

# BEHAVIOUR ANALYSIS AND CLONING IN SECURITY DOMAINS

Aleš Tavčar

**Doctoral Dissertation**  
**Jožef Stefan International Postgraduate School**  
**Ljubljana, Slovenia**

**Supervisor:** Prof. Dr. Matjaž Gams, Jožef Stefan Institute, Ljubljana, Slovenia

**Evaluation Board:**

Prof. Dr. Jurij Tasič, Chair, Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia

Dr. Mitja Luštrek, Member, Jožef Stefan Institute, Ljubljana, Slovenia

Dr. Tomaž Šef, Member, Jožef Stefan Institute, Ljubljana, Slovenia

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA  
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL



Aleš Tavčar

BEHAVIOUR ANALYSIS AND CLONING IN SECURITY DOMAINS

**Doctoral Dissertation**

UČENJE IN KLONIRANJE OBNAŠANJA V VARNOSTNIH  
DOMENAH

**Doktorska disertacija**

**Supervisor:** Prof. Dr. Matjaž Gams

Ljubljana, Slovenia, April 2018



*To my family for always supporting me*



# Acknowledgments

First and foremost, I would like to thank my supervisor Prof. Dr. Matjaž Gams for his invaluable contributions to my research and work advancements. He provided continuous support, guidance, suggestions for research improvements, and professional assistance throughout all my years at the Department of Intelligent Systems at the Jožef Stefan Institute. Without his commitment and insightful observations this work could not have been carried out to such an extent.

I would also like to thank the past and present colleagues at the Department of Intelligent Systems who helped me with suggestions, research inputs, and provided a pleasant working environment. In particular, I would like to thank the members of the Agents group that I am a part of. Our discussions about project-related challenges have helped me in the work area and improved my professional competences.

Thanks to the EUSAS project group that spawned most of the research challenges covered in this thesis. In particular Marc Contat, Marcel Kvassay, Stefan Dlugolinsky, and Bernhard Bracker. Moreover, thanks to Damjan Kužnar and Blaž Mahnič for all the provided help and collaboration in the EUSAS project. In addition, I would like to thank the European Defence Agency that partially funded this research.

I would like to thank the thesis Evaluation Board Prof. Dr. Jurij Tasič, Dr. Mitja Luštrek, and Dr. Tomaž Šef for their invaluable comments and guidelines that helped improve this doctoral thesis.

Many thanks to my dear friends and colleagues Miha Mlakar and Boštjan Kaluža for our discussions, brainstorming ideas during group meetings and lunches that provided new research insights, ideas and solutions that helped me with my research work.

Next, I would like to thank my parents Tatjana and Marčelo Tavčar for always supporting me in my decisions, providing me financial support during all my years of study, teaching me the importance of hard work and perseverance and creating an encouraging family environment.

I wish to thank Tanja for her immense support in my personal and professional life for the last six years and for the continuous encouragements to complete the doctoral thesis.

Many thanks to my brother Edvin, to my family, and all my friends for their support, encouragement, and advice throughout the years.

Last, but not least, I acknowledge all the financial support received from the Jožef Stefan Institute, the Department of Intelligent Systems. All the interesting research areas we are involved in, the opportunities to be a part of challenging European projects, providing the ability to attend project meetings at various research institutions, attending the world's best renowned conferences allowed me to evolve as a researcher as well as a person.



# Abstract

Skill training in real-life circumstances for specific personnel (such as military, police, surgeons, and pilots) can be expensive, dangerous to the trainees and others, and time-consuming. Moreover, it is often impossible for the trainee to explore all the possible circumstances or deal with problems that might occur. In addition, the emotional and psychological states and well-being of people should be taken into consideration. Due to these reasons, extensive skill training using simulations for improving decision making has become common practice in a multitude of domains and professions.

In this dissertation, we present a novel training approach called PATDS (Performance Analysis of Training through Data-Farming Simulations) that introduces a training cycle, where each iteration identifies the weaknesses of the trainee and another round of training is performed to improve the performance on the identified problematic situations. This is achieved by constructing a simulated agent as a virtual copy of a human trainee and testing it in all circumstances to provide feedback and decision support for another round of human learning. The main concepts of the procedure resemble the boosting methodology from the machine learning domain, but instead of improving a learning model, the human surrogate and the human trainee are being improved.

PATDS is composed of three main phases: (i) behavior analysis; (ii) pattern selection and behavior cloning; and (iii) performance evaluation and identification of weaknesses. In the behavior analysis phase, the relevant behavior patterns, demonstrated by the trainee in simulations, are identified, extracted, and described. These patterns are obtained using the HMASDA algorithm, which creates a set of identified patterns from basic observations. In the second phase, all obtained patterns are clustered in order to obtain a smaller subset of key patterns describing the physical and cognitive behavior of observed agents. Moreover, the patterns are transformed into one or many models of surrogate agents of the trainees, with the goal of enabling as accurate simulated behavior of the trainees as possible in a reasonably short learning period. In this way, artificial copies of the human trainees are obtained, which are capable of active performance under general conditions. In the final phase, the created agents are evaluated within data-farming experiments on a variety of simulation parameters to discover the key weaknesses in the trainees' learned strategies.

The motivation behind PATDS is to replace the need for the trainees to extensively test a huge set of possible specific conditions and scenarios, including various behaviors of the competing agents. The automated process can, in turn, test all options in a reasonable amount of time and detect circumstances causing weak performances. Consequently, in each subsequent round of training, the trainee can improve the behavior based on the identified weaknesses of the surrogate copy of the trainee. We evaluated the proposed approach on two asymmetric urban-terrain threat scenarios, where a squad of security forces interacts with hostile crowds under various conditions. We first show that the discovered behavior patterns are rich enough to successfully reproduce the observed behavior of the trainee, and then demonstrate how the data-farming-based evaluation on the complete simulation parameter space is beneficial for improving the trainee's capabilities.



## Povzetek

Usposabljanje v resničnem okolju za izboljšanje pripravljenosti v specifičnih poklicih (vojaška, policija, kirurgi, piloti itd.) je lahko drago, nevarno tako za njih kot za druge in zamudno. Pripravniki se zaradi tega večinoma ne uspejo pripraviti na vse mogoče situacije, ki se lahko pojavijo tekom izvajanja nalog. Poleg tega je potrebno pri obravnavanju ljudi skrbno upoštevati njihovo čustveno in mentalno stanje ter trenutno počutje. Zaradi vseh teh razlogov je obsežno usposabljanje s simulacijami za izboljšanje procesa odločanja postalo ustaljena praksa na številnih področjih in poklicih.

V pričujoči disertaciji predlagamo nov pristop usposabljanja, imenovan PATDS (Performance Analysis of Training through Data-Farming Simulations), ki uvaja iterativen cikel usposabljanja, pri katerem se v vsakem krogu treninga odkrijejo glavne pomanjkljivosti v sposobnostih pripravnika in se na podlagi tega sestavi nov načrt usposabljanja z namenom izboljšanja učinkovitosti pri prepoznanih problematičnih situacijah. To se doseže z izgradnjo simuliranega agenta, virtualno kopijo pripravnika, ki se testira v vseh mogočih okoliščinah in se s tem zagotovi povratna informacija ter podpora pri odločanju za naslednji krog treninga pripravnikov. Glavni koncepti predlaganega postopka so podobni metodologiji »boosting«, s področja strojnega učenja z razliko, da se namesto učnega modela izboljšuje simulirani agent ter pripravnik.

PATDS sestavljajo tri glavne faze: (i) analiza obnašanja; (ii) izbira strateških vzorcev in kloniranje obnašanja; (iii) ovrednotenje uspešnosti in odkrivanje slabosti v znanju pripravnikov. V fazi analize obnašanja se izvaja odkrivanje, izgradnja in opis ustreznih vzorcev obnašanja (strategij), ki jih udeleženci treninga prikažejo v simulacijah. Odkrivanje vzorcev poteka z uporabo algoritma HMASDA, ki generira množico pomembnih strateških vzorcev, pridobljenih iz osnovnih opazovanj. V drugi fazi metode se, z uporabo gručenja, iz celotne množice vzorcev pridobi manjša podmnožica ključnih vzorcev, ki opisujejo vedenje opazovanih agentov. Tako pridobljene vzorce preoblikujemo v enega ali več modelov nadomestnih agentov, ki opisujejo opazovane pripravnike s ciljem, da čim bolj verodostojno reproducirajo njihovo obnašanje. Na ta način ustvarimo umetne kopije človeških pripravnikov, ki so sposobni aktivnega delovanja v normalnih pogojih. V zadnji fazi nato ustvarjene nadomestne agente ovrednotimo v obsežnih računalniških na različnih pogojih z namenom odkritja ključnih pomanjkljivosti v znanju pripravnika.

Cilj je odpraviti potrebo po obsežnih usposabljanjih pripravnikov na velikem številu scenarijev, simulacijskih pogojev, ter različnih odzivih nasprotnikov. Avtomatizirani postopek lahko izvede vse te možnosti v sprejemljivem času in zazna okoliščine, kjer so rezultati simulacije slabši. Na ta način se lahko v vsakem naslednjem krogu usposabljanja znanje in sposobnosti pripravnikov izboljšajo glede na ugotovljene dosedanje pomanjkljivosti nadomestnega agenta. Predlagani postopek smo preverili na dveh asimetričnih scenarijih bojevanja v mestnem okolju, kjer se skupina varnostnih sil sooča s sovražnimi množicami v različnih okoliščinah. Najprej pokažemo, da so odkriti vzorci obnašanja semantično dovolj bogati za uspešno reproduciranje opazovanega vedenja pripravnikov. V nadaljevanju pa pokažemo, kako je vrednotenje na podlagi obsežnih računalniških simulacij koristno za izboljšanje pripravljenosti ter sposobnosti pripravnika.



# Contents

|                                                                            |              |
|----------------------------------------------------------------------------|--------------|
| <b>List of Figures</b>                                                     | <b>xv</b>    |
| <b>List of Tables</b>                                                      | <b>xvii</b>  |
| <b>List of Algorithms</b>                                                  | <b>xix</b>   |
| <b>Abbreviations</b>                                                       | <b>xxi</b>   |
| <b>Symbols</b>                                                             | <b>xxiii</b> |
| <b>1 Introduction</b>                                                      | <b>1</b>     |
| 1.1 Challenges .....                                                       | 2            |
| 1.2 Approach and Hypothesis .....                                          | 3            |
| 1.3 Scientific Contributions .....                                         | 5            |
| 1.4 Impact and Publications .....                                          | 5            |
| 1.5 Thesis Overview .....                                                  | 5            |
| <b>2 Background and Problem Definition</b>                                 | <b>7</b>     |
| 2.1 Problem and Domain Related Concepts .....                              | 7            |
| 2.1.1 Multi-Agent Systems.....                                             | 7            |
| 2.1.2 Behavior Analysis.....                                               | 8            |
| 2.2 Target Military Domain .....                                           | 9            |
| 2.2.1 Riot Domain .....                                                    | 11           |
| 2.2.2 Looting Domain .....                                                 | 13           |
| 2.3 Problem Definition .....                                               | 14           |
| 2.3.1 Agents and Other Entities.....                                       | 15           |
| 2.3.2 Behavior Analysis.....                                               | 16           |
| 2.3.3 Domain-Specific Definitions .....                                    | 17           |
| <b>3 Human Behavior Analysis</b>                                           | <b>18</b>    |
| 3.1 Related Work.....                                                      | 18           |
| 3.2 Hybrid Multi-Agent Strategy Discovering Algorithm .....                | 20           |
| 3.2.1 Learning Data.....                                                   | 21           |
| 3.2.2 Action Graph Construction .....                                      | 22           |
| 3.2.3 Abstraction of Actions .....                                         | 23           |
| 3.2.4 Frequent Pattern Mining .....                                        | 26           |
| <b>4 Performance Analysis of Training Through Data-Farming Simulations</b> | <b>31</b>    |
| 4.1 Related Work.....                                                      | 31           |
| 4.2 General Overview.....                                                  | 32           |
| 4.3 Pattern Clustering and Selection.....                                  | 36           |
| 4.4 Behavior Cloning .....                                                 | 38           |
| 4.5 Data Farming and Evaluation .....                                      | 42           |
| <b>5 Experimental Setup and Evaluation</b>                                 | <b>45</b>    |
| 5.1 Experimental Setup .....                                               | 45           |
| 5.1.1 Ontologies.....                                                      | 47           |
| 5.1.2 Behavior Graphs .....                                                | 49           |

|                   |                                         |            |
|-------------------|-----------------------------------------|------------|
| 5.2               | Discovered Behavior Patterns .....      | 51         |
| 5.2.1             | Riot Domain .....                       | 51         |
| 5.2.2             | Looting Domain.....                     | 59         |
| 5.2.2.1           | Simulated behavior.....                 | 60         |
| 5.2.2.2           | Human play.....                         | 62         |
| 5.2.2.3           | Distribution evaluation .....           | 65         |
| 5.3               | Evaluation of Pattern Clustering.....   | 66         |
| 5.4               | Evaluation of Behavior Cloning .....    | 68         |
| 5.5               | Training Improvement.....               | 71         |
| <b>6</b>          | <b>Conclusions</b> .....                | <b>75</b>  |
| 6.1               | Summary and Discussion.....             | 75         |
| 6.2               | Original Contributions .....            | 76         |
| 6.3               | Future Directions .....                 | 77         |
| <b>Appendix A</b> | <b>HMASDA Additional Content</b> .....  | <b>79</b>  |
| A.1               | List of Learning Features .....         | 79         |
| A.1.1             | Riot Domain .....                       | 79         |
| A.1.2             | Looting Domain.....                     | 80         |
| A.2               | Action and Abstract Action Graphs ..... | 81         |
| A.2.1             | Riot Domain .....                       | 81         |
| A.2.2             | Looting Domain.....                     | 83         |
| <b>Appendix B</b> | <b>Software Mockups</b> .....           | <b>89</b>  |
|                   | <b>References</b> .....                 | <b>91</b>  |
|                   | <b>Bibliography</b> .....               | <b>99</b>  |
|                   | <b>Biography</b> .....                  | <b>101</b> |

## List of Figures

|                                                                                                                                                                                  |    |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| Figure 1.1: General overview of the PATDS learning cycle presenting the difference between regular training sessions and training using PATDS. ....                              | 3  |
| Figure 2.1: A bird's eye view of the MASON simulation screen of the Riot Domain. ....                                                                                            | 11 |
| Figure 2.2: The 3D representation of the scenarios as seen by the trainees in VBS2. ....                                                                                         | 12 |
| Figure 2.3: A bird's eye view of the MASON simulation screen of the Looting Domain.....                                                                                          | 13 |
| Figure 2.4: The 3D representation of the scenarios as seen by the trainees in VBS2. ....                                                                                         | 14 |
| Figure 3.1: A general representation of an ontology used by HMASDA, where concepts are linked using the is_a relation. ....                                                      | 25 |
| Figure 3.2: A simplified representation of a behavior pattern containing three actions. ....                                                                                     | 27 |
| Figure 3.3: The selection mechanism of positive and negative examples from an AAG node, used as learning examples for the rule inducer. ....                                     | 27 |
| Figure 4.1: The schematic overview of the whole PATDS training cycle, related to Algorithm 5.1, showing the main modules of the procedure.....                                   | 33 |
| Figure 4.2: A simplified scheme of the behavior cloning procedure listing the main modules (related to Algorithm 4.3). ....                                                      | 39 |
| Figure 4.3: An example plot of all Key Simulation Parameters for one simulation run that represent how the simulation evolved in time.....                                       | 43 |
| Figure 5.1: The scheme of the experimental setup for the behavior analysis method.....                                                                                           | 46 |
| Figure 5.2: The taxonomy of agent actions, representing the relations between all actions that are available in the domain.....                                                  | 47 |
| Figure 5.3: The taxonomy of agent roles. ....                                                                                                                                    | 48 |
| Figure 5.4: The taxonomy of cognitive and mental states of the agents. ....                                                                                                      | 48 |
| Figure 5.5: HMASDA Action Graph of human play in the looting domain.....                                                                                                         | 50 |
| Figure 5.6: The Abstract Action Graph 4 of human play in the looting domain, containing abstracted nodes from the Action Graph. ....                                             | 50 |
| Figure 5.7: A discovered soldier pattern, occurring close to the camp entry in the riot domain, representing the load gun event.....                                             | 53 |
| Figure 5.8: A discovered soldier pattern in the riot domain, representing movements from the tower area to the camp entry and performing warning shots. ....                     | 54 |
| Figure 5.9: A discovered soldier pattern occurring in the tower area of the riot domain, describing warning shot events. ....                                                    | 55 |
| Figure 5.10: A discovered civilian pattern in the riot domain, describing provoke actions towards soldier agents occurring close to the camp entry.....                          | 56 |
| Figure 5.11: A discovered civilian movement pattern in the riot domain. ....                                                                                                     | 57 |
| Figure 5.12: A discovered civilian pattern in the riot domain, describing attack events in front of the camp entry. ....                                                         | 58 |
| Figure 5.13: A discovered soldier pattern, describing movements in the looting domain.....                                                                                       | 61 |
| Figure 5.14: A discovered soldier pattern in the looting domain, describing the response to an attack with a warning shot.....                                                   | 61 |
| Figure 5.15: A discovered civilian attack pattern in the looting domain. ....                                                                                                    | 62 |
| Figure 5.16: A discovered soldier pattern obtained for the human training session in the looting domain. The pattern describes movements towards insurgents looting a shop. .... | 62 |
| Figure 5.17: A discovered soldier pattern obtained for the human training session in the looting domain. The pattern describes approaching civilians near the looted shop. ....  | 63 |
| Figure 5.18: A discovered soldier pattern obtained for the human training session in the looting domain, where soldiers are protecting the looted shop. ....                     | 64 |

|                                                                                                                                                                |    |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| Figure 5.19: A discovered civilian pattern obtained for the human training session in the looting domain. The pattern describes civilians looting a shop. .... | 64 |
| Figure 5.20: A discovered civilian pattern obtained for the human training session in the looting domain, where insurgents are attacking soldier agents. ....  | 65 |
| Figure 5.21: Mean silhouette size for different cluster sizes. Higher values denote more homogeneous clusters that contain similar patterns. ....              | 67 |
| Figure 5.22: Comparison of similarities between the original simulation and cloned behavior for different cluster sizes (number of patterns used). ....        | 69 |
| Figure 5.23: Comparisons of similarities between different types of behaviors. ....                                                                            | 70 |
| Figure 5.24: Comparisons of the 10 worst simulations in the first data-farming run with retraining in the second run. ....                                     | 72 |
| Figure 5.25: The distribution of different soldier actions for the initial retrain phase over the simulation execution time, segmented in 5s intervals. ....   | 72 |
| Figure 5.26: The distribution of different soldier actions for the last retrain phase over the simulation execution time, segmented in 5s intervals. ....      | 73 |
| Figure A.1: The Action Graph for all agent traces in the riot domain. ....                                                                                     | 82 |
| Figure A.2: Abstract Action Graph with abstraction level 3 for the riot domain. ....                                                                           | 82 |
| Figure A.3: Abstract Action Graph with abstraction level 5 for the riot domain. ....                                                                           | 83 |
| Figure A.4: Abstract Action Graph with abstraction level 8 and the visualization of information for one abstracted node. ....                                  | 83 |
| Figure A.5: Abstract Action Graph with abstraction level 1 for data collected from the fixed-policy strategy in the looting domain. ....                       | 84 |
| Figure A.6: Abstract Action Graph with abstraction level 2 for data collected from the fixed-policy strategy in the looting domain. ....                       | 84 |
| Figure A.7: Abstract Action Graph with abstraction level 4 for data collected from the fixed-policy strategy in the looting domain. ....                       | 85 |
| Figure A.8: Abstract Action Graph with abstraction level 6 for data collected from the fixed-policy strategy in the looting domain. ....                       | 85 |
| Figure A.9: Abstract Action Graph with abstraction level 1 for data collected during human training in the looting domain. ....                                | 86 |
| Figure A.10: Abstract Action Graph with abstraction level 3 for data collected during human training in the looting domain. ....                               | 86 |
| Figure A.11: Abstract Action Graph with abstraction level 6 for data collected during human training in the looting domain. ....                               | 87 |
| Figure A.12: Abstract Action Graph with abstraction level 7 for data collected during human training in the looting domain. ....                               | 88 |
| Figure B.1: Software mockup – Increasing abstraction and exploring AAG. ....                                                                                   | 89 |
| Figure B.2: Software mockup – Exploring and visualization of patterns. ....                                                                                    | 90 |
| Figure B.3: Software mockup – Exporting behavior patterns. ....                                                                                                | 90 |

## List of Tables

|                                                                                                                                                   |    |
|---------------------------------------------------------------------------------------------------------------------------------------------------|----|
| Table 2.1: List of all possible soldier and civilian actions in the Riot Domain.....                                                              | 12 |
| Table 2.2: List of all possible soldier and civilian actions in the Looting Domain. ....                                                          | 14 |
| Table 3.1: Event log entries for the two main agent roles.....                                                                                    | 22 |
| Table 5.1: The distribution of soldier actions in the discovered patterns for the fixed-policy strategy in the riot domain. ....                  | 59 |
| Table 5.2: The distribution of soldier actions in the discovered patterns for the strategy adopted by operational people in the riot domain. .... | 59 |
| Table 5.3: The distribution of all soldier actions in the behavior patterns discovered for the fixed-policy strategy in the looting domain. ....  | 65 |
| Table 5.4: The distribution of all soldier actions in the behavior patterns discovered for human play in the looting domain. ....                 | 66 |
| Table 5: Comparison of all similarities between different simulation runs. ....                                                                   | 70 |



## List of Algorithms

|                |                                                                      |    |
|----------------|----------------------------------------------------------------------|----|
| Algorithm 3.1: | Hybrid Multi-Agent Strategy Discovering Algorithm.....               | 21 |
| Algorithm 3.2: | Action graph construction.....                                       | 23 |
| Algorithm 3.3: | Abstract Action Graph construction.....                              | 24 |
| Algorithm 3.4: | Extraction of frequent strategic patterns.....                       | 29 |
| Algorithm 4.1: | Performance Analysis of Training using Data Farming Simulations..... | 35 |
| Algorithm 4.2: | Pattern selection mechanism through clustering.....                  | 38 |
| Algorithm 4.3: | Pattern selection algorithm during cloning in PATDS.....             | 40 |
| Algorithm 4.4: | Pattern matching algorithm during cloning in PATDS.....              | 42 |



# Abbreviations

|         |                                                                                     |
|---------|-------------------------------------------------------------------------------------|
| ABS     | Agent-Based Simulation                                                              |
| AG      | ... Action graph                                                                    |
| AAG     | ... Abstract action graph                                                           |
| CORBA   | ... Common Object Request Broker Architecture                                       |
| DFE     | ... Data-farming experiment                                                         |
| DTW     | ... Dynamic time warping                                                            |
| HMASDA  | ... Hybrid Multi-Agent Strategy Discovering Algorithm                               |
| IDF     | ... Inverse Document Frequency                                                      |
| KPI     | ... Key performance indicator                                                       |
| KSP     | ... Key simulation parameter                                                        |
| LD      | ... Looting domain                                                                  |
| MAS     | ... Multi-Agent systems                                                             |
| MASON   | ... Multi-Agent simulation of networks                                              |
| MOUT    | ... Military operations urban terrain                                               |
| PATDS   | ... Performance Analysis of Training through Data-Farming Simulations               |
| PECS    | ... Physical conditions, Emotional state, Cognitive capabilities, and Social status |
| RD      | ... Riot domain                                                                     |
| SA      | ... Surrogate agent                                                                 |
| SD      | ... Simulation domain                                                               |
| SiA     | ... Simulated agent                                                                 |
| SLIPPER | ... Simple Learner with Iterative Pruning to Produce Error Reduction                |
| T       | ... Trainee                                                                         |
| TF      | ... Term Frequency                                                                  |
| UKB     | ... Unified Knowledge Base                                                          |
| VBS2    | ... Virtual Battlespace 2                                                           |



# Symbols

|                           |                                                                       |
|---------------------------|-----------------------------------------------------------------------|
| $BP$                      | ... set of behavior patterns                                          |
| $ BP $                    | ... cardinality of the set of patterns                                |
| $C$                       | ... cognitive envelope                                                |
| $C_A$                     | ... set of cognitive states                                           |
| $c_A$                     | ... cognitive state of agent $A$                                      |
| $D$                       | ... symbolic descriptions                                             |
| $D_A$                     | ... set of decision states                                            |
| $d_T(i, j)$               | ... distance in taxonomy between concepts $i$ and $j$                 |
| $\text{dist}_f(n_1, n_2)$ | ... distance function $f$ between nodes $n_1$ and $n_2$ .             |
| $\Delta$                  | ... SA's asserted knowledge                                           |
| $\delta_A$                | ... agent decision function                                           |
| $DM_A$                    | ... decision model of agent $A$                                       |
| $E$                       | ... trajectory envelope                                               |
| $\eta(n, i)$              | ... number of instances in node $n$ including the attribute value $i$ |
| $F$                       | ... set of features composing symbolic descriptions                   |
| $f_i$                     | ... feature $i$ composing symbolic descriptions                       |
| $I_A$                     | ... set of internal states                                            |
| $i_A$                     | ... internal state of agent $A$                                       |
| $M_A$                     | ... set of agent $A$ actions                                          |
| $m_A$                     | ... active action of agent $A$                                        |
| $O$                       | ... ontology                                                          |
| $O_a$                     | ... ontology of agent actions                                         |
| $O_r$                     | ... ontology of agent roles                                           |
| $O_c$                     | ... ontology of agent cognitive states                                |
| $\omega_f$                | ... weight for feature $f$                                            |
| $P$                       | ... agent traces                                                      |
| $p_i$                     | ... vector of term identifiers                                        |
| $\Phi$                    | ... strategic pattern                                                 |
| $R_A$                     | ... set of agent roles                                                |
| $r_A$                     | ... current role of agent $A$                                         |
| $S$                       | ... spatial envelope                                                  |
| $S'$                      | ... set of finite domain states                                       |
| $SP$                      | ... simulation play                                                   |
| $ST$                      | ... strategy                                                          |
| $\text{sim}(n_1, n_2)$    | ... similarity between nodes $n_1$ and $n_2$                          |
| $T$                       | ... temporal envelope                                                 |
| $T_x$                     | ... taxonomy $x$                                                      |
| $\theta(n, e)$            | ... a set of distinct feature values $e$ that are present in node $n$ |
| $\tau_{sim}$              | ... node merging threshold                                            |
| $w_{i,j}$                 | ... term attribute weight                                             |



# Chapter 1

## Introduction

Several domains require extensive training of personnel in simulated environments in order to increase their efficiency in a variety of operating conditions, scenarios, and simulation-parameter settings. For example, in military operations in urban terrain, mission-analysis simulations are employed to prepare the trainees according to a specific military doctrine, for a difficult environment, facing diverse enemies with different cultural and social backgrounds. Moreover, flight simulations are used extensively in the aviation industry to train pilots, other flight crew, or airport staff for both civil and military domains [1, 2, 3, 4].

Training using simulations is increasing in importance in various real-life domains, since it makes it possible to repeat the same scenario until a satisfactory level of expertise is achieved. Modern variations of training simulations involve the use of serious games, which are essentially interactive high-fidelity video games that reproduce the environment, the physics of objects, and the definition of the scenario. Simulations in medicine are used to train doctors for various medical specializations [5, 6, 7], and research shows that training in simulations is better for residents than lectures [8]. Similar simulation-based training solutions were proposed for management [9, 10], maintenance of different equipment [11, 12, 13], road traffic simulation [14], novice and professional driver training [15, 16, 17], fluvial navigation training [18], and many others areas. Whereas simulation alone cannot factually reproduce specific phenomena, such as feelings or micro-timing in the physical world, it does usually enhance strategic decision making and pattern recognition [19, 20].

Most of the current simulated training approaches utilize fixed simulation policies that mimic how trainees should perform in certain scenarios [21], but they lack the ability to personalize or further analyze the training performance. Data-farming evaluations are performed using predefined, fixed-policy behavior, while we propose the use of dynamic policies that describe the specific behavior of observed individuals. Precise methodological procedures for defining, setting up, and using game models, as they arise from the simulation use, are not often addressed. Conceptual models have been introduced as a framework for building simulation-based serious gaming for decision support systems [22, 23], which are used to define ordered sequences of actions for model specification, definitions of modeling objectives, requirements for a conceptual model, etc., but they lack the ability to systematically validate the abilities of the trainees on a large variety of conditions. In more recent efforts, operative simulations and war gaming, commercial, off-the-shelf, computer-game characters are able to adapt the difficulty level based on opponent performance [24], but still follow a predefined scenario and are not able to serve specific tasks to address the weaknesses of the trainees. In addition, various authors have attempted to improve the simulation-based training process by designing various methodologies to analyze and evaluate team and trainees' [25, 26, 27] or system [28, 29] performance before, during, and after training activities. The current main challenges of simulation-based training [30] remain the ability to adapt to specific types of trainees, leveraging learner control to accurately assess their knowledge level and train on the most problematic scenarios, and managing the costs of customizations and adaptations of simulations.

## 1.1 Challenges

This thesis deals with the problem of designing a time-efficient and effective system for advanced training of repetitive tasks. The main focus is on simulated training where two **antagonistic and asymmetric groups of agents compete with each other**, and later test the general method on specific **urban-security domains**. The simulated agent can represent one human or a group of individuals: security forces. Since humans are involved in the training loop, it is important to consider the social interactions and cognitive processes that lead to the adoption of specific behavior during the performance evaluation of the trainees. Moreover, cognitive properties also play an important role; that is, they influence both the effectiveness of learning and the preparedness of the trainees [31]. The introduction of these human-specific aspects into the training loop is relevant for a number of possible use cases and applications in real-world domains, ranging from medicine, security, and peace-keeping to sports and cultural heritage.

In classical operational training using simulations, the training sessions are usually repeated several times for a specific set of parameters or scenarios; this is not only time-consuming but also makes it practically impossible for the trainee to explore the complete set of scenarios under all initial conditions and circumstances. This can result in **shortcomings in the learned behavior and weak responses of the trainee for specific scenarios and conditions**, which can be dangerous to the trainee and to others. Therefore, the common challenge is to **design a mechanism that can identify these problematic scenarios**, based on the **specific behavior, psychological, and cognitive traits** of the trainee without the need of his or her active involvement. The designed mechanism will **reduce the training time** in a simulator by expanding the **tested parameter settings with a simulated surrogate of the trainee, thus improving the training efficiency**. This task is difficult, for four reasons. Firstly, the high-level knowledge of the trainee has to be extracted from observing the performance in the simulation environment with as little prior or additional knowledge as possible. Secondly, when dealing with real people or simulation models representing them, cognitive aspects, motives, and emotions must be taken into consideration when analyzing their behavior. Thirdly, the trainee's knowledge should be modeled and used in a way that supports clear graphical representation for non-technical personnel and enables improvements of the models. Fourthly, the extracted knowledge must be evaluated on as many parameter settings as possible in order to identify weaknesses in the trainee's knowledge. Information regarding the identified weaknesses should be used to design new strategies and guidelines to cope with problematic situations and improve overall performance.

The novel approach presented in this thesis, called **PATDS (Performance Analysis of Training through Data-Farming Simulations)**, addresses the above-mentioned challenges using the following three key ideas. First, the high-level behavior patterns of the trainee are extracted from simulation logs using the Hybrid Multi-Agent Strategy Discovering Algorithm (HMASDA) [32], which exploits hierarchically ordered domain knowledge to generate graphical and symbolic descriptions of strategic behavior in the form of behavior patterns. The algorithm is adapted to consider the psychological states and cognitive reasoning of all the agents included in the simulation, which makes it suitable for analyzing behavioral strategies when real people are included in the training (the observed or the opposing group). Moreover, the generated behavior patterns with HMASDA provide a mechanism with which to assess the performance of the trainees over the entire progress of the simulation, to analyze the constructed behavior strategies on their own, and to provide symbolic descriptions for the specific physical and cognitive behavior traits identified for the observed trainees during simulation experiments. Experts can use the provided information to understand the conditions that trigger a specific behavior by the trainees, to learn how they perform under psychological strain, to follow their learning curve, and to create new training scenarios based on their progress. Second, the identified behavior patterns are transferred into a surrogate agent (SA) that is able to reproduce the behavior of the trainee; in other words, to generally perform similarly in similar circumstances. This process is often termed **"behavior cloning"** [33]. This is a difficult problem that requires the creation of precise behavioral models in order to realistically reproduce the observed behavior. Third, a data-farming procedure validates the cloned behavior of the trainee throughout the entire simulation parameter space. This provides a time-efficient mechanism to test the behavior of the trainees on all possible simulation configurations, initial conditions, and

scenario parameters. The method was developed during the execution of the EUSAS (“European Urban Simulation of Asymmetric Scenarios”) project financed by 20 nations under the Joint Investment Program Force Protection of the European Defence Agency. Some of the content presented in this thesis was contributed by other project partners (implementation of the simulation, ontologies, definition of scenarios, validation of behavior), but we describe that content in more detail in several section in order to improve the understanding of the proposed approach, which is an original contribution.

The idea behind the approach is that, from observing a human team leader (and team members) performing the tasks, a **surrogate agent is automatically designed and tested under all possible circumstances**. Even though the agent can learn on its own, the major advantage of PATDS is to enable the human trainee to review his/her training session in a clear and transparent way, observe the weak points of the current skills he/she possesses and the overall performance quality of his/her learned strategies, and, based on that, design more efficient tactics to improve his/her performance. On this basis, another round of the learning cycle is performed. The review and performance assessment can be performed either by the trainee, expert, or training instructor who understands how the applied strategies and behavior influences the outcome of the scenarios. Figure 1.1 presents the comparison between the classic training session performed on simulators and the PATDS training mechanism proposed in this thesis. The benefit of this is an additional training and verification cycle that does not require the direct involvement of human trainees, thus speeding up the entire training process.



Figure 1.1(a): Regular training sessions.

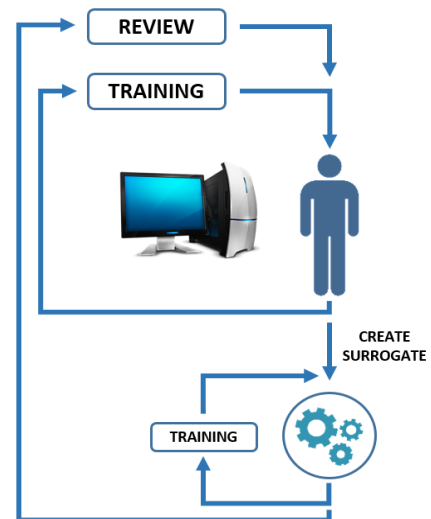


Figure 1.1(b): Enhanced training session using PATDS.

Figure 1.1: General overview of the PATDS learning cycle presenting the difference between regular training sessions and training using PATDS.

## 1.2 Approach and Hypothesis

This thesis investigates one main hypothesis and several sub-hypotheses.

**Hypothesis: Given a predefined training scenario in the form of a computer serious game and several executions of the scenario by the trainees, it is possible to construct surrogate agents that will play similarly to the observed trainees and identify weaknesses in the learnt strategies. The hypothesis is tested in Sections 5.4 and 5.5.**

In addition to the main hypothesis, three sub-hypotheses are proposed:

- (SH1) The surrogate agent will perform similarly and display similar strategies as the observed entity on the target domain. Tested in Sections 5.2 and 5.4.
- (SH2) The developed method for clustering extracted behavior patterns can increase the similarity between the behavior of the surrogate agent and the observed entity. Tested in Sections 5.3 and 5.4.
- (SH3) Given a surrogate agent, the proposed method will provide a quantitative and qualitative assessment of the agent's performance and identify weaknesses in the agent's strategy. Tested in Section 5.5.

In addition to verifying the hypotheses, the following goals are expected to be attained in this doctoral thesis:

- Provide a comprehensive survey of related work in the areas of the doctoral thesis
- Design a behavior analysis algorithm that considers the psychological aspects of the trainee and can thus efficiently process human behavior
- Implement mechanisms for pattern clustering that take into consideration different types of features included in the symbolic descriptions created with the behavior analysis algorithms
- Evaluate of the PATDS components and analyze the efficiency of the PATDS method over one learning cycle of the trainee.

To verify the validity of the hypotheses and attain the set goals in this doctoral thesis, standard methods from antagonistic multi-agent systems, machine learning, and simulations will be applied. A comprehensive list of methodologies and approaches is provided below:

- **Survey of related work.** Literature related to the research areas of the doctoral thesis will be covered and reviewed. Literature related to behavior analysis, behavior libraries, transfer learning, and behavior cloning will be reviewed in more detail. Related work regarding behavior analysis is presented in Section 3.1, while related work regarding behavior cloning and validation is presented in Section 4.1.
- **Defining formal definitions and descriptions** for entities and concepts included in the PATDS learning cycle. Moreover, definitions to support behavior analysis and the automatic development of surrogate agents will be created to theoretically support the presented methodology. The formal definitions are provided in Section 2.3.
- **Design and implement components** for the behavior analysis methods that will enable efficient extraction of behavior patterns from real operational people and will consider cognitive and psychological aspects when dealing with civilian crowds. The design includes optimized data structures for traces, time-efficient graph traversal algorithms, definition of cognitive ontologies, new abstraction mechanisms, feature and instance selection for learning. The behavior analysis algorithm is presented in Section 3.2.
- **Design and implement** a new method for the selection of typical behavior pattern among the entire set of created patterns during behavior analysis. The method is based on clustering and implements custom distance measures. Details of the implementation are presented in Section 4.3.
- **Design and implement** the PATDS method, which combines different mechanisms to achieve quantitative evaluation of the learnt behavior and identify scenarios with poor performance. To achieve this, PATDS relies on behavior cloning that realistically reproduces the performance of the observed operational people and uses data-farming to explore the entire domain space and all possible simulation outcomes. The presentation of the overall method is provided in Section 4.2.
- **Perform empirical evaluations** of the main PATDS components. Qualitative evaluation of several discovered behavior patterns is presented, as is quantitative evaluation of

pattern clustering, behavior cloning, and behavior evaluation through data-farming. Evaluations are presented in Sections 5.2, 5.3, 5.4, and 5.5.

### 1.3 Scientific Contributions

While this dissertation makes several original scientific contributions delivered in this doctoral dissertation, the main one is a **novel, computer-supported training method** applicable to various domains that require extensive training and is composed of the following elements:

- The novel PATDS method, which encompasses a novel behavior cloning and its evaluation through data-farming experiments, combining graphical and symbolic reasoning, based on domain and cognitive features. The general overview of the approach is provided in Section 4.2 and the individual components are presented in Sections 4.3, 4.4, and 4.5. Published in a first-quarter SCI journal [34].
- A behavior learning algorithm (HMASDA) based on graphical and symbolic representation that exploits **not only physical but also cognitive aspects and features**. The contribution is presented in Section 3.2 and the validation in Section 5.2. Published in a first-quarter SCI journal [32].
- A novel mechanism for **pattern clustering** and **behavior cloning** based on graphical and symbolic representation, domain, and cognitive features. The contribution is presented in Section 4.4 and is published in the same journal paper as PATDS.

