5. Testing Multi-Agent Systems

As every computer program, multi-agent systems and their agents need to be tested.
And all the usual techniques and tools are available and should be used:
Using use cases to develop
- Unit tests
- Interface tests
- Integration tests
Statistical testing
...

Multi-Agent Systems

Jörg Denzinger
Additional problems for testing

- Observing several agents/processes working in parallel is difficult
  - we need development environments that
    - Protocol MAS runs in such a manner to allow repeating them (racing conditions)
    - Visualize the agents and their actions
    - Help with comprehension of the system

- Many MAS aim at achieving emergent behavior for solving the given problem
  - what about emergent misbehavior?
Detecting emergent misbehavior: learning of events/behaviors

Multi-Agent Systems

Jörg Denzinger
Learning to test for unwanted behavior

- Define unwanted behavior (including how we can measure how near we are to such an unwanted behavior)
- Create simulation of the environment in which $A_{\text{tested},i}$ are acting
- Determine what agents should be controlled by tester and what agents are additionally in simulation
- Determine start conditions (or if they should also be learned)
- Select appropriate learner
Example 1: testing rescue teams for ARES (I)

Kidney, Denzinger (2006)

- Tested MAS:
  rescue team written by students for ARES competition in CPSC 567/609

- Type of problems looked for:
  Low number of survivors rescued in a competitive world scenario

- Success measure/fitness:
  number of survivors rescued, aggregated over each turn
Example 1: testing rescue teams for ARES (II)

- Learning method: Genetic algorithm with
  - Individual: sequence of actions for each agent in the attack team
  - Genetic operators: standard crossover and mutation on strings + targeted crossover and mutation
Round 4:

Multi-Agent Systems

Jörg Denzinger
Round 21:

Tested Team
Round 32:

Tested Team

Multi-Agent Systems

Jörg Denzinger
Example 2: testing computer players for ORTS (I)

Atalla, Denzinger (2009)

- **Tested MAS:**
  Computer players developed for ORTS, a real time strategy game; using the simple combat scenario

- **Type of problems looked for:**
  ways to loose the combat scenario

- **Success measure / fitness:**
  various measures combined as weighted sum, like surviving units, health of units, strategic positions, etc.
  for each round of the game
Example 2: testing computer players for ORTS (II)

- Learning method:
  The Genetic Algorithm from Example 1, enhanced with macro actions
  - Macro action triggers a (predefined) behavior for all agents for a certain numbers of rounds (or until a condition is fulfilled)
  - Macros are introduced sparsely by specialized operators
### Other results

<table>
<thead>
<tr>
<th>Player</th>
<th>Start situation</th>
<th>Success in generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blekinge</td>
<td>1</td>
<td>15.0</td>
</tr>
<tr>
<td>Blekinge</td>
<td>2</td>
<td>16.8</td>
</tr>
<tr>
<td>WarsawA</td>
<td>1</td>
<td>3.3</td>
</tr>
<tr>
<td>WarsawA</td>
<td>2</td>
<td>11.3</td>
</tr>
<tr>
<td>WarsawB</td>
<td>1</td>
<td>9.2</td>
</tr>
<tr>
<td>WarsawB</td>
<td>2</td>
<td>8.3</td>
</tr>
</tbody>
</table>
Example 3: testing harbor security strategies (I)

Flanagan, Thornton, Denzinger (2009)

- Tested MAS:
  A group of harbor patrol vessels following a given patrol and interception strategy (in a GIS simulation of a harbor with a simple physics engine)

- Type of problems looked for:
  Attack on harbor that has attacker reach predefined spot without being intercepted

- Success measure/fitness:
  various measures (taken in regular intervals) combined in so-called ordering structure
Example 3: testing harbor security strategies (II)

- Learning method:
  Particle Swarm Optimizer with low-level path planner
  - PSO individual: sequence of high-level waypoints with associated speeds (vector of real numbers)
  - Update: Create a velocity of a particle out of old velocity, best individual, best historic point of the individual and some random factors. Add this velocity to current individual to create new point for the individual.
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1:</strong> Intruder 1 and Intruder 2 are spotted by Patroller 1, and tasked to Interceptor 1 and Interceptor 2</td>
<td><strong>2:</strong> Intruder 1 and Intruder 2 are intercepted by Interceptor 1 and Interceptor 2</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Intruder 2</td>
<td>Patroller 1</td>
</tr>
<tr>
<td><strong>Interceptor 2</strong></td>
<td><strong>Patroller 2</strong></td>
</tr>
<tr>
<td>Intruder 1</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3:</strong> Intruder 3 is spotted by Patroller 2 and assigned to Interceptor 1</td>
<td><strong>4:</strong> Intruder 3 reaches target before it can be intercepted by Interceptor 1</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Interceptor 1</strong></td>
<td>Patroller 2</td>
</tr>
<tr>
<td>Intruder 3</td>
<td></td>
</tr>
<tr>
<td>Intruder 2</td>
<td>Patroller 1</td>
</tr>
<tr>
<td><strong>Interceptor 2</strong></td>
<td></td>
</tr>
</tbody>
</table>
Testing surveillance of a room: simulation
Testing surveillance of a room: real life
Current research goals

- When is this type of testing affordable?
- Can we come up with generic fitness functions to use?
- How to also evolve start situations? (layered learning)
- How to find new problems without having fixed already found ones? → EvoStar 2017
- Can we use search for a problem to focus search for establishing that it is really fixed?