Key Ideas:

– A nogood is generalized for optimization.
– Perform an opportunistic best-first search based on (generalized) nogoods for a DFS tree.
Generalized Nogood

Associate a threshold for each nogood, e.g.,:
\[
[\{(x_1, r),(x_5, r)\}, 10],
\]
\{(x_1, r),(x_5, r)\} is a nogood, if we want a solution whose
cost is less than 10

Resolve a new nogood as follows:
• for red: \[\{(x_1, r),(x_4, r)\}, 10\]
• for yellow: \[\{(x_2, y),(x_4, y)\}, 7\]
• for green: \[\{(x_3, g),(x_4, g)\}, 8\]
• then, \[\{(x_1,r), (x_2,y), (x_3,g)\}, 7\],
  where 7 is a minimal value among 10, 7, & 8.

Nogoods and thresholds increase monotonically!
Opportunistic Best-first Search in ADOPT

• Each agent assigns a value that minimizes the cost based on currently available information.

• The information of the total cost is aggregated/communicated via generalized nogoods.

• Agents eventually reach an optimal solution.

• Some nogoods can be thrown away after aggregation, thus the memory space requirement is polynomial.

• Utilize threshold values to avoid excessive value changes.
ADOPT: Performance

- Asynchronous algorithm.
- Guaranteed to find an optimal solution.
- Each message has a linear size.
- The required memory space for each agent is also linear.
- The number of total messages can be exponential.
DPOP: Distributed Pseudo-tree Optimization Procedure (Petcu & Faltings, IJCAI-2005)

• Perform Dynamic Programming style propagation from leaf nodes toward the root node in the DFS tree.

• Then, the root node knows which value is the best. The root node tells its decision to children. Next, each child chooses the best value based on the decision of the root node, and so on.
DPOP Example: No Back-edge Case

- Requires a linear number of messages.
- The size of each message is constant ($O(d)$, where $d$ is the domain size).
DPOP Example: General Case

• Requires a linear number of messages.
• The message size can be exponential (O(d^w), where w is the tree width).
DPOP phases/messages

**PHASES**

1. DFS tree construction
2. Utility phase: from leaves to root
3. Value phase: from root to leaves

**MESSAGES**

- **token passing**
- **util** (\textit{child} -> \textit{parent}, \textit{constraint table} \([-\textit{child}]\))
- **value** (\textit{parent} -> \textit{children}, \textit{pseudochildren}, \textit{parent value})
DPOP: Performance

• Synchronous algorithm, linear number of messages
• \textbf{util} messages can be exponentially large: main drawback
• If the tree width is small, an optimal solution can be obtained very quickly.
• The max-sum algorithm (Farinelli, Rogers, Petcu, and Jennings, AAMAS-08) uses a similar idea for no back-edge case, but it does not use the tree structure.
P-optimal algorithm (Okimoto, Joe, Iwasaki, Y, Faltings, CP-2011)

• Basic Idea:
  Simplify a problem instance by removing some constrains/edges so that:
  – We can solve the simplified problem efficiently, and
  – We can bound the difference between the solution of the simplified problem and an optimal solution.

• More specifically,
  – We remove edges so that the tree width of the remaining graph is at most $p$.
  – Then, the simplified problem can be solved in $o(n \cdot d^{p+1})$, where $d$ is the domain size of variables.
Example (reward maximization) 
(p = 2)

Reward = 11

1, 2, 3, 4, 5

induced width = 2

1, 2, 3, 4, 5
Example

- we need to be careful to determine which edges to remove

Induced width 3

\[ p = 2 \]
P-optimality: Performance

• Can be solved in $o(n \cdot d^{p+1})$
  – Use DPOP for solving a simplified problem.
  – Require a linear number of messages, whose size is bounded by $d^{p+1}$

• The difference between the obtained solution and an optimal solution is bounded by the number of removed edges.
Outline

• Constraint Satisfaction Problem (CSP)
  – Formalization
  – Algorithms
• Distributed Constraint Satisfaction Problem (Dis-CSP)
  – Formalization
  – Algorithms
• Distributed Constraint Optimization Problem (DCOP)
  – Formalization
  – Algorithms
• Advanced Topics
  – Coalition Structure Generation based on DCOP
Coalition Structure Generation based on DCOP

(Ueda, Iwasaki, Y, Silaghi, Hirayama, Matsui, AAAI-2010)
What is a Coalition Structure Generation (CSG)?