### 1.4 Impact and Publications

Most of the activities, scientific contributions, and results were performed and obtained during the execution of the project EUSAS (“European Urban Simulation of Asymmetric Scenarios”), which was financed by 20 nations under the Joint Investment Program Force Protection of the European Defence Agency. The system was designed and implemented as a prototype to address the issue of inadequate training procedure both in the sense of time efficiency and quality of achieved knowledge during mission rehearsal. The EUSAS system delivered: (i) a highly realistic 3D cyber environment VBS2; (ii) human behavior models on the basis of the latest findings from psychology; and (iii) a novel training system for validation and flexibility verification of the rules of engagement. The behavior analysis and cloning components were two of the core activities within this system, and thus one of the primary pillars that enabled a successful deployment of the EUSAS system.

The general overview of the project is presented in the journal *Computing and Informatics* [35] and two conference papers [36, 37]. The different mechanisms used to model human behavior were presented in various conference papers. A general overview of human behavior modeling is presented in two workshop papers [38, 39]. More specific methods for behavior analysis that were applied in EUSAS are presented in the *Information Society* conference [40] and in the *European Conference on Artificial Intelligence* [27]. A comprehensive overview of the HMASDA method [32] was published in the *Expert Systems with Applications: An International Journal*. The PATDS method for improved training [34] was published in the journal *Engineering Applications of Artificial Intelligence*. Methods for heuristic reasoning based on behavior recognition and path finding are presented in the conference paper [41] and in the *Informatica : an international journal of computing and informatics* paper [42]. A method for behavior cloning using supervised learning is presented in the *Information Society* paper [43].

Methods for analysis of human behavior based on traces were also implemented in the smart home [44] and smart city [45] domains. Generation of behavior strategies and improving driving skills were also conceptually defined for a car driving simulator [46].

### 1.5 Thesis Overview

This dissertation is composed of six chapters. Chapter 2 describes a general overview of the main concepts, fields covered in this dissertation, and the motivating domain. We start with a general

overview of agents and multi-agent systems in which we cover the main characteristics of agent systems and the two main types of interactions between agents. Next, the area of behavior analysis related to dissertation is presented. Third, the two target military domains – the Riot and Looting Domains – are discussed in more detail. Finally, we present the problem definitions used throughout the dissertation. First, we discuss the definitions for agents and other entities present in the simulation and implemented in the PATDS algorithm. We then provide definitions linked to the behavior analysis part of the PATDS algorithm and about domain-specific concepts.

Chapter 3 describes the behavior analysis component (HMASDA) that provides one of the key functionalities to enable the extraction of executable agent behavior models and later reuse them throughout the entire PATDS learning cycle. First, a survey of related work about behavior analysis is presented. In the remainder of the chapter, the technical details about the four main parts of HMASDA are provided.

Chapter 4 describes the PATDS algorithm that supports the implementation of the whole learning cycle. The chapter begins by surveying related work linked to behavior cloning, transfer learning, simulation-based training systems, and performance evaluation. This is followed by the main contribution of the dissertation. In its first step, PATDS involves operational people or software agents playing a few simulated rounds of a specific scenario. Traces from the simulations are captured and analyzed to extract behavior patterns that describe the behavior displayed in the simulations. Patterns are then processed to remove redundancy and are reused during data-farming experiments in order to evaluate the captured behavior of the observed trainees.

The experimental setup and evaluation of the results are presented in Chapter 5.

Chapter 6 summarizes the dissertation, provides the outline of the main original scientific contributions, and presents several ideas for future work.

## Chapter 2

# Background and Problem Definition

In this chapter we provide the general description of concepts related to the research areas of the doctoral thesis and an overview of the two target domains where the proposed algorithm has been applied. The chapter starts with an overview of concepts related to agents and multi-agent systems and presents how they are applied to cooperative and antagonistic domain problems. Next, the area of behavior analysis is presented and its main research challenges. The chapter concludes with a presentation of the two target military domains and the simulation tools used to obtain learning data from training sessions.

## 2.1 Problem and Domain Related Concepts

### 2.1.1 Multi-Agent Systems

The concept of agents is frequently used in computer science. However, a general definition of an agent is still not generally accepted yet and various explanations can be found in the literature. We will use the most complete definition provided by Russel and Norvig [47]. They define an agent as an entity that exhibits a high degree of autonomy, is able to perceive the environment through any kind of sensors, process the information, infer conclusion, and perform action based on reasoning. Other definitions build on the former by adding human-like features that agents must exhibit in order to be deemed intelligent: knowledge, purpose, cognition, and commitment [48, 49].

Even though, there are a multitude of agent systems, there are a handful of key properties [50] linked to agent entities:

- Limited access to information and incomplete capabilities to solve problems on their own. There can be too much information to be accessible or stored in one agent and the problem formulation might require a large number of agents in various locations to attain the goals and solve problems.
- Data are spread among a network of agents, thus can be accessed only indirectly through other agents. The system must adopt mechanisms that can identify the adequate data providers.
- Communication with other agents through defined protocols to exchange information (in the form of predefined messages) about the environment and transfer of required data stored locally.
- Asynchronous execution of actions in the system. Execution of one action in one agent is independent from others. This feature can be beneficial, since it can speed-up the overall execution; however, it can introduce synchronization problems with shared resources and with concurrent execution of dependent tasks.

- Lack of a central or global control, which requires agents to coordinate and monitor the execution of common tasks. This often requires a decomposition of the main problem into more easily manageable subproblems that are solved by individual agents.

The definition for a multi-agent system is similar as for a single agent, with an addition that there are multiple agents in the observed environment, where they interact with each other. The key difference between a multi-agent and a general system/environment is that agents are autonomous and intelligent entities that operate in a given environment in order to attain specific goals, which would be harder to achieve without any interaction between the entities. We can distinguish between two types of interactions: (i) cooperative [51], when agents in a multi-agent system collaborate to solve a joint task, attain specific goals, or maximize a specific utility function; and (ii) competitive interactions [52], where agents do not have the same goals and they seek to maximize their own utility function or gains. Researchers usually deal with one type of interaction, since they design one party, entity or team that competes with others. On the one hand, in cooperative settings these are usually teams. In competitive, on the other hand, we are usually dealing with individuals in a market-based environment, where game theory is applied for negotiations, market mechanism, auctions, and restricted matrix games.

In this thesis, we interchangeably use the term agent for different entities involved in training and simulations: human trainees, simulated characters, avatars in serious games and their cloned versions. The target domain and the two defined scenarios include both types of interactions. Two asymmetric groups of agents are included in the scenarios, where members of each group generally collaborate to solve the defined tasks (cooperative interactions) and might be in conflict with the members of the opposite group (competitive interactions).

## 2.1.2 Behavior Analysis

Behavior analysis is a general concept related to various research areas from ambient assisted living, activity recognition, social networks, to multi-agent analysis. This chapter deals with the latter.

### Opponent analysis and modelling

Opponent modelling is frequently used in antagonistic settings, where predicting the intentions of an unknown adversary and adapting to the changes in the domain caused by his actions is crucial to win or achieve specific goals. Opponent modelling is intended to provide predictions about the strategy adopted by the opponent in the next rounds of play, reason about its intentions, and design counteractions that will provide an advantage to the first player [53].

### Plan recognition

Schmidt et al. [54] defined plan recognition as a mechanism to infer characteristic sequences of actions, motives, and agent goals from their observed actions in the environment. There are several possible interactions during the plan recognition: (i) agents might be unaware of being observed, that is, their behavior is not influenced; (ii) agents cooperate and help the recognition process by selecting actions that make their intentions transparent; and (iii) agents might deliberately try to mask their intentions and impede the recognition process. The target plans are usually predefined and stored in a plan library, which implicitly specifies the behavior of interest that needs to be recognized and matched against the agent observations. This requires a large amount of predefined expert knowledge and limits the abilities of the mechanisms to discover novel knowledge.

The early attempts in plan recognition were based on predefined rules that were applied to describe the relations and observations in the target environment. Next, generalized frameworks for plan recognition were proposed [55], where underlying formal models were introduced to improve the detection mechanism, time-efficiency, and enable efficient model scaling. In recent years, a lot of work has been done using Hierarchical Hidden Markov Models (HHMMs) and Conditional Random Fields (CRF) that deliver efficient approaches, consider the likelihood information, and use machine learning support to automatically acquire plan models. In addition, novel systems are able to deal with analyzing the behavior of agents pursuing more than one plan at a time.

### Behavior identification and behavior libraries

One possible way of discovering agent behavior patterns is by developing behavior libraries that encode the complete repertoire of expected behavior. This technique focuses on matching the observations of agent behavior with library prototypes that are defined at design time. This approach has been successfully used in domains such as intelligent user interfaces [56], in virtual environment training [57], and games [58]. However, this approach is for the most part inappropriate for open or adversarial settings in which the behavior repertoire of the observed agents is too complex and unknown at design time. A key challenge is to enable on-the-fly modelling and strategic behavior acquisition. This is often linked to opponent analysis, since the system must be able to understand the observed behavior, identify key action sequences, and store them in a reusable manner.

The standard approach using behavior libraries adopts two phases: (i) off-line, where for a larger set of observed games the behavior of a team is analyzed and stored in a behavior library. These are usually low-level action sequence that represent tactical behavior; (ii) on-line, where the goal is to classify the behavior of an observed team into one of the patterns previously stored in the library. Although some concepts in this area are similar to our approach, the tasks we need to solve and the methods applied are significantly different. Extraction of strategic behavior is a more complex problem, since it requires observing the behavior for longer intervals. The mechanism we use for this problem is to recognize repetitive behavior, which indicates strategic behavior intrinsic to the observed agents. In addition, the patterns must contain enough information to enable behavior cloning, which is not a standard task in this area.

## 2.2 Target Military Domain

Among the most demanding simulation domains are security and military domains, mainly due to the opposition of two human groups, lack of clear rules of engagement, and many unknown or hidden variables enabling a huge number of possible outcomes, even from similar initial conditions. In these applications, an important part of training is engagement in Military Operations on Urbanized Terrain (MOUT) domains. Exhaustive training is necessary due to the unfeasible constraints on real-life testing: among other constraints, real-life conflicts are relatively rare making training in real life an infrequent event, and it is not sensible to send untrained personnel into real-life situations with potentially damaging consequences. Besides some practical exercises in pre-set physical polygons, MOUT multi-agent simulations are often introduced where a human is assigned the role of a squad leader, while the roles of peacekeepers are assigned to other humans or agents that interact with hostile humans, represented as cognitive agents [59].

In cooperation with security experts (partners in the EUSAS project), we chose two urban asymmetric domains for the simulated experiments, broadly inspired by actual missions. The domains represent confrontational situations, or open conflicts in urban areas, in which small groups of trained and equipped security forces are confronted by much larger groups of mainly civilian rioters and sometimes also a small number of insurgents. However, in all cases the event looks like a civil riot.

The implementation of the scenarios for simulated and human play was performed in collaboration with other project partners and is not part of the contributions of this dissertation. However, we describe the details of the simulations and agent modelling in more details in this section to improve the comprehension of the proposed method. Both scenarios were implemented using MASON (Multi-Agent Simulator Of Networks [60]), which is a multi-agent simulation tool where security staff and civilian agents interact. For human play, the 3D serious game Virtual Battlespace 2 (VBS2) as a human-computer interface between the trainee and the multi-agent simulation was used. The game was modified to provide feedback to the trainees regarding the emotional and cognitive states of their adversaries. The trainees observe the values of the cognitive levels of the opposing agents in a simplified way. While MASON computes the values from 0 to 100 and sends them back to VBS2, the cognitive state is presented with only three different markers on top of the avatars (see Figure 2.2). Using this approach, the trainees can interpret the overall cognitive state of the crowd and act accordingly. During the expert or later trainee review, however, exact numbers are more informative and can help better define rules for engagement in subsequent training sessions. The behavior and cognitive

states of civilian agents was defined based in the sociological and cultural background of the defined scenarios by experts in the field of behavioral psychology. The various behaviors were modeled using the PECS (Physical conditions, Emotional state, Cognitive capabilities, and Social status) reference model [61], which is able to characterize the civilians with several personality factors and internal processes that generate situational-dependent behavior that is driven by their motivational and emotional states. To accommodate various behaviors a generic agent structure within PECS was developed. The internal structure of generic PECS agents is defined using concepts from system theory: (i) *inputs* are sensory perceptions in the form of filtered messages from the connector linked to MASON; (ii) *internal states* consist of various factors that drives or modulates the agent, for example, Cognition, Motives, Social Status, Emotion, Physical conditions, etc. These components are connected in a network of causal dependencies and information flows, following a state transition function, defined for each internal state. Since the mathematical form of the functions is not constrained, the behavior of the agent can be tailored to specific needs. Functions can be composed of a continuous (differential equations) and discrete (spikes in values due to external events) part. (iii) *outputs* are atomic actions selected from a predefined sequence (patterns) that are activated at a specific time based on the internal states and the current motive of the agent. The generic agent can represent simulated agents, where actions are generated, or just follow the behavior of real operational people. As seen by MASON, both types of behavior are the same and regarded as an agent. The project team and reviewers regard the PECS simulation as a reasonably realistic representation of the real world since it was validated by security experts. The PECS model, in the context of this work, is just a mechanism to define the behavior of the simulated agents. Moreover, MASON and in turn PECS can be regarded as a black box component, since, in theory, PATDS can analyze and reproduce any behavior that follows defined rules and is thus not random, as long as it is able to export a trace in the form of detailed simulation logs.

The behavior of security agents, on the other hand, is not modeled in the same way as that of the civilians. Their cognitive state is not relevant since their performance is what matters, and there is no feedback from their cognitive state to the simulated rioting agents' others than through their actions, as in computer games. However, the trainees have to interpret and respond to the cognitive elements demonstrated by the civilians since, for example, dealing with an aggressive opponent might demand different treatment than dealing with a frightened one. Additional information regarding the simulation can be found in Kvassay et al. [35].

A single scenario run can evolve in different ways, as it depends on the instantiation of the specific parameter settings of a domain, for example, it can evolve during a bright day, or during a rainy night. Having set the parameters, one simulation is one execution of that scenario. Due to the stochastic nature of the PECS models, simulations of the same scenario start similarly, but the evolution of the event can vary up to a point or even substantially as in the real world; in particular, it varies since the conflict can take one of several paths based on the participants' decisions. For example, a single violent action in a tense situation may or may not trigger a major violent outbreak. Our motivation in this paper was to demonstrate that the algorithm is able to capture various behaviors displayed by human trainees, and to test whether the discovered integrated behavior patterns are meaningful enough for experts to use them to evaluate all aspects of the overall performance of trainees during the scenario training. For example, psychological factors have a very important role in the event, and they are generated, observed and analyzed throughout the whole simulation.

In our experiments, two different types of behavior were analyzed in each scenario: one using a fixed-policy simulated in MASON and executed by artificial agents, and the other obtained from a group of security staff members that played both domains several times using a similar strategy. The simulated security agents in the Riot domain fixed-policy simulation stochastically followed the scenario and modified their performance mainly in response to the actions of the civilians. Squad members responded proportionately to the anger level and violence displayed by the rioters. The strategy in the fixed-policy simulation was not considered adequate by the experts, since it too often resulted in a full-scale conflict. Somehow, the expansion of violence was an emergent property of our simulated groups of agents.

Similarly, the Looting domain fixed-policy simulation involved the squad moving through the scenario without active interference until the first direct interaction or provocation by a looting civilian. As in the first domain, most simulations typically resulted in a full-scale conflict unless the simulated agents representing civilians were rather peaceful. In summary, the full simulation

of all agents provided a rather limited, one-sided performance. In contrast, two strategies were obtained from security forces that tried to resolve the situation according to the standard European rules of engagement, and to use as few aggressive actions as possible until needed. Each squad member played the scenarios on a computer running a VBS2 client that was connected to the MASON simulation through a real-time CORBA-based interface (Common Object Request Broker Architecture). A DLL plugin in C++ that integrates in VBS2 was developed. The plugin receives data from MASON, encodes an SQF (Status Quo Function) string and sends it to VBS2. Similarly, any changes, events and states in the VBS2 game are periodically sent to the plugin as a string object, which is decoded and the data is sent via the CORBA communication protocol to MASON. Even though, the use of CORBA might be outdated and inefficient, the performance of the prototype setup was not significantly affected. Moreover, due to the commercial limitations of the VBS2 API (VBS2Fusion) the use of CORBA represented a reasonable trade-off between performance and price. However, in case of a future deployment in a production environment, VBS2Fusion should be purchased and implemented instead of the CORBA-based interface.

### 2.2.1 Riot Domain

In the first domain, Riot Domain (RD), a tussle develops between security forces and civilians demanding to enter a compound. The primary goal of the gate guards is to prevent forced entry, and the second goal is to either calm or if necessary disperse the crowd.

The bird's eye view of the riot domain is shown in Figure 2.1. The entry to the camp is on the top-left side, where five security personnel (two close to the gate and three on the tower), denoted with white circles, guard the entry point. Green, yellow, blue, and red circles represent civilians in different mental states. For example, green represents a normal state of the civilian, yellow is slightly agitated, blue is afraid, and red is angry. The cognitive process is highly affected by emotions and mental states of individuals [62]. In our case, we simplify our view of the two different concepts and regard the mental states as tightly linked to the cognitive process, since they directly influence the behavior adopted during the training and simulations. At each point or cognitive state specific PECS equations guide the behavior of the agents.

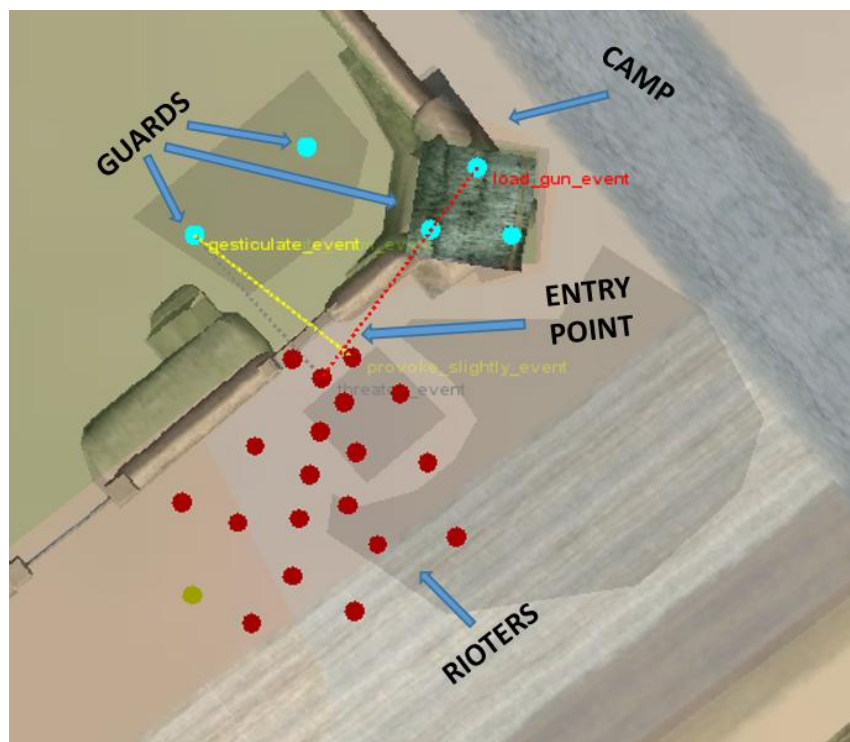


Figure 2.1: A bird's eye view of the MASON simulation screen of the Riot Domain.

For easier interpretation, every interaction between each type of agent in the simulation is represented with a dotted line from the origin to the target agent. The type of interaction is written next to the line. The color denotes the severity of the action: Grey for calm actions both from the civilian or soldier agents, yellow are stricter actions, and red denote strict actions, which could also be lethal.

Soldiers and civilian agents can perform different actions within the scenario. A complete list of actions for the riot domain is presented in Table 2.1. The left column lists soldier actions, while the right column lists civilian actions. All actions are self-explanatory, only two civilian actions require some additional explanation: *attack\_event\_ts* represents a throw stone attack; *attack\_event\_ww* represents an attack with weapon.

Table 2.1: List of all possible soldier and civilian actions in the Riot Domain.

| Soldier actions [Riot Domain] | Civilian actions [Riot Domain] |
|-------------------------------|--------------------------------|
| moving_event_SA               | moving_event                   |
| communicate_calm_event        | negotiate_event                |
| communicate_warning_event     | provoke_slightly_event         |
| gesticulate_event             | provoke_aggressive_event       |
| load_gun_event                | threaten_event                 |
| show_of_weapon_event          | attack_event_ts                |
| gun_shot_warning_event        | attack_event_ww                |
| gun_shot_effective            | attack_failed                  |

Note that the MASON bird's eye view simulation is an internal presentation for expert and later trainee analysis. During normal training sessions, soldier trainees played the same scenario in a serious game environment, implemented in VBS2, where they see human avatars with objects as close to real world as possible, with weapons, either hidden or exposed, and bearing facial and bodily expressions to identify their cognitive states, as can be seen in Figure 2.2. However, as with computer games, the amount of cognitive information available from avatars is too inferior to real life, so some additional information was provided to obtain better cognitive clues.



Figure 2.2: The 3D representation of the scenarios as seen by the trainees in VBS2.

### 2.2.2 Looting Domain

The second domain, Looting Domain (LD), was designed to enhance the dynamic cooperation of squad members, since they must work together to successfully resolve a conflict. The scenario involves a large crowd of civilians looting a shop in an urban environment. The approaching security patrol is supposed to stop the looting while avoiding unnecessary escalation, and to avoid causing resentment among civilians, but at the same time to respond firmly if needed. The patrol achieves this by constantly assessing the threat level at the individual and crowd levels, and adopting actions based on the cognitive and emotional states of the civilian agents. In both domains, the aim of the squad is on the one hand to achieve the objective with the least possible damage in both the physical and cognitive senses to all participants in the conflict, and on the other hand to show determination, effectiveness, and strength when needed, in particular toward very hostile participants, such as hidden terrorists.

In Figure 2.3, a bird's eye view of the domain is presented. Squares represent buildings, gray belts represent streets, the orange and yellow zones represent various areas of interest (conflict, looting, spawning), the purple building is the looted shop, and the conflict zone is in front of the shop. Small circles represent human or artificial agents; medium-sized purple circles are security forces, while all others are civilian agents labeled in different colors that denote their cognitive traits. Green represents looting civilians, blue represents those who are frightened/afraid, red is angry, orange is frustrated, and yellow represents calm bystanders.

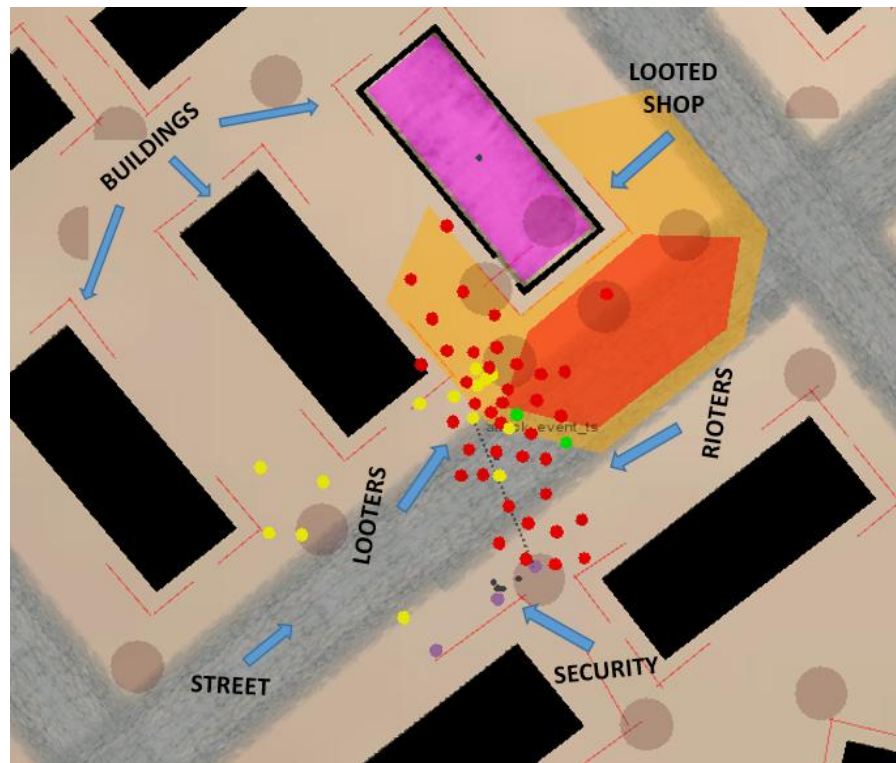


Figure 2.3: A bird's eye view of the MASON simulation screen of the Looting Domain.



Figure 2.4: The 3D representation of the scenarios as seen by the trainees in VBS2.

A list of all possible actions in the Looting Scenario is presented in Table 2.2. The left column lists soldier actions, while the right lists civilian actions. There are a handful of differences between the two scenarios. The main is the introduction of additional looting-related actions for civilians (shop\_entry and take\_event) that can occur near a specific building of interest. Next, there are fewer actions available for soldiers and civilians; however, the focus in the Looting domain is more on the movement dynamics as compared to the Riot domain, where the key is to perform appropriate actions at the right time. The soldier actions, in the left column, marked with blue, are not basic actions that trainees can perform, but are higher-level actions that the analysis algorithm detects from basic actions. Any custom action can be defined and considered as the basic action. This provides additional semantics to better explain the strategic behavior of the observed agents.

Table 2.2: List of all possible soldier and civilian actions in the Looting Domain.

| Soldier actions [Looting Domain] | Civilian actions [Looting Domain] |
|----------------------------------|-----------------------------------|
| moving_event_SA                  | moving_event                      |
| loudspeaker_event                | take_event                        |
| load_gun_event                   | shop_entry                        |
| show_of_weapon_event             | threaten_event                    |
| gun_shot_warning_event           | attack_event_ts                   |
| gun_shot_effective               | attack_event_ww                   |
| approaching_civilian             | attack_failed                     |
| approaching_looting_civilian     |                                   |
| protecting_object                |                                   |

## 2.3 Problem Definition

This section provides formal definitions and descriptions of concepts used throughout this thesis. First, definitions of the entities performing in multi-agent simulations and those involved during behavior analysis are provided. Next, we define key concepts introduced during behavior analysis. Finally, we provide formal definitions linked to behavior cloning and its evaluation.

### 2.3.1 Agents and Other Entities

The formal definitions of agent entities are taken from Lettmann et al. [63], with extensions related to the specifics of the domain and the inclusion for cognitive features that resemble human behavior.

**Definition 2.1 (Agent A).** Agent  $A$  is in general a tuple  $(S', I_A, R_A, C_A, D_A, M, \delta_A)$ , where:

- $S'$  is a set of finite domain states that empirically describe the current situation of the environment an agent is in,
- $I_A$  is a set of internal states, represented with attributes, of the agent that influences its decision process,
- $R_A$  is a set of roles the agent  $A$  can adopt (domain specific agent roles described in an ontology),
- $C_A$  is a set of cognitive states that influence the behavior of agent  $A$  (domain specific cognitive and mental states of the agent described in an ontology),
- $D_A$  is a set of states linked to the decision process of the agent that defines the circumstances when specific actions are triggered,
- $M$  is a set of actions that agent  $A$  can perform in the target environment (domain specific actions described in an ontology),
- $\delta_A$  is a decision function that maps from the current decision state to a new decision state of agent  $A$ .

A decision state  $d_A \in D_A$  is a complex representation of the agent's cognitive state and decision process, which changes according to the internal state  $i_A \in I_A$  of agent  $A$ , its current role  $r_A \in R_A$ , its cognitive state  $c_A \in C_A$ , and the current action  $m_A \in M_A$  performed by the agent. This is required to define discrete transitions between decision states when applying the decision function  $\delta_A$ .

The basic notion in the domain is an action. It represents which an agent carries out in the domain, as well as the location where these actions occur. Actions in the domain are represented as spatial ( $S$ ) and temporal ( $T$ ) envelopes. The former contains a set  $S = \{M_1, \dots, M_n\}$  of similar actions that occur in a constrained region of the domain. The envelope is defined to accommodate actions at different levels of generality, from low-level actions that agents can perform to high-level actions that represent strategic behavior. This is required in order to enable the extraction and description of strategic behavior displayed by the agents. For each spatial polygon  $s_i \in S$ , a temporal constraint  $t_i \in T$  is assigned which denotes the sequential order of the spatial polygons in  $S$ .

**Definition 2.2 (Trainee T).** Trainee  $T$  is an individual or a group of individuals performing a training task as one entity.

**Definition 2.3 (Simulated Agent SiA).** Simulated agent  $SiA$  is a computer program with a fixed-policy strategy, which consists of predefined behavior that is triggered based on reactive percepts from the simulation.

**Definition 2.4 (Surrogate Agent SA).** Surrogate agent  $SA$  is a behavioral simulated clone of a trainee or of the simulated agent. It is an extension of agent  $A$  (Definition 3.1) to accommodate the generated behavior model from observations. The  $SA$  can only adopt one specific role  $r_A$  and only actions from  $O_o$ . Moreover, the decision function  $\delta_A$  is specifically defined for the problem domain. It is represented by a tuple  $(S', I_A, i_A, r_A, \Delta, O_o, \delta_A)$  where:

- $S'$  is a set of finite domain states,

- $I_A$  is a finite set of internal states,
- $i_A$  is the current internal state of SA used in the decision function to compute the next internal state,
- $r_A$  is the role of SA from the ontology of agent roles  $O_r$ ,
- $\Delta$  is SA's asserted knowledge, i.e., policy in the form of patterns extracted during behavior analysis. The provided policy contains all information relevant to the agent's decision making,
- $O_a$  is the ontology of agent actions, where the agent can perform only leaf actions,
- $\delta_A : S_i \times i_A \times r_A \times \Delta \mapsto i'_A \in I_A$  is a decision function that calculates the next internal state  $i'_A$  of the agent based on the current domain state  $s_i \in S$ , the internal state of the agent  $i_A$ , its role  $r_A$ , and the policy function.

### 2.3.2 Behavior Analysis

**Definition 2.5 (Trajectory Envelope).** A trajectory envelope is a triple  $E = (S, T, C)$ , where:

- $S = \bigcup_i K_{Ai}$  is the spatial envelope of agent  $A$ ,
- $T = \bigcup_i T_i$  is the temporal envelope of agent  $A$ ,
- $C = \bigcup_i C_{Ai}$  is the cognitive envelope of agent  $A$ .

The spatial envelope  $S$  comprises a set of similar actions performed in a confined area of the domain map. This area may vary in size, depending on the number of included actions, and the way they are fragmented. Actions can be presented at different levels of generality; therefore,  $m_A \in O_a$ , where  $O_a$  is an ontology with hierarchically-structured concepts. Next, the temporal envelope represents time constraints for executing the actions present in the envelope  $T = \{T_1, \dots, T_n\}$ . The type of  $T_i$  can be either numerical or categorical. Lastly, the cognitive envelope represents the cognitive states of the agents'  $C = \{C_{Ai}, \dots, C_{An}\}$ . The cognitive states define the circumstances for agent  $A$  to perform specific actions  $m_A$  at certain time constraints  $T_i$ . As before, the cognitive states can take any value from the cognitive ontology  $C_{Ai} \in O_c$ .

**Definition 2.6 (Strategic Pattern).** A strategic behavior pattern of an agent  $A$  is a tuple  $\Phi_A = (E, D, T)$ , where:

- $E = \bigcup_i E_i$  is a set of trajectory envelopes that compose the strategic pattern,
- $D = \bigcup_i D_i$  is a set of symbolic descriptions for each trajectory envelope in the pattern,
- $T = \bigcup_i T_i$  is a set of target agents for each trajectory envelope in the pattern.

One pattern represents part of the strategic behavior displayed by an agent. The overall strategy is usually composed of several patterns that fully define the high-level intentions of the agent. At a lower level, a strategic pattern is composed of several trajectory envelopes that contain movements or actions that the agent performs. For each envelope, a set of symbolic descriptions  $D = \{(conditions, action)_1, \dots, (conditions, action)_n\}$ , where *conditions* is a string of domain features  $f_i$  created during behavior analysis using a rule inducer algorithm, linked together with logical operators  $conditions = \{f_1, \dots, f_n\} \times \{<, >, >=, <=, =\}$  that is provided to ease the comprehensibility of the envelope. In addition, each envelope may contain actions directed towards an agent; therefore, the ordered set  $T_A = \{V_1, \dots, V_n\}$  is included in the pattern.

**Definition 2.7 (Decision Model).** The entire set of extracted patterns for an agent represents the Decision Model  $DM_A = \{\Phi_1, \dots, \Phi_n\}$  of the agent  $A$ .

The Decision Model  $DM_A$  is a set of strategic behavior patterns  $\Phi_i$  that comprehensively describe the strategic behavior displayed by the agent in terms of movements, performed actions and the underlying intentions.

**Definition 2.8 (Agent traces).** Traces  $P$  of agent  $A$  are all relevant attributes collected during simulated or real training experiments. They include movements of agent  $A$ , performed actions, its internal states, and domain states, including attribute descriptions of all other entities included in the simulation.

**Definition 2.9 (Behavior analysis).** The behavior analysis algorithm maps the collected agent traces  $P$ , from simulated or real training experiments, and domain knowledge, in the form of Ontologies  $O$ , to a Decision Model  $DM_A$  for each observed agent. The algorithm is defined as a function  $f: P \times \{O_1, \dots, O_n\} \mapsto DM_A$ .

**Definition 2.10 (Action Graph).** The Action Graph ( $AG$ ) is a directed graph constructed from all agent traces for an arbitrary number of simulation plays:  $f: P \mapsto AG$ . Vertices of the  $AG$  are positions of agents in the domain, which include domain state information at that specific time step. Edges are transitions from one position to the other in the domain and include information about the performed actions and interactions.

**Definition 2.11 (Abstract Action Graph).** The Abstract Action Graph ( $AAG$ ) uses domain knowledge in the form of ontologies  $O_i$  in order to create generalized or abstracted concepts from  $AG$ :  $f: AG \times \{O_1, \dots, O_n\} \mapsto AAG$ .

### 2.3.3 Domain-Specific Definitions

**Definition 2.12 (Simulation play).** A Simulation Play  $SP = \{e_1, \dots, e_n\}$  is defined as a collection of low-level events  $e_i$  representing discrete changes of a multi-agent system. The individual events are computed by the simulator and contain the snapshot of the entire domain state for a specific time step.

**Definition 2.13 (Simulation domain SD).** Simulation domain  $SD$  represents a real-life simulated environment with all constraints and relations between entities.

**Definition 2.14 (Data-farming).** Data-farming experiment  $DFE = \{SP_1, \dots, SP_n\}$  is a collection of simulation plays  $SP$  under all reasonable circumstances

## Chapter 3

# Human Behavior Analysis

This section provides a description of HMASDA, the behavior analysis algorithm that is used to generate behavior patterns based on the collected training examples. The algorithm comprises the initial steps of PATDS. In Section 3.1, a survey of the related work from the area of behavior analysis is provided. Section 3.2 presents the four main components of HMASDA and how they interact to construct the behavior patterns.

### 3.1 Related Work

This work is also closely linked to agent-behavior modeling, which has been studied in various domains. For example, the RoboCop simulation soccer game spurred many successful efforts towards the use of learned-opponent models to adapt the team for adversarial strategies. Given a set of segments from simulation logs, agglomerative clustering was applied to classify the team's behavior, and to generate counter tactics ([64]). Support vector machines have been used to model defensive strategies in a simulated football game [65]. Both papers address behavior analysis, as does this paper, but from a different point of view. They classify the behavior of the opponents into predefined strategies, and select the most appropriate counter-strategy for their team. We, on the other hand, do not use predefined behavior libraries, and do not provide counter-strategies.

Stracuzzi et al. [66] demonstrated a knowledge transfer method by observing videos of American football in order to control agent players in a game simulator. It included object tracking, dynamic time warping for action recognition, and an Icarus cognitive architecture to infer strategies in the form of rules. Even though it has similarities to this paper, it does not address the cognitive aspects of agent behavior, it does not provide a graphical representation of the observed strategies, and does not provide a feedback that humans can understand.

The COLTS algorithm that tackles a similar problem has been presented in [67]. The method learned team behavior from observations by creating behavior maps that bind together state-action pairs extracted from simulation logs. The behavior maps are used to determine which actions should be performed at various domain and agent states. This information is later used to reproduce the behavior of the observed team. Several steps of the learning algorithm involve manual definition of domain components and context segmentation. In contrast, all steps of HMASDA are automated and expert knowledge is required only during domain definition, for the creation of ontologies. Moreover, COLTS does not focus on the comprehensibility of the model, while HMASDA aims to create human understandable strategy explanations.

Several contributions in the area of behavior analysis adopt plan recognition algorithms. The common denominator for this group of methods is a predefined library of action sequences that are matched to the observed trace. Kaminka et al. [68] presented an algorithm for the recognition of actions performed by a team of agents. This is important since agents can recognize what their opponents or team members are doing, and this enables better cooperation, better coordination, or improved adoption of appropriate response actions. Their approach introduces several important characteristics: it can handle missing observations, it is probabilistic, it is single-goal driven, and it uses prior knowledge of the team to infer the overall performance. Later, Zilberbrand and Kaminka [69] improved the plan recognition algorithm to consider complex multi-featured observations, and to address situations where matching

observations to the plan library, and computing a recognition hypothesis, becomes expensive. The algorithm constructs a decision tree that efficiently matches multi-feature observations to the plan library. Geib and Goldman [70] later advanced their work and presented PHATT, an algorithm for plan recognition based on tree structures, which is goal-driven and is able to cope with interleaved goals, partially ordered plans, and missing observations. Contrary to the above-mentioned papers, our work does not use predefined behavior libraries to recognize intentions and plans of the team.

Iglesias et al. [71] merged various approaches, and proposed an opponent modeling approach for the Robot Soccer domain. Opponent modeling gives an important advantage in Robotic Soccer, since it provides insights into the performance and behavior of your opponent. Identifying what the opposing team is doing can improve the performance of your own team, since it gives you the ability to devise countermeasures, and to execute specific actions best suited to counter the detected strategic play. The authors propose a two-phase method. Firstly, an opponent model is constructed from an ordered sequence of recognized atomic events (traces). These sequences are then segmented into relevant subsequences of the same length, named *tries*. Their relevance is computed using the relative frequency and the level of support. The discovered sequences are stored in a pattern library, and later used on-line to detect specific plays adopted by the opposing team. Even though their approach enables the identification of behavior without any predefined libraries, they can discover only basic sequences of actions, while our method provides graphical and symbolic descriptions for the high-level strategic behavior and intentions of agents.

AI has been widely used in games to model and analyze dynamic situations for decision support in military operations. Brynielsson [72] proposed a decision support tool for providing command and control support based on game theory, coupled with traditional AI methods. The author presented a scenario involving two conflicting groups. However, in contrast to our approach, he suggested actions to the decision makers on how to tackle the problem, whereas we analyzed the observed behavior and provided a comprehensive explanation about the strategic behavior encountered during the play.

Cil and Mala [73] proposed a two-layer hybrid agent architecture to match the needs of future large-scale disaster evacuation and military operations. Their approach involves modeling the agents involved in the conflict and providing a simulation environment for them, while ours analyzes their behavior and extracts additional knowledge from their actions.

Perry et al. [74] explored the benefits of using decision support methodologies for improving the performance of less-experienced personnel in emergency response situations. They showed that the adopted decision support tools, providing feedback on the most appropriate actions to adopt or the resources and methods to employ in emergency situations, improved the response and communication of less-experienced emergency responders. When compared with experienced commanders, the differences in behavior were less evident. They obtained the data for defining the decision support system (DSS) manually, whereas our contribution presents a mechanism to extract the patterns to be considered during the decision-making process automatically.

