#13: DCOP (AE4M36MAS tutorial)

- **Tutorial time:** 11 Dec 2012 @ 14:30
- **Notes by:** Jan Hrnčíř

DCOP algorithms

Revision from the lecture

- DCOP problem = <X, D, C, A> … variables, domains, constraints, agents; minimise overall cost
- Soft and hard constraints (how to model hard constraints using soft constraints)
- Algorithms
    - Systematic search (DPOP, ADOPT)
        - need variable ordering
        - no anytime behavior: have to wait for termination
        - often (too) costly
    - Local search (MGM/min-conflicts, breakout algorithm)
        - Exchange local values for variables
            - Similar to search based methods (e.g. ADOPT)
        - Consider only local information when maximizing
            - Values of neighbors
        - Anytime behavior
        - But: Could result in very bad solutions
    - Stochastic search
        - Greedy local search with activation probability to mitigate issues with parallel executions

Meeting Scheduling – modelling

- **problem specification**
    - **Variables:** variable for each meeting an agent is attending to
    - **Domains:** 1..8
    - **Agents:** managers, each is controlling a set of variables
    - **Constraints:** manager schedules (private constraints), meetings overlapping (equality, no overlap)
    - (Additional constraints: e.g., order of meetings)
    - **Constraint graph with hard and soft constraints**
- preference constraint in XCSP
    - `<relation name="agent_pref" semantics="soft" nbTuples="3" arity="1">infinity:1 | 50:2 | 50:3</relation>`

FRODO

- Docs
    - User manual
    - Installation
        - **frodo2.zip**
            - **+JDOM**
            - **+JaCoP**
            - **+graphviz**
                - Win: graphviz-2.28.0.msi
                - Linux: sudo apt-get install graphviz
- Architecture
    - Communications Layer (Queue, Message)
    - Solution Space Layer
    - Algorithms Layer (algorithm = one or more modules)
- File Formats

- Problem File Format
  - XCSP 2.1 format + which agent owns which variable(s)
    - <agents>
    - <domains>
    - <variables>
    - <relations> … defines generic relations over variables … e.g. NEQ
      - all constraints as soft
      - hard constraint are modelled as soft with cost +infinity
    - <constraints> … constraints in the DCOP, by referring to previously dened relations, and applying them to specic variable tuples
- Agent configuration – algorithms
  - Performance Metrics
  - MAS problem file

Example problem

```
<instance>
  <presentation name="sampleProblem" maxConstraintArity="2"
                maximize="false" format="XCSP 2.1_FRODO" />

  <agents nbAgents="3">
    <agent name="agentX" />
    <agent name="agentY" />
    <agent name="agentZ" />
  </agents>

  <domains nbDomains="1">
    <domain name="three_colors" nbValues="3">1..3</domain>
  </domains>

  <variables nbVariables="3">
    <variable name="X" domain="three_colors" agent="agentX" />
    <variable name="Y" domain="three_colors" agent="agentY" />
    <variable name="Z" domain="three_colors" agent="agentZ" />
  </variables>

  <relations nbRelations="1">
    <relation name="NEQ" arity="2" nbTuples="3" semantics="soft" defaultCost="0">
      infinity: 1 1|2 2|3 3
    </relation>
  </relations>

  <constraints nbConstraints="3">
    <constraint name="X_and_Y_have_different_colors" arity="2" scope="X Y" reference="NEQ" />
    <constraint name="X_and_Z_have_different_colors" arity="2" scope="X Z" reference="NEQ" />
    <constraint name="Y_and_Z_have_different_colors" arity="2" scope="Y Z" reference="NEQ" />
  </constraints>
</instance>
```

Example predicates

```
<predicates nbPredicates="2">
<predicate name="P0">
  <parameters>int X0 int X1</parameters>
  <expression>
    <functional>eq(X0,X1)</functional>
  </expression>
</predicate>
<predicate name="P1">
  <parameters>int X0 int X1 int X2 int X3 int X4 int X5 int X6</parameters>
  <expression>
    <functional>or(eq(add(X0,X1),X2),and(eq(X3,X4),eq(X5,X6)))</functional>
  </expression>
</predicate>
</predicates>
```

Resources

- Constraint Programming (COP) On-line Resources