Force Protection Call 4
A-0938-RT-GC

EUSAS
European Urban Simulation for Asymmetric Scenarios

Behaviour analysis and cloning – part 1
Matjaz Gams, Ales Tavcar
Jozef Stefan Institute
Presentation outline

- Cognitive Multi-Agent Strategy Discovering Algorithm (CMASDA)
  - Features
  - Algorithm structure
  - Analysis results
- Pattern Discovery Replay (PDR)
Cognitive MASDA
Features

- Algorithm for advanced data analysis and **extraction of behaviour patterns** from low-level observations of two agent groups.

- Behaviour is analyzed:
  a) To provide descriptions of relevant behaviour patterns of humans
  b) To transfer behaviour patterns of observed live persons into simple behaviour models
Cognitive MASDA
Algorithm structure
Cognitive MASDA

Action graph construction

• An example:

(15,31) → action → (12,48)

(15,31) → move → agent position

agent position

civilian

soldier

civ. leader
Cognitive MASDA / Vignette1.1

Action graph construction

cognitive features
agents have cognitive states
cognitive taxonomy
colour modifications:
  nodes: cognitive states (red - anger, green - need, blue - fear)
  edges: actions (move_events – gray,..)
Taxonomies – action
Taxonomies – agent roles
Taxonomies – cognition
Multi-Agent Strategy Discovering Algorithm

Abstract action graph

• Abstraction process merges nodes together.
• Iteratively, the nearest two nodes are merged.
• Modified distance definition:
  – Weighted sum of distances between node positions and graph distances between role, action and cognitive concepts.

\[
\text{dist}(a,b) = w_{pos} \cdot \text{dist}_{pos}(a,b) + w_{role} \cdot \text{dist}_{role}(a,b) + w_{action} \cdot \text{dist}_{action}(a,b) + w_{state} \cdot \text{dist}_{state}(a,b),
\]
Cognitive MASDA
Abstract action graph

- An example:

communicate_calm_event

communicate

communicate_warning_event

communicate_calm_event

communicate_warning_event

negotiate_event
Multi-Agent Strategy Discovering Algorithm

Abstract action graph
Language attributes

anger
anger_level
fear
fear_level
need
need_level
evaluation
evaluation_level
standby
energy_level
average_civilian_anger
CALC_anger_value
CALC_motive_value
NearProvokeSlightlyEvent
inSoldierArea

inSoldierAreaSPA
inFightAreaEntrance
inFightArea
inFightAreaInterest
inFightAreaInterestEntrance
inTowerArea
inTowerFightArea
Moving
HasCACLNear
HasCommunicatedCalmEvent
HasGesticulateEvent
HasShowedWeapon
HasLoadedGun
HasPerformedWarningShot
HasPerformedEffectiveShot
Multi-Agent Strategy Discovering Algorithm

Machine learning

- Machine learning: providing symbolic explanation of a graphical pattern
- Machine learning from abstract action graph + taxonomies + description of attributes
- Area of interest around graphical patterns
- ML learns all interesting patterns inside area of interest
Multi-Agent Strategy Discovering Algorithm

Civilian pattern

IF CACL Needy THEN moving_event
IF energy_level >= 100 THEN moving_event
Multi-Agent Strategy Discovering Algorithm

Civilian pattern

IF energy_level = 92 AND CAO Angry THEN attack_event_ww
IF energy_level = 97 AND anger_level >= 100 AND need_level >= 68 THEN attack_event_ww
Multi-Agent Strategy Discovering Algorithm
Soldier pattern

IF calc_anger_value >= 64 AND evaluation_level = 55 THEN load_gun_event
IF SASM NearProvokeSlightlyEvent THEN load_gun_event
IF calc_anger_value >= 67 THEN load_gun_event
Multi-Agent Strategy Discovering Algorithm

Soldier pattern

IF calc_anger_value >= 100 AND SASM HasLoadedGun THEN
gun_shot_warning_event
Multi-Agent Strategy Discovering Algorithm
Soldier pattern

IF evaluation_level >= 62.5 THEN gun_shot_warning_event
IF evaluation_level >= 65 AND SASM HasLoadedGun THEN gun_shot_warning

- Number of actions: 17
- Abstracted action: gun_shot_warning_event
- Rule:
  IF evaluation_level >= 62.5 THEN gun_shot_warning_event
  IF evaluation_level >= 55 AND SASM HasLoadedGun THEN gun_shot_warning_event
- Actions:
  17 x gun_shot_warning_event
Pattern editor

load_gun_event: IF evalution_level >= 55 THEN load_gun_event;
load_gun_event: IF evalution_level >= 70 THEN load_gun_event;
load_gun_event: IF evalution_level >= 55 AND calc_anger_value <= 63 THEN load_gun_event;
load_gun_event: IF evalution_level >= 35 AND evalution_level >= 64.2857 THEN load_gun_event; IF evalution_level >= 35 THEN load_gun_event;
communicate_calm_even: IF evalution_level >= 35 AND SASM HasPerformedWarningShot THEN communicate_calm_even;
communicate_calm_even: IF evalution_level >= 39 AND SASM Standby THEN communicate_calm_even; IF evalution_level >= 39 THEN communicate_calm_even;
XML output of patterns