Loh et al. [75] have analyzed navigational action sequences in order to differentiate between expert and novice performance, and to track how the behavior of novices improves throughout training rounds. They divided the domain map into several smaller grids, and compared the similarity of navigational patterns by using the Jaccard similarity. They showed that it is possible to successfully distinguish between the two types of players and that there is an optimal size to divide the game map into in order to obtain the best detection results. Our work is also based on basic movements of agents; however, we do not compare trajectories of different participants, but we extract their high-level strategic behavior.

Camacho and Liberatore [76] developed a DSS for predictive police patrolling, with the aim of reducing criminal acts throughout the city of Madrid. The DSS can predict future incidents, and can determine the most efficient police patrolling distribution in various districts. The authors noted that such intelligent systems are an effective addition to the classical experience-based systems, and to police intuition. Our work is related to theirs, since the output of our strategy discovery method can be used as part of a decision support tool; however, we provide a more complex algorithm capable of extracting temporal and spatial behavior patterns, and providing explanations for the discovered patterns.

Kaluža et al. [77, 78] also analyzed the behavior of individuals from observations of low-level movements. However, their motivation was to detect suspicious behavior by combining evidence from multiple detected events, and not to extract complex movement patterns and strategic behavior, as in our case. They proposed a two-step detection system, where they firstly detected trigger events from interactions of a group of agents, and then translated the detected evidence to a degree of suspicion. They developed a novel utility-based plan recognition detector with arbitrary utility functions. They compared their approach using three detectors in the domains of a simulated airport, and in a dangerous-driver situation. The focus of their work is not to analyze and describe the high level strategic intentions of the agents, but to detect suspicious behavior based on agent traces.

A similar contribution has been presented in Arroyo et al. [79] where they tracked the movement of agents using segmentation and Kalman predictions. Next, they classified basic behaviors (exits, looting, etc.) in the video-surveillance systems using SVM kernels to compute distances between appearance features such as GCH, LBP, and HOG. Their analysis of human behavior is limited to basic actions, while ours focuses on the strategic, more complex behavior.

A related research field is the problem of domain model acquisition from action traces alone. Gregory and Cresswell [80] presented an approach based on the LOCM and LOCM2 family of algorithms [81] that generates relations and static domain predicates for the Peg Solitaire and Freecell card games by analyzing player actions captured during game-play. The constructed domain model can explain the dynamic and static relations of arbitrary board games. Their approach differs from ours in two ways: (i) besides action, we consider also movements that occur in the domain and include them in the model; (ii) since we analyze humans, we added cognitive and emotional features to the algorithm.

Champalle et al. [82] proposed a set of models and the D3KODE prototype for sharing knowledge of expert tutors using training simulators. For the knowledge sharing task, observation, analysis and debriefing of the interactions of trainee operators are essential for young tutors who do not have the expertise of confirmed tutors. Their approach is based on digital traces and a visual synthesis of trainees' activities by using the knowledge of experts.

The algorithm for behavior analysis builds upon MASDA [83, 84], which the authors originally demonstrated on the RoboSoccer domain that transforms multi-agent sequences to a graphical form, and applies hierarchically-ordered domain knowledge to generate graphic and symbolic strategy descriptions in the form of rules. We have designed a new system by extending the MASDA approach to address the cognitive dimensions when modeling human players. Moreover, we provide a new mechanism for graph construction and pattern extraction that can cope with larger trace graphs. HMASDA [32] also constructs complex patterns with a novel procedure that differs from MASDA by applying a more straightforward machine-learning approach for learning data selection. HMASDA was defined and implemented on its own from scratch, using only the knowledge and experience obtained from the MASDA algorithm.

## 3.2 Hybrid Multi-Agent Strategy Discovering Algorithm

The discovery of informative strategic integrated behavior is performed using the HMASDA algorithm. The central task is to transform raw multi-agent physical and mental action traces and states to a sequence of strategic action concepts that represent the strategy human trainees adopt in order to achieve certain objectives. The introduction of cognitive ontologies and features, an upgrade of the basic algorithm, allows to combine physical and cognitive aspects during the behavior analysis in order to better grasp the influences that the psychological aspects have on the actions performed.

The practical task is twofold: a) to provide help to decision-makers, experts and mission planners when examining the behavior of opposing forces and designing the rules of engagement for a specific mission; and b) to improve the training of the trainees. The constructed behavior models consist of graphical and symbolic parts, facilitating the comprehension of the strategic concepts dealing with physical and cognitive aspects. The key addition to the MASDA algorithm is the introduction of the psychological/cognitive properties. HMASDA treats all aspects of behavior and all states as being equally important, which enables natural integration in algorithmic, graphical and symbolic ways.

Algorithm 3.1 presents a general overview of key algorithm components. We can segment the algorithm into four groups, based on the tasks it needs to solve. First, it observes how human

or simulated agents perform specific tasks (line 1) by creating low-level attributes from agent traces. In practice, it parses all the simulation logs and creates time series with data-value pairs. Second, it creates a connected network (Action Graph) of low-level actions taking into account the influence the actions have on each other (line 4). In addition, each node in the Action Graph is semantically tagged using the set of domain ontologies in order to traverse and process each node during abstraction and feature extraction more efficiently (line 6). Third, low level actions in the Action Graph are abstracted in order to obtain a high-level definition of the strategic action concepts (lines 9–14); Finally, the most important sequences of actions are identified, labeled as cognitive and physical behavior patterns, and explained using symbolic descriptions (lines 17–21). The four main parts of the algorithm are presented in more details in the next subsections.

---

**Algorithm 3.1:** Hybrid Multi-Agent Strategy Discovering Algorithm

---

**Data:**  $P$  – Traces of agent play,  $O$  - Set of Ontologies,  $len$  – target length of the pattern  
**Result:**  $DM$  – The decision model composed of strategic patterns of length  $len$

```

1:   $AL \leftarrow \text{parse}(P)$  //Parse and process agent traces
2:
3:  //Create an action graph from all agent traces
4:   $AG \leftarrow \text{createAG}(AL)$ 
5:
6:   $AG \leftarrow AG.\text{mergeOntologies}(O)$ 
7:
8:  //Create an AAG by merging nodes of the AG
9:   $AAG \leftarrow \text{createAAG}(AG)$ 
10: for each  $x, y \in AAG$  do
11:   if  $\text{similar}(x, y)$  then
12:   |    $\text{merge}(x, y)$  //Merge nodes x and y into z
13:   end if
14: end for
15:
16: //Find repetitive action sequences
17:  $\text{sequences} \leftarrow \text{findRepetitiveSequences}(AAG)$ 
18: for each  $\text{seq} \in \text{sequences}$  do
19:   //Create learning examples and induce IF-THEN rules
20:    $\text{describe}(\text{seq})$ 
21: end for

```

---

### 3.2.1 Learning Data

In order to extract meaningful behavior patterns the algorithm requires data exports regarding all relevant changes, actions and traces of basic agent movements. These action sequences are obtained during the agent-based simulations or virtual training. All the event logs are stored in the Unified Knowledge Base (UKB). Regardless of the way the log files were obtained, HMASDA can analyze them and store the results in the XML format or back to the UKB. The event logs comprehensively describe situations that can occur in a domain context. Different domains can support different behaviors. Since we are dealing with two antagonistic groups of agents (civilians and security forces), both with different actions, goals and states, log entries are different for each group of agents. All attributes specific for each agent role are enlisted in Table 3.1.

Table 3.1: Event log entries for the two main agent roles

| Security force       | Civilians            |
|----------------------|----------------------|
| timestamp            | timestamp            |
| role                 | role                 |
| id number            | id number            |
| target id number     | target id number     |
| atomic action        | atomic action        |
| coordinate x         | coordinate x         |
| coordinate y         | coordinate y         |
| area                 | area                 |
| standby level        | anger level          |
| evaluation level     | fear level           |
| energy level         | need level           |
| leading motive       | arousal level        |
| leading motive level | energy level         |
|                      | leading motive       |
|                      | leading motive level |

Even though most of the listed attributes are rather simple to understand, a short description might be useful to understand the type of information contained in the learning data. Each trace has a precise timestamp to identify when it occurred during the simulation. For the observed agent a role and its id number is provided. In case the agent interacts with another agent, the target id number of the second agent is provided. Next, the performed action, the coordinates and the area where the action occurred are listed. When the trace describes the security force agent, specific values for computed internal states are provided. Note that some are computed only during simulated play. The standby level is provided by the differential equations in the PACS model and describe how strong is the need of the agent to wait and assess the situation in the domain. The evaluation level is an estimate of the situation severity in the domain and is based on the number of aggressive actions, and the level of key civilian parameters (anger, fear, arousal, etc.). This value is computed for both simulated play and training. The energy level is computed using the PACS model and estimates the remaining energy of the agents. The leading motive is defined only during simulated play and is based on the behavior routines that are triggered in response to the stimulus from the environment (patrol motive, secure gate motive, attack, etc.). The leading motive level provides an estimate of how confident the PACS model is regarding the execution of the behavior routine. The civilian agents are defined with four additional attributes computed by the PACS model (anger, fear, need, and arousal) at specific time steps. Numerical attributes for both roles can take real values from the interval [0,100]: 0 - low influence, 100 - high influence.

The parser module generates a list of objects describing agent actions. The Action object is implemented as a data structure (linked list) containing all the relevant information obtained from the simulation logs: timestamp, agent role, basic agent action, position, area, cognitive state, etc. This data structure is in the next step of HMASDA used to generate an Action Graph.

### 3.2.2 Action Graph Construction

The main data structure of HMASDA contains all information about the strategic concepts in the AG. It is constructed in the initial part of the algorithm from basic agent traces. The  $AG = \langle V, E \rangle$  is a directed graph constructed by linking together sequential actions from the agent traces. For example, the nodes  $v_i$  and  $v_{i+1}$  represent actions  $m_i$  and  $m_{i+1} \in M_A$ . If the action represented by node  $v_i$  from the set of vertices  $V$  ( $v_i \in V$ ) is followed by an action represented by the node  $v_{i+1} \in V$ , then  $v_i$  and  $v_{i+1}$  are connected by edge  $e_n$  from the set of edges  $E$  ( $e_n \in E$ ). This data structure is part of the graphical representation of HMASDA; thus node positions correspond to the locations where agents performed the represented action in the domain space. Moreover, each node contains all relevant information that can be extracted from the traces. This enables

HMASDA to generate more general concepts, and symbolic descriptions in the next steps of the algorithm. The procedure that creates the AG is described in Algorithm 3.2. The algorithm processes every movement (position in the domain) of every agent and (lines 4–14) creates AG nodes for each position (lines 6 and 8). The created nodes are enriched with additional information: agent role, performed action, the specific instance, the target entity, and a set of domain features from Table 3.1. Next, for each created pair of nodes an edge is created (line 12). The end of the sequence is marked as an end node and an edge that connects the is created (line 10).

---

**Algorithm 3.2:** Action graph construction.

---

**Data:**  $P$  – Traces of agent play

**Result:** AG – An Action Graph linking action concepts from  $P$

---

```

1:  $TR \leftarrow \text{parse}(P)$ 
2:  $AG \leftarrow \text{new AG}()$  //Create an empty action graph
3:
4: for each  $a \in TR.\text{agents}$  do
5:   for each  $p \in TR[a].\text{position}$  do
6:      $n_1 \leftarrow AG.\text{addNode}(a, p.\text{role}, p.\text{action}, p.\text{start}, p, p.\text{target}, p.\text{feat})$ 
7:     if  $p.\text{next}.\text{next} \neq \text{null}$  then
8:        $n_2 \leftarrow AG.\text{addNode}(a, p.\text{next}.\text{role}, p.\text{next}.\text{action}, p.\text{next}.\text{start}, p.\text{next},$ 
9:          $p.\text{next}.\text{target}, p.\text{next}.\text{feat})$ 
10:      else
11:         $n_2 \leftarrow AG.\text{addFinal}(a, p.\text{next}.\text{role}, p.\text{next}.\text{action}, p.\text{next}.\text{start}, p.\text{next},$ 
12:           $p.\text{next}.\text{target}, p.\text{next}.\text{feat})$ 
13:      end if
14:       $AG.\text{addEdge}(n_1, n_2, a, p.\text{role}, p.\text{action}, p.\text{duration})$ 
15:    end for
16:  end for

```

---

### 3.2.3 Abstraction of Actions

Humans cannot reason well with large amounts of data; instead we use generalization to make intelligent judgments. The HMASDA AG may contain a huge number of nodes and edges that are usually difficult to analyze and visualize in a timely and comprehensible manner. Therefore, an abstract action graph is designed from an AG as described in Algorithm 3.3.

The key concept of the defined procedure to address the comprehensibility issue is to reduce the number of nodes while at the same time preserving the strategic behavior patterns. This can be accomplished by the abstraction process, i.e., hierarchical clustering of the graph nodes. This is an iterative process (lines 3–22) that merges together two similar graph nodes in a new node that represents the common role and action concepts of the two original nodes. The algorithm exploits the provided taxonomies and searches for the first common ancestor in the taxonomy hierarchy (lines 6–7). Spatial attributes are computed as averages of the two original nodes (line 10), all basic instances are collected and transferred to the new node (line 12), and the connections in the graph are rewired and removed (lines 14–18). Input and output connections of both nodes are transferred to the new node. The merge process continues until the minimal distance between any two nodes is greater than a threshold  $\tau_{\text{sim}}$  (line 3). We denote the resulting graph as an abstraction action graph (AAG) (line 1). The main difference is that a node in the AAG contains a set of AG nodes and it is expected that an AAG describes agent behavior in a more abstract way than the original AG.

---

**Algorithm 3.3:** Abstract Action Graph construction.
 

---

**Data:** AG - Action Graph,  $O$  - Set of Ontologies

**Result:** AAG - Abstract Action Graph at different abstraction levels.

```

1: AAG  $\leftarrow$  new AAG(AG) //Create an abstract action graph from the current AG
2:
3: while ( $n_1, n_2 \in \text{AAG}$ ): distance( $n_1, n_2$ ) <  $\tau_{\text{sim}}$  do //until there are similar nodes
4:    $z \leftarrow$  create node  $z \in \text{AAG}$ 
5:   //Find the first common ancestor of  $x$  and  $y$  in the corresponding ontologies
6:    $z.\text{action} \leftarrow \text{commonAncestor}(O_A, n_1, n_2)$ 
7:    $z.\text{role} \leftarrow \text{commonAncestor}(O_r, n_1, n_2)$ 
8:    $z.\text{cognitive} \leftarrow \text{commonAncestor}(O_c, n_1, n_2)$ 
9:   //Compute the new position of node  $z$ 
10:   $z.\text{pos} \leftarrow \text{average}(n_1.\text{pos}, n_2.\text{pos})$ 
11:  //Copy all learning instances
12:   $z.\text{instances} \leftarrow n_1.\text{instances} \cup n_2.\text{instances}$ 
13:  //Rewire connections of  $x$  and  $y$  with  $z$ 
14:  move_connections( $n_1, z$ )
15:  move_connections( $n_2, z$ )
16:  //Remove  $x$  and  $y$  nodes
17:  remove( $n_1$ )
18:  remove( $n_2$ )
19: end while

```

---

The HMASDA algorithm is domain-independent to the point that only domain knowledge needs to be provided in order to extract meaningful information. In the current application, the domain knowledge is introduced in the form of ontologies  $O$ . A general representation of an ontology used in HMASDA is presented in Figure 3.1.

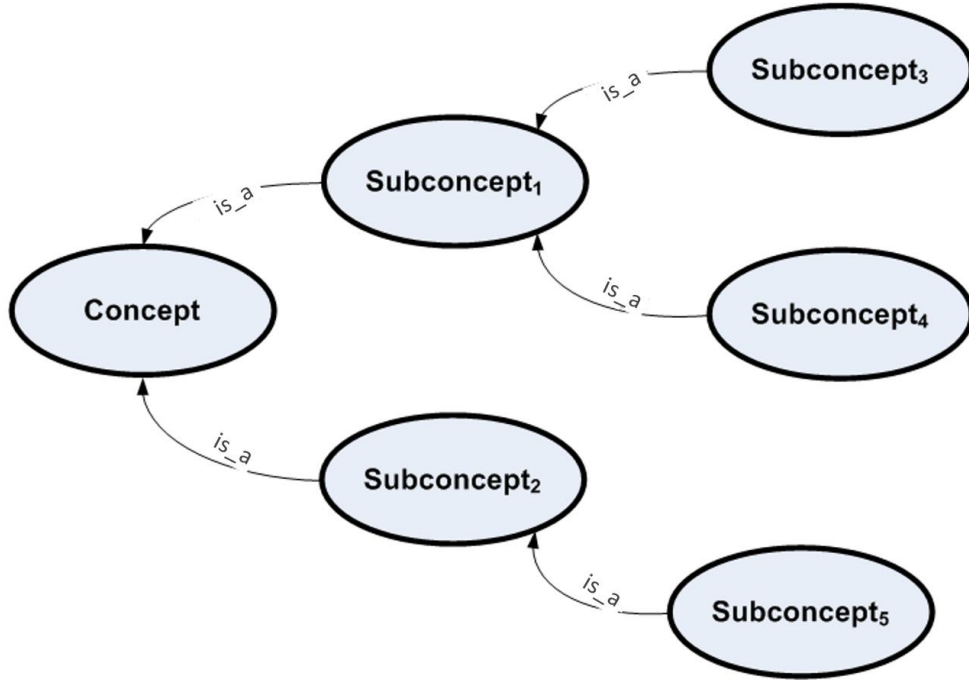


Figure 3.1: A general representation of an ontology used by HMASDA, where concepts are linked using the *is\_a* relation.

Each node in an AG is described by a set of features  $F$  that specify the role of an agent, its cognitive and physical state, actions taken, and so on. The shell of the AG nodes is generalized in such a way that each attribute relates to a corresponding ontology that specifies the hierarchical structure of the attribute values symbolizing concepts in the ontology. The specific ontology instantiation introduced in HMASDA is a taxonomy  $T_x$  that is a hierarchical representation of domain knowledge, where *is\_a* is the relation that binds the concepts of the taxonomy. A concept  $x \in T_x$  is an ancestor of a concept  $y \in T_x$  if it denotes a more general concept than the concept  $y$ . Ontologies supporting other relations that can be used to improve the calculation of similarity between concepts could also be introduced in the algorithm (See Equation 3.1). The relation *is\_similar* could include the information that tells us how similar two concepts are, since currently we use a weight of 1 as a measure of similarity between two linked concepts. Taxonomies are the only domain knowledge provided to the learning algorithm HMASDA.

The leaf nodes of the taxonomy contain specific concepts, whereas nodes higher in the hierarchy denote increasingly more general concepts. The hierarchy color is used for visualization of the AG and the AAG, since modeling the behavior of humans requires a consideration of the cognitive properties during behavior analysis. For example, the civilians perform aggressive actions when they are angry or frustrated; therefore, we introduced a cognitive taxonomy for the estimation of the similarity between nodes, in such a way that only nodes representing similar physical actions and similar cognitive states will be linked during the learning process.

We generalize the similarity function  $s(n_1, n_2)$  between nodes  $n_1$  and  $n_2$ , based on the following features in the nodes and Equation:

$$s(n_1, n_2) = \sum_{f \in F} \omega_f \cdot \text{dist}_f(n_1, n_2), \quad (3.1)$$

where  $F = \{\text{pos}, \text{action}, \text{role}, \text{cognitive}\}$  is a set of node features.

The feature *pos* contains the information regarding the absolute position of the agent in the domain space. The *action* feature describes the action of the agent in that node and can take

any value from the nodes of the taxonomy of agent actions (see Figure 5.2 for the complete domain taxonomy of actions). The basic actions that agents can perform are stored in the leaves, while more abstracted concepts are stored in the internal nodes of the structure. Similarly, the *role* feature of the agent in the observed node can adopt any value from the taxonomy of agent roles (see Figure 5.3). Finally, the *cognitive* feature denotes the cognitive state of the node agent. It is obtained from the taxonomy of cognitive states (see Figure 5.4). For two arbitrary nodes  $n_1$  and  $n_2$ , values for features are collected from the internal node structure and the equation computes the similarity between these two nodes. The calculation is performed in the following way: if the feature is position, the distance between the two feature values is defined as the Euclidean distance between the two nodes in the domain space. Otherwise, it is defined as a sum of distances in a taxonomy ( $d_T(i, j)$ ), normalized by the number of distinct instances with the particular attribute  $i$  in the AG node  $n$  ( $\eta(n, i)$ ), for all distinct feature values  $e$  that are present in an AAG node  $n$  ( $\theta(n, e)$ ); for example, all different agent roles of the basic instances included in an AAG node:

$$\text{dist}_f(n_1, n_2) = \sum_{i \in \theta(n_1, f)} \sum_{j \in \theta(n_2, f)} \frac{d_T(i, j)}{\eta(n_1, i) \cdot \eta(n_2, j)}, \quad (3.2)$$

The distance  $d_T(i, j)$  between two concepts in a taxonomy  $T$  is defined as follows:

$$d_T(i, j) = \frac{\frac{\text{path}(i, j)}{\text{depth}_{T_{\max}}} + (\text{depth}_{T_{\max}} - \text{depth}_T(\text{ancestor}(i, j)))}{\text{depth}_{T_{\max}} + 1}, \quad (3.3)$$

where  $\text{depth}_{T_{\max}}$  is the maximum taxonomy depth. This metric evaluates the similarity of two elements based on the level of their common ancestor. A common ancestor higher in the taxonomy indicates that the elements are less similar, and vice versa. If the common ancestor of two pairs of compared elements has the same level, the distance metric is higher for the pair whose shortest connecting path,  $\text{path}(i, j)$ , in the taxonomy is the longer.

The abstraction process is iterative and continuously merges similar AAG nodes until the minimum similarity between any two nodes in the graph is greater than a predefined threshold  $\tau_{\text{sim}}$ . The appropriate  $\tau_{\text{sim}}$  value depends on the domain, and should be set experimentally. The value does not need to be set exactly, and it is usually given as an interval since we are interested in patterns at different abstraction levels. A lower  $\tau_{\text{sim}}$  value results in more specific patterns, while a higher value results in more general, abstract patterns. The resulting AAG graph contains significantly fewer nodes than the original AG graph.

### 3.2.4 Frequent Pattern Mining

The third part of the algorithm involves extracting sequences of actions relevant to the trainees under observation from the Abstract Action Graph constructed in the previous section. These sequences are behavior patterns demonstrated by individuals during the training sessions, representing their personal traits, their mental and cognitive features prompting the execution of the pattern, and their knowledge. The patterns can be analyzed later offline by experts, and compared with the desired predefined behavior, which enables the identification of weaknesses and strengths in the learned knowledge of the individuals. In a simplified version, the strategic behavior pattern is a tuple  $\Phi$  composed of several trajectory envelopes and symbolic descriptions. A graphical representation is provided in Figure 3.2. This pattern is a subgraph of an AAG and is composed of three trajectory envelopes, each describing the abstracted action, the temporal constraints and the cognitive and domain features. The size of the nodes represents the number of basic action instances that are included in the nodes. More action instances result in a bigger node.

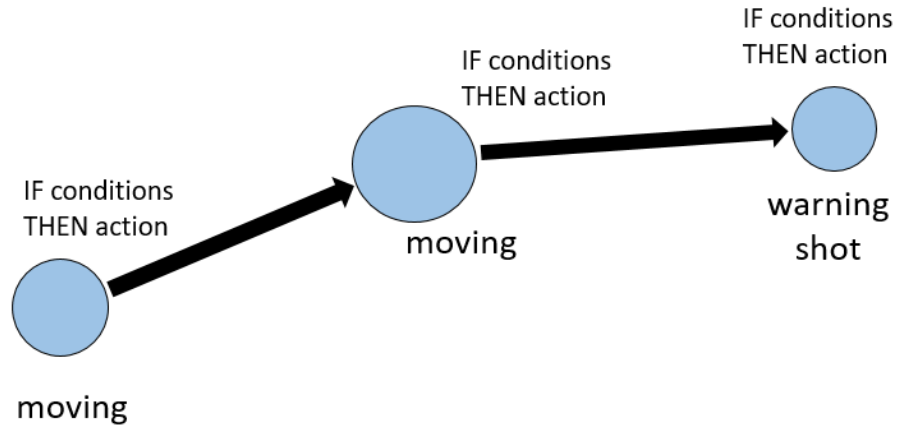


Figure 3.2: A simplified representation of a behavior pattern containing three actions.

The patterns are represented uninterrupted paths in the AAG and in order to identify only the most relevant trajectories, the  $connection_{level}$  measure was defined as the number of basic action sequences of the same length as the pattern from the original AG that are fully included in the pattern. Patterns that have more basic action sequences of the same length and are fully included in the pattern tend to be more strategic than those that eventuate from local decisions. The trajectories with the highest  $connection_{level}$  are macro-actions that represent strategic behavior.

Next, each node contained in the identified pattern is enriched with symbolic descriptions in the form of decision rules. These rules are combined with the trajectories and provide a set of conditions, that include features from the domain and agent's internal states, describing when a specific trajectory occurs in the domain. The values for all features (see Appendix A) that are included in the decision rules are extracted from the observed trajectory envelope in order to create learning datasets, which include both basic domain concepts, abstracted features from ontologies, internal states of the agents, and complex features computed from domain states. In general, features are domain-dependent, and have to be defined for each scenario in order to capture the relations between the concepts of the simulation. The induced rules determine the situation in which the spatial envelope can occur. We use the spatial and the constructed symbolic information during the cloning sessions of the PATDS method to identify which patterns can be executed at any given time.

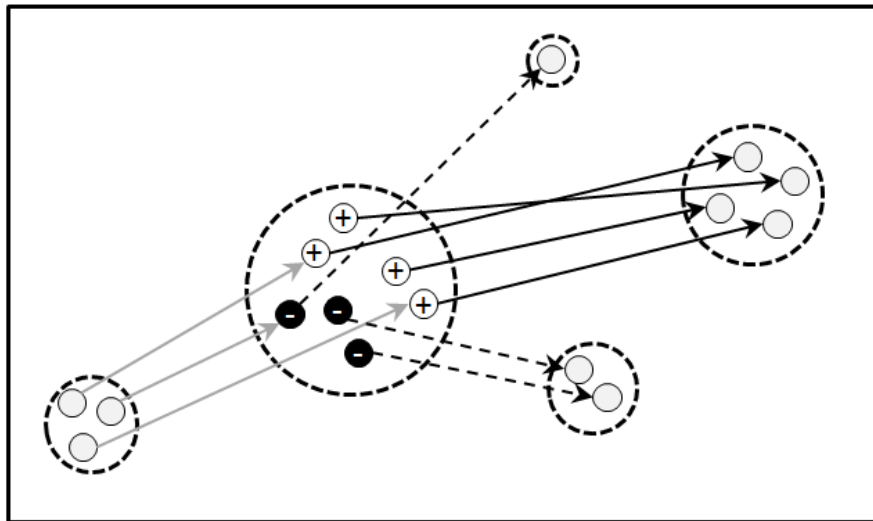


Figure 3.3: The selection mechanism of positive and negative examples from an AAG node, used as learning examples for the rule inducer.

For the rule inducing task, we applied the set-valued rule inducer SLIPPER (Simple Learner with Iterative Pruning to Produce Error Reduction) [85], which generates IF-THEN rules describing the action concepts in a specific pattern. We have changed the selection of positive examples from the original MASDA. Figure 3.3 shows how the examples are labeled in HMASDA in a different way from the previous algorithm. Positive examples (annotated with '+') represent all high-level action instances in a node that corresponds to a connection of interest. Negative examples (annotated with '-'), on the other hand, start at the same node, but continue along any of the connections of the remaining nodes. Negative examples are further refined by selecting only examples describing action concepts that are similar to the action concept of interest, based on the location in the taxonomy of actions. The original selection mechanism in MASDA considered all non-positive nodes in Figure 3.3 as negative examples. This resulted in a much higher number of negative examples than positive, which skewed the class distribution, and reduced the performance of the induction algorithm for the looting domain. Next, for all identified positive and negative examples values for all features are computed from the information contained in the nodes of the original AG. This entire dataset is then used by SLIPPER to generate symbolic descriptions for each trajectory. The entire procedure is presented in Algorithm 3.4.

First, the algorithm discovers relevant sequences of agent actions (i.e., behavior patterns) present in the AAG (line 2). The areas of interest are the AAG sequences that agents have executed more often. The algorithm extracts all the AAG subsequences of length  $len$  containing an arbitrary number of AG subsequences supporting that path. For example, an AAG subsequence with three AAG nodes  $n_1-n_2-n_3$  may have three AG subsequences following the same path. Frequent sequences represent the strategy the agents were executing, in contrast to single actions corresponding to reflex reactions or random events. The original MASDA used sub-graph matching to find meaningful patterns, which has exponential time complexity. This was replaced by graph path-matching with polynomial time complexity [86] (additional information regarding the implemented matching algorithm is provided in Section 5.1.2). Rarely, it happens that complex sub-graph patterns best describe a particular sequence in which case HMASDA constructs several simple paths that correspond to the sub-graph pattern.

The next step is to generate symbolic descriptions for the edges in the most frequent behavior patterns (lines 5-17). Learning examples (line 8) are created by identifying instances in a node that correspond to the observed sequence  $seq$ , as described at the beginning of this section. Learning data is generated (line 12) for each instance in the identified examples by computing values for all features. The last important step is the induction of rules (line 15) using SLIPPER from the constructed dataset. The symbolic descriptions comprise a variety of domain-dependent features obtained from observations (e.g., location, physical, and psychological features) and computed from the AAG nodes, e.g., the relative positions of agents or objects, average anger and fear values for close opponent groups. The entire list of features for both domains is listed in Appendix A. The goal is to create a list of rules describing the contextual precondition and the action that follows.

---

**Algorithm 3.4:** Extraction of frequent strategic patterns.

---

**Data:** AAG - Abstract Action Graph,  $len$  – target length of the pattern**Result:** DM – The decision model composed of strategic patterns of length  $len$ 

```

1: //Discover repetitive strategic concepts in AAG
2: relevantSeq  $\leftarrow$  findRepetitiveSequences(AAG)
3:
4: //generate symbolic descriptions
5: for each seq  $\in$  relevantSeq do
6:   for each e  $\in$  seq.nodes do
7:     //Select positive and negative learning examples
8:     e.examples  $\leftarrow$  selectLearningExamples(e,AAG)
9:     //Generate learning data
10:    for each instance  $\in$  e.examples do
11:      //Extract features from AAG and create learning data
12:      dataset  $\leftarrow$  generateData(instance)
13:    end for
14:    //Induce IF-THEN rules from examples
15:    e.description  $\leftarrow$  induceRules(dataset)
16:  end for
17: end for
18: DM  $\leftarrow$  postProcessPatterns(relevantSeq)

```

---



## Chapter 4

# Performance Analysis of Training Through Data-Farming Simulations

This section provides a description of the PATDS method, used to evaluate the performance of the observed individuals. The three components (contributions) of the method are presented. In Section 4.1, a survey of the related work is presented. Section 4.2 provides a general overview of the method. Section 4.3, 4.4, and 4.5 present the three components of PATDS.

### 4.1 Related Work

Training groups of operators to maintain different tasks is a topic of wide interest. Researchers showed that simulated training for operators in a drinking water treatment plant on regular tasks related to water treatment significantly improves the performance of the operators during trials and lowers the overall required training time to achieve the appropriate level of expertise [87]. Winter et al. [88] analyzed the relationships between the performance during driver training sessions in a simulator and actual driving results. They showed that users that trained in the simulator had a higher chance of passing the driving test the first time and performed better in three key evaluation measurements. Vengust et al. [46] proposed a motion car driving simulator and several concepts for mechanism that would analyze the behavior of the user and adapt the scenarios based on their reactions and skill level.

Zhongkun et al. [89] developed a training system in a virtual environment for training workers in a nuclear fuel processing plant and for optimizing the entire production process. In cultural heritage, serious games and simulations are often used to train and teach various participants.

Mortara et al. [90] presented an overview of serious games intended for educational purposes in cultural heritage applications. They analyzed the relations between games, context of use, solutions, and learning effectiveness. Simulation-based training has become widespread in medicine and health care to improve staff performance.

Hegland et al. [91] presented a review of simulation-based training tools for training nurses, which showed a noticeable advantage over other learning strategies.

Pasquier [92] presented a tool based on serious games for training Combat Life Savers that enables users to create unlimited training scenarios and provides the ability to repeatedly train in a safe and consequence-free environment.

The calibration of systems and simulators has been extensively studied in the literature. ATABS (Automatic Training of Agent-Based Simulations) [93] delivers a framework that enables automatic training of agents. The system learns from examples of real-case situations that describe how certain simulation input combinations reflect in the simulation outcome. ATABS then in turn focuses on the calibration of random number generators for training the ABS. Moreover, it computes the simulation parameters needed to obtain as similar a performance as possible to the real operating domain. Although the goal of this work is similar to ours, it differs in several ways: (i) we learn from real operational people and create complex behavior patterns that are in turn used as the strategy in the ABS, and we do not set existing simulation parameters on a hardcoded fixed-policy strategy; and (ii) the approach in this work is to discover if it is

possible to train the simulation to perform similarly. In addition, we try to evaluate the extracted behavior (strategy) on all possible simulation parameters to identify its weaknesses.

Data farming was originally used in the Albert project [94], where agent-based models were created to capture a specific military challenge and run several times to allow high-performance computers to gain insight into the potential outcomes. In the iterative procedure, the modeler, that is, the decision maker, specified a model in the form of a fixed policy, executed the model on a variety of parameters, and analyzed the results. On that basis, strategies were explored about how to use data farming for the calibration and validation of simulation models [59].

Recent efforts are focused on generative analysis, where the goal is to generate novel solutions and find “interesting” regions in the space of the simulation parameters [95]. For example, researchers used genetic algorithms to discover effective strategies for crowd control that consisted of the deployment of several police barriers that varied in strength and position [96]. In our work, instead of using a fixed policy, we model the behavior of a trainee that is afterward executed on a variety of simulation parameters.

Connolly et al. [97] presented a comprehensive review of papers dealing with learning using computer and serious games. Some of the reviewed papers also deal with expert-trainee relations. However, none of the papers provides an elaborated mechanism for cognitive learning.

A method of knowledge transfer from American-football video observations to control in a game simulator was demonstrated in the literature [66]. It included object tracking, dynamic time warping for action recognition, and an Icarus cognitive architecture to infer strategies in the form of rules. Even though it is similar to the method in this paper, it addresses only the aspects of hardcoded agent behavior without the cognitive aspects that are crucial when dealing with people. Moreover, those researchers evaluate the quality of transfer learning and do not apply data-farming experiments to systematically analyze the performance of the trainees.

Qudrat-Ullah [98] presented a study where the intention was to improve the performance of people in dynamic tasks through transfer learning. Evaluation was performed by incorporating three levels of facilitation (pre-task, in-task, and post-task) and the outcome was simulated in an interactive learning environment. It has been shown that all levels of facilitation helped users to develop transfer learning skills. The methodology in our work is different, since we analyze the behavior, create models and use them in data-farming experiments in order to identify weaknesses in the behavior of trainees and consequently improve their performance.

Campuzano et al. [99] presented a methodology to create computational models of human behavior (CHMB) and they validated them in a hospital-based scenario. They used questionnaires and domain experts to create an ontology of key concepts that was used to induce CHMBs. These models are then used in simulations in order to validate them and check if the captured behaviors correspond to the real ones. The methodology in our work is different, since we tried to automate the whole modeling and training process. We gather the behavior of people from simulations or serious games sessions, automatically create models and use them in data-farming experiments in order to identify weaknesses in the behavior of trainees and consequently improve their performance in several training rounds.

## 4.2 General Overview

The general methodology, presented in this thesis, represents a novel procedure to expedite the training sessions of human trainees – The PATDS training cycle. The schematic overview in Figure 4.1 presents the entire workflow, from training, analysis, evaluation of learned skills, to the redesign of training scenarios.

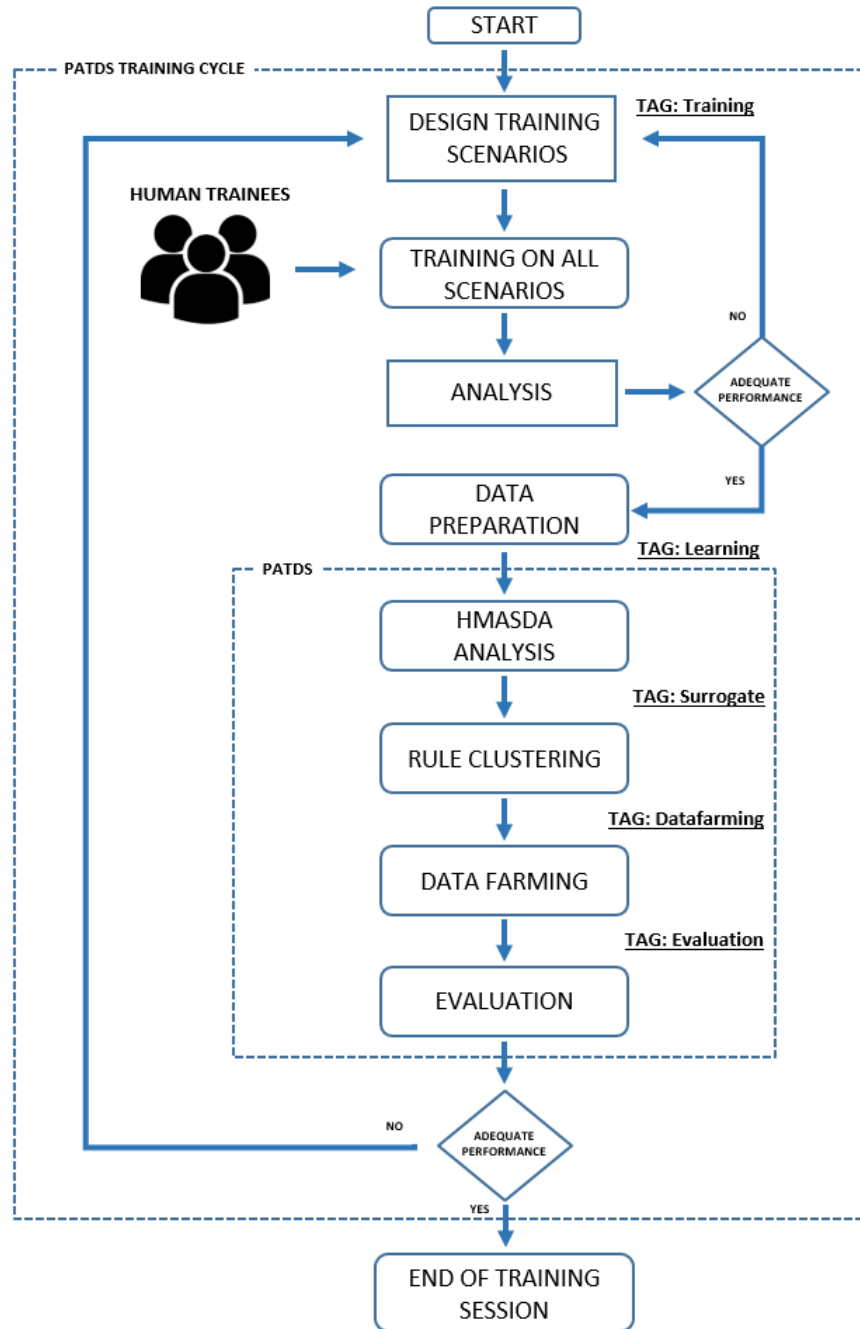


Figure 4.1: The schematic overview of the whole PATDS training cycle, related to Algorithm 5.1, showing the main modules of the procedure.