• Assume you are a president of a company and considering the organization of groups...
  – Only Alice ( Blonde) can handle Becky (Pink).
  – They should be in different groups.

• Your goal is to find the best division of the personnel so that the sum of productivities of groups is maximized.
Example of CSG

**Characteristic function:** $v$

- $v(\text{Alice}) = 5$
- $v(\text{Becky}) = 3$
- $v(\text{Carol}) = 4$
- $v(\text{Alice, Becky}) = 9$
- $v(\text{Alice, Carol}) = 7$
- $v(\text{Becky, Carol}) = 7$
- $v(\text{Alice, Becky, Carol}) = 12$

**Set of all agents:** $T$

- Alice
- Becky
- Carol

**Coalition:** $S \subseteq T$

- 7

**Coalition Structure:** $CS$

- 13

- 9
- 4
Existing works on CSG

• Many algorithms for finding optimal/approximate solutions have been developed.
  – [Sandholm et al., 1999]: anytime, $O(n^n)$
  – [Rahwan et al., 2007]: DP-based, $O(3^n)$

• Almost all existing works assume that a characteristic function is given as a black box function (oracle).
  – A notable exception is [Ohta et al., 2009].
  – The value of a coalition is calculated by applying a set of rules (ex. MC-nets, SCG).
Meaning of the value of a coalition

• The value of a coalition (reward) represents the optimal gain achieved by agents in the coalition.
  – It’s natural to think that agents need to solve some combinatorial optimization problem to coordinate their activities.
Representation based on DCOP (I)

• Each agent has a variable.

• There exist unary constraints/rewards.
  – Alice’s action is “Active” ⇒ 5
  – Alice’s action is “Passive” ⇒ 0

• There also exist binary constraints/rewards.
  – (Active, Passive) ⇒ 4
  – (Active, Active) ⇒ -2

• The value of a coalition is that of an optimal solution of a DCOP among the coalition.
When two agents are in different coalitions, the constraint between them becomes ineffective.

- We have to calculate optimal assignments at each coalition.

CSG = an optimal partition + optimal value assignments
Is this approach feasible?

• We need to solve an NP-hard problem instance just to obtain the value of a single coalition.
  – In existing algorithms, we need to find the value of all coalitions ($\Theta(2^n)$). ($n$ is the number of agents)
  – So, we need to solve NP-hard problem instances $\Theta(2^n)$ times --- sounds infeasible...

• Quite surprisingly, we obtained approximation algorithms with quality guarantees, whose complexity is about the same as obtaining the value of a single coalition (which contains all agents).
Approximation algorithm (Basic)

• Main idea: only search for a restricted subset of all CSs without calculating the value of coalitions.
  – Search for CS that contains only one coalition with multiple agents; other agents act independently.
  – Slightly modify the original DCOP
    • For each variable, we add new value “independent”, which means the agent acts independently.
    • This new value has the max unary reward and no binary reward.
  – By solving this modified DCOP, we can obtain an optimal solution in this restricted search space.
Example of algorithm application

Let an agent who has negative relations work independently ⇒ better CS.
Approximation algorithm (Generalized with parameter k)

- Consider CS that contains at most k multi-agent coalitions

Domain

Active₁
Passive₁

... 

Activeₖ
Passiveₖ

Independent

Coalition 1
Coalition 2
Coalition 3

Coalition k

Passive₃
Quality Bound

• We can bound the worst case ratio.

• The ratio between the solution obtained by the approximation algorithm and the optimal solution is more than

\[
\frac{k}{w^* + 1}
\]

– \(w^*\) is the induced width of a constraint graph (\(w^*=1\) for a tree, usually small for a sparse graph).

– If we set \(k = w^* + 1\), we can obtain an optimal solution.
Number of Coalitions in an Optimal Coalition Structure

Can be bounded by $w^* + 1$
Quality bound

- CS* contains at most \( w^* + 1 \) multiagent coalitions.
- If we break \( w^* + 1 - k \) multiagent coalitions into single-agent coalitions, the obtained value is reduced at most

\[
\frac{k}{w^* + 1}
\]
History of DCR Research

• Started working on this topic around 1988
  – Initially, not very well accepted from MAS and CSP communities.
• Gradually noticed by two communities
• Both of the communities expanded (so that they have their own conferences/journals).
• The research community of DCR is growing.
  – A popular topic in AAMAS.
  – Workshops specialized on DCR have been held every year since 2000.
Further Readings