Contains graphical and symbolic description

```
<library>
  <pattern id="1">
    <description>Test</description>
    <result>ALARM</result>
    <graph>
      <edges destination="1" rule="0" source="0"/>
      <vertices/>
      <rules>
        <ruleSet/>
      </rules>
    </graph>
  </pattern>
</library>
```
Pattern discovery replay / vignette1
Conclusion and discussion

• Explanation how CMASDA works

• The CMASDA algorithm discovers relevant patterns in the Vignette1.1 scenario as planned (civilian and soldier)

• MASDA discovered patterns on its own from raw data and taxonomies

Thank you
Force Protection Call 4
A-0938-RT-GC

EUSAS
European Urban Simulation for Asymmetric Scenarios

Behaviour analysis and cloning – part 2
Matjaž Gams, Aleš Tavčar
Jozef Stefan Institute
Presentation outline

- Application of CMASDA on Vignette1.1
  - Rule clustering
- Cloning module implementation and verification:
  - Algorithm
  - Structure
  - Results

Major improvement:
Cloning implemented and tested
Presentation outline

• Programs for:
  – advanced data analysis
  – extraction of behaviour patterns from low-level observations of two agent groups
  – behaviour cloning
  – validation

• Task: Learn behaviour from around 10 games on its own without prepared patterns/plans and clone the learned behaviour.

• Purpose:
  a) To analyze events (discover patterns)
  b) To train squads (improve squad performance)
  c) To clone behaviour during data-farming (search optimal strategies)
Application of CMASDA on Vignette1.1

Adaptations

• Input modification (scene, functions, logs)
• Add new taxonomies:
  – action, role, cognition
• Add new language terms (e.g. loadedGun)

Upgrades (see next slides)

• Clustering
Rule clustering

- CMASDA generates a huge amount of patterns
- Solution: apply clustering to select most informative patterns, use representative patterns for analysis and cloning

Technically:
- Create term vectors from patterns
- Calculate a vector of term TFIDF weights
- Use of a clustering algorithm kNN
- Measure similarity of cloning behaviour to the original behaviour using DTW
DTW Algorithm
For measuring similarity between original and cloned behaviour

• Compare time series
• Similarity between 0 and 1, 0.5 border
• Compare each of 10 games to each of 10 games and compute average
DTW for comparing 2 games

• Compute DTW of all time series and average the results
Influence of no. of clusters

- Simulation log similarity
- All extracted patterns
- 10 clusters
- 15 clusters
- 16 clusters
- 20 clusters
Influence of no. of clusters
Behaviour cloning

1. Obtain 10+ game logs
2. Apply CMASDA on the logs
3. Apply clustering on CMASDA patterns
4. Play 10+ cloned games using cluster representative rules (see next slide)
5. Compare each of the original games to each of the cloned games using DTW
Behaviour cloning

4. Play 10+ cloned games using cluster representative rules:

4.1 Run the ABS simulator with the implanted cloning module

4.2 On each squad member’s turn the cloning module is called

4.3 Cloning module:

- apply the longest pattern found given circumstances
- execute the pattern by involving simulator in turns
- each step during the sequence check for better patterns (e.g. if the pattern fails, a new one is selected)
Behaviour cloning
Behaviour cloning – Validation

10 games played by the simulator
Cloned by CMASDA

10 games played by an aggressive human player
Cloned by CMASDA

Validation:
- Viewing Simulation Variables and Statistics
- Computing similarity by DTW
- Visual comparison of actual games
Behaviour cloning – simulation, human
Computing similarity by DTW

<table>
<thead>
<tr>
<th></th>
<th>simulation</th>
<th>human play</th>
<th>cloned simulation play</th>
<th>cloned human play</th>
</tr>
</thead>
<tbody>
<tr>
<td>simulation</td>
<td>0.7133</td>
<td>0.2642</td>
<td>0.5913</td>
<td>0.2965</td>
</tr>
<tr>
<td>human play</td>
<td>-</td>
<td>0.6324</td>
<td>0.3907</td>
<td>0.5857</td>
</tr>
<tr>
<td>cloned</td>
<td>-</td>
<td>-</td>
<td>0.7884</td>
<td>0.3822</td>
</tr>
<tr>
<td>simulation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cloned</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>human play</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>human play</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cloned</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>human play</td>
<td></td>
<td></td>
<td></td>
<td>0.8514</td>
</tr>
</tbody>
</table>
Computing similarity by DTW
Visual comparison of actual games

1. **Simulation scenario 1**
   Cloned simulation scenario 1
   (crowd gathers, conflict, dispersion, throwing)

2. **Simulation scenario 2**
   Cloned simulation scenario 2
   (crowd gathers, conflict, angry panic flight, return)

3. **Human play**
   Cloned Human play
   (crowd gathers, conflict, panic flight)
Conclusions

• CMASDA successfully applied to Vignette1.1
• Rule clustering improves quality of the behaviour patterns
• Successfully cloned simulator and human behaviour in Vignette1.1

Thank you