The first part of the methodology proceeds in a usual manner. Experts define the training scenarios for the trainees, which are replayed several times during training. Next, a simple evaluation is performed based on training monitoring and assessment of key KPI's. This procedure is repeated until satisfactory performance. The second part of the PATDS training cycle is the novel PATDS algorithm that enables a comprehensive automatic evaluation of behavior on a multitude of training scenarios. The algorithm introduces a novel mechanism to improve training sessions both in terms of time and effectiveness of training. The novel contributions of the algorithm are mechanisms to (i) exploit observed behavior patterns of trainees or simulated agents to infer their strategies and reproduce their performance; (ii) identify and select key operational patterns for behavior cloning; (iii) evaluate agent behavior in simulated scenarios during data-farming experiments; and (iv) identify key weaknesses and

overall performance of the exhibited trainee strategy through the data farming of the simulated agents to improve the trainee performance in several learning iterations.

The entire algorithm is summarized in Algorithm 4.1. The TAG lines in the algorithm are linked to the tags in Figure 4.1 to better represent the connection between the overall PATDS scheme and the individual parts of the algorithm and facilitate the comprehension of the overall method. The algorithm is composed of several key parts. First, the training part, represented in the repeat loop (lines 5–12) one or several trainees play a set of simulation plays until satisfied with the performance. Simulation logs are stored for simple performance evaluation (line 6) to detect when their performance is adequate. Second, in the learning part all simulated plays of the trainee are analyzed and behavior patterns are extracted using HMASDA (line 15). Third, in the surrogate part the pattern selection is performed and a surrogate agent of the trainee (lines 18–19) is created. Next, the surrogate agent is tested, in the data-farming part, during data-farming experiments on the entire collection of simulation plays under all reasonable circumstances (lines 22–25). Finally, all plays are evaluated in the evaluation part (lines 28–29) and sorted based on the performance achieved (line 29). If the overall performance is not yet satisfactory, a subset of the worst simulation plays is selected and the corresponding parameter settings are stored to be replayed by trainees in the next training round of the PATDS algorithm (lines 31–35). Otherwise, the performance evaluation of the entire data-farming set of experiments is returned by the algorithm.

**Algorithm 4.1:** Performance Analysis of Training using Data Farming Simulations.

**Data:**  $SD$  - Simulated domain;  $N_{\text{train}}$ ,  $N_{\text{datafarm}}$  - Set of training and data-farming simulation parameters

**Result:**  $d$  - performance (damage) on the domain  $SD$  with  $N_{\text{datafarm}}$  parameters

```

1: start:
2: // TAG: Training
3: //Trainee (one or a group) repeats playing a set of simulation plays until reasonably
4: satisfied with performance
5: repeat
6:    $L' \leftarrow \{\}$  //Set of simulation logs
7:   for each  $p \in N_{\text{train}}$  do
8:     //A simulation play on selected parameter
9:      $L' \leftarrow \text{simulation\_play}(p, T)$ 
10:   end for
11:    $d' \leftarrow \text{damage}(L')$  //damage of all simulation plays in  $L$ 
12: until  $\text{small}(d')$  //trainee satisfied with performance
13:
14: // TAG: Learning
15:  $BP \leftarrow \text{extractBehavior}(L')$ 
16:
17: // TAG: Surrogate
18:  $B \leftarrow \text{selectPatterns}(BP)$ 
19:  $SA \leftarrow \text{createSA}(B)$ 
20:
21: // TAG: Data-farming
22:  $L \leftarrow \{\}$ 
23: for each  $p \in N_{\text{datafarm}}$  do //perform data-farming
24:    $L \leftarrow \text{sim}(p, SA)$  //execute a simulation run with  $SA$ 
25: end for
26:
27: // TAG: Evaluation
28:  $d \leftarrow \text{damage}(L)$  //damage of all simulation plays in  $L'$ 
29:  $\text{sort}(L, d)$  //sort logs based on performance
30:
31: if  $\text{large}(d)$  then //unsatisfactory performance
32:   //store parameter settings of worst simulation plays to be replayed in another
33:   round
34:    $N_{\text{train}} \leftarrow k\_worst(k, N_{\text{datafarm}}, d)$ 
35:   go to start
36: else
37:   return  $d$ 
38: end if

```

PATDS is composed of four main parts that are linked in order to provide a mechanism for continuous training sessions. The first part, analysis of simulated or training data that results in behavior patterns characteristic of the observed group of trainees was already presented in the previous chapter. In the next subsections we present the remaining three parts of the algorithm. First, we present a mechanism that removes the redundancy from all discovered patterns and selects the most appropriate subset to be used in training evaluation. Next, we present the

behavior cloning module and provide technical details on how the module is included in the whole learning cycle. The last section deals with data-farming experiments and the problems of how to compare similarities in behavior for different simulation runs and how to evaluate the overall performance in a scenario. Besides the main concepts behind the methods, definitions and pseudocode are also provided to further explain each key module of PATDS.

### 4.3 Pattern Clustering and Selection

The pattern selection mechanism in PATDS identifies and selects a minimum viable subset of behavior patterns from the entire set created with HMASDA in order to remove redundant and less-informative rules while maintaining a set large enough to reasonably reproduce the observed behavior. The assumption is that a smaller subset of meaningful behavior patterns will improve the behavior cloning process by covering as much of the surrogate agent action space as possible and at the same time will eliminate meaningless noise that corrupts the strategy behind the performance. In our tests, at the end of the behavior analysis phase a large set of patterns was typically obtained, on the order of a few hundred. Moreover, many of the generated patterns were redundant, describing similar patterns with slightly different rules. To avoid this problem, we applied rule clustering to identify centroid patterns and enabled the removal of redundant or too similar patterns.

The key concepts of the rule clustering method are borrowed from text mining and information retrieval. The goal in these domains is to cluster similar textual documents based on their topic. Each document is represented as vectors of identifiers; for example, index terms. In PATDS, each pattern is represented as a vector  $p_i$ :

$$p_i = (w_{1,i}, w_{2,i}, \dots, w_{j,i}), \quad (4.1)$$

where each dimension of the vector ( $w_{j,i}$ ) corresponds to a separate attribute weight. Term frequency-inverse document frequency (TF-IDF) is the most common weighting method used to describe documents, and the same principles were applied to describe rule attributes in the patterns.

TF-IDF( $a, p$ ) measures the importance or weight of the attribute  $a$  within a particular pattern  $p$  and is defined as

$$\text{TF-IDF}(a, p) = \text{TF}(a, p) \times \text{IDF}(a), \quad (4.2)$$

where term frequency  $\text{TF}(a, p)$  is defined as the occurrence count of an attribute in a pattern, and inverse document frequency  $\text{IDF}(a)$  is a measure of the general importance of the attribute (obtained by dividing the total number of patterns by the number of patterns containing the attribute, and then taking the logarithm of the quotient).

$$\text{IDF}(a) = \log \frac{|P|}{1 + |\{p: a \in P\}|}, \quad (4.3)$$

where  $|BP|$  is the total number of patterns in the pattern set and  $|\{p: a \in BP\}|$  is the number of patterns where attribute  $a$  appears.

The TF-IDF weights associated with each pattern are used to cluster the patterns based on the action space that they cover. The clustering algorithm [100] partitions  $N$  objects (patterns) into  $K$  disjoint subsets  $S_j$ , containing  $N_j$  data points that minimize the sum-of-squares criterion (see Equation 5.1):

$$J = \sum_{j=1}^K \sum_{n \in S_j} |\chi_n - \mu_j|^2, \quad (4.4)$$

where  $\chi_n$  is a vector representing the  $n$ -th data point,  $\mu_j$  is the geometric centroid of the data points in  $S_j$ , and  $|\chi_n - \mu_n|^2$  is a chosen distance measure between the two data points. By iteratively updating the mean vector the algorithm minimizes the criterion  $J$ .

The general clustering algorithm is composed of the following steps:

- Define the initial group centroids by randomly selecting  $K$  points and place them into the space represented by the objects that are being clustered.
- Assign each element to the closest centroid.
- Assign all elements and recalculate the new positions of the  $N$  centroids so that the distance between all elements in the group is minimized.
- Repeat the previous two steps until the recalculated positions of the centroids do not change significantly.

We identified the appropriate  $K$  using two methods: (i) the value that maximizes the similarity of objects inside all clusters (see Section 5.3); (ii) the value where behavior cloning performs best (see Section 5.4).

To effectively assign instances to clusters, the clustering algorithm uses a measure that calculates the similarity between two vectors (patterns). Cosine similarity is often used in combination with TF-IDF weighting since it is suitable for metric learning: the resulting similarity measure is always within the range from -1 to +1 (-1 meaning exactly opposite, +1 meaning exactly the same, with 0 usually indicating independence, and in-between values indicating intermediate similarity or dissimilarity). Moreover, it has been observed that in high-dimensional data, the cosine similarity performs better than other distance metrics (for example, Euclidian distance).

Given two vectors of attributes,  $x$  and  $y$ , the cosine similarity is represented using a dot product and magnitude as

$$\cosSim(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| \cdot |\vec{y}|} = \frac{\sum_{i=1}^n x_i \times y_i}{\sqrt{\sum_{i=1}^n x_i^2} \times \sqrt{\sum_{i=1}^n y_i^2}} \quad (4.5)$$

where  $\vec{x} \cdot \vec{y}$  is the dot product of two vectors and  $|\vec{x}|$  is the vector magnitude.

The entire pattern clustering mechanism is described in Algorithm 4.2. The input to the algorithm is the entire set of behavior patterns discovered using HMASDA, and the output is the strategy  $S$  that will be used for efficient behavior cloning. The first step of the algorithm is to collect all IF-THEN rules in the patterns and store them into rulesets (line 2). The main data structure is the vector of identifiers  $W$  where TD-IDF weights are stored (line 3). Next, for each ruleset in  $BP$  (lines 5–14) a vector of identifiers is created, where each element is an attribute from the ruleset (line 6). The next part is important, since it involves the transformation of rule data. Categorical attributes are left unchanged, while numerical attributes are discretized and transformed into categorical attributes. Numerical attributes are in the form of

*attribute\_name con N,*

where  $con \in \{>, <, \leq, \geq, =\}$  and  $N \in [0, 100]$ . For discretization, the interval  $[0, 100]$  is divided into five subintervals, and each numerical attribute is assigned to a specific subinterval based on the attribute value. The unique label of the assigned interval is used for the new attribute name. This way we retain the information about the attribute type and the value it represents. Next, TF-IDF weights are created for all identifiers in the  $w_r$  vector and stored in the vector identifiers set. Once the rule preprocessing part is performed, a clustering algorithm is used to distribute rulesets into unique clusters and return the computed centroids (line 17). The last step of the algorithm is to reassign rulesets to the corresponding patterns (line 18), which is the strategy set used for the next phase of PATDS.

---

**Algorithm 4.2:** Pattern selection mechanism through clustering.
 

---

**Data:**  $BP$  – Set of behavior patterns  
**Result:**  $ST$  – Strategy

```

1: //Collect symbolic descriptions from patterns
2:  $BP_S \leftarrow \text{collect\_rules}(BP)$ 
3:  $W \leftarrow \{\}$  //Set of vector identifiers
4:
5: for each  $r \in BP_S$  do //For each term vector
6:    $w_r \leftarrow \text{create\_vector}(r)$  //Create vector of identifiers
7:   //Discretize numerical attributes to categorical
8:   for each  $term \in w_r$  do
9:     if  $\text{type}(term)$  is numerical then
10:      discretize_term( $term$ )
11:     end if
12:   end for
13:    $W \leftarrow W \cup \text{tf-idf}(w_r)$ 
14: end for
15:
16: //Perform clustering on the entire set of TF-IDF vector weights
17:  $SBP_{\min} \leftarrow \text{clustering}(W)$ 
18:  $ST \leftarrow \text{reassign\_patterns}(SBP_{\min})$ 

```

---

## 4.4 Behavior Cloning

The clustered patterns extracted with HMASDA are on the one hand used to get insights about the strategies of the trainees or simulated agents and on the other hand used for behavior cloning; that is, creating a surrogate agent capable of reproducing the behavior of the observed agent. The whole design was inspired by the human cognitive process that is based on memories and a set of assertions [101]. The scheme of the cloning procedure is presented in Figure 4.2. At each simulation step, the MASON simulator sends an action request to the behavior cloning module, and the module provides a non-action response or the most appropriate action for execution. The decision is based on the context provided in the request and the past events, which can be regarded as memories of the surrogate agent. They are stored in simulation logs and continuously delivered to the behavior cloning module. When a new observation is created in the simulation, it is sent to the module for interpretation. The set of assertions of the surrogate agent is stored in the behavior patterns library.

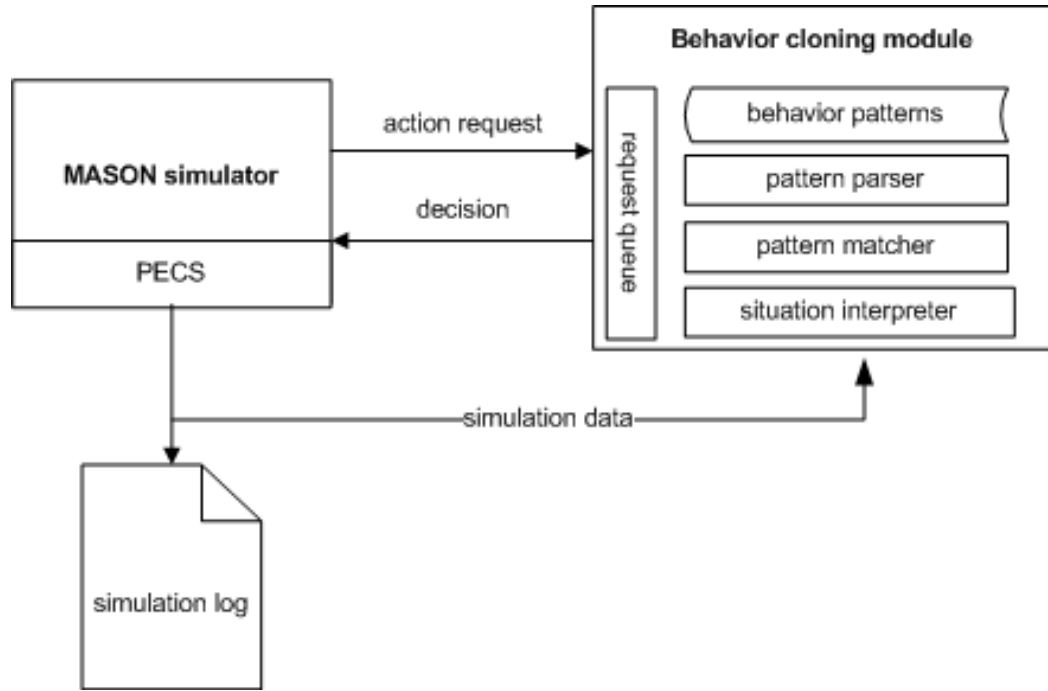


Figure 4.2: A simplified scheme of the behavior cloning procedure listing the main modules (related to Algorithm 4.3).

The behavior cloning module consists of the pattern parser, the pattern matcher, and the situation interpreter. The pattern parser translates the graphical and symbolic part of the patterns into an executable form. Agent actions, roles, and cognitive states are linked to the corresponding node in the taxonomies  $T_A$ ,  $T_r$ , and  $T_c$  to enable better matching through generalization. The IF-THEN rules are interpreted and stored into an attribute-value vector. The situation interpreter parses the simulation log and creates a new instantiation of the domain state  $S'$  (set of finite domain states). The pattern matcher implements a decision function  $\delta$  (see Equation 5.6) that identifies the next action  $M_A$  of agent A based on its current state  $I_A$ , the valid domain state  $S'$ , and the policy  $\Delta$ , which is the set of the agent's asserted knowledge and fetches patterns from the behavior pattern library.

$$\delta: S' \times I_A \times \Delta \longrightarrow M_A, \quad (4.6)$$

The decision function  $\delta$  matches both the graphical and symbolic form of the behavior patterns with the current situational context. The context is established from the action graph (AG) that was generated from the current simulation, transforming low-level activities into several layers of more and more abstract behavior.

The decision function implements a fast matching algorithm, presented in Algorithm 4.3. The task of the algorithm is to select, among the entire set of behavior patterns, the most appropriate pattern based on the current domain state and the internal state of the agent. The domain state is obtained from the current agent traces, based to the simulation step, that are transformed in a partial Action Graph. The behavior patterns of an agent can be matched directly to the constructed partial Action Graph, since they are extracted from AAG, an abstraction of AG. Matching is performed in two steps: (i) first, the graphical part (subgraph matching is performed); (ii) second, the induced rules for each edge in the pattern are used to verify that the pattern can be executed based on the domain and agent's internal state. The graphical part of the patterns is matched against the partial Action Graph (lines 5–34) by considering three situations:

- No soldier nodes are present in the AG (lines 5–12).

- Perform additional matching for the patterns that were already selected for execution in the previous step and are stored in the priority queue (lines 14–23).
- New patterns that has not been selected for execution yet are discovered (lines 25–35).

---

**Algorithm 4.3:** Pattern selection algorithm during cloning in PATDS.
 

---

**Data:**  $ST$  – Strategy (set of behavior patterns  $BP$ );  $P$  – Agent traces (full or partial)

**Result:**  $M_A$  – Agent A action

---

```

1:  $AL \leftarrow \text{parse}(P)$ 
2:  $AG \leftarrow \text{createAG}(AL)$  //Create the Action graph
3:  $E_{BP} \leftarrow \text{priority\_queue}()$  //Empty priority queue based on BP length
4:
5: if no_soldier_nodes( $AG$ ) then //Without soldier actions yet
6:   for each  $BP \in ST$  do //For each pattern in the strategy
7:     //Store the first node of the pattern and a null as the reference to the node in the
8:      $AG$ .
9:      $E_{BP} \leftarrow \text{create\_pattern}(BP.\text{first}, \text{null})$ 
10:    go to sd //Algorithm 5.4
11:   end for
12: end if
13:
14: for each  $e \in E_{BP}$  do //for each executable pattern stored in the queue
15:    $N \leftarrow e.N.\text{next}$  //get next node in AG of the partially matched BP
16:   if  $N.\text{action} = e.\text{action}$  then //Check if same action
17:     // add the newly matched action to the pattern.
18:      $E_{BP}.\text{update\_pattern}(e, N)$ 
19:   else
20:     //remove pattern from queue since cannot continue with the execution
21:      $E_{BP}.\text{remove\_pattern}(e)$ 
22:   end if
23: end for
24:
25: for each  $N \in AG$  do //traverse every node in AG
26:   if soldier_node( $N$ ) and is_terminal( $N$ ) then //if soldier node
27:     for each  $BP \in ST$  do //for each pattern in the strategy
28:       //Check if the current node action is the same as the first pattern action
29:       if  $N.\text{action} = BP.\text{first}.\text{action}$  then
30:         //Store the next action of the selected pattern and the reference to the
31:         //matched node in AG.
32:          $E_{BP} \leftarrow \text{create\_pattern}(BP.\text{next}, N)$ 
33:       end if
34:     end for
35:   end if

```

---

In the first case, there are no soldier nodes already present in the AG. This occurs at the beginning of the simulation, when only simulated agents are active and providing traces. In this case, the mechanism takes the first action from every behavior pattern in the library, creates a

new executable pattern (line 9), and stores the patterns in the priority queue. The selected patterns are stored in the priority queue based on the length of the entire pattern. This priority queue is maintained throughout the entire matching process, since the algorithm favors longer patterns. The mechanism then moves to the second phase of the matching mechanism, where the current domain settings are matched against the IF-THEN rules of the pattern in the queue to confirm possible candidates for execution.

Next, the algorithm tries to continue with the execution of patterns that were already selected, thus the priority queue already contains a set of patterns that have been deemed adequate for execution. For all active patterns in the priority queue an additional step of matching is performed based on their current node matches in the AG (lines 14–23). The next action node from the pattern (the next step of execution) is applied to the successor of the last match in the AG (line 15–19). If the transition from the pattern node to the next AG node is not possible, which means that execution conditions have changed and the pattern has become invalid, then the pattern is removed from the priority queue (lines 19–22).

Next, the entire graph needs to be traversed (lines 25–35) and all soldier nodes (line 26) need to be matched against the first node of all behavior patterns (lines 27–33). The algorithm thus identifies the pattern candidates for the next action and stores them in the priority queue with the reference to the node in the AG where the match occurred (line 31). Since matching is performed only on the first node of the pattern, the result is a set of actions that might evolve in a longer strategic behavior in the next iterations.

The constructed priority queue (based in graphical matching) and the situational AG from Algorithm 4.3 is the input for the next phase (see Algorithm 4.4). The decision function of the surrogate agent tries to identify which patterns from the priority queue are valid by evaluating the symbolic part of the pattern on the situational AG (domain state). To that end, instances containing values for attributes included in the symbolic descriptions are computed for nodes of the provided AG that were linked to the patterns during the previous phase (line 3). Next, for each pattern node (lines 5–13), all feature values that describe the pattern are obtained and used to validate the pattern. This is achieved by classifying (line 9) the previously created instance using IF-THEN rules of the pattern's symbolic description (line 7). If the classification is successful, then we deem the pattern as executable. In contrast, if the classification fails, the pattern is removed from the priority queue (line 11).

The first and longest pattern in the queue is selected for execution (line 14). If several patterns have the same length, the decision function  $\delta$  chooses the most frequent action from the selected patterns as the response.

**Algorithm 4.4:** Pattern matching algorithm during cloning in PATDS.**Data:**  $E_{BP}$  – Priority queue of matched BPs;  $AG$  – Situational Action graph**Result:**  $M_A$  – Agent A action

---

```

1: sd:
2: //Create features and situational descriptions
3:  $SD \leftarrow \text{symbolic\_desc}(AG)$ 
4:
5: for each  $e \in E_{BP}$  do //for each pattern in the priority queue
6:   //Obtain attribute-value vector for all nodes in the pattern
7:    $PD \leftarrow \text{parse}(e.\text{description})$ 
8:   //Match current vector to situational description from the AG
9:   if not  $\text{is\_match}(PD, SD)$  then
10:    //If there is no match, remove the pattern
11:     $E_{BP}.\text{remove\_pattern}(e)$ 
12:   end if
13: end for
14: return  $\text{select\_action}(E_{BP})$ 

```

---

## 4.5 Data Farming and Evaluation

Evaluation is performed for three parts of the PATDS algorithm: the cloning module, pattern selection, and data farming. The evaluation of the behavior cloning module needs to show how well the behavior patterns discovered with HMASDA describe the observed behavior of real operational people or simulated agents. The evaluation is performed through the comparison of the Key Simulation Parameters (KSPs) that are logged during simulation runs and describe how key parameters changed throughout one single run. Figure 4.3 shows a plot of the KSPs for a single MASON simulation run of the RD. Different colors represent time series for a subset of KSPs listed above the graph. In this specific case, the simulation started to escalate at the beginning and global anger and arousal started to increase; however, after 20s (simulation is speeded up) of the simulation, due to the strict response of the security personnel, the fear level started to rise and at around 30s took over as the main motive. This resulted in a sharp decrease of the global arousal, the adoption of the panic flight behavior and a gradual decrease of the global energy. The escalation measure at the end of the simulation reached a steady level. This is just an example of a possible outcome. Some scenarios might resolve in a full scale conflict, where *Global\_anger*, *Global\_escalation*, *Global\_injured*, etc. rise considerably. Since two significantly different simulations can resolve in a similar manner, we wanted to compute the similarity for the entire simulation run by comparing two entire sets of KSPs. The comparison is performed with Dynamic Time Warping (DTW) [102], a state-of-the-art method for comparing time series. In general, DTW is a method that allows a computer to find an optimal match between two given sequences that may vary in time or speed. DTW returns a real value from the interval  $[0,1]$ , where 1 is obtained when comparing two identical sequences and 0 when comparing two completely different sequences. Since several KSPs are obtained during one simulation run, we performed the DTW comparison for all corresponding time series and computed a weighted average of their similarity. The weights for each KSP were determined by domain experts according to the influence they provide to the final outcome of the simulation. *Global\_Anger* time-series obtained from one simulation run is compared to the *Global\_Anger* time-series from a simulation run with different scenario settings. This enables the algorithm to: (i) capture the similarity of the compared KSP throughout the whole simulation run and not just

compare the final value at the end; (ii) grasp a comprehensive evaluation of all important KPSs at the same time and not individually. For example, in one simulation run, *Global\_Need* rises and *Global\_Fear* declines throughout the simulation, while in a different simulation both *Global\_Need* and *Global\_Fear* are rising. Only the appropriate concurrent comparison of both KSPs yields the correct similarity of both simulation runs.

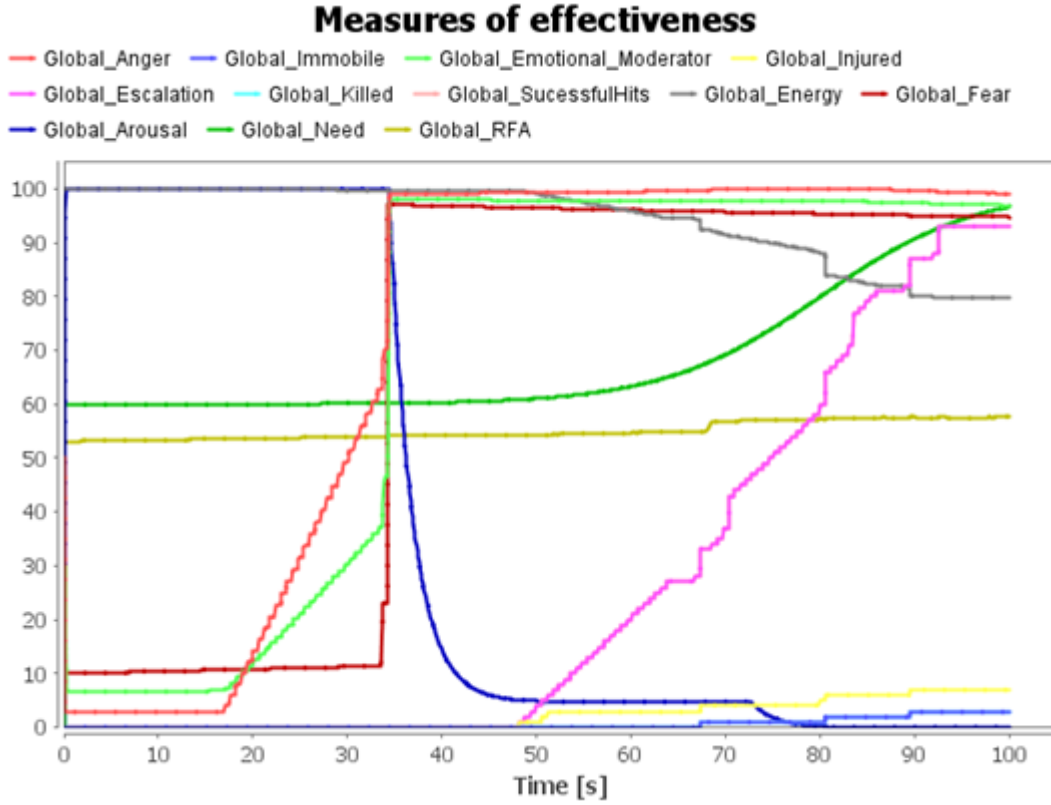


Figure 4.3: An example plot of all Key Simulation Parameters for one simulation run that represent how the simulation evolved in time.

Next, we wanted to evaluate our pattern selection mechanism and identify the optimal number of clusters (patterns) where behavior cloning performs best. The former evaluation employs silhouette plots [103] that estimate the separation distance between the resulting clusters and can identify which objects lie well within the cluster and which ones are merely somewhere in between clusters. This measure ranges from +1, indicating points that are very distant from neighboring clusters, through 0, indicating points that are not distinctly in one cluster or another, to -1, indicating points that are probably assigned to the wrong cluster. A higher average silhouette value denotes a better structure of the clusters. Moreover, it identifies the set of unique patterns that should perform better during behavior cloning.

The final pattern evaluation is again performed through the DTW comparison. The set of patterns that yields the highest similarity when compared to the original plays denotes the appropriate number of clusters.

The overall behavior evaluation was performed on the data-farming experiments after the appropriate set of behavior patterns was determined. Each simulated scenario was evaluated using a damage function designed by experts (see Equation 5.7):

$$damage(L) = \omega_d \cdot num_d + \omega_i \cdot num_i + \omega_a \cdot level_a + \omega_f \cdot level_f, \quad (4.7)$$

where  $num_d$  is the number of people who died,  $num_i$  is the number of injured, and  $level_a$  is the anger level and  $level_f$  is the fear level at the end of the simulation. The weights  $\omega_d$ ,  $\omega_i$ ,  $\omega_a$ , and  $\omega_f$  were also defined by security experts in accordance with the rules of engagement. The

damage function is general, and can be applied to different domains, and can follow different rules of conduct enforced by the domain.

## Chapter 5

# Experimental Setup and Evaluation

### 5.1 Experimental Setup

The goal of the experimental evaluation was to discover if a computer algorithm is able to grasp and meaningfully present the strategic behavior of real human experts solely from observations of their physical actions and the provided cognitive features of the civilian opponents. Moreover, we wanted to show that we are able to set up the PATDS training cycle and verify the learned behavior of trainees in subsequent learning cycles during data-farming simulations. The task was made more complicated by introducing two distinct overall strategies for each domain. In this way, some low-level patterns, for example, how to deal with a single rioter, were the same in both strategies, while there were two different overall strategies to investigate.

The evaluation presented here was performed in experiments with human operational personnel and analysis of simulated play, which is represented as a predefined fixed-policy behavior. This behavior is not optimal, since was defined by project partners and does not follow the standard rules of engagement. It was defined to test the ability of the analysis algorithm to discover different strategic behaviors. Training with human operational people; however, was performed with a group of experts, consisting of one squad leader providing strategic advice and instructions (commands) to three squad members, who used the VBS2 simulation interface. They were instructed to play the RD and LD scenarios ten times and devise strategies they deemed the most effective to cope with the two scenarios. Next, they replayed each scenario ten times using the same strategy in each scenario. We collected the simulation logs from all simulation runs and used them as the principal input for the entire evaluation cycle.

The scheme of the experimental setup for the behavior analysis part is presented in Figure 5.1. It encompasses the top activities from the schematic overview in Figure 4.1 and the main goal is to generate credible behavior patterns that are needed in the PATDS method. VBS2 provides a first-person 3D interface environment for the scenario simulations. The interaction is as usual in computer games; for example, if a stone is thrown by the opposing agent, it can damage the simulated trainee and, by accident, another opposing agent. All these actions are computed by the MASON simulator and displayed by VBS2. VBS2 is connected through the CORBA interface to the MASON simulator, which computes the next action in real-time where the scenario parameters are computed in real-time based on the inputs provided by the trainees through VBS2 and the response obtained from the PECS model for human behavior. VBS2 is a commercial product that cannot be modified. The key intelligent modules for the serious games execution are in the MASON simulator and the PACS model. Together they enable the interaction between human trainees and opposing cognitive agents. The output from MASON is twofold: (i) it computes the control commands for avatars and the environment in VBS2; and (ii) it generates simulation logs where all relevant parameters regarding the conduct of the simulation are stored.

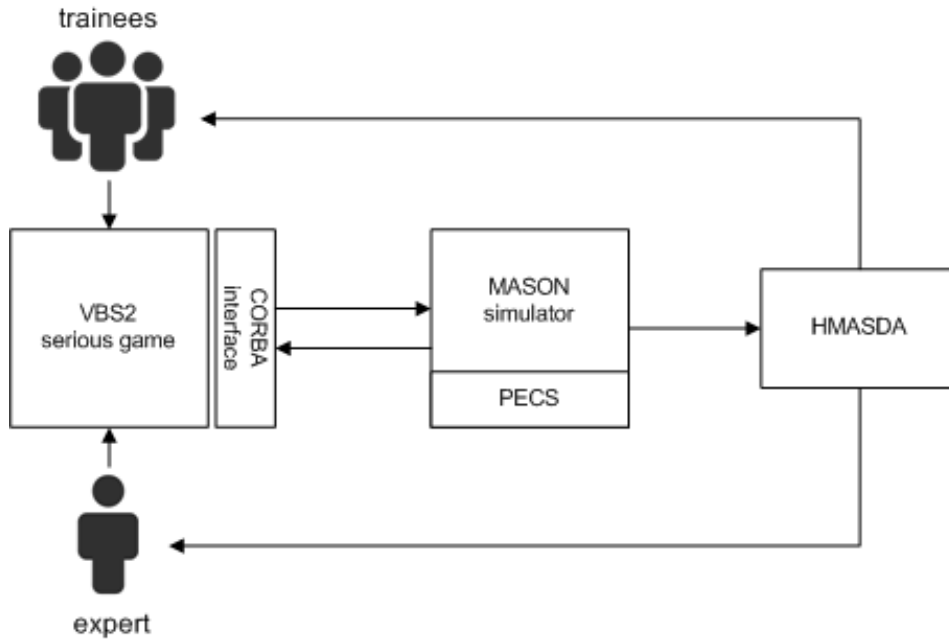


Figure 5.1: The scheme of the experimental setup for the behavior analysis method.

The other type of experiments used simulated agents for soldiers, with a simple fixed-policy strategy. That is hardcoded reactive behavior that selects actions based on the interactions and the current domain state at specific simulation steps. This enabled us to: (i) verify and comprehensively test all components of PATDS; (ii) enable automated repetition, since human experts most of the time were not at our disposal; (iii) reproduce different strategies to be analyzed by the behavior analysis algorithm. After the training or simulated session, the obtained logs are used in HMASDA to generate behavior patterns and thus discover strategies used by the participants in the simulation. These patterns are presented to the trainees and the expert who identifies the trainee's mistakes and instructs them how to perform correctly in the next training session.

To verify whether HMASDA can discover different strategic behavior and behavior patterns, we defined two experiments. First, the trainees played both domains several times, following the standard rules of engagement. Next, the naïve fixed-policy strategy was used in simulated play to generate significantly different behaviors. The motivation for this experiment is to verify whether HMASDA can differentiate between various strategic behaviors adopted in the simulation and thus generate significantly different behavior patterns. In the first experiment trainees tried to resolve the situation according to the standard rules of engagement, while the second experiment involved moving through the domain without active interference and only to reacting when directly provoked. For each scenario, ten simulations were performed by the trainees and simulated agents and the generated logs were used by HMASDA, with the aim of discovering mid-level and strategic behavior patterns. The expert role in the experiments is to evaluate the plays of the trainees from the behavior patterns created with HMASDA and provide guidelines to the trainees. The discovered patterns were used to create surrogate agents and tested in the same simulated environment setting to obtain a quantitative evaluation of how well the patterns describe the observed trainees or simulated agents. The experiments were executed on an Intel Xeon based system, with 16GB of RAM. The algorithm implementation had been optimized to support larger action graphs. Moreover, the computational complexity had been significantly decreased and the algorithm is able to extract behavior patterns in a reasonable amount of time (within an hour).

Finally, the overall PATDS learning cycle was initiated to verify if it enables to identify key weaknesses in the learned behavior of the trainees and facilitates the learning process, learning time, and the overall performance quality. The created surrogate agents were verified in data-farming simulations under all possible circumstances to identify the scenario settings where there is a high probability that the human trainees would perform poorly. These scenarios would be replayed in the subsequent training sessions and another learning cycle is introduced.

In the next subsections, the evaluations of all key parts of the PATDS cycle are presented. First, the experimental setup for HMASDA and the created patterns are provided. Second, clustering of behavior patterns is presented. Next, we describe the results for behavior cloning and how well surrogate agents reproduce the original behavior. Finally, we show the overall benefit of the PATDS cycle.

### 5.1.1 Ontologies

Ontologies, as defined in Chapter 4, are the only domain knowledge provided to the learning algorithm HMASDA. They describe the relations between key concepts in the observed domain. The factual instantiation of the ontologies in HMASDA are taxonomies that enable efficient abstraction of traces. To that end, we defined three taxonomies in collaboration with security and psychology experts that were part of the EUSAS project: actions, roles and cognitive states. The end nodes of all three taxonomies contain basic concepts or states from the domain that agents can adopt, while elements closer to the root node are more generic and abstracted, which facilitates hierarchical clustering during the abstraction process. Figure 5.2 presents the taxonomy of agent actions that we defined for the specific domain. For easier interpretation, the colors denoting the concepts in the taxonomy correspond to the action nodes in the Action Graph.



Figure 5.2: The taxonomy of agent actions, representing the relations between all actions that are available in the domain.

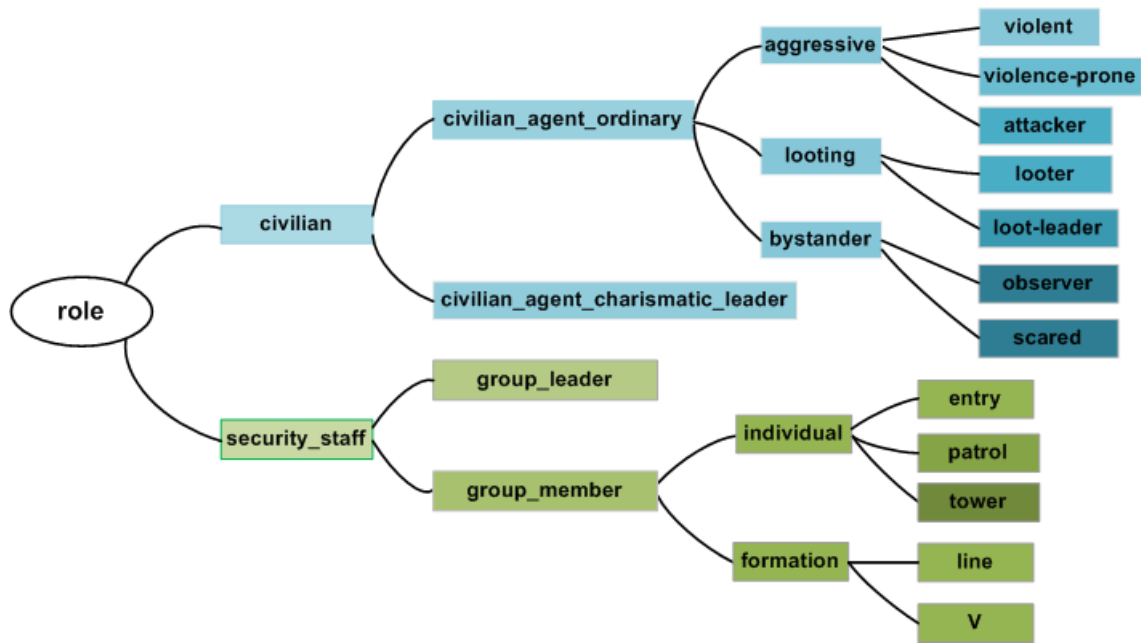


Figure 5.3: The taxonomy of agent roles.

The cognitive performance is captured through cognitive and mental feature values, which are part of the behavior rules. There are 15 attributes implemented in MASON simulations, as presented in Figure 5.4, with the corresponding value sets. Some can be deemed as mental states of the observed entity and some as cognitive states that express intentions and are determined by the PACS model, which resemble cognitive reasoning. The values of the cognitive and mental attributes in a rule that describe the patterns are being determined by HMASDA during the learning process. The experts observe the rules and the attributes and infer how well the trainees capture the cognitive and mental state of the opposing agents, and whether the level of the trainee's aggression was appropriate; that is, not too high and not too low.

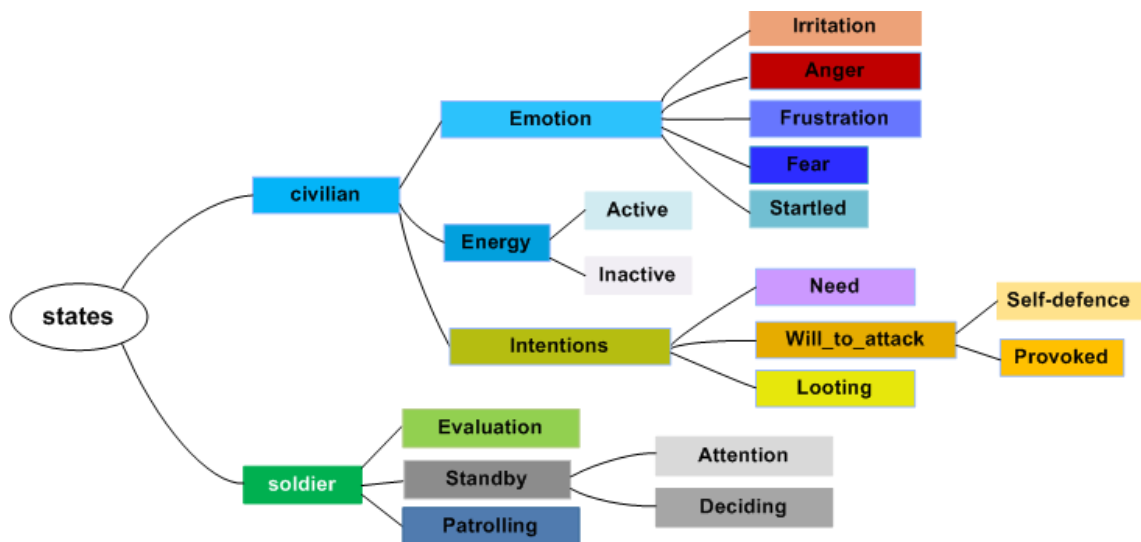


Figure 5.4: The taxonomy of cognitive and mental states of the agents.

### 5.1.2 Behavior Graphs

In the next step of the algorithm, the AG and AAG are constructed using the three defined taxonomies and simulation logs. They represent all captured movements, and the abstracted concepts, for the different strategic behavior that can be observed during the simulation. The constructed AG contained approximately 150,000 nodes. In HMASDA, advanced algorithms for improved graph traversal and subgraph matching methods made it possible to process the domain, thus speeding up the process while losing the theoretical assumptions that all patterns would be examined. The process optimization methods were introduced in two key parts of the behavior analysis algorithm and during pattern selection in PATDS. First, during the abstraction process in order to optimize the search for similar nodes, which originally compared every node to every other. In HMASDA, graph search is performed locally, based on heuristics that consider the distance between nodes and the similarity between concepts from the taxonomy. The second optimization was introduced for the identification and extractions of relevant sequences. We limited the search to linear sequences, without multiple interactions. The only counterexamples we could design by hand were artificial cases lacking realism, and with several interactions at the same time (for example, warn an opponent and shoot him at the same time). This assumption of linearity also simplifies the pattern selection algorithm, since it is possible to iteratively match each node of the pattern to the partially constructed AG.

An important step in behavior analysis is to construct behavior graphs that are then used to identify strategic behavior of the observed group of agents. Figure 5.5 shows the constructed AG for the standard peacekeeping approach, where rules of engagement are followed to resolve the conflict. The shapes of the nodes represent the two main agent roles: squares for security experts and circles for civilians. Security experts approach the building in formation, secure it, and patrol the perimeter of the building. Depending on the timing and actions adopted, the majority of the civilians may get scared and flee the area during this maneuver. Some of them may start adopting aggressive actions towards the soldiers. Security experts isolate and apprehend the remaining aggressive individuals. The colors are translated from the taxonomy of cognitive states to make the graph easier to understand, and to identify groups of locations where the appearance of certain states is more frequent. Gray squares represent only the position of peacekeepers, since their behavior is provided by human trainees; therefore, there is not a leading motive guiding their behavior. Yellow circles represent the looting motive, red the aggressive motive, and white nodes represent inactive (injured) civilian agents. Additional information regarding the looting scenario can be found in Section 2.2.2.



Figure 5.5: HMASDA Action Graph of human play in the looting domain.

Figure 5.6 shows the Abstract Action Graph at abstraction level 4. The increased abstraction highlights the most important and repetitive strategic paths. At a higher abstraction level it is possible to more easily observe the strategy that the soldier agents adopted and how they move in the scenario. They start moving towards the shop entrance, where most of the interaction between civilians occurs.

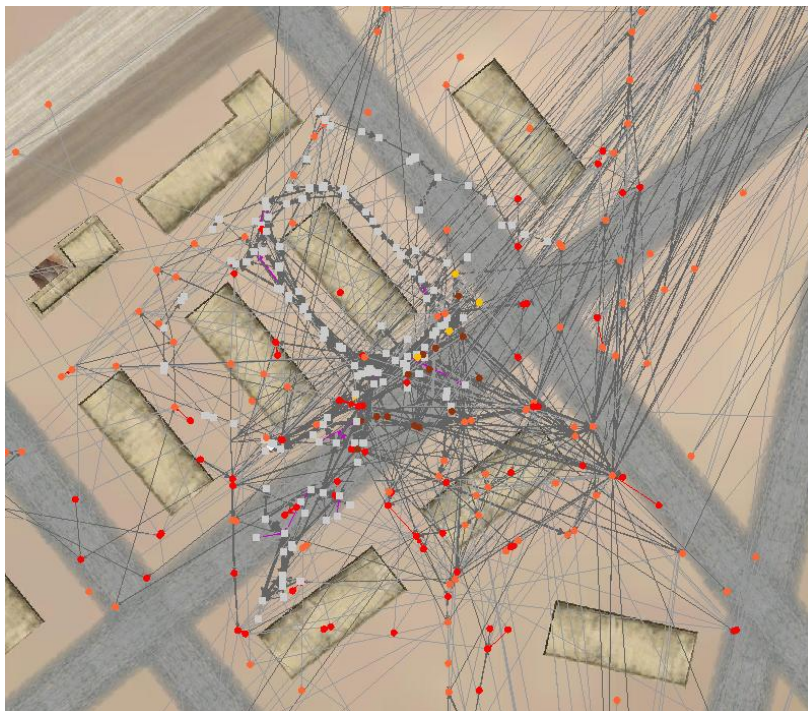


Figure 5.6: The Abstract Action Graph 4 of human play in the looting domain, containing abstracted nodes from the Action Graph.

The nodes in both graphs deliver the same type of information. They provide the position where the event occurred in the domain and contain a detailed description of the domain and agent's internal state. However; nodes in AAG are presented at a higher abstraction process, which means that they contain information from several nodes of the original AG and describe more general concepts from the taxonomies. The importance of a specific node in the AAG is denoted by its size. Bigger nodes contain more AG nodes that are similar, which means that the events and concepts are relevant for the observed group of agents. The detailed procedure of the abstraction process is presented in Subsection 3.2.3.

In this section we presented only one AG and AAG at abstraction level 4 for the human play in the looting domain due to practical reasons. All Action and Abstract Action Graphs for the two domains and different behaviors can be found in the appendix of the thesis.

## 5.2 Discovered Behavior Patterns

The intention of the qualitative evaluation presented in this subsection is to demonstrate that HMASDA is able to capture different complex integrated behavior displayed by interacting groups of humans at the tactical and strategic levels, and to present it in a transparent way. Under the given conditions, HMASDA constructed around 100 middle-level and five semi-strategic patterns. In the next subsections, we present a subset of the discovered patterns. The algorithm does not differentiate between soldier and civilian patterns and creates both types. We present and explain here both, since it gives an insight how the algorithm works and creates the patterns; however, only soldier patterns are used later on for evaluation of the other PATDS modules.

### 5.2.1 Riot Domain

The ability of HMASDA to analyze plays and discover meaningful behavior patterns that describe the characteristic situations in the simulation was first performed for the Riot domain. The domain is less complex than the Looting domain, since in principle it does not require from the participants to carefully plan movements in the scenario as the scenario can be resolved with minimal movement and an accurately coordinated sequence of actions. The lack of dynamics and additional complexity arises from the fact that this was the first developed scenario that was intended to verify the HMASDA algorithm and the concepts of the whole PATDS learning cycle.

The identified best strategy to cope with the scenario is to properly perceive the cognitive state of hostile individuals, subgroups and the whole group and select the appropriate level of response. Longer response time and milder actions for calmer crowds and fast response and stricter actions for more agitated crowds. In the remainder of the subsection we present six patterns, three of soldier and the remaining of crowd behavior. The patterns are not divided and presented individually for simulated or human play, due to the above-described lack of movement dynamics. The difference between the two types of play can be observed in the simple statistical analysis of the patterns (see Tables 5.1 and 5.2), presented at the end of the subsection.

The first pattern, presented in Figure 5.7, is a soldier pattern occurring close to the camp entry and describes the soldier agents responding to a more aggressive crowd behavior with a load gun event intended for intimidation. The pattern is located in the area in front of the camp entrance, where soldiers guard the entry and rioters gather to provoke conflicts. This area is where most of the conflict occurs and where most of the patterns are discovered. The soldier movement included in the pattern is limited. Soldiers move from the south part of the gate to the north to create a formation. The different parts of the pattern are enriched with symbolic descriptions, presented in the blue square, that provide additional explanations regarding the overall pattern and the conditions for the pattern to occur:

- The first field (Number of actions) displays the number of basic actions that are abstracted and assigned to a specific node of the pattern. The more actions that support the pattern, the more representative that pattern is. Moreover, the number of listed actions represents the learning examples that were created during frequent node

mining and thus used to induce symbolic descriptions. In the presented pattern 113 actions are included in the observed node.

- The second field (Abstracted action) provides the information on how much have the basic actions been abstracted and which is the mapped abstracted concept from the taxonomy of agent actions. In this pattern the abstracted node is shot, the fourth level concept from the action taxonomy (see Figure 5.2).
- The Rule field contains symbolic descriptions in the form of IF-THEN rules that provide factual conditions under which the observed action inside the pattern can occur. The values for all attributes are computed during the pattern mining phase in HMASDA for all basic actions in the observed node (learning instances). A list of all possible attributes is provided in Appendix A. The type of the attributes can be categorical or numerical. Categorical are regarded as Boolean (TRUE or FALSE) attributes and they determine its influence under the observed spatio-temporal circumstances. Numerical attributes can take values from the interval [0, 100] – values in the interval [0, 40] denote a low influence of the parameter on the simulation, those in the interval [40, 60] denote an important influence, and those in the interval [70, 100] denote a high impact influence of the observed attribute. Three rules were induced for the pattern in Figure 5.7. Rules can be evaluated separately, since they independently describe the selected part of the pattern. The first rule is simple and states that the soldier loads his gun when the average anger of civilian agents (*calc\_anger\_value*) nearby is very high (greater or equal 90). The second rule describes that the same action can be performed when the average civilian anger is greater or equal to 64 and the soldier evaluation level (*evaluation\_level*) is greater or equal to 55. This attribute is created from values outputted by the PESC model and simulation measures and can be regarded as an estimation of the current overall status in the scenario run. It tries to resemble the cognitive process of humans when assessing the severity of the observed situation. A higher value denotes harsher, while lower values represent milder current circumstances in the simulation. The last rule contains one categorical attribute that is TRUE if the soldier agent has been provoked by the surrounding civilian agents (*NearProvokeSlightlyEvent*). This is a combined attribute that can be extracted from surrounding civilian nodes in the Action Graph. The second part of the rule states that besides the soldier agent being provoked, the average civilian anger must be greater or equal 67.
- The last field (Actions) of the pattern description contains a list of all actions included in the observed pattern node and provides their occurring frequencies. This provides a mechanism for training supervisors to inspect which actions were adopted in this specific situation and deduce whether or not the actions taken were appropriate. In this example, the actions adopted by soldier agents were homogeneous and involve only load gun events (113 examples).



Figure 5.7: A discovered soldier pattern, occurring close to the camp entry in the riot domain, representing the load gun event.

The next example is another soldier pattern (see Figure 5.8). The pattern starts in the observation tower, north of the main gate. The soldier agent starts the scenario in the tower and moves to the main conflict area, north of the gate, when a turmoil evolves. The selected node of the pattern contains 31 basic actions, of which 29 are *gun\_shot\_warning* events and 2 *gun\_shot\_effective* events. The actions are abstracted into the *shot* node, the third level concept from the action taxonomy. For this specific part of the pattern, one IF-THEN rule was induced that explains the conditions under which this action was performed. The rule states that the soldier performs a *gun\_shot\_warning\_event* when the average anger of civilian agents (*calc\_anger\_value*) nearby is very high (greater or equal 90) and the soldier agent has loaded his gun before this event (*HasLoadedGun*).

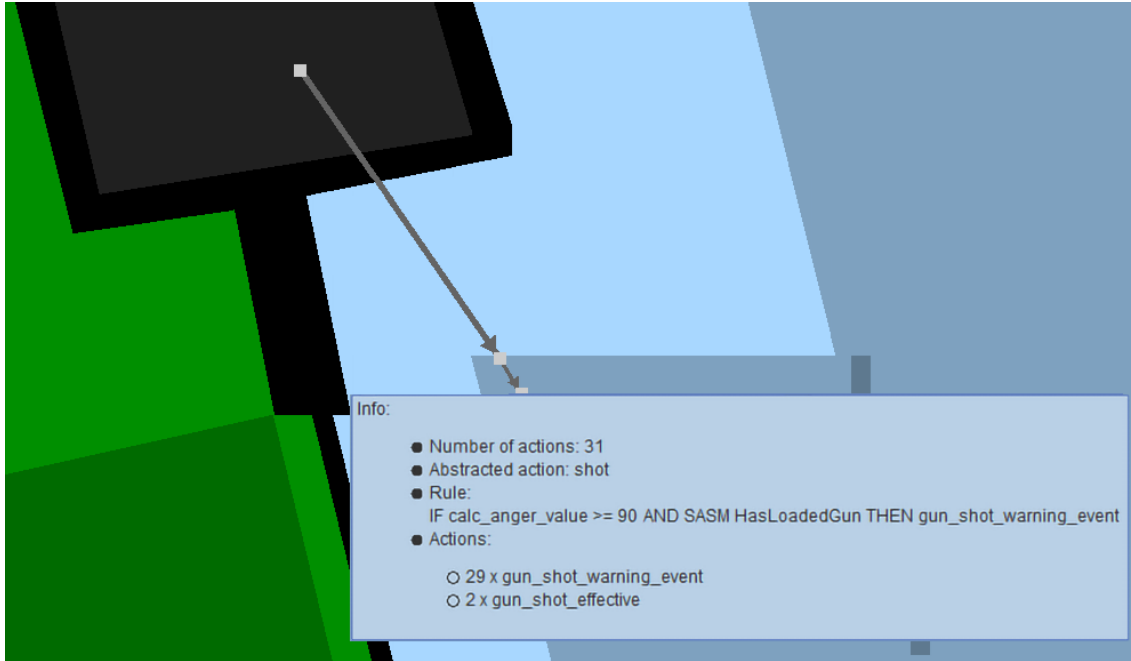


Figure 5.8: A discovered soldier pattern in the riot domain, representing movements from the tower area to the camp entry and performing warning shots.

In Figure 5.9 another soldier pattern representing a *gun\_shot\_warning\_event* is presented. In this case the pattern is short and fully located in the tower area, where some of the soldier agents start the scenario. From the previous pattern we learned that some soldiers moved from the tower to the conflict area when the turmoil started. In this example, we can observe a different strategy or a tendency of a subgroup of soldiers that remain in the tower when conflicts arise in the area around the camp entry to monitor the situation from higher ground and enforce strict response actions. The observed sub-pattern node contains 17 basic *gun\_shot\_warning\_event* actions and the same abstracted node in the taxonomy of agent actions (fourth level). The learning algorithm was able to induce two IF-THEN rules. The first rule relies only on the *evaluation\_level* attribute. The soldier in the tower area performs a warning gunshot when his evaluation of the overall scenario situation is that interactions between the soldier and civilian agents are getting increasingly more severe (value greater or equal to 72). The other rule also contains the *evaluation\_level* attribute, with a lower value (greater or equal 65), but also contains the categorical attribute *HasLoadedGun*. The interpretation behind the rule is that the soldier agent must have been already provoked during the scenario and in this case the evaluation of the overall condition could be lower to trigger the warning gunshot.

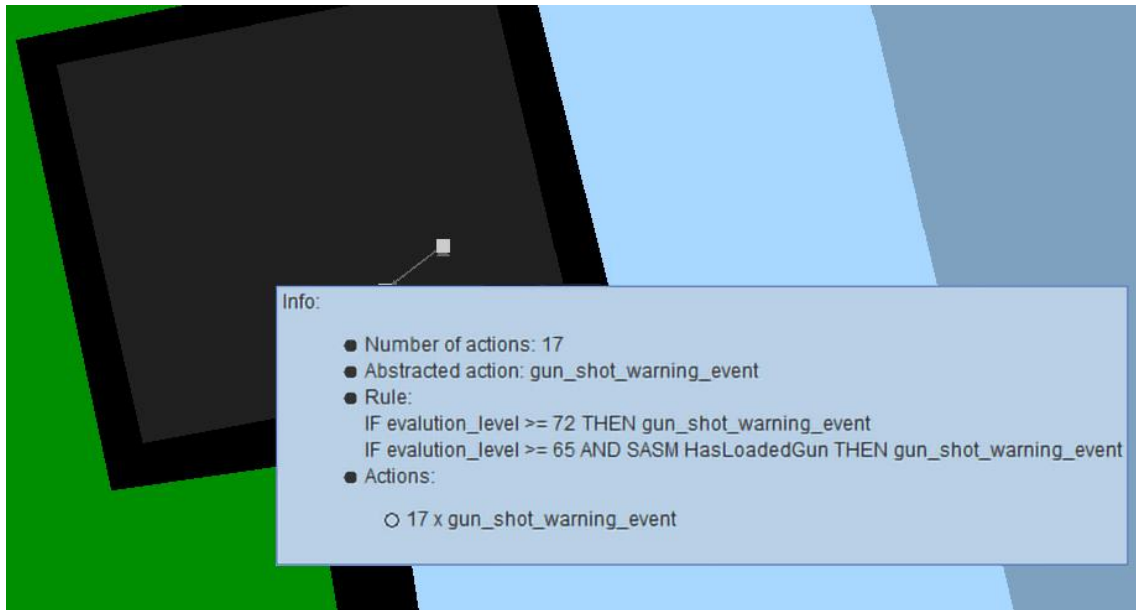


Figure 5.9: A discovered soldier pattern occurring in the tower area of the riot domain, describing warning shot events.

The remaining three patterns that will be presented in this subsection are different civilian patterns that represent the typical actions taken by the civilian agents using the fixed-policy behavior: (i) provoking soldier agents, (ii) movement, and (iii) attack events. The shape of the pattern nodes represents the two different roles of civilian agents. A triangle denotes a civilian agent charismatic leader (CALC) that has an important influence on the cognitive state and actions of other ordinary civilian agents (CAO), denoted with circles. The color of the node, on the other hand, represents the dominant cognitive state that influences the selection of actions as a response to soldier interactions. Red nodes indicate anger as the lead motive, green nodes indicate the need motive, and blue the fear motive (see Appendix A for a complete list of features).

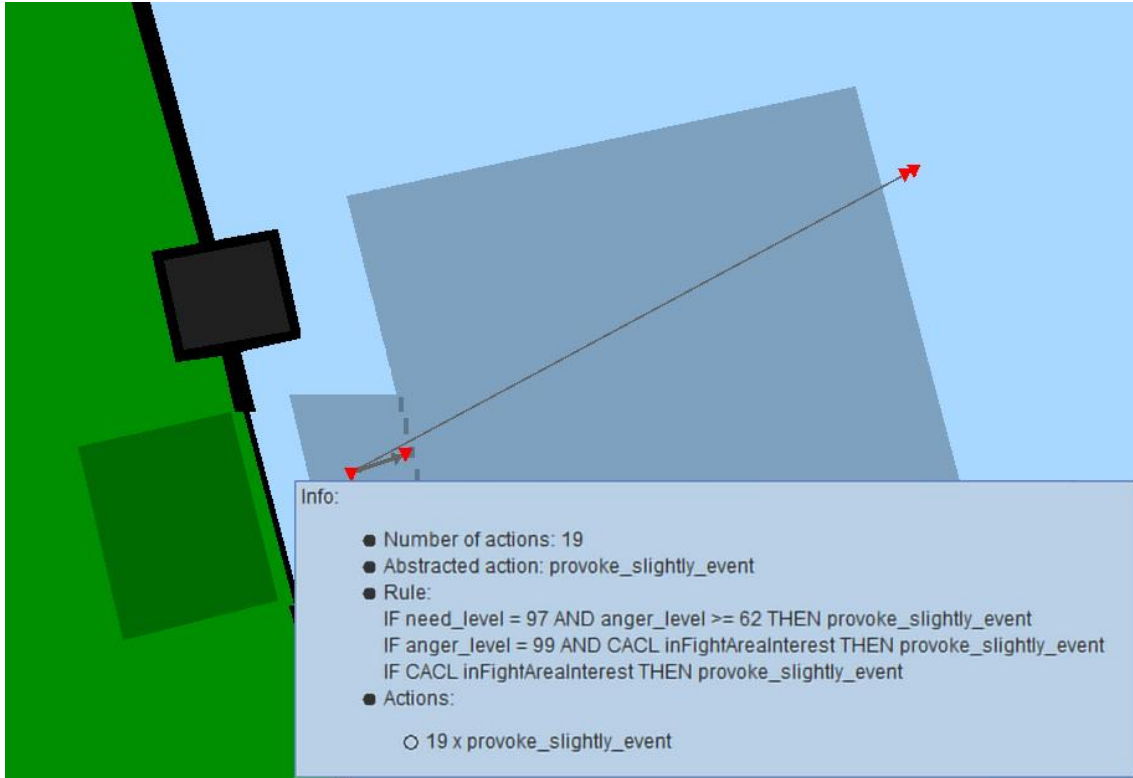


Figure 5.10: A discovered civilian pattern in the riot domain, describing provoke actions towards soldier agents occurring close to the camp entry.

Figure 5.10 shows one of the patterns where civilian agents provoke soldiers inside the fight area, near the camp entry (*provoke\_slightly\_event*). The pattern starts with a provoke event and then a flee event outside the area of interest. The actions are performed by the charismatic leader agent that directly interacts and negotiates with soldier agents. The red color of the nodes indicates anger as the main cognitive state of the civilian agents in the presented pattern. The symbolic description of the selected node provides information that 19 basic *provoke\_slightly* events are included in the node of the pattern. The actions are not abstracted and are linked to the third level leaf node *provoke\_slightly\_event* in the taxonomy of agent actions. Three IF-THEN rules were induced for the observed node. The first states that provoking starts when the need level of the civilian leader is very high (97) and the agents start becoming angry (anger greater than 62). This is an appropriate representation of the civilian agent fixed-policy strategy. The pattern starts at the beginning of the scenario, since the need level is still high and the agent gets frustrated and later angry since his need to enter the camp is not realized. The response for this combination is to provoke the soldier agents and instigate disorder. The second rule describes that provocations start when the civilian leader is very angry and he is located in the fight area (categorical attribute *inFightAreaInterest*) close to the gate entry. A similar explanation is provided by the third rule.

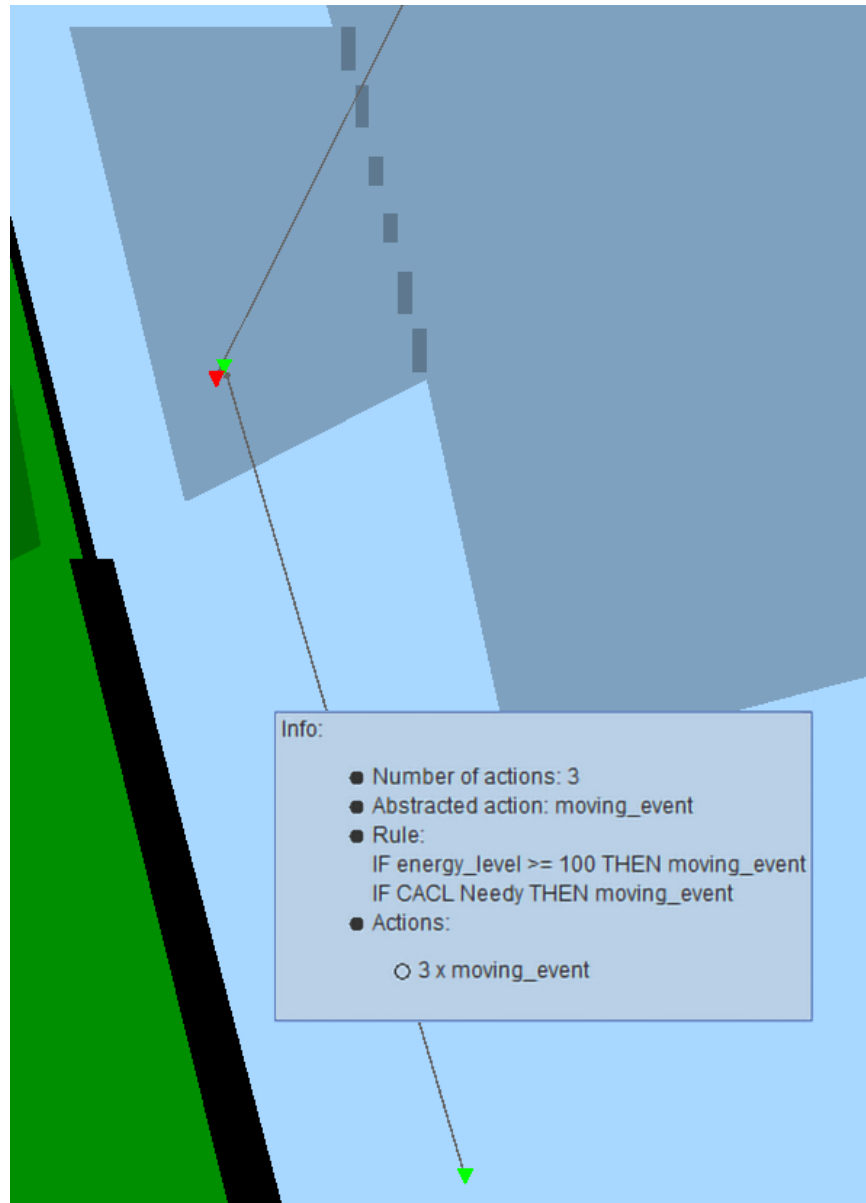


Figure 5.11: A discovered civilian movement pattern in the riot domain.

The second presented civilian pattern (see Figure 5.11) is a characteristic movement pattern. Civilians start the scenario at the southern part of the domain and start moving (*moving\_event*) north towards the area of interest near the camp entry. This is the only sequence of actions that occurs every scenario run, independently of how the scenario evolves. The presented pattern describes the movement of the civilian charismatic leader (CALC), the green triangle provides the information that the leading cognitive state of the civilian leader is needed and he performs predefined action routines according to this: movement to the area of interest and starting the interaction/negotiation with soldier agents. There are only three basic actions, *moving\_event*, included in the pattern action node since the observations are regarding the individual civilian leader in the entire group. The observed action is not abstracted and is the third level leaf concept in the taxonomy of agent actions. Two rules were generated for the pattern which capture both prerequisites for the selected action routine. The first one states that the leader civilian agent moves towards north when his energy level is at maximal value (*energy\_level* greater or equal 100). The second rule determines that the leading cognitive state of the leader agent must be needed in order to start the movement event.

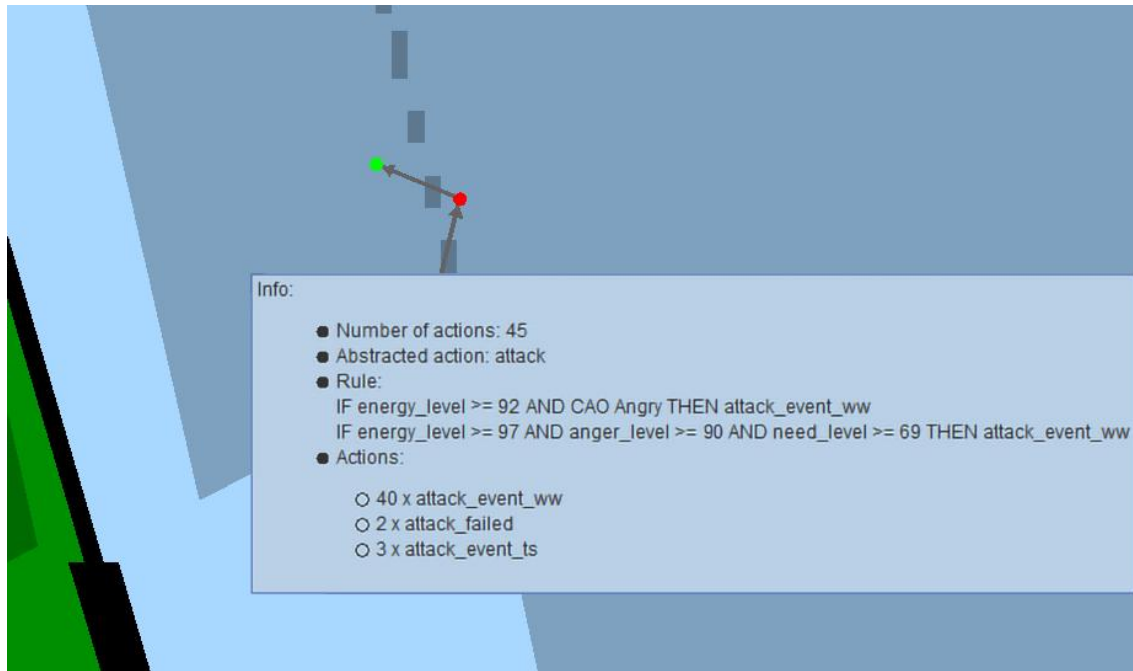


Figure 5.12: A discovered civilian pattern in the riot domain, describing attack events in front of the camp entry.

The last civilian pattern is an attack event (see Figure 5.12), where ordinary civilian agents (CAO) start to verbally instigate the soldier agents and throw stones at them. The attack usually occurs in the area of interest close to the camp entry as a response to the frustration and anger due to the inability to enter the camp and thus resolve their need. The red circle as part of the observed pattern node tells us that the leading cognitive state of the ordinary civilian is anger. 45 basic actions are included in the attack node, of which 40 are *attack\_event\_ww*, 2 *attack\_failed*, and 3 *attack\_event\_ts*. The first attack type is verbal, the second denotes a failed attack with stones and the last type denotes throwing stones and successfully hitting the soldier agents. All these basic actions are abstracted to the attack concept, a second level concept in the taxonomy of agent actions. Two rules were induced from the learning examples to further explain the pattern action node. The first rule states that the attack occurs when the energy level of the ordinary civilian is high (*energy\_level* greater or equal 92) and his leading cognitive state is *Angry*. The second rule explains the conditions of the pattern with three numerical attributes: the energy level is high (greater or equal to 97), as is the CAO anger level (greater or equal to 90) and the need level is getting lower, compared to the levels at the beginning of the scenario (greater or equal to 69), but still contributing to the overall civilian behavior model.

The number of discovered behavior patterns in the riot domain is between 50-100, depending on the strategy and the selected abstraction level. Since a visual comparison between patterns discovered for the fixed-policy and human play does not provide good distinctions, due to a limited set of possible outcomes, we performed a statistical analysis about the frequency of which all possible basic soldier actions occur in the patterns. The distribution of actions in pattern instances for the fixed-policy strategy is presented in Table 5.1 and the distribution of actions in patterns for human play is presented in Table 5.2.

The first column lists all seven possible different basic soldier actions in the riot domain. The second column contains the most common length (the mode) of all the patterns where the current action from the first column is included in the pattern. The third column contains the number of pattern instances that have at least one pattern node with the current action. A specific pattern might be assigned to instances under multiple actions, if it contains basic actions of different types.

Table 5.1 presents the distribution of soldier actions in patterns for the fixed-policy strategy. There are not many movement actions present in the patterns, which is in line with the fixed-policy, since the focus is to apply direct actions towards individual civilian agents based on the

severity of the current domain state (*evaluation\_level*). It can be observed that the frequency distribution of all actions is declining with stricter conditions in the specific scenario run. *Communicate\_calm* event is the most frequent action in all the patterns. The explanation is that thresholds are not set equidistantly, but both communicate events take roughly 50% of the entire interval space. Moreover, the *evaluation\_level* attribute does not increase linearly, but exponentially; therefore, stricter actions are achieved quicker with increasing attribute levels (Table 5.1).

Table 5.1: The distribution of soldier actions in the discovered patterns for the fixed-policy strategy in the riot domain.

| Pattern type        | Modal length | Instances |
|---------------------|--------------|-----------|
| Moving event        | 3            | 27        |
| Communicate calm    | 2            | 42        |
| Communicate warning | 2            | 32        |
| Show of weapon      | 3            | 31        |
| Load gun            | 2            | 29        |
| Warning shot        | 3            | 17        |
| Effective shot      | 3            | 14        |

Table 5.2 presents the distribution of soldier actions for the strategy adopted by operational people. There were less discovered patterns overall compared to the fixed-policy strategy, thus less instances in the table. In addition, there are more movement actions present in the patterns of operational people, which correspond to the learned behavior of soldiers, since they often gather in formations and move towards points of interest. Moreover, the frequency of actions in the distribution is skewed towards less lethal actions. *Communicate\_warning* and *Show\_of\_weapon* events are the most frequent actions appearing in all patterns. When taking into account the time component of the scenario it can be observed that soldiers adopted these actions sooner during the scenario: as soon as the civilians started to get agitated. This resulted in a behavior change of civilians, since anger was at a manageable level and fear was still able to take over as the main cognitive process. For this reason, there was not a need to resort to lethal actions and only a handful of warning shots were performed and none of the effective shots.

Table 5.2: The distribution of soldier actions in the discovered patterns for the strategy adopted by operational people in the riot domain.

| Pattern type        | Modal length | Instances |
|---------------------|--------------|-----------|
| Moving event        | 2            | 47        |
| Communicate calm    | 3            | 26        |
| Communicate warning | 3            | 31        |
| Show of weapon      | 2            | 35        |
| Load gun            | 2            | 21        |
| Warning shot        | 2            | 9         |
| Effective shot      | 0            | 0         |

### 5.2.2 Looting Domain

The Looting domain is more complex and dynamic compared to the Riot domain, since it allows for more movement actions and different interactions. Soldiers start the scenario along the road leading to the shop building that is being looted. They start moving towards the building to

prevent looting, secure it, and prevent any possible conflicts. Besides moving to strategic positions, the key to successfully cope with the scenario is, as in the previous, to time the appropriate actions based on the overall state of the crowd. However, since there are more possible actions and relations between locations and the distance between agents of different types the scenario is more complicated to successfully resolve.

We divide the analysis into two cases. In the first, we analyze the simulated, fixed-policy behavior implemented in the simulation. The implemented strategy in this case is to move along the road, next to the looted shop and only respond to the aggressive actions of civilian agents. In the second, the analysis of human play is presented, where operational people devised an optimal strategy to resolve it with minimal damage. The patterns discovered in both experiments are significantly different and are presented in the next two subsections. In each, soldier and civilian patterns are presented.

### 5.2.2.1 Simulated behavior

In the simulated behavior experiment, soldier agents use a simple fixed-policy strategy, based on reactive behavior. Since they only react to interactions from civilian agents and the actions soldiers adopt are proportional to the aggressiveness of civilian agents, the scenario almost always escalates to excessively aggressive behavior from both groups. Regardless of how the scenario evolves, we are interested detecting the behavior that leads to such an outcome. We start with two patterns describing the behavior of security experts adopting the passive behavior strategy. The first pattern, presented in Figure 5.13, shows the initial movements performed by the experts from the starting position. Exploring the graphical part of the pattern provides some information regarding the positions of the experts and their performed movements. Soldiers start their patrol next to a building south-west of the looted shop, and continue to move along the street. The symbolic description in the blue square provides additional explanations regarding the overall pattern and the conditions when this pattern occurred:

- The first parameter (in this case 55) displays the number of actions that are included in a specific node of the pattern. The more actions that support the pattern, the more representative that pattern is.
- The second parameter (abstracted action moving) provides information on which action in the taxonomy of actions the algorithm has reached. This information reveals how abstracted the pattern is.
- Next, we present the two discovered symbolic IF-THEN rules that determine the conditions under which the selected actions in the pattern occur. Numerical attributes can take values from the interval [0, 100] – values in the interval [0, 40] denote a low influence of the parameter on the simulation, those in the interval [40, 60] denote an important influence, and those in the interval [70, 100] denote a high impact influence of the observed attribute. Attributes that take up values from the last interval contribute significantly to the course of the simulation. In the first rule, the *evaluation\_level* parameter is high, which is true since the soldier agent is computing the movement plan, and assessing the surrounding conditions. The *average\_civilian\_anger* value tells us that it is not yet a leading parameter in the simulation, but it might result in a problematic situation if not handled promptly. Other categorical attributes can take values TRUE or FALSE. In the second rule, we learn that the civilians are currently looting, and the soldiers are moving towards them. SASM (Soldier Agent Squad Member) is the role of the current agent taken from the taxonomy of agent roles.
- The fourth parameter lists all the actions included in the pattern node, and provides their occurrence number. In this way, the training supervisors can deduce whether or not the actions taken were appropriate for the specific situation. In this example, the peacekeepers considered and adopted various moving events (six *moving\_event\_SAs* and 49 general moving events) during similar situations.
- The last parameter shows the leading cognitive motive for the selected action. Soldiers always have this parameter set to 'focused' since their cognitive state is not displayed explicitly; however, they are cognitive aware in two ways: First, the patterns include cognitive states of the opposing agents and thus the soldier acts based on cognitive

perception of the situation. Second, the patterns reflect the cognitive state of the trainee. A more aggressive player usually utilizes a more aggressive behavior, which is reflected in more aggressive behavior patterns and vice versa.

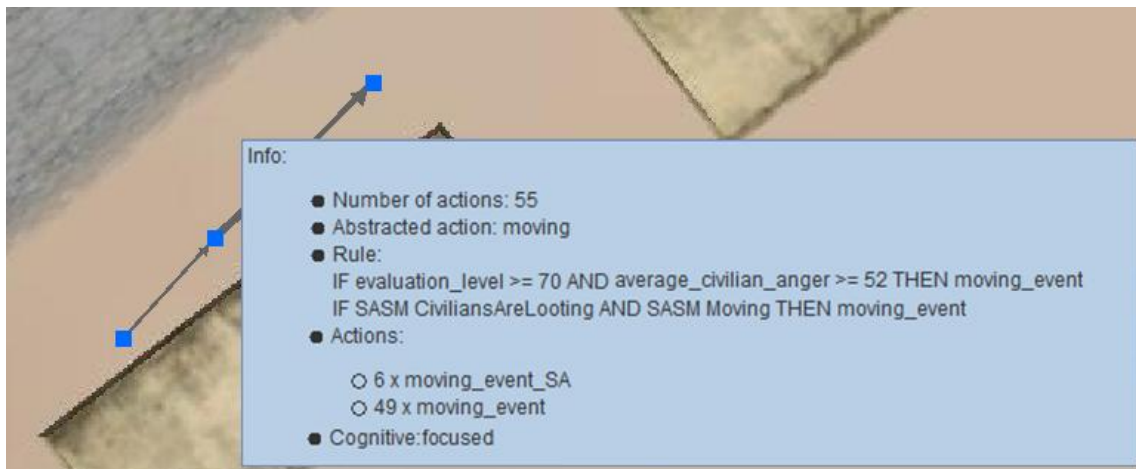


Figure 5.13: A discovered soldier pattern, describing movements in the looting domain.

Figure 5.14 shows the second characteristic pattern of the strategy adopted by security experts. Soldiers passively patrol and respond when they are attacked. This occurs when soldiers move close to the looted shop and civilians start to attack them. This specific behavior is successfully captured by the pattern, which explains why the warning shot is fired close to the shop. The induced rule describes that the soldier responds with fire when attacked, and the average civilian anger attribute becomes one of the important parameters of the scenario (with anger level above 74). We can also see that, in these circumstances, the peacekeepers responded 10 times with a warning shot and 17 times with an effective shot. In this particular situation, the training supervisor might instruct the trainees that less severe actions should be used in the following training sessions.

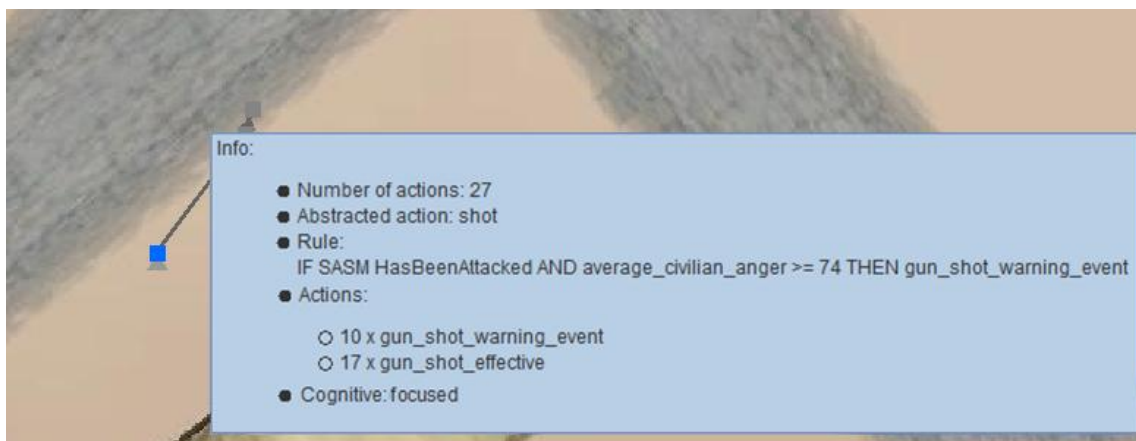


Figure 5.14: A discovered soldier pattern in the looting domain, describing the response to an attack with a warning shot.

The last figure in this subsection (Figure 5.15) shows an attack pattern performed by the civilian agents (CAO -- Civilian Agent Ordinary). Their main motive is to loot the shop; however, they attack the security forces if they feel threatened or are not appropriately approached (in some cases, insurgents mix with looters to ambush the security forces). They attack the patrolling forces close to the shop building when their cognitive process is affected by anger, and their anger levels increase considerably (in this case to 78).

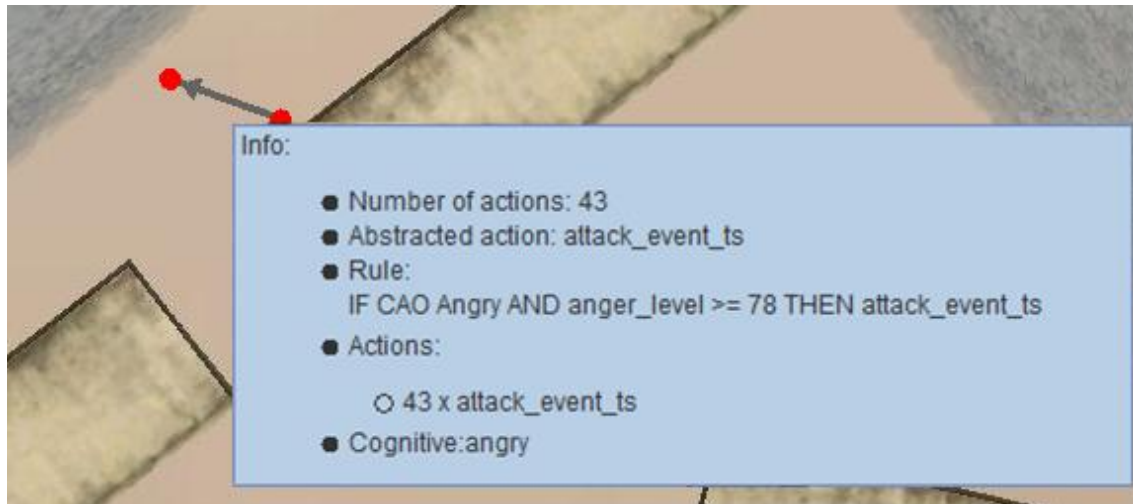


Figure 5.15: A discovered civilian attack pattern in the looting domain.

### 5.2.2.2 Human play

The next subsection presents three soldier behavior patterns, and two civilian ones, that were obtained by analyzing the strategy that follows appropriate Measures of Effectiveness (MoE) defined by experts to cope with the presented scenario.

The first soldier pattern, presented in Figure 5.16, occurs under the same conditions as the pattern of Figure 5.13. Both represent the movement of soldiers from their initial position. By inspecting the two patterns, the HMASDA discovers that in this situation the pattern is positioned more to the north, towards the shop building. This complies with the adopted strategy, since the experts moved towards the shop when the looting started. The induced rule explains this exact situation – the soldiers move towards the shop when they are still halfway to the shop. The civilians are looting, but their average anger level is still low.

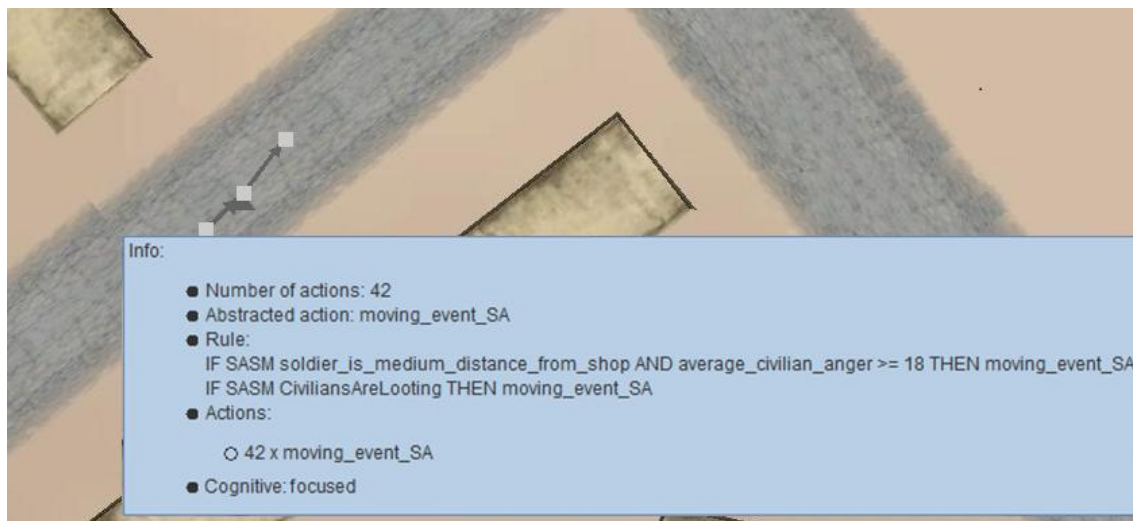


Figure 5.16: A discovered soldier pattern obtained for the human training session in the looting domain. The pattern describes movements towards insurgents looting a shop.

The second soldier pattern (Figure 5.17) represents the characteristic approach that soldiers adopt when they try to scare away the remaining looters and rioters. This is not a basic action, but a complex routine adopted by the experts composed of several basic actions. The novel introduction of routines into the HMASDA algorithm improves the expressiveness and the

comprehensibility of the extracted patterns. In this case, the approaching civilian sequence is composed of several move events towards a civilian agent with the intention of inducing fear. The pattern occurs close to the looted shop, where most of the civilians are located. The induced symbolic description explains that this pattern is executed either when the average civilian anger is quite high, or when the security forces have been attacked. The latter highlights the similarity between this and the pattern in Figure 5.14, since both occur when the security forces are attacked. In the latter situation, the security forces are located further away from the shop, and they immediately respond with a warning shot. Again, the difference between the two behaviors is well-emphasized.

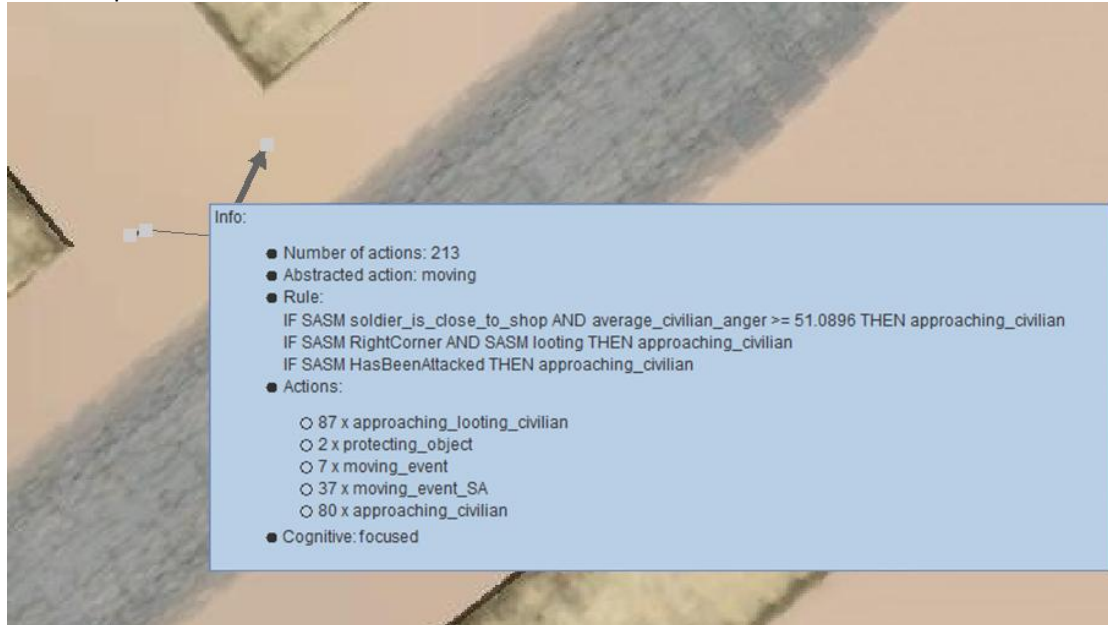


Figure 5.17: A discovered soldier pattern obtained for the human training session in the looting domain. The pattern describes approaching civilians near the looted shop.

The last soldier pattern is the protecting object pattern (Figure 5.18). As in the previous pattern, it shows a complex routine that is detected when the soldier is moving around a building, trying to secure it, and it usually occurs close to the conclusion of the scenario. The rule induced for the last part of the move around the building describes that the pattern usually occurs when the soldier is close to the left corner of the building, and the civilian anger level is around 60%. Such patterns were not discovered during the analysis of the passive strategy, since such behavior was not demonstrated during the simulation. This confirms that the algorithm can differentiate between different strategic plays.

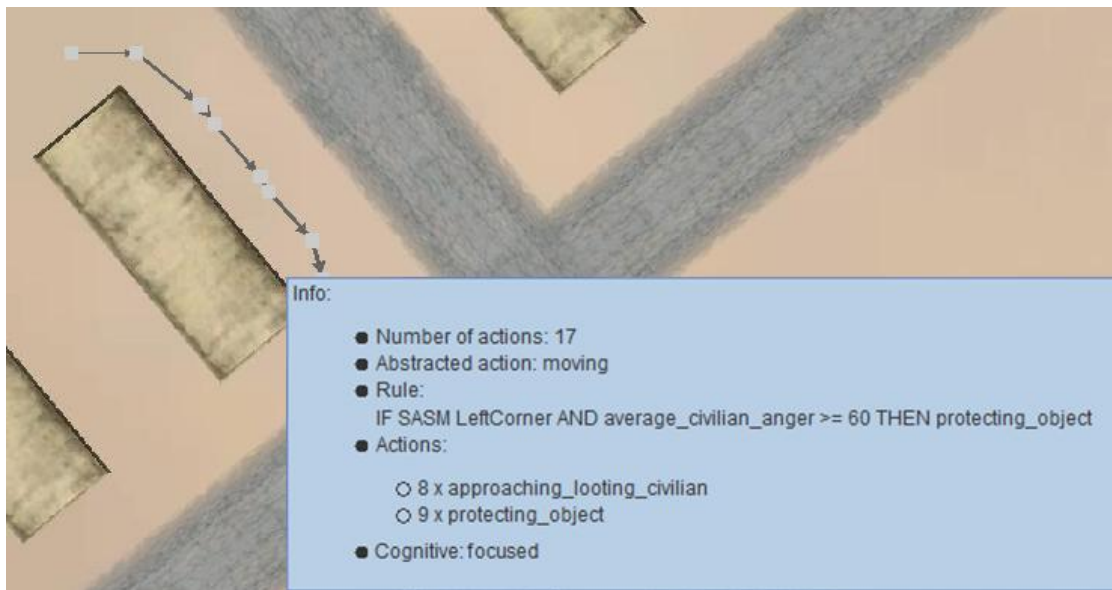


Figure 5.18: A discovered soldier pattern obtained for the human training session in the looting domain, where soldiers are protecting the looted shop.

The first civilian pattern, presented in Figure 5.19, shows the looting and move pattern. The pattern starts in front of the shop entrance where civilians start looting before they move along the street. The presented rule explains that looting begins when the anger and fear levels of civilians are still low. Their cognitive process is driven by the looting motive that is higher than the fear and anger motives, and when their energy levels were at their maximum. We discovered similar patterns in the passive strategy behavior, since it is inherent to the used scenario simulator, and is exhibited at the beginning of the scenario.

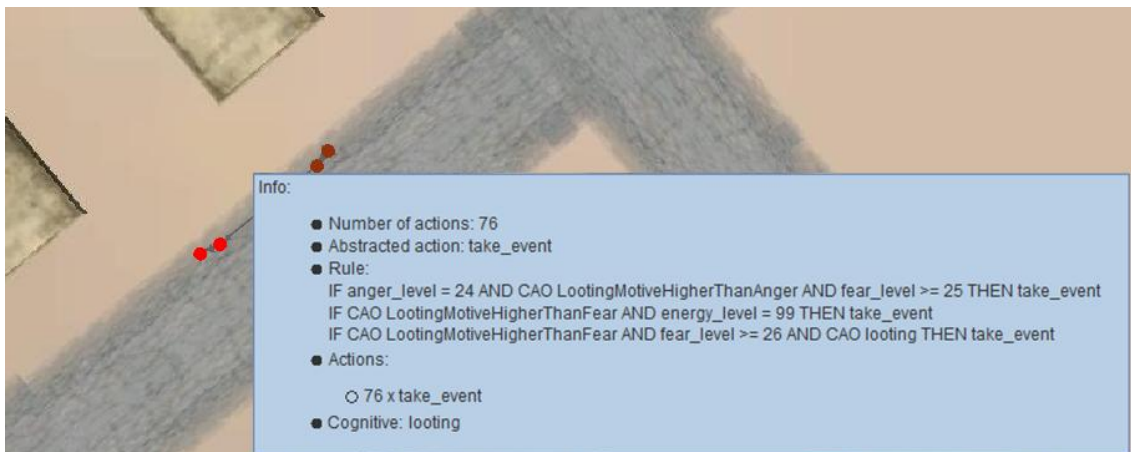


Figure 5.19: A discovered civilian pattern obtained for the human training session in the looting domain. The pattern describes civilians looting a shop.

The civilian attack pattern is presented in Figure 5.20. The general conditions for this pattern to occur were similar for each simulation run; however, the occurrence of the attack pattern is closely related to the actions performed by the security forces, and the location where it occurs varies depending on how the scenario evolves. In this case, the pattern occurs close to the building next to the looted shop. The rule explains that the attack actions are triggered by the high anger level of the civilians, which is higher than their fear. Their anger and energy levels are close to their maximum values.

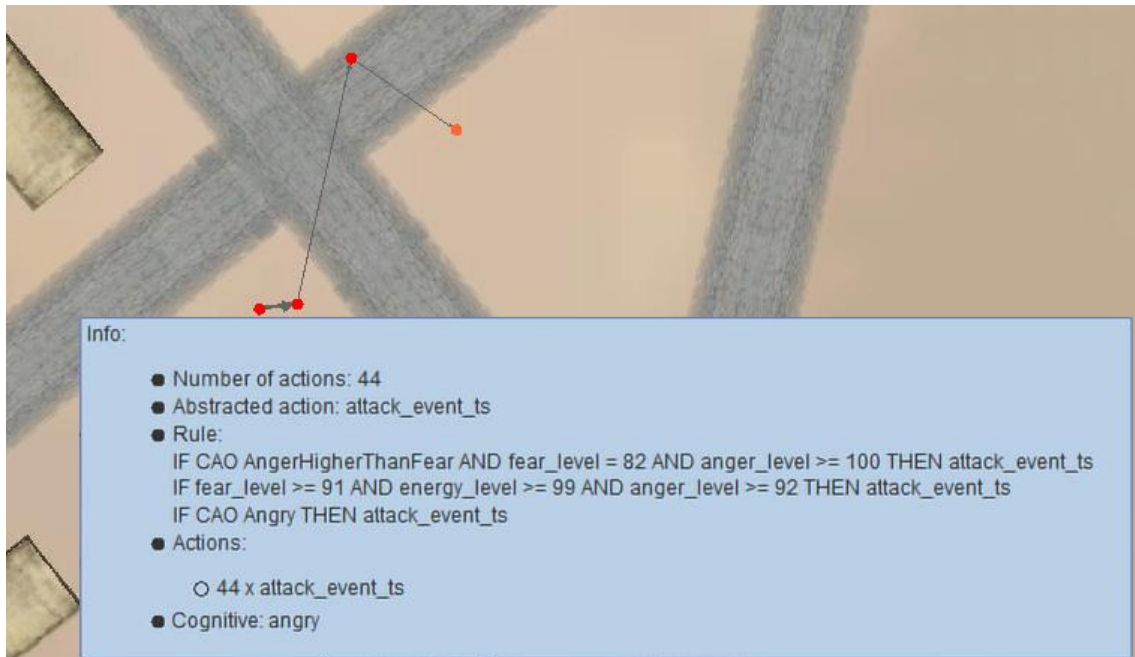


Figure 5.20: A discovered civilian pattern obtained for the human training session in the looting domain, where insurgents are attacking soldier agents.

### 5.2.2.3 Distribution evaluation

The presented patterns are just a small subset of more than 100 patterns discovered by HMASDA, and they demonstrate the ability of the algorithm to capture complex human strategic behavior in a non-trivial domain. It should be noted that the cognitive features represent an important part of the patterns, and provide valuable insights into the behavior of the participants of the game. The two tables below summarize the information regarding the discovered patterns for both strategies used during the simulation. The first column lists the types of pattern detected, which are based on the actions they contain. We have identified six main groups to which all patterns were assigned. The second column contains the most common length of each of the patterns (the mode) for each group, and the third column contains the number of instances included in the group. The total number of discovered patterns is similar in both experiments; however, the distribution along various types of patterns is significantly different. This indicates that the algorithm was able to identify the difference between the two behaviors, and to create diverse patterns. Table 5.3 presents the pattern information for the passive behavior strategy. Most of the included patterns are soldier movements and gunshots, which is consistent with the passive strategy.

Table 5.3: The distribution of all soldier actions in the behavior patterns discovered for the fixed-policy strategy in the looting domain.

| Pattern type          | Modal length | Instances |
|-----------------------|--------------|-----------|
| Soldier movement      | 3            | 64        |
| Soldier communication | 0            | 0         |
| Approaching civilian  | 2            | 5         |
| Protecting object     | 0            | 0         |
| Handling gun          | 3            | 12        |
| Shooting gun          | 2            | 37        |
|                       | 2            | 118       |

The summary of the patterns discovered during the analysis of the active behavior strategy is presented in Table 5.4. As before, the majority of the discovered patterns are movement patterns; however, the distribution between the other types of patterns is different. There are several communication and protecting patterns that were not present in the previous experiment. Moreover, there are more approaching civilian and weapon handling patterns. These patterns represent the behaviors defined by security experts, and were used during the simulation.

Table 5.4: The distribution of all soldier actions in the behavior patterns discovered for human play in the looting domain.

| Pattern type          | Modal length | Instances |
|-----------------------|--------------|-----------|
| Soldier movement      | 3            | 36        |
| Soldier communication | 3            | 13        |
| Approaching civilian  | 2            | 18        |
| Protecting object     | 5            | 9         |
| Handling gun          | 2            | 21        |
| Shooting gun          | 3            | 13        |
|                       | 2            | 110       |

The presented experiments and provided results confirm part of the main hypothesis, since it supports the creation of behavior patterns used to evaluate the observed behavior. Moreover, it supports the first scientific contribution from Section 1.3.

### 5.3 Evaluation of Pattern Clustering

The second part of the overall evaluation deal with the remaining three components of PATDS. We present here the evaluation of pattern clustering. For each scenario analysis, HMASDA discovers a big set of patterns, in the range of several hundred. A significant part of the patterns is similar; therefore, possibly redundant. We wanted to verify if it is possible to efficiently cluster patterns based on similarity through multiple dimensions and how the patterns set size influences the quality of the reproduced behavior. To that end, the behavior patterns are clustered into groups using the k-means algorithm [104] with the k-means++ [105] revision for choosing the initial values or *seeds* for the clustering algorithm. The algorithm was run several times for  $k \in [2, 20]$ . The obtained clusters were evaluated in terms of the similarity of patterns assigned to the clusters. For each run we created silhouette plots [103] that estimate the separation distance between the resulting clusters and can identify which patterns lie well within the cluster and which ones are somewhere in between clusters. This measure ranges from +1, indicating patterns that are safely distant from neighboring clusters, through 0, indicating patterns that are not distinctly in one cluster or another, to -1, indicating patterns that are probably assigned to the wrong cluster.

A sample of the most interesting silhouettes is presented in Figure 5.21. Each cluster is represented by one silhouette. The plot is a horizontal line, whose length indicates the value of the similarity measure for each instance in the cluster. A wide silhouette indicates a large similarity between instances and hence a well-pronounced cluster. The height of a cluster is simply equal to the number of objects in the cluster. The entire clustering is then displayed by plotting all silhouettes into a single diagram, from which the quality of the clusters can be compared. The average silhouette width can be used to select the most adequate number of clusters, by choosing the  $k$  value that yields the highest silhouette width. As a rule of thumb, an average silhouette value between 0.5 and 0.7 denotes reasonably well-structured clusters, and values higher than 0.7 indicate a strong cluster structure.

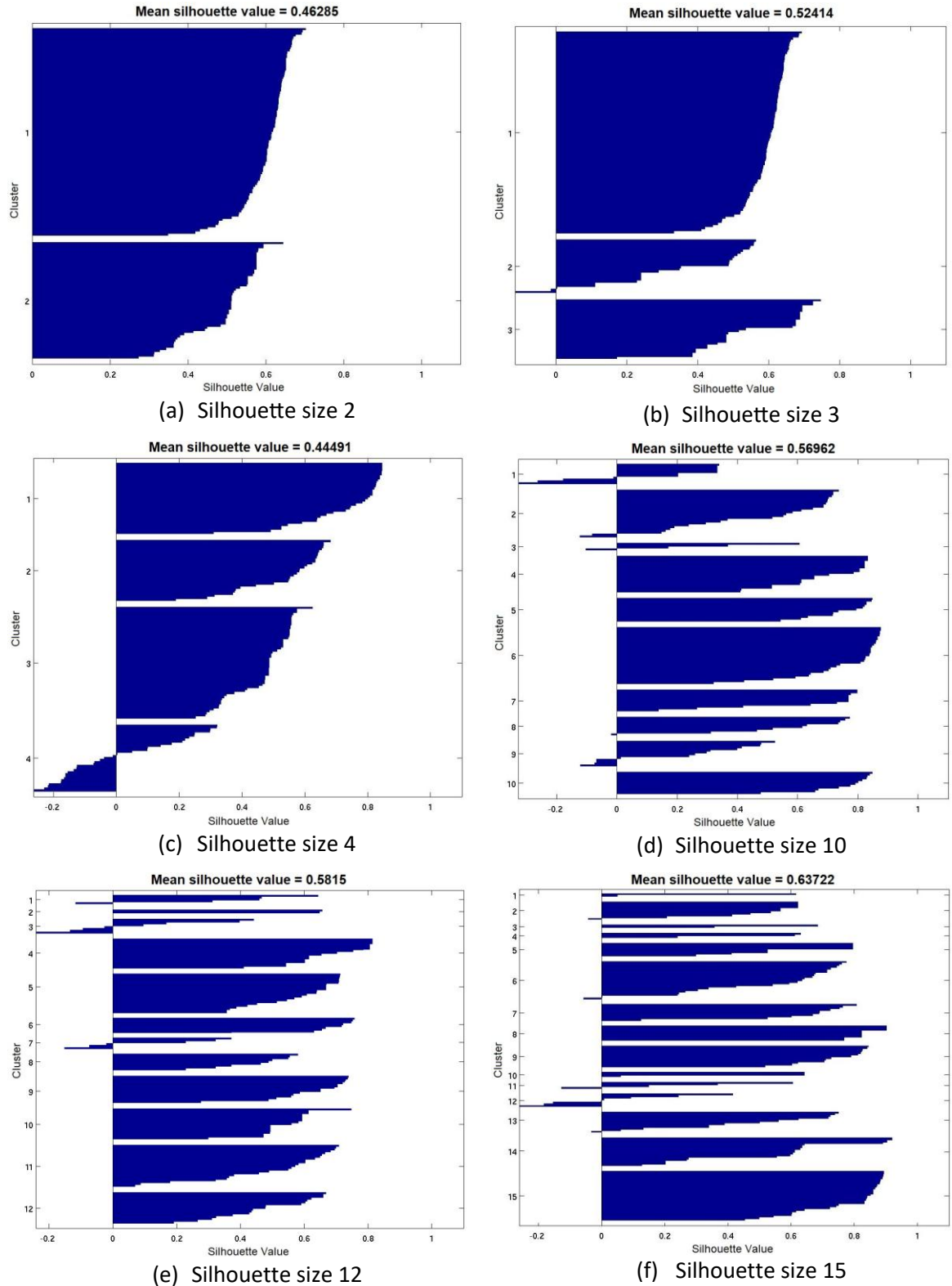


Figure 5.21: Mean silhouette size for different cluster sizes. Higher values denote more homogeneous clusters that contain similar patterns.

The six figures above represent different groups of patterns. Subfigure (a) roughly divides patterns into soldier patrolling patterns and non-movement actions, which is a reasonable split for 2 clusters. Subfigure (b) further splits between moving, calming, and stricter soldier actions. All other clusters further partition the patterns into more specific actions that represent moving

under various circumstances and different degrees of soldier intervention. The cluster with a higher  $k$  value tends to differentiate between more pattern types performed under different situations.

Through the comparison of all constructed silhouettes the best value for  $k$  was 15. We also applied the X-means algorithm [106], an extension of the k-means algorithm that automatically calculates the optimal number of clusters. The algorithm starts with  $k$  equal to the lower bound of the given range and continues to add centroids where they are needed until the upper bound for  $k$  is reached. During this process, the centroid set that achieves the best score is recorded, and at the end, the algorithm outputs the optimal number of clusters and the resulting clusters. The computed number of clusters using X-means was  $k=16$ , which is close to the number of clusters observed using silhouettes.

Our analysis was not focused on the patterns included in each cluster, since it is not relevant for the overall PATDS method. It is, however, important how the cluster size affects the performance during behavior cloning. In the next subsection, the evaluation of behavior cloning for different sizes of clusters is presented.

The presented experiments confirm sub-hypothesis 2 and support the third scientific contribution from Section 1.3.

## 5.4 Evaluation of Behavior Cloning

The next evaluation in the overall experimental setup is the cloning procedure that aims to reproduce as credibly as possible the behavior of the observed individuals. The evaluation of the similarity between the original and the cloned performances is computed using DTW on the KSPs or in our case the Measures of Effectiveness (MoEs) that are created during simulation runs. An example of the MoEs for the LD was presented in the previous section. The simulations for both domains output 16 parameters, but we only use a subset of the MoEs that were selected by security experts and are deemed important for the evaluation of mission success. The DTW similarity was computed for each pair of the corresponding Mo's and averaged over all simulation runs.

In the previous subsection we analyzed clustering based on the cluster size. In this experiment we evaluate how cluster sizes influence the performance of the surrogate agent. The results of the evaluation are presented in the graph in Figure 5.22. We compared each MoE log with all the others and averaged the similarities. The first column from the left (green) represents the similarity between the simulation logs of the same type and experiment. In this case, the simulation logs obtained from real training session with security personnel. The value 0.7038 of the first column is the reference point for the other similarity experiments since it represents an estimation of the maximal similarity for simulation experiments due to the stochasticity of the simulation. The second column (red) represents the similarity between the original simulation logs and cloned simulation logs, where the entire set of extracted behavior patterns was used to create the surrogate agent. A similarity value 0.3101 was obtained by computing the similarity between each simulation log of the surrogate agent with all the original simulation logs and averaging the similarities. The obtained similarity value is low, which implies that using all extracted patterns during cloning affects the performance of the cloning module. To obtain the remaining seven columns (blue), clustering was performed on the entire set of behavior patterns with various numbers of clusters.

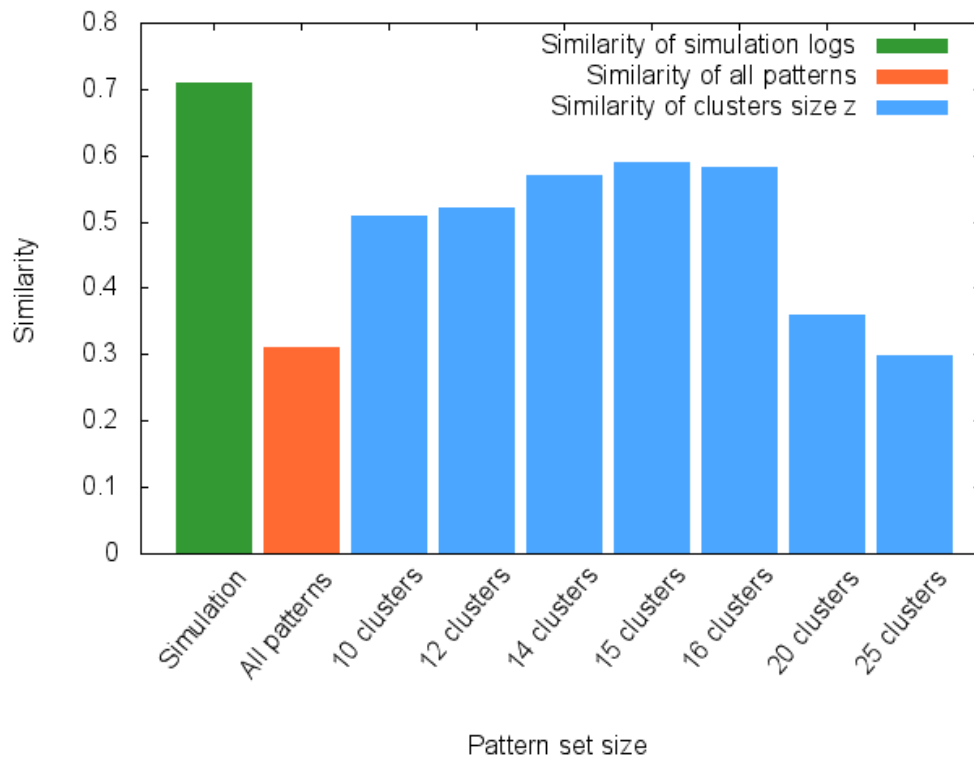


Figure 5.22: Comparison of similarities between the original simulation and cloned behavior for different cluster sizes (number of patterns used).

As can be seen, the best performance of the cloning module occurred when 15 clusters were used for the creation of the surrogate agent. The results indicate that clustering improves the performance of the cloning procedure. The tentative explanation for this observation is that the number of clusters corresponds to significantly different patterns, with too small a number being too general and not containing enough information about the behavior, and too big a number being too specific and containing too many similar patterns that introduce noise. For the two remaining evaluations, clustering was applied after pattern extraction with HMASDA to obtain a reduced and more efficient set of patterns for the creation of the surrogate agent.

The results in Figure 5.23 present the similarity measure between different behaviors. RD and LD denote the rioting and the looting domains; F the fixed-policy agent (that is, the simulation system); FS the fixed-policy agent surrogate (clone of the simulation system); T the trainee (human); and TS the trainee surrogate (clone of the trainee). The first bar (F-F) in Figure 5.23 shows the similarity between the different simulations of the fixed-policy agent, while the second bar (F-FS) shows the similarity between the fixed-policy agent and its clone (SA) in RD. The four left-most bars show that surrogate modeling reproduces behavior that is very similar to the fixed-policy agent in the RD and quite similar for the LD. The next four columns represent the similarity of behavior of the trainees and their surrogates in both domains. For verification purposes, the last four bars compare the behavior of the fixed-policy agent to that of the human trainees and the trainee surrogates in both domains.

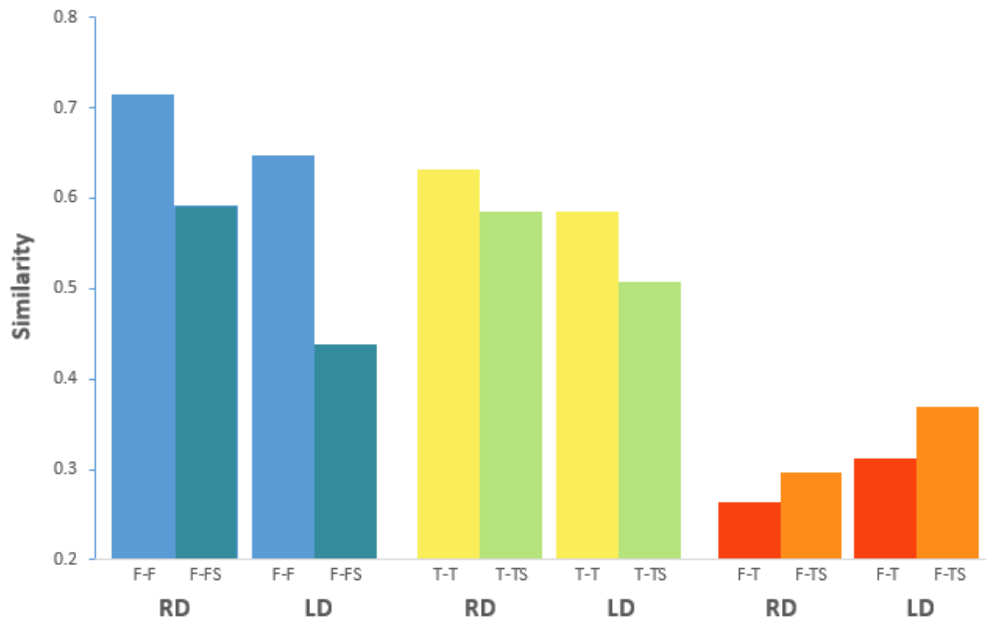


Figure 5.23: Comparisons of similarities between different types of behaviors.

The computed similarities between different experiments and simulation runs are presented in Table 5. The top part of the table lists the similarities for the Riot Domain (RD), while the bottom part of the table lists the similarities for the Looting Domain (LD). The first row (F) contains the results of comparisons between the fixed-policy simulation strategy and all other types of experiments. The highest similarity is obtained when the comparison is between the same type setup. The closest to this value the surrogate agent (fixed-policy surrogate - FS) can get, the more credibly it is able to reproduce the observed fixed-policy strategy. The similarities for the other two comparisons (trainee - T, and trainee surrogate - TS) are significantly lower, which shows that the strategy adopted by the trainee is noticeably different than the fixed-policy strategy. Similar results can be observed for the comparisons in the second row (trainee session).

The same pattern can also be observed for the LD, with the computed similarities slightly lower. This can be attributed to a more complex domain, where more actions and variations can be applied during the simulation/training session.

| Riot domain - RD    |       |       |       |       |
|---------------------|-------|-------|-------|-------|
|                     | F     | T     | FS    | TS    |
| F                   | 0.714 | 0.264 | 0.591 | 0.297 |
| T                   | —     | 0.632 | 0.391 | 0.586 |
| FS                  | —     | —     | 0.788 | 0.382 |
| TS                  | —     | —     | —     | 0.851 |
| Looting domain - LD |       |       |       |       |
|                     | F     | T     | FS    | TS    |
| F                   | 0.647 | 0.313 | 0.439 | 0.370 |
| T                   | —     | 0.586 | 0.298 | 0.508 |
| FS                  | —     | —     | 0.522 | 0.350 |
| TS                  | —     | —     | —     | 0.657 |

Table 5: Comparison of all similarities between different simulation runs.

The presented experiments confirm part of the main hypothesis and sub-hypothesis 1. In addition, it supports part of the first scientific contribution from Section 1.3.

## 5.5 Training Improvement

The last group of experiments were focused on measuring the overall benefit of the PATDS algorithm for the training sessions and how can the method help to identify weaknesses in the learnt strategies. Due to time constraints regarding the availability of operational trainees, we were able to evaluate PATDS on only one domain; we selected RD due to a better observed performance during the cloning experiments. The learning data were obtained from the previous training experiments. Behavior patterns were generated using HMASDA, filtered through clustering, and transferred to new surrogate agents in the cloning module. The surrogate agents were then used during data-farming experiments. Since the created surrogate agents perform reasonably similarly to the security members, as demonstrated in the previous section, the surrogate agents should provide a good indication of how the trainees would perform on a variety of different scenario settings. For humans, testing under all possible circumstances is infeasible, but for surrogate agents it is feasible. We evaluated the surrogate agents on approximately 15,000 different simulation runs with several parameter settings; for example, starting position, crowd number, and initial cognitive variables such as anger level, fear level, initial need level, frustration level, etc. The ten simulations with the highest damage values emerging during data-farming experiments are presented in the left-hand (blue) bars in Figure 5.24. The average damage of these ten simulations is 1252, while 5% of all simulations resulted in a damage higher than 1000.

Next, the human trainees were asked to repeat the training on the ten worst simulations, thus improving their skills in the circumstances where the surrogate agent performed the worst. The logs of newly obtained simulations were input to HMASDA, and the extracted patterns were added to the surrogate agent created in the first cycle. The data-farming experiments were run again, resulting in a significant improvement of the overall performance. First, none of the simulation runs scored damage higher than 1000, and the average damage value for the same ten worst simulations was 337. The right-hand (green) bars in Figure 5.24 represent the new damage levels for the same ten simulations that performed the worst during the first training cycle. Overall, the average damage over the complete data-farming simulation was reduced by over 31%, from 251 to 171. The reduction over the entire parameter space is statistically significant with a paired t-test with a 0.05 significance level. The main insight in terms of the semantic difference in behavior was that when a crowd is agitated in a specific way, the unit leader should immediately resort to more aggressive defensive actions.

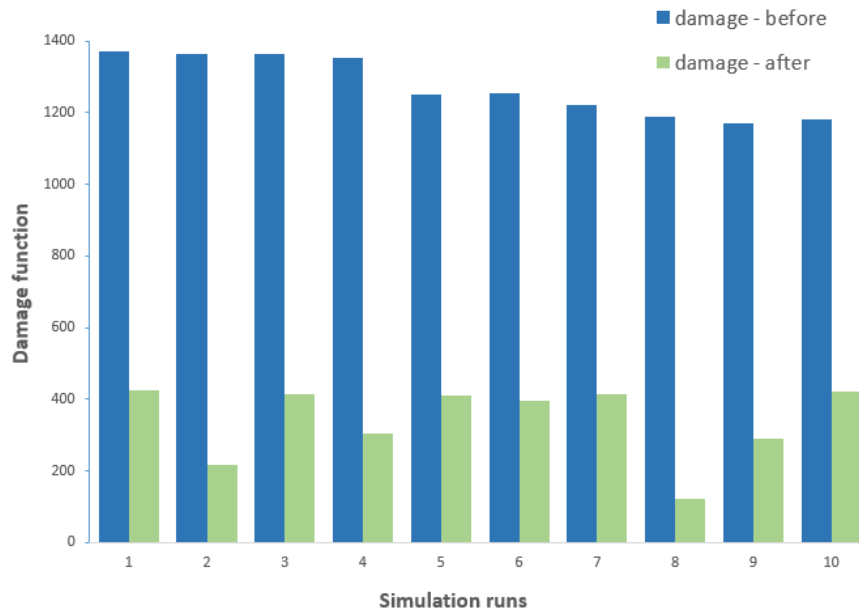


Figure 5.24: Comparisons of the 10 worst simulations in the first data-farming run with retraining in the second run.

Next, we wanted to verify whether the identified scenarios during data-farming, where the surrogate agents performed worst, would result in a similar outcome when human trainees would play them and if the strategy needs to be changed in order better cope with these scenarios. The main problematic scenarios were those with a higher initial arousal and anger level of the opponents. To this end, we analyzed the simulation logs that were created during the second PATDS cycle (retrain on the worst scenarios). We segmented the execution time of the simulation into 5 second intervals and counted the occurrence of all events associated to soldier agents for each segment. The plot for the first replayed scenario is presented in Figure 5.25.

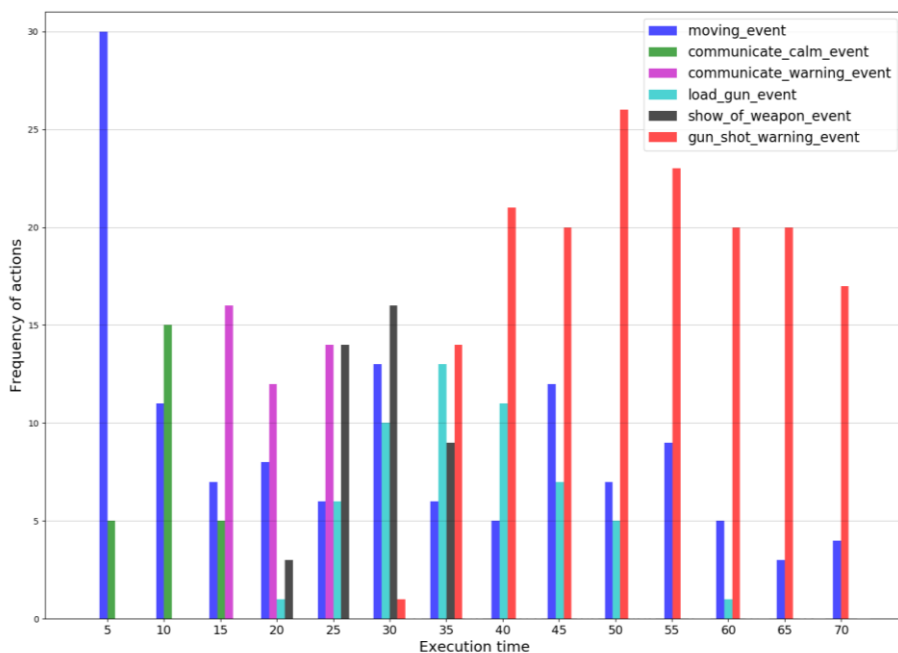


Figure 5.25: The distribution of different soldier actions for the initial retrain phase over the simulation execution time, segmented in 5s intervals.

The trainees tried to follow the standard course of actions that was adopted in the first training cycle. Their first interactions and responses were milder (mostly verbal interactions); however, due to the increasing escalations they quickly adapted and resorted to stricter actions (gun loading and gunshot warnings). Even though they promptly adapted to the new situations, the scenario did not end favorably, since the soldiers were repeatedly attacked and two soldiers were killed and one injured.

To verify whether the adopted strategy to cope with this scenario was changed, we also analyzed the last training of the second PATDS cycle. The distribution of actions is presented in Figure 5.26. It can be observed that there is a notable difference between the two distributions. The soldiers started to adopt stricter actions at the beginning of the simulations and the scenario concluded with less gun handling actions. We can attribute this to the fact that some of the insurgents fled, due to the change in their mental status, computed by the PACS model, which caused fear to prevail over anger. The analyzed scenario ended in a more favorable manner, compared to the previous, since only 1 soldier was injured and less attack actions occurred.

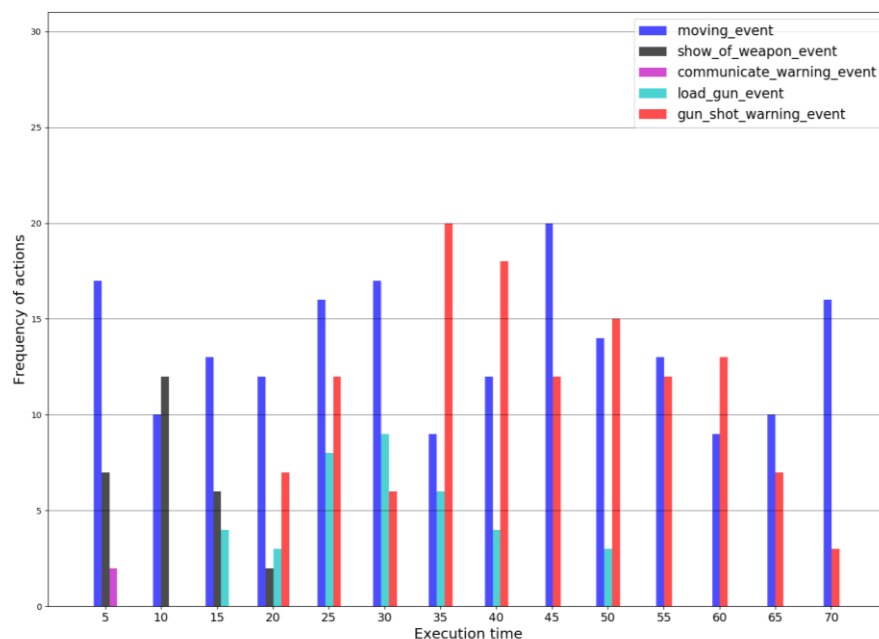


Figure 5.26: The distribution of different soldier actions for the last retrain phase over the simulation execution time, segmented in 5s intervals.

The presented experiments confirm the main hypothesis and sub-hypothesis 3; that is, the PATDS method supports the automatic verification of behavior strategies and enables improvements of strategies on the problematic scenarios. Moreover, they support part of the first and last scientific contribution from Section 1.3



## Chapter 6

# Conclusions

### 6.1 Summary and Discussion

In this thesis, we have presented a novel approach to facilitate the training of one or several trainees in simulated environments dealing with cooperating versus adversarial agents with cognitive properties. The main concepts of the procedure resemble the boosting methodology from the machine learning domain. In boosting, improvements are achieved by focusing on the areas where the system is not performing well and where the model needs further training. The difference between ML boosting and our approach is that, in the former, classifier models are being improved, while in the latter, human trainees improve their performance in each round of “boosting”.

The introduced PATDS algorithm is composed of several methods to enable the whole training cycle. First, the algorithm learns the specific strategic behavior of a trainee. It then transforms the learned physical and cognitive behavior patterns into a surrogate cognitive agent that is capable of reproducing the performance of the trainee; that is, the learned strategies of the trainee. The surrogate agent is then thoroughly tested through data farming to obtain a performance evaluation of the learned behavior strategies under all circumstances, thereby providing insights into weak parts of the strategies and enabling improvements in the next round of the training. The rationale of this approach is that humans can test only a limited amount of all possible scenarios, whereas the surrogate agent can test all scenarios due to the speed of data-farming simulations and their capability to work efficiently nonstop. The improvement of the approach is mainly related to the quality of the surrogate agent.

One of the key parts of PATDS is the HMASDA algorithm for the analysis of the behavior of trainees from simulation runs in serious games. The algorithm considers cognitive features of the behavior, which is important because interactions between two opposing groups of humans are largely based on psychological, cognitive, and emotional factors. HMASDA provides a way for experts to assess the physical and cognitive performance of the trainees in a graphical and symbolic form throughout the entire course of the simulation, and allows them to identify key strategic patterns they adopt for solving specific tasks. We demonstrated the application of the algorithm on the two security domains, where the behavior of security experts was captured in the form of meaningful graphical and symbolic patterns (decision rules), including psychological aspects.

We have demonstrated that the discovered behavior patterns can create a realistic surrogate agent that reliably reproduces the observed behavior of either fixed-policy agents or the complex behavior of human trainees in two domains. A significant similarity between the original simulation logs and the logs of cloned behavior was observed, while a clear differentiation between the various behaviors was also demonstrated. Moreover, the rule clustering method delivered an efficient pattern selection mechanism that improves the quality of cloned behavior, since it removes redundant and ambiguous information. Finally, PATDS in one domain identified key weaknesses in the behavior of trainees, enabling faster and more complete training.

The proposed approach can provide experts with helpful insights into the behavior and capabilities of the trainees; thus, specific and personalized training sessions can be designed to improve the identified weaknesses. Furthermore, comprehensive behavior verification through data farming and the necessary number of learning loops can provide a method to rank trainees

based on their performance and can help select squad members that learn fastest and whose expertise is best suited for certain tasks. In addition, the module for computing the similarity of behavior can reveal whether the learned strategies of the current trainee are similar to those of the previous trainees and experts. Experts can also devise semantic understanding as well as adequate rules of engagement based on data-farming observations. For both RD and LD, the semantic conclusion from the cloning and data-farming experiment was that the art of dealing successfully with hostile crowds is to capture the cognitive state of the crowd and act accordingly, sometimes by negotiating and sometimes by demonstrating determination.

The dissertation presents empirical evaluations to validate the main hypothesis and the three sub-hypotheses. The main hypothesis was demonstrated in Sections 5.4 and 5.5, where we confirmed that it is possible to construct surrogate agents that play similarly to the observed individuals and enable the identification of weaknesses in the learnt strategies. In Sections 5.2 and 5.4, we show that the discovered behavior patterns can credibly describe the strategic behavior of the observed individuals and can be used to create realistic surrogate agents, thus confirming sub-hypothesis 1. We demonstrate in Sections 5.3 and 5.4 that clustering improves the performance of the surrogate agent thus confirming sub-hypothesis 2. We tested the method for the assessment of the agent's performance and identification of weaknesses in Section 5.5. We showed that it is possible to identify problematic scenarios, identify weaknesses and provide a mechanism to improve the behavior of the trainees, thus confirming sub-hypothesis 3.

## 6.2 Original Contributions

**In addition to the scientific contributions** presented in Subsection 1.3, which represent the main contributions of the dissertation, the research work led to additional original contributions presented below. These are more technical contributions focusing on the design implementation of algorithms.

- **Definition, implementation and demonstration of a computer-assisted training procedure.** This is the main contribution of this thesis, and all other original contributions are introduced to support this primary notion. We propose a mechanism for training humans in domains where repetitive sessions, to acquire the appropriate level of expertise, are time-consuming or dangerous to perform in real-life conditions. The computer-assisted method takes care of the comprehensive behavior verification and identifies all weaknesses in the expertise of the trainees that can be appropriately addressed in the subsequent rounds of training. This sequence can be repeated several times until satisfactory performance is achieved or the trainees' expertise converges.
- **Designing problem definitions and formal descriptions of methods adopted in PATDS.** We provided new definitions that encompass all important aspects required for behavior analysis and identification of strategic behavior. These include data structures for behavioral traces, spatio-temporal envelopes to contain the relevant information about general behavior, and strategic patterns that capture the intention of the observed agents.
- **Design and implementation of the behavior analysis algorithms.** The algorithm was improved by introducing cognitive aspects and features to capture the interactions between groups of people. This was implemented through additional taxonomies that provide domain knowledge and mechanisms that exploit them in order to extract patterns efficiently. Moreover, we improved the time-efficiency of graph traversal, modified the abstraction process, and enriched the features for data mining.
- **Design of the pattern clustering mechanism.** The method takes into consideration spatial transitions, actions of pattern nodes, and the symbolic rules generated for each transition in order to create a complete subset of behavior patterns that cover a significant part of the displayed behavior of the trainees.

- **Contribution to behavior cloning and evaluation of behavior.** We designed and implemented a new comprehensive approach to improving training procedures based on a novel mechanism for behavior cloning and evaluation of behavior through data-farming experiments. Behavior cloning utilizes strategic behavior patterns obtained during behavior analysis. Cloning exploits both the graphical and symbolic descriptions to realistically reproduce the behavior of the observed agents. Evaluation is performed on a multitude of possible scenarios and conditions during data-farming experiments. Using the DTW method, we provided a means of comparing and evaluating the performance on single experiment runs.
- **Software framework in Java implementing the entire PATDS methodology.** We implemented a unified framework covering all aspects and algorithms needed to enable the PATDS cycle. See Appendix B for screenshots of the user interface.

### 6.3 Future Directions

The work and obtained insights presented in this thesis open up multiple directions for future work stemming from the PATDS algorithm. First of all, the behavior analysis algorithm might be further improved to capture more informative patterns. Defining additional features from data traces would provide richer pattern descriptions. Certain domains might require more user-tailored patterns, which could be constructed by refining the abstraction process. Next, introducing a more complex filtering algorithm for redundant and non-informative patterns could further improve the behavior cloning process. Similar improvements could be achieved with a novel algorithm for graphical patterns and IF-THEN rules clustering that is not based on TF-IDF weights similarity, instead on a novel mechanism for computing similarities between patterns. Evolutionary algorithms could be applied to identify the subset of patterns where the surrogate agent performs the best.

Future research directions also include various upgrades to the cloning process. A mechanism for generating new patterns, improving them by removing, adding, or modifying parts – rule attributes or values – could be developed. This is a suitable problem for evolutionary algorithms that can efficiently search the domain space and modify the population (set of rules) in order to optimize the performance on this domain. An interesting research problem, arising from the previous idea is to enable the surrogate agent to be able to learn on its own during data farming, thus providing improvements to the trainee's strategies in the form of transparent rules. The benefit of this would be that the surrogate agent would be able to theoretically find the most efficient strategies to cope with any scenarios or tasks in a complex spatio-temporal domain. The implications of this are immense; however, the biggest problem, as foreseen at this point, is the computational complexity of the simulation running as the evaluation function of the evolutionary algorithm. Moreover, a decision support system could be designed and deployed on tablets, that soldiers carry on during missions, which would provide recommendations based on the current situational descriptions of the scenario and the database of processed scenario runs, obtained during data-farming evaluations. The recommendations would consist of guidelines and instructions on how to cope in different situations that the soldier might encounter to optimize the final outcome.



## Appendix A

# HMASDA Additional Content

### A.1 List of Learning Features

This section provides a comprehensive list of features that are constructed from the AG and AAG for each domain and are used to describe the learning examples in the last step of the behavior analysis algorithm. The rule inducer then constructs the relevant rules using these features.

#### A.1.1 Riot Domain

| attributes               | description                                                                |
|--------------------------|----------------------------------------------------------------------------|
| anger                    | TRUE if the civilian is angry, FALSE otherwise                             |
| anger_level              | The numeric value of the anger parameter                                   |
| fear                     | TRUE if the civilian is afraid, FALSE otherwise                            |
| fear_level               | The numeric value of the fear parameter                                    |
| need                     | TRUE if the civilian is needy, FALSE otherwise                             |
| need_level               | The numeric value of the need parameter                                    |
| evaluation               | TRUE if the soldier is evaluating, FALSE otherwise                         |
| evaluation_level         | The numeric value of the evaluation parameter                              |
| standby                  | TRUE if the soldier is on standby, FALSE otherwise                         |
| energy_level             | The numeric value of the energy parameter                                  |
| average_civilian_anger   | The average value of anger parameter of the civilians close to the soldier |
| Moving                   | TRUE if agent is moving, FALSE otherwise                                   |
| CALC_anger_value         | The anger value of the civilian charismatic leader (CALC)                  |
| CALC_motive_value        | The value of the leading motive of the charismatic leader (CALC)           |
| NearProvokeSlightlyEvent | TRUE if CACL has provoked the soldier, FALSE otherwise                     |
| inSoldierArea            | TRUE if agent is located in soldier area, FALSE otherwise                  |

|                             |                                                                               |
|-----------------------------|-------------------------------------------------------------------------------|
| inSoldierAreaSPA            | TRUE if agent is located in soldier area (SPA), FALSE otherwise               |
| inFightAreaEntrance         | TRUE if agent is located close to the camp entrance, FALSE otherwise          |
| inFightArea                 | TRUE if agent is located in fight area, FALSE otherwise                       |
| inFightAreaInterest         | TRUE if agent is located in the fight area interest, FALSE otherwise          |
| inFightAreaInterestEntrance | TRUE if agent is located in the fight area interest entrance, FALSE otherwise |
| inTowerArea                 | TRUE if agent is located in the tower area, FALSE otherwise                   |
| inTowerFightArea            | TRUE if agent is located in the tower fight area, FALSE otherwise             |
| HasCACLNear                 | TRUE if the CACL is close to the soldier, FALSE otherwise                     |
| HasCommunicatedCalmEvent    | TRUE if the soldier has performed communicate calm event, FALSE otherwise     |
| HasGesticulateEvent         | TRUE if the soldier has performed gesticulate event, FALSE otherwise          |
| HasShowedWeapon             | TRUE if the soldier has performed show of weapon, FALSE otherwise             |
| HasLoadedGun                | TRUE if the soldier has performed load gun, FALSE otherwise                   |
| HasPerformedWarningShot     | TRUE if the soldier has performed warning shot, FALSE otherwise               |
| HasPerformedEffectiveShot   | TRUE if the soldier has performed effective shot, FALSE otherwise             |

### A.1.2 Looting Domain

| attributes             | description                                                                |
|------------------------|----------------------------------------------------------------------------|
| anger                  | TRUE if the civilian is angry, FALSE otherwise                             |
| anger_level            | The numeric value of the anger parameter                                   |
| fear                   | TRUE if the civilian is afraid, FALSE otherwise                            |
| fear_level             | The numeric value of the fear parameter                                    |
| need                   | TRUE if the civilian is needy, FALSE otherwise                             |
| need_level             | The numeric value of the need parameter                                    |
| evaluation             | TRUE if the soldier is evaluating, FALSE otherwise                         |
| evaluation_level       | The numeric value of the evaluation parameter                              |
| standby                | TRUE if the soldier is on standby, FALSE otherwise                         |
| energy_level           | The numeric value of the energy parameter                                  |
| average_civilian_anger | The average value of anger parameter of the civilians close to the soldier |
| Moving                 | TRUE if agent is moving, FALSE otherwise                                   |

|                                      |                                                                                      |
|--------------------------------------|--------------------------------------------------------------------------------------|
| inSaACivilianHouse                   | TRUE if agent is located in civilian house, FALSE otherwise                          |
| InsideShop                           | TRUE if agent is located inside shop, FALSE otherwise                                |
| inLootingViolenceProne               | TRUE if agent is located in looting violence area, FALSE otherwise                   |
| looting                              | TRUE if civilian agent is looting, FALSE otherwise                                   |
| LeftCorner                           | TRUE if civilian agent is located close to the left corner of shop, FALSE otherwise  |
| RightCorner                          | TRUE if civilian agent is located close to the right corner of shop, FALSE otherwise |
| Entrance                             | TRUE if civilian agent is located close to the entrance of shop, FALSE otherwise     |
| patrolling_motive_level              | The numeric value of the patrolling motive parameter                                 |
| IsApproachingToAngerCivilian         | TRUE if soldier agent is approaching an angry civilian, FALSE otherwise              |
| CiviliansAreLooting                  | TRUE if civilian agents are looting, FALSE otherwise                                 |
| soldier_is_far_from_shop             | TRUE if soldier agent is far away from shop, FALSE otherwise                         |
| soldier_is_medium_distance_from_shop | TRUE if soldier agent is at a medium distance from shop, FALSE otherwise             |
| soldier_is_close_to_shop             | TRUE if soldier agent is close to the shop, FALSE otherwise                          |
| AngerHigherThanFear                  | TRUE if the civilian agent motive anger is higher than fear, FALSE otherwise         |
| FearHigherThanAnger                  | TRUE if the civilian agent motive fear is higher than anger, FALSE otherwise         |
| AngerHigherThanLootingMotive         | TRUE if the civilian agent motive anger is higher than looting, FALSE otherwise      |
| FearHigherThanLootingMotive          | TRUE if the civilian agent motive fear is higher than looting, FALSE otherwise       |
| LootingMotiveHigherThanAnger         | TRUE if the civilian agent motive looting is higher than anger, FALSE otherwise      |
| LootingMotiveHigherThanFear          | TRUE if the civilian agent motive looting is higher than fear, FALSE otherwise       |

## A.2 Action and Abstract Action Graphs

### A.2.1 Riot Domain

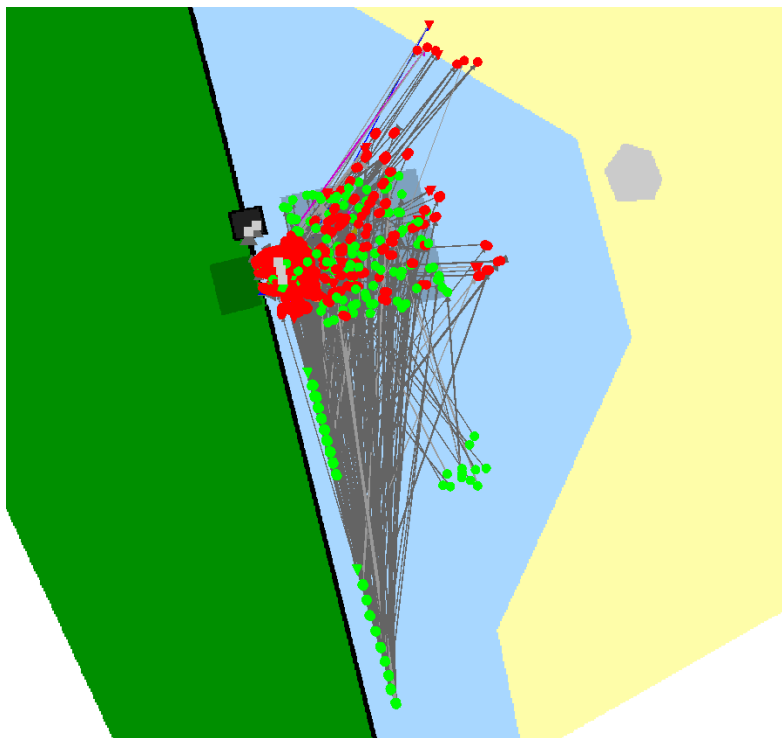


Figure A.1: The Action Graph for all agent traces in the riot domain.

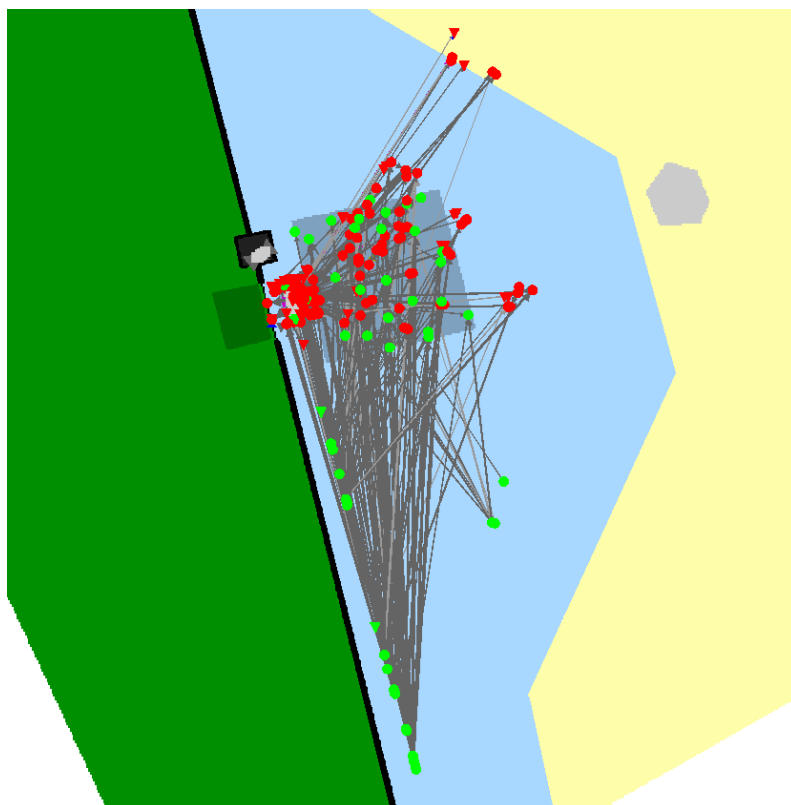


Figure A.2: Abstract Action Graph with abstraction level 3 for the riot domain.

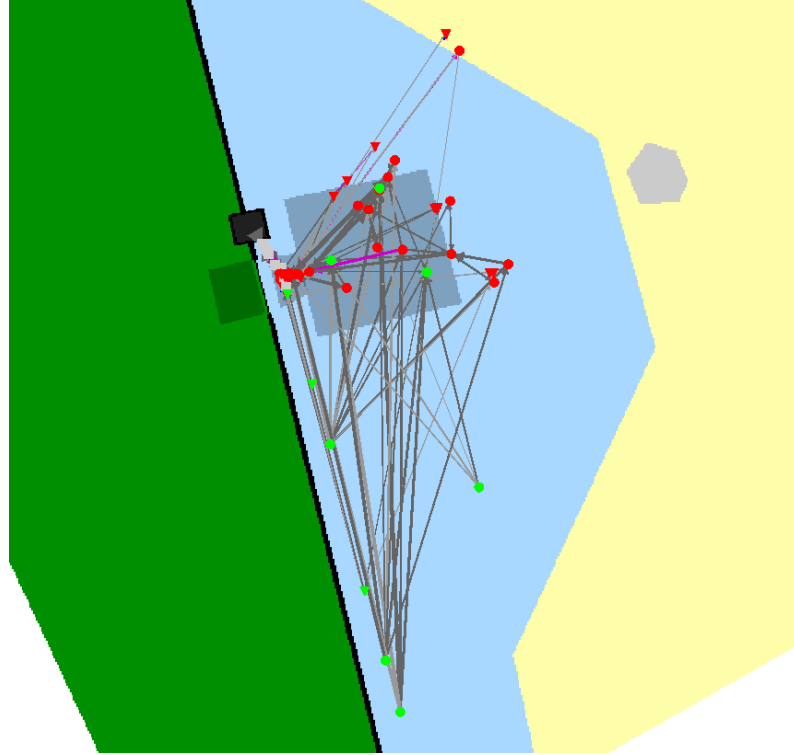


Figure A.3: Abstract Action Graph with abstraction level 5 for the riot domain.

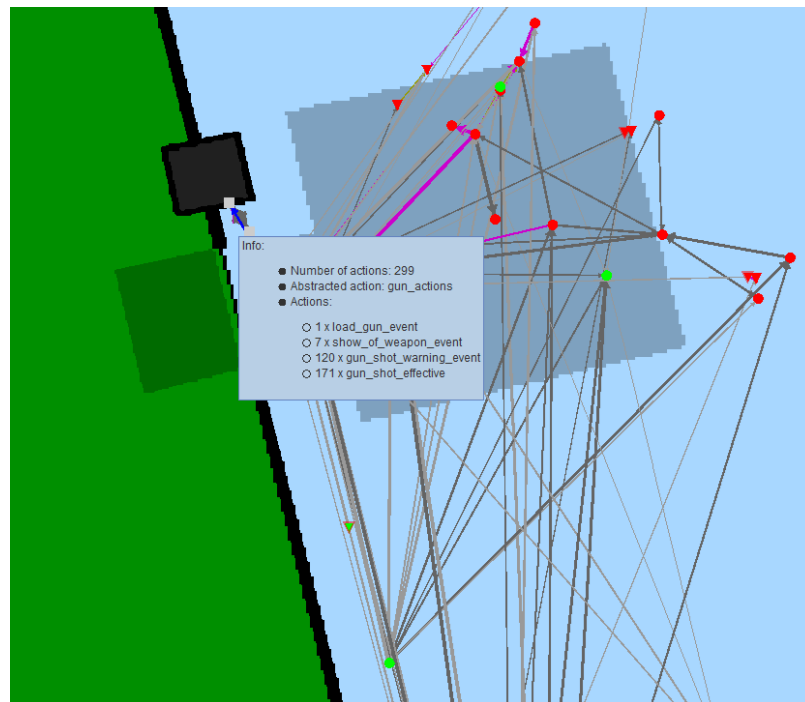


Figure A.4: Abstract Action Graph with abstraction level 8 and the visualization of information for one abstracted node.

## A.2.2 Looting Domain

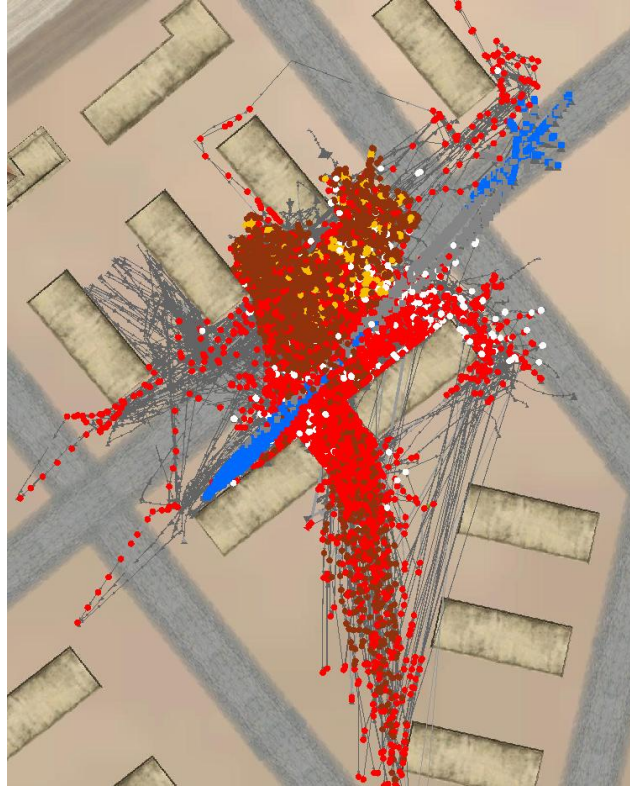


Figure A.5: Abstract Action Graph with abstraction level 1 for data collected from the fixed-policy strategy in the looting domain.

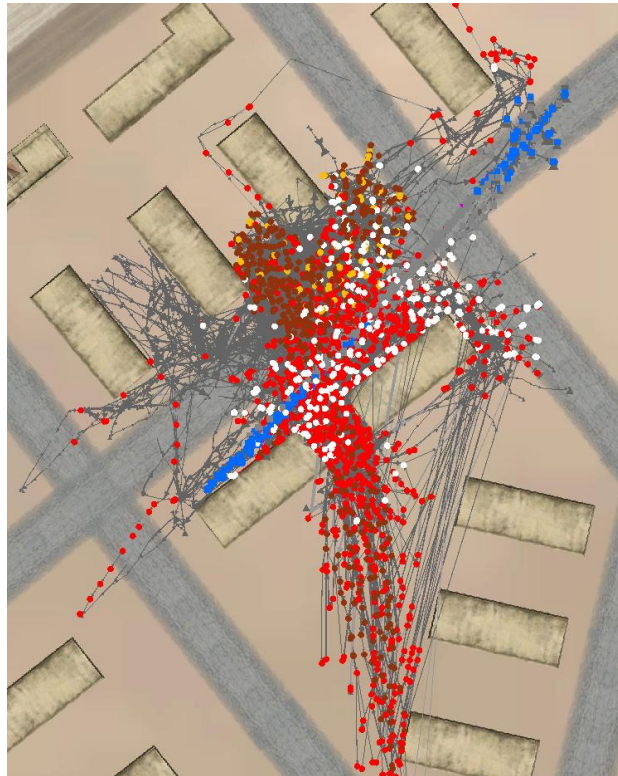


Figure A.6: Abstract Action Graph with abstraction level 2 for data collected from the fixed-policy strategy in the looting domain.

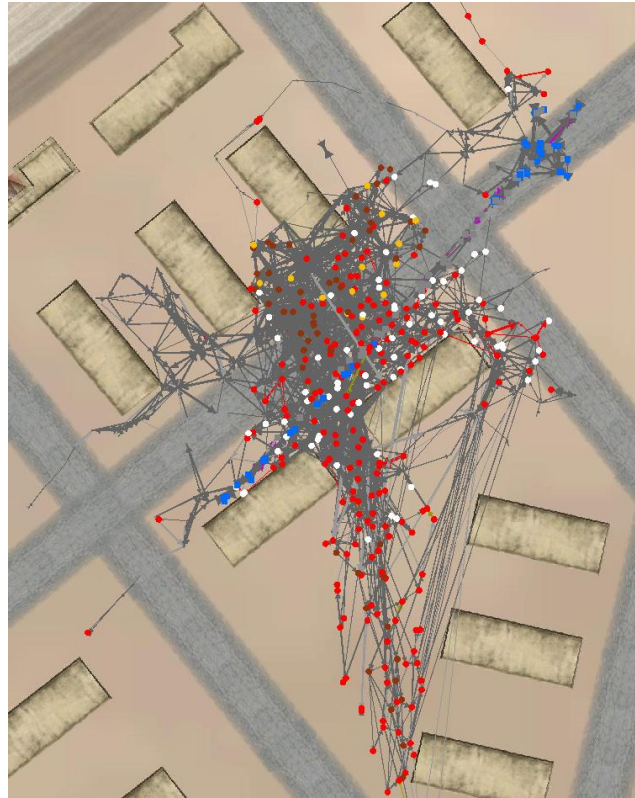


Figure A.7: Abstract Action Graph with abstraction level 4 for data collected from the fixed-policy strategy in the looting domain.

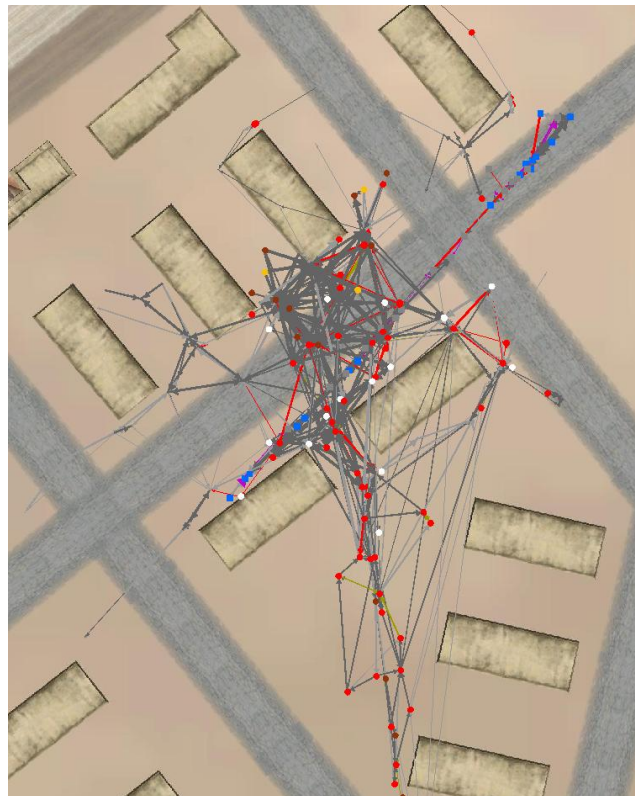


Figure A.8: Abstract Action Graph with abstraction level 6 for data collected from the fixed-policy strategy in the looting domain.



Figure A.9: Abstract Action Graph with abstraction level 1 for data collected during human training in the looting domain.

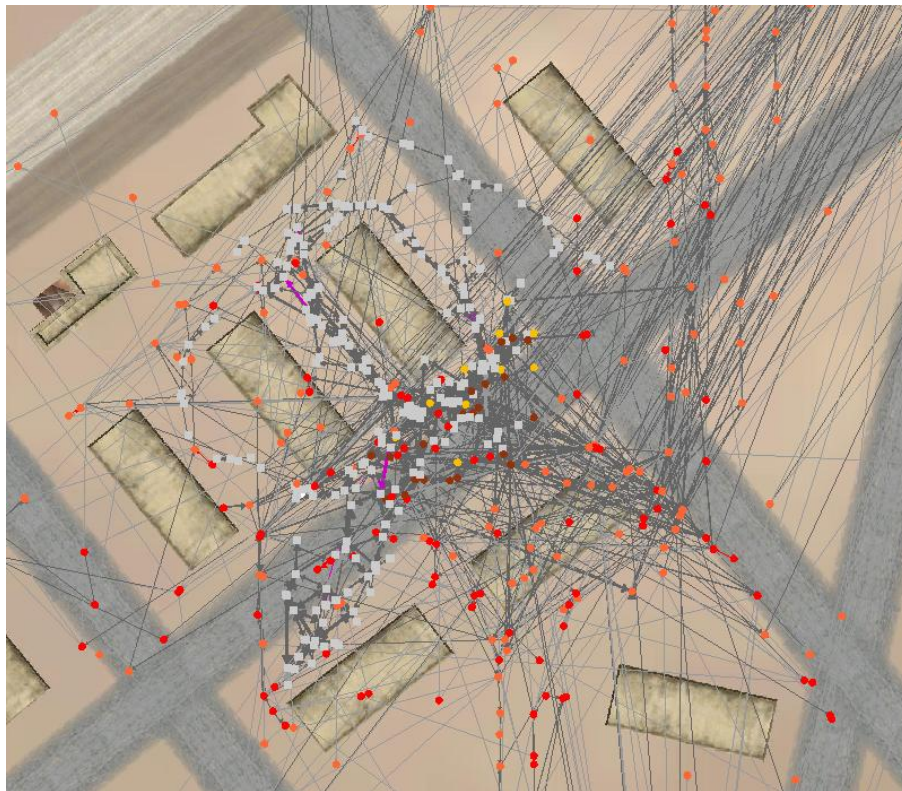


Figure A.10: Abstract Action Graph with abstraction level 3 for data collected during human training in the looting domain.

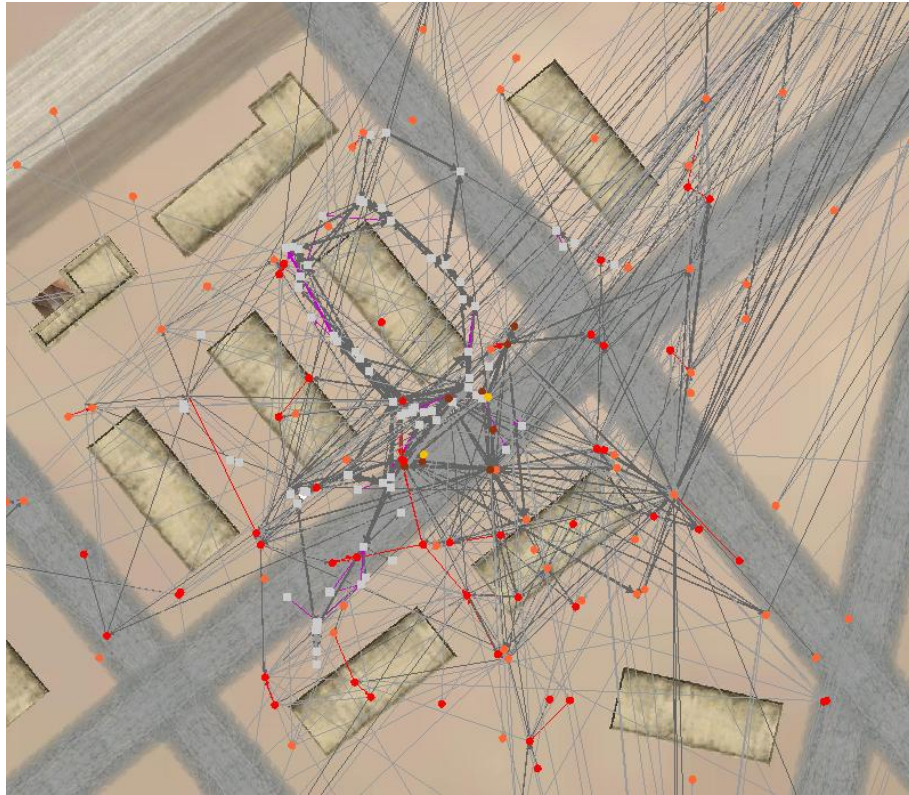


Figure A.11: Abstract Action Graph with abstraction level 6 for data collected during human training in the looting domain.

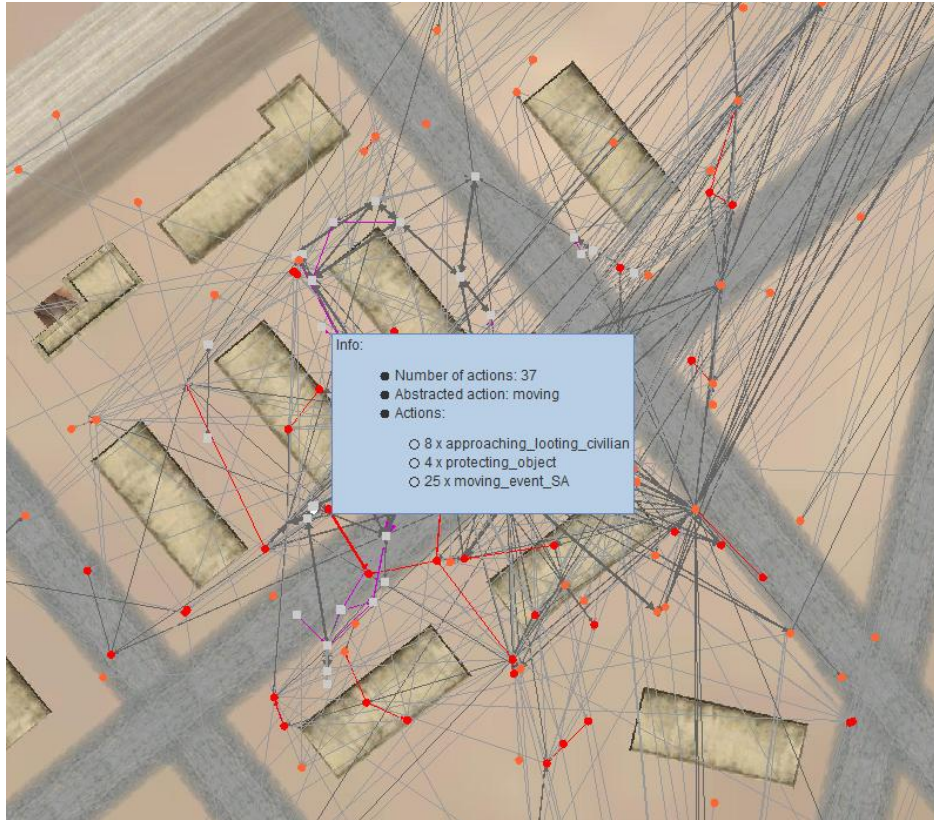


Figure A.12: Abstract Action Graph with abstraction level 7 for data collected during human training in the looting domain.

## Appendix B

# Software Mockups

We present here three screenshots of the user interface for the software framework in Java that we implemented and is an original contribution of this dissertation.

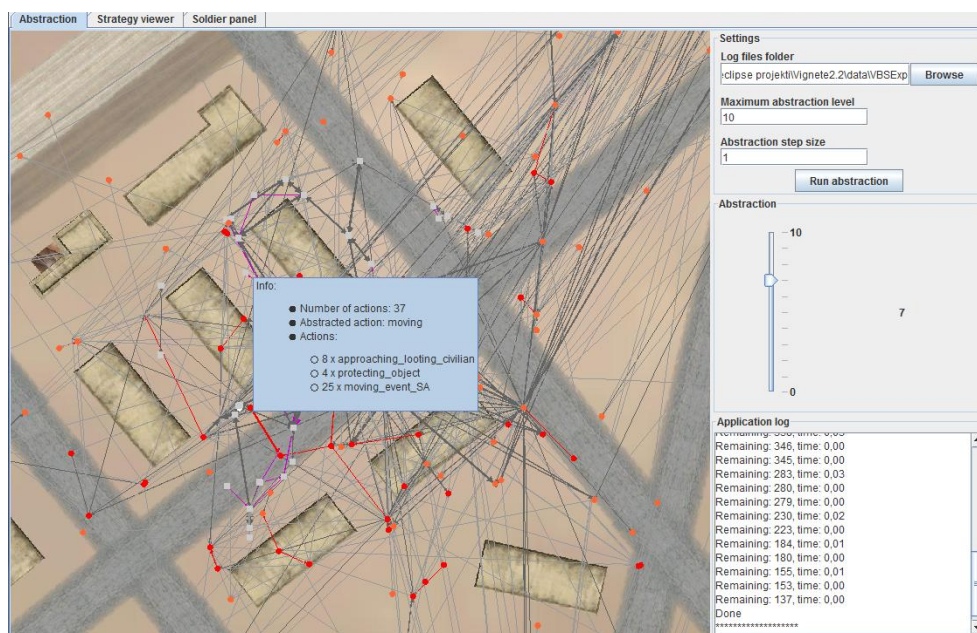


Figure B.1: Software mockup – Increasing abstraction and exploring AAG.

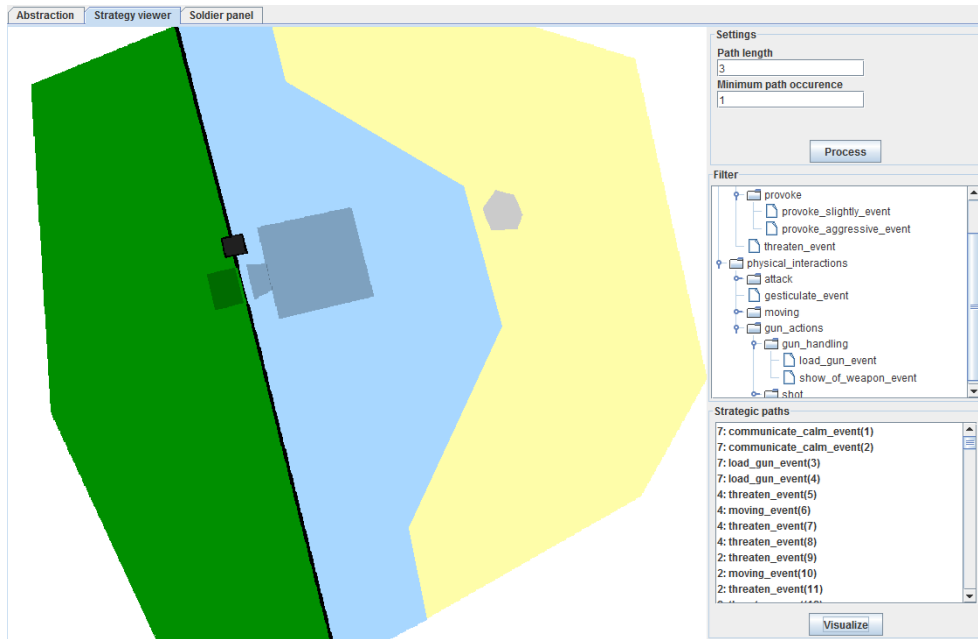


Figure B.2: Software mockup – Exploring and visualization of patterns.

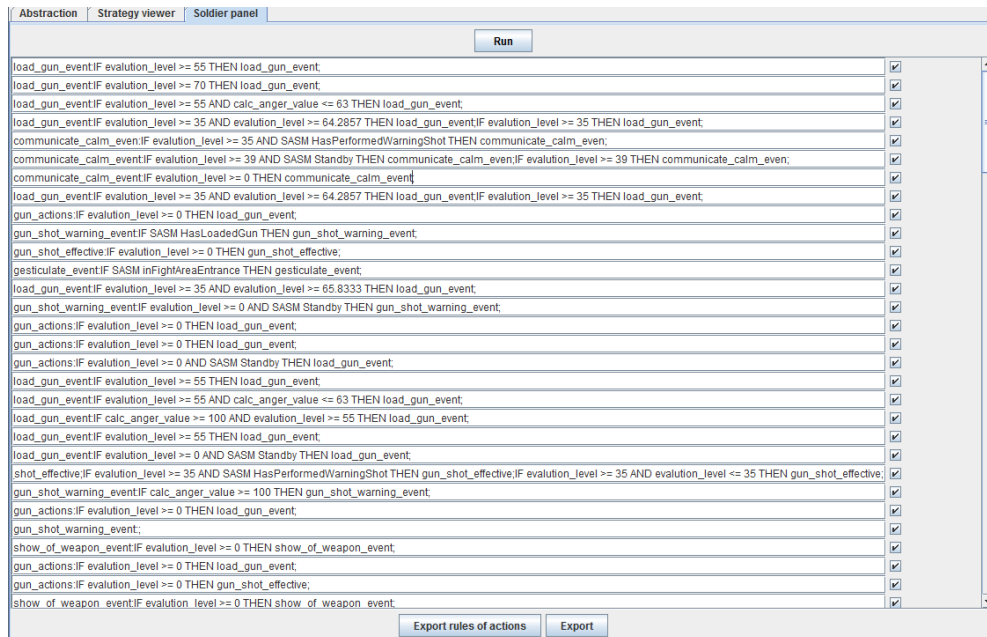


Figure B.3: Software mockup – Exporting behavior patterns.

## References

- [1] R. Davidrajuha and B. Lin, "Exploring airport traffic capability using Petri net based model," *Expert Systems with Applications*, vol. 38, no. 9, pp. 10923-10931, 2011.
- [2] I. Koglbauer, "Training for Prediction and Management of Complex and Dynamic Flight Situations," in *Procedia - Social and Behavioral Sciences*, 2015.
- [3] J. Kozuba and A. Bondaruk, "Flight simulator as an essential device supporting the process of shaping pilot's situational awareness," *Scientific Research & Education in the Air Force*, vol. 1, pp. 41-60, 2014.
- [4] X. Chen, J. Li and Q. Gao, "A simple process simulation model for strategic planning on the airside of an airport: A case stud," *Journal of Simulation*, vol. 9, no. 1, pp. 64-72, 2015.
- [5] A. Karadag, N. Caliskan, H. Korkut, Z. Baykara and D. Ozturk, "The Effect of Simulation Training on the Learning of some Psychomotor Skills by First Year Nursing Students: The Case of Turkey," in *Procedia - Social and Behavioral Sciences*, 2012.
- [6] V. Vattanavanit, J. Kawlaied and R. Bhurayanontachai, "High-fidelity medical simulation training improves medical students' knowledge and confidence levels in septic shock resuscitationy," *Open Access Emerg Med*, vol. 9, pp. 1-7, 2017.
- [7] F. Lateef, "Simulation-based learning: Just like the real thing," *Journal of Emergencies, Trauma and Shock*, vol. 3, no. 4, pp. 348-352, 2010.
- [8] A. Trivedi and M. Kollef, "Case Based Lectures Vs Simulation Based Training: Which Is Better for Resident Education?," *American Journal of Respiratory and Critical Care Medicine*, vol. 193, 2016.
- [9] E. Salas, J. Wildman and R. Piccolo, "Using Simulation-Based Training to Enhance Management Education," *Academy of Management Learning & Education*, vol. 8, no. 4, pp. 559-573, 2009.
- [10] F. Pasin and H. Giroux, "The impact of a simulation game on operations management education," *Computers & Education*, vol. 57, no. 1, pp. 1240-1254, 2011.
- [11] F. Ganier, C. Hoareau and J. Tisseau, "Evaluation of procedural learning transfer from a virtual environment to a real situation: a case study on tank maintenance training," *Ergonomics*, vol. 57, no. 6, pp. 828-843, 2014.
- [12] F. De Crescenzo, M. Fantini, F. Parsiani, L. Di Stefano, P. Azzari and S. Salti, "Augmented Reality for Aircraft Maintenance Training and Operations Support," *IEEE Computer Graphics and Applications*, vol. 31, no. 1, pp. 96-101, 2011.
- [13] N. Gavish, T. Gutierrez, S. Webel, J. Rodriguez, M. Peveri, U. Bockholt and F. Tecchia, "Evaluating virtual reality and augmented reality training for industrial maintenance and assembly tasks," *Interactive Learning Environments*, vol. 23, no. 6, pp. 778-798, 2015.
- [14] A. Doniec, R. Mandiau, S. Piechowiak and S. Espiec, "A behavioral multi-agent model for road traffic simulation," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 8, pp. 1443-1454, 2008.

- [15] Y. Wang, W. Zhang and G. Salvendy, "Effects of a Simulation-Based Training Intervention on Novice Drivers' Hazard Handling Performance," *Traffic Injury Prevention*, vol. 11, no. 1, pp. 16-24, 2010.
- [16] M. Sullman, L. Dorn and P. Niemi, "Eco-driving training of professional bus drivers -- Does it work?," *Transportation Research Part C: Emerging Technologies*, vol. 58, no. D, pp. 749-759, 2015.
- [17] P. van Leeuwen, R. Happee and J. de Winter, "Vertical field of view restriction in driver training: A simulator-based evaluation," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 24, pp. 169-182, 2014.
- [18] F. Fricoteaux, I. Thouvenin and D. Mestre, "GULLIVER: A decision-making system based on user observation for an adaptive training in informed virtual environments," *Engineering Applications of Artificial Intelligence*, vol. 33, pp. 47-57, 2014.
- [19] K. Hudson, L. Taylor, S. Kozachik, S. Shaefer and M. Wilson, "Second Life simulation as a strategy to enhance decision-making in diabetes care: a case study," *Journal of Clinical Nursing*, vol. 24, no. 5-6, pp. 797-804, 2015.
- [20] E. AbuKhoussa, J. Al-Jaroodi, S. Lazarova-Molnar and N. Mohamed, "Simulation and Modeling Efforts to Support Decision Making in Healthcare Supply Chain Management," *The Scientific World Journal*, vol. 2014, p. 16 pages, 2014.
- [21] D. Kallfass and T. Schlaak, "NATO MSG-088 case study results to demonstrate the benefit of using data farming for military decision support," in *Proceedings of the 2012 Winter Simulation Conference*, 2012.
- [22] D. van der Zee, B. Holkenborg and S. Robinson, "Conceptual modeling for simulation-based serious gaming," *Decision Support Systems*, vol. 54, no. 1, pp. 33-45, 2012.
- [23] D. Van der Zee and J. Slomp, "Simulation as tool for gaming and training in operations," *Journal of Simulation*, vol. 3, no. 1, pp. 17-28, 2009.
- [24] O. Missura and T. Gärtner, "Player Modeling for Intelligent Difficulty Adjustment," *Discovery Science*, vol. 5808, pp. 197-211, 2009.
- [25] M. Rosen, S. Weaver, E. Lazzara, E. Salas, T. Wu, S. Silvestri, N. Schiebel, S. Almeida and H. King, "Tools for evaluating team performance in simulation-based training," *Journal of Emergencies, Trauma and Shock*, vol. 3, no. 4, pp. 353-359, 2010.
- [26] L. Chittaro and R. Sioni, "Serious games for emergency preparedness: Evaluation of an interactive vs. a non-interactive simulation of a terror attack," *Computers in Human Behavior*, vol. 50, no. 1, pp. 508-519, 2015.
- [27] A. Tavcar, B. Kaluza, M. Kvassay, B. Schneider and M. Gams, "Surrogate-Agent Modeling for Improved Training," in *ECAI 2014* 263, 2014.
- [28] S. Zikos and H. Karatza, "Performance and energy aware cluster-level scheduling of compute-intensive jobs with unknown service times," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 239-250, 2011.
- [29] M. Gueriau, B. Dafflon and F. Gechterc, "VIPS: A simulator for platoon system evaluation," *Simulation Modelling Practice and Theory*, vol. 77, pp. 157-176, 2017.
- [30] B. Bell, A. Kanar and K. S.W.J, "Current Issues and Future Directions in Simulation-Based Training," *CAHRS Working Paper Series*, vol. WP08, no. 13, pp. 8-13, 2008.
- [31] D. Worley, A. Wahlman and D. Gleeson Jr, "Military Operations in Urban Terrain: A Survey of Journal Articles," *IDA Document D-2521, Institute for Defense Analyses*, 2000.
- [32] A. Tavcar, D. Kuznar and M. Gams, "Hybrid Multi-Agent Strategy Discovering Algorithm for human behavior," *Expert Systems with Applications: An International Journal*, vol. 71, no. C, pp. 370-382, 2017.

- [33] M. Bain and C. Sammut, "A framework for behavioral cloning," *Machine Intelligence*, vol. 15, pp. 103-129, 1995.
- [34] A. Tavcar and M. Gams, "Surrogate-agent modeling for improved training," *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 280-293, 2018.
- [35] M. Kvassay, L. Hluchy, S. Dlugolinski, M. Laclavik, B. Schneider, H. Bracker, A. Tavcar, M. Gams, D. Krol, M. Wrzeszcz and J. Kitowski, "An integrated approach to mission analysis and mission rehearsal," in *Proceedings of the Winter Simulation Conference*, 2012.
- [36] V. Mirchevska, A. Tavcar and M. Gams, "Behavioral cloning of asymmetric conflicts in urban environment using supervised learning," in *Proceedings of the 15th International Multiconference Information Society - IS 2012*, 2012.
- [37] M. Kvassay, L. Hluchy, S. Dlugolinsky, B. Schneider, H. Bracker, A. Tavcar, M. Gams, M. Contat, L. Dutka, D. Krol, M. Wrzeszcz and J. Kitowski, "A Novel Way of Using Simulations to Support Urban Security Operations," *Computing and Informatics*, vol. 34, no. 6, pp. 1201-1233, 2015.
- [38] A. Tavcar, "The use of agents in modeling human behavior," in *Proceedings of the 2nd Jožef Stefan International Postgraduate School Students Conference*, Ljubljana, 2010.
- [39] A. Tavcar, D. Kuznar and M. Gams, "Modeliranje večagentnih sistemov," in *Proceedings of the 3rd Jožef Stefan International Postgraduate School Students Conference*, Ljubljana, 2011.
- [40] A. Tavcar and M. Gams, "Analiza obnašanja v večagentnem sistemu," in *Proceedings of the 15th International Multiconference Information Society - IS 2012*, 2012.
- [41] A. Tavcar and M. Gams, "Hevristično odločanje," in *Proceedings of the 14th International Multiconference Information Society - IS 2011*, Ljubljana, 2011.
- [42] A. Tavcar, "Analysis of a single-agent search," *Informatica : an international journal of computing and informatics*, vol. 36, no. 2, pp. 177-183, 2012.
- [43] A. Tavcar, R. Piltaver, D. Zupancic, T. Sef and M. Gams, "Modeliranje navad uporabnikov pri vodenju pametnih hiš," in *Proceedings of the 16th International Multiconference Information Society - IS 2013*, 2013.
- [44] R. Kerkhoff, A. Tavcar and B. Kaluza, "Inferring models for subsystems based on real world traces," in *Proceedings of the 17th International Multiconference Information Society - IS 2014*, 2014.
- [45] M. Vengust, D. Mozina, N. Pusenjak, L. Zevnik, J. Sodnik, B. Kaluza and A. Tavcar, "NERVteh 4DOF motion car driving simulator," in *Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 2014.
- [46] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995.
- [47] M. Wooldridge and N. Jennings, "Intelligent Agents: Theory and Practice," *Knowledge Engineering Review*, vol. 10, pp. 115-152, 1995.
- [48] G. Sukthankar and K. Sycara, "Hypothesis Pruning and Ranking for Large Plan Recognition Problems," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-08)*, Chicago, Illinois, USA, 2008.
- [49] K. Sycara, "Multiagent Systems," *AI Magazine*, vol. 19, no. 2, 1998.
- [50] L. Panait and S. Luke, "Cooperative Multi-Agent Learning: The State of the Art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387-434, 2005.
- [51] P. Hoen, K. Tuyls, L. Panait, S. Luke and J. La Poutre, "An overview of cooperative and competitive multiagent learning," in *LAMAS'05 Proceedings of the First international conference on Learning and Adaption in Multi-Agent Systems*, Utrecht, The Netherlands, 2005.

- [52] R. Fathzadeh, V. Mokhtari, M. Kangavari and M. Mousakhani, "Opponent Provocation and Behavior Classification: A Machine Learning Approach," *RoboCup 2007: RoboCup 2007: Robot Soccer World Cup XI*, pp. 540-547, 2007.
- [53] C. Schmidt, N. Sridharan and J. Goodson, "The plan recognition problem: An intersection of psychology and artificial intelligence," *Artificial Intelligence*, vol. 11, pp. 45-83, 1978.
- [54] H. Kautz and J. Allen, "Generalized plan recognition," in *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI-86)*, 1986.
- [55] N. Lesh, C. Rich and C. Sidner, "Using Plan Recognition in Human-Computer Collaboration," in *Proceedings of the 7th International Conference on User Modeling*, 1998.
- [56] M. Tambe and P. Rosenbloom, "An Approach for Real-time, Dynamic Agent Tracking," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.
- [57] E. Dourado and C. Martinho, "I Mean It! Detecting User Intentions to Create Believable Behavior for Virtual Agents in Games," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2010.
- [58] B. Schneider, T. Lampe, G. Schwarz and S. S., "Calibration as Prerequisite for the Validation of the German Agent-Based Model PAX," in *Proceedings of the NATO RTO-MP-SAS-071 conference*, 2009.
- [59] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan and G. Balan, "MASON: A Multi-Agent Simulation Environment," *Simulation: Transactions of the society for Modeling and Simulation International*, vol. 82, no. 7, pp. 517-527, 2005.
- [60] C. Urban, "PECS: A reference model for the simulation of multiagent systems," in *Tools and techniques for social science simulation*, 2000, pp. 83-114.
- [61] N. R. Council, *Human Behavior in Military Contexts*, Washington, DC: The National Academies Press, 2008.
- [62] T. Lettmann, M. Baumann, M. Eberling and T. Kemmerich, "Modeling Agents and Agent Systems," in *Transactions on Computational Collective Intelligence, Volume 6910 of the series Lecture Notes in Computer Science*, 2011.
- [63] C. Erdogan and M. M. Veloso, "Action Selection via Learning Behavior Patterns in Multi-Robot Systems," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 2011.
- [64] K. Laviers and G. Sukthankar, "A real-time opponent modeling system for rush football," in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, 2011.
- [65] D. J. Stracuzzi, A. Fern, K. Ali, R. Hess, J. Pinto, N. Li, T. Konik and D. G. Shapiro, "Application of Transfer to American Football: From Observation of Raw Video to Control in a Simulated Environment," *AI Magazine*, vol. 32, no. 2, pp. 107-125, 2012.
- [66] L. C. Johnson and J. A. Gonzales, "Learning collaborative team behavior from observation," *Expert Systems with Applications*, vol. 41, no. 5, pp. 2316-2328, 2014.
- [67] G. A. Kaminka, P. D. V. and M. Tambe, "Monitoring Teams by Overhearing: A Multi-Agent Plan-Recognition Approach," *Journal Of Artificial Intelligence Research*, vol. 17, pp. 83-135, 2002.
- [68] D. Avrahami-Zilberbrand and G. A. Kaminka, "Fast and complete symbolic plan recognition," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2005.
- [69] C. W. Geib and R. P. Goldman, "A probabilistic plan recognition algorithm based on plan tree grammars," *Artificial Intelligence*, vol. 173, no. 11, pp. 1101-1132, 2009.

- [70] J. A. Iglesias, J. A. Fernandez, I. R. Villena, A. Ledezma and A. Sanchis, "The Winning Advantage: Using Opponent Models in Robot Soccer," *Intelligent Data Engineering and Automated Learning*, vol. 5788, pp. 485-493, 2009.
- [71] J. Brynielsson, "Using AI and games for decision support in command and control," *Decision Support Systems*, vol. 43, no. 4, pp. 1454-1463, 2007.
- [72] I. Cil and M. Mala, "A multi-agent architecture for modelling and simulation of small military unit combat in asymmetric warfare," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1331-1343, 2010.
- [73] N. C. Perry, M. W. Wigginsb, M. Childsc and F. G., "Can reduced processing decision support interfaces improve the decision-making of less-experienced incident commanders?," *Decision Support Systems*, vol. 52, no. 2, pp. 497-504, 2012.
- [74] C. S. Loh, Y. Sheng and L. I., "Predicting expert–novice performance as serious games analytics with objective-oriented and navigational action sequences," *Computers in Human Behavior*, vol. 49, no. 1, pp. 147-155, 2015.
- [75] H. Camacho-Colladosa and F. Liberatore, "A Decision Support System for predictive police patrolling," *Decision Support Systems*, vol. 75, no. 1, pp. 25-37, 2015.
- [76] B. Kaluža, G. A. Kaminka and M. Tambe, "Detection of suspicious behavior from a sparse set of multiagent interactions," in *Proceedings: AAMAS 2012, The 11th International Conference on Autonomous Agents and Multiagent Systems*, 2012.
- [77] B. Kaluža, G. A. Kaminka and M. Tambe, "Towards detection of suspicious behavior from multiple observations," in *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence and the Twenty-Third Innovative Applications of Artificial Intelligence*, 2011.
- [78] R. Arroyo, J. J. Yebes, L. M. Bergasa, I. G. Daza and J. Almazan, "Expert video-surveillance system for real-time detection of suspicious behaviors in shopping malls," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7991-8005, 2015.
- [79] P. Gregory and S. Cresswell, "Domain Model Acquisition in the Presence of Static Relations in the LOP System," in *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling*, 2015.
- [80] S. Cresswell and P. Gregory, "Generalised Domain Model Acquisition from Action Traces," in *International Conference on Automated Planning and Scheduling*, 2011.
- [81] O. Champalle, K. Sehaba and A. Mille, "Facilitate Sharing of Training Experience by Exploring Behavior Discovery in Trainees Traces," in *Design for Teaching and Learning in a Networked World, Volume 9307 of the series Lecture Notes in Computer Science*, 2015.
- [82] A. Bezek and M. Gams, "From Basic Agent Behaviour to Strategic Patterns in a Robotic Soccer Domain," *Informatica (Slovenia)*, vol. 29, no. 4, pp. 461-468, 2005.
- [83] A. Bezek, M. Gams and I. Bratko, "Multi-agent strategic modeling in a robotic soccer domain," in *Proceedings: AAMAS 2006*, 2006.
- [84] W. W. Cohen and S. Y., "A Simple, Fast, and Effective Rule Learner," in *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.
- [85] F. Wenfei, L. Jianzhong, M. Shuai, N. T., W. Yinghui and W. Yunpeng, "Graph Pattern Matching: From Intractable to Polynomial Time," in *Proceedings of the VLDB Endowment*, vol. 3,3 Issue 1-2, 2010.
- [86] G. Worm, M. van der Wees, J. de Winter, L. de Graaf, P. Wieringa and L. Rietveld, "Training and assessment with a faster than real-time simulation of a drinking water treatment plant," *Simulation Modelling Practice and Theory*, vol. 21, no. 1, pp. 52-64, 2012.
- [87] J. de Winter, R. Happee, M. Martens and N. Stanton, "Effects of adaptive cruise control and highly automated driving on workload and situation awareness: A review of the

- empirical evidence," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 27, pp. 196-217, 2014.
- [88] L. ZHongkun, Z. Qiang, Z. Haishan and S. Peitao, "The virtual simulation system for training and demonstrating the design of the head-end of spent nuclear fuel reprocessing," *Annals of Nuclear Energy*, vol. 108, pp. 310-315, 2017.
  - [89] M. Mortara, C. Catalano, F. Bellotti, G. Fiucci, M. Houry-Panchetti and P. Petridis, "Learning cultural heritage by serious games," *Journal of Cultural Heritage*, vol. 15, no. 3, pp. 318-325, 2014.
  - [90] P. Hegland, H. Aarlie, H. Stromme and G. Jamtvedt, "Simulation-based training for nurses: Systematic review and meta-analysis," *Nurse Education Today*, vol. 54, pp. 6-20, 2017.
  - [91] P. Pasquier, "Use of serious games to facilitate the predeployment training of Combat Life Savers," *Injury*, vol. 47, no. 6, pp. 1357-1358, 2016.
  - [92] I. Magarino and G. Navarro, "ATABS: A technique for automatically training agent-based simulators," *Simulation Modelling Practice and Theory*, vol. 66, pp. 174-192, 2016.
  - [93] G. Horne and S. Johnson, "Maneuver Warfare Science 2003," USCM Project Albert, 2003.
  - [94] T. Graves, R. Picard and S. Upton, "Improving rule bases for agent-based simulations," Los Alamos National Laboratory Report, LA-UR-00-2566, 2009.
  - [95] J. Schubert and R. Suzic, "Decision Support for Crowd Control: Using Genetic Algorithms with Simulation to Learn Control Strategies," in *IEEE Military Communications Conference*, 2007.
  - [96] T. M. Connolly, E. A. Boylea, E. MacArthura, T. Haineya and J. M. Boyle, "A systematic literature review of empirical evidence on computer games and serious games," *Computers and Education*, vol. 59, no. 2, pp. 661-686, 2012.
  - [97] H. Qudrat-Ullah, "Yes we can: Improving performance in dynamic tasks," *Decision Support Systems*, vol. 61, no. 1, pp. 23-33, 2014.
  - [98] F. Campuzano, T. Garcia-Valverde, E. Serrano and J. Botia, "Generation of human computational models with knowledge engineering," *Engineering Applications of Artificial Intelligence*, vol. 35, pp. 259-276, 2014.
  - [99] J. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations," in *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1*, 1967.
  - [100] P. Johnson-Laird, "Mental models and human reasoning," in *Proceedings of the National Academy of Science of the United States of America* 107 (43), 2010.
  - [101] T. Giorgino, "Computing and Visualizing Dynamic Time Warping Alignments in R: The DTW package," *Journal of Statistical Software*, vol. 31, no. 7, pp. 1-24, 2009.
  - [102] P. Rousseeuw, "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis," *Computational and Applied Mathematics*, vol. 20, pp. 53-65, 1987.
  - [103] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129-137, 1982.
  - [104] D. Arthur and S. Vassilvitski, "k-means++: the advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms -- SODA '07*, 2007.
  - [105] D. Pelleg and L. Moore, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters," in *Proceedings of the Seventeenth International Conference on Machine Learning - ICML '00*, 2000.

- [106] A. Tavcar, V. Mirchevska and M. Gams, "EUSAS : European urban simulation for asymmetric scenarios," in *Proceedings of the 13th International Multiconference Information Society - IS 2010*, Ljubljana, 2010.



# Bibliography

## Publications Related to the Thesis

### Journal Articles

- A. Tavcar, D. Kuznar and M. Gams, "Hybrid Multi-Agent Strategy Discovering Algorithm for human behavior", *Expert Systems with Applications: An International Journal*, vol. 71, no. C, pp. 370-382, 2017.
- A. Tavcar and M. Gams, "Surrogate-Agent Modeling for Improved Training", *International Scientific Journal Engineering Applications of Artificial Intelligence*, 74, pp. 280-293, 2018.
- M. Kvassay, L. Hluchy, S. Dlugolinsky, B. Schneider, H. Bracker, A. Tavcar, M. Gams, M. Contat, L. Dutka, D. Krol, M. Wrzeszcz and J. Kitowski, "A Novel Way of Using Simulations to Support Urban Security Operations," *Computing and Informatics*, vol. 34, no. 6, pp. 1201-1233, 2015.
- A. Tavcar, "Analysis of a single-agent search," *Informatica : an international journal of computing and informatics*, vol. 36, no. 2, pp. 177-183, 2012.

### Conference Papers

- A. Tavcar, B. Kaluza, M. Kvassay, B. Schneider and M. Gams, "Surrogate-Agent Modeling for Improved Training," in *ECAI 2014* 263, 2014.
- A. Tavcar, V. Mirchevska and M. Gams, "EUSAS : European urban simulation for asymmetric scenarios," in *Proceedings of the 13th International Multiconference Information Society - IS 2010*, Ljubljana, 2010.
- M. Kvassay, L. Hluchy, S. Dlugolinski, M. Laclavik, B. Schneider, H. Bracker, A. Tavcar, M. Gams, D. Krol, M. Wrzeszcz and J. Kitowski, "An integrated approach to mission analysis and mission rehearsal," in *Proceedings of the Winter Simulation Conference*, 2012.
- A. Tavcar, "The use of agents in modeling human behavior," in *Proceedings of the 2nd Jožef Stefan International Postgraduate School Students Conference*, Ljubljana, 2010.
- A. Tavcar, D. Kuznar and M. Gams, "Modeliranje večagentnih sistemov," in *Proceedings of the 3rd Jožef Stefan International Postgraduate School Students Conference*, Ljubljana, 2011.
- A. Tavcar and M. Gams, "Analiza obnašanja v večagentnem sistemu," in *Proceedings of the 15th International Multiconference Information Society - IS 2012*, 2012.
- A. Tavcar and M. Gams, "Hevristično odločanje," in *Proceedings of the 14th International Multiconference Information Society - IS 2011*, Ljubljana, 2011.
- V. Mirchevska, A. Tavcar and M. Gams, "Behavioral cloning of asymmetric conflicts in urban environment using supervised learning," in *Proceedings of the 15th International Multiconference Information Society - IS 2012*, 2012.

## Other Publications

- A. Tavčar, C. Antonya, V. Butila, "Recommender system for virtual assistant supported museum tours", *Informatica : an international journal of computing and informatics*, vol 40, no. 3, pp. 279-284, 2016.
- D. Kužnar, A. Tavčar, J. Zupančič, M. Duguleana, "Virtual assistant platform". *Informatica : an international journal of computing and informatics*, vol. 40, no. 3, pp. 285-289, 2016.
- D. Fišer, A. Tavčar, "Več glav več ve : uporaba množičenja za čiščenje sloWNeta", *Družbena funkcijskost jezika : (vidiki, merila, opredelitve)*, (Obdobja, ISSN 1408-211X, 32), Ljubljana: Znanstvena založba Filozofske fakultete, pp. 125-132, 2013.
- R. Kerkhoff, A. Tavcar and B. Kaluza, "Inferring models for subsystems based on real world traces", in *Proceedings of the 17th International Multiconference Information Society - IS 2014*, 2014.
- J. Zupančič, G. Grasselli, A. Tavčar, M. Gams, "Virtual assistants for the Austrian-Slovenian Intelligent Tourist-Information Center", in *proceedings of the 20th International Multiconference Information Society - IS 2017*, pp. 27-30, 2017.
- T. Šef, A. Tavčar, M. Mlakar, M. Gams, "Spletno svetovanje študentom v stiskah", in *proceedings of the 20th International Multiconference Information Society - IS 2017*, pp. 49-52, 2017.
- M. Vengust, D. Možina, L. Zevnik, B. KALUŽA, A. Tavčar, N. Pušenjak, J. Sodnik, "NERVteh 4DOF motion car driving simulator", in *6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, in cooperation with ACM SIGCHI Special Interest Group on Computer-Human Interaction, Seattle, USA, pp. 1-6, 2014.
- D. Fišer, A. Tavčar, T. Erjavec, "sloWCrowd : a crowdsourcing tool for lexicographic tasks", in *LREC 2014 : proceedings, Ninth International Conference on Language Resources and Evaluation*, 2014.
- A. Tavčar, D. Fišer, T. Erjavec, "sloWCrowd: orodje za popravljanje wordneta z izkoriščanjem moči množic", in *Proceedings of the Eighth Language Technologies Conference, proceedings of the 15th International Multiconference Information Society - IS 2012*, pp. 197-202, 2012.
- A. Tavčar, J. Šorn, T. Tušar, T. Šef, M. Gams, "Arhitektura sistema OpUS", in *proceedings of the 17th International Multiconference Information Society - IS 2014*, pp. 93-96, 2014.
- A. Tavčar, C. Antonya, E. Butilă, M. Gams, "Providing recommendations for virtual museum tours using intelligent assistants", in *Workshop e-Heritage : proceedings of the 19th International Multiconference Information Society - IS 2016*, pp. 24-26, 2016.
- A. Tavčar, J. Šorn, M. Gams, "Platforma za pametna mesta", in *Workshop Electronic and Mobile Health : proceedings of the 19th International Multiconference Information Society - IS 2016*, pp. 84-86, 2016.
- A. Tavčar, C. Antonya, G. B. Răzvan, L. N. Jovan, M. Gams, "Computer-generated knowledge base for virtual assistants" in *Workshop e-Heritage : proceedings of the 19th International Multiconference Information Society - IS 2016*, pp. 31-32, 2016.

## Biography

Aleš Tavčar was born in Šempeter pri Gorici in 1984 and he spent his childhood in a small village Ivanji Grad. He attended the primary school Komen where he obtained several awards in national competitions in mathematics, history, logic, and practical technical tasks. After finishing primary school in 1998 he attended the Šolski center Srečka Kosovela grammar school in Sežana from 1999 until 2003.

He received his Diploma from the Faculty of Computer and Information Science, University of Ljubljana, in 2009. His thesis covered a comprehensive analysis of the occurrence and intensity of pathology in single agent games using Minimin algorithm, by creating synthetic OR trees and observing the pathology effect under various search conditions.

In 2009, he enrolled in the Information and Communication Technologies program at the Jožef Stefan International Postgraduate School. His research work was focused in the field of behavior analysis of individuals and groups and accurate reproductions of the observed behavior. The research work was performed under the supervision of Prof. Dr. Matjaž Gams.

He started working at the Department of Intelligent Systems, Jožef Stefan Institute, as a student in 2007, where he later obtained a student grant. Since 2009, he is an assistant researcher at the department, where, for the last three years, he is the head of the Agents group. His main field of research covers agent technologies, human behavior analysis using advanced machine learning techniques and the development of agent-based solutions. He has 65 entries in COBISS. He was involved in European projects dealing with the detection and recognition of strategic behavior from basic movement of crowds and individuals. Moreover, he led national and industry projects focusing on dynamic control algorithms for smart homes using optimization and data analysis methods, behavior identification and optimization of city subsystems, intelligent web-based services, and development of intelligent information retrieval and recommendation systems for the cultural heritage domain.