Computing and Informatics, Vol. 33, 2014, 79–108

DISCOVERING STRATEGIC BEHAVIOUR OF MULTI-AGENT SYSTEMS IN ADVERSARY SETTINGS

Violeta Mirchevska

Result d.o.o., Celovška cesta 182
1000 Ljubljana, Slovenia
&
Jožef Stefan International Postgraduate School, Jamova 39
1000 Ljubljana, Slovenia
e-mail: violeta.mircevska@ijs.si

Mitja Luštrek

Jožef Stefan Institute, Jamova 39
1000 Ljubljana, Slovenia

Andraž Bežek

Jožef Stefan Institute, Jamova 39
1000 Ljubljana, Slovenia
&
Marg d.o.o., Tržaška cesta 515
1351 Brezovica pri Ljubljani, Slovenia

Matjaž Gams

Jožef Stefan Institute, Jamova 39
1000 Ljubljana, Slovenia
&
Jožef Stefan International Postgraduate School, Jamova 39
1000 Ljubljana, Slovenia

Abstract. Can specific behaviour strategies be induced from low-level observations of two adversary groups of agents with limited domain knowledge? This paper presents a domain-independent Multi-Agent Strategy Discovering Algorithm (MASDA), which discovers strategic behaviour patterns of a group of agents under the described conditions. The algorithm represents the observed multi-agent activity as a graph, where graph connections correspond to performed actions and graph nodes correspond to environment states at action starts. Based on such data representation, the algorithm applies hierarchical clustering and rule induction to extract and describe strategic behaviour. The discovered strategic behaviour is represented visually as graph paths and symbolically as rules. MASDA was evaluated on RoboCup. Both soccer experts and quantitative evaluation confirmed the relevance of the discovered behaviour patterns.

Keywords: Agent modelling, strategy discovery, behaviour analysis, multi-agent system, RoboCup

Mathematics Subject Classification 2010: 68T05

1 INTRODUCTION

Understanding agent behaviour is beneficial for many applications. First, it allows agent's behaviour to adapt to the behaviour of the agents with which it interacts. Knowing the plans of other agents, agents can better plan their actions, whether in competitive or cooperative settings [1, 2, 3]. Second, it allows users to understand and study agents' behaviour in a multi-agent system of interest, determining its strengths and weaknesses [4]. This can be used to improve the behaviour of the system of interest. Finally, understanding agents' behaviour can be observed as a step towards behaviour cloning [5, 6, 7]. It allows to reproduce the observed behaviour in virtual worlds and serious games, thus enabling user training, testing the behavioural performance of the multi-agent system and searching for an optimal strategy.

We present the Multi-Agent Strategy Discovering Algorithm (MASDA), an algorithm for discovering strategic patterns from raw multi-agent action data. MASDA represents the observed multi-agent activity as a graph, where graph connections correspond to performed actions and graph nodes correspond to environment states at action starts. Each action is described with a set of features, including the place where it was performed and the role of the agent that performed it. Feature taxonomies and low-level pre-processing routines for identifying performed actions in game traces are the only required domain knowledge. Hierarchical clustering is applied to merge instances of similar actions. Recurring activity is thus represented by single graph paths. Graph paths with the highest number of merged actions are labelled as strategic behaviour, because actions that frequently occur near in the domain

space define important strategic concepts. MASDA presents the discovered strategic behaviour visually, depicting the extracted graph paths, and symbolically, using rules that describe characteristic patterns of the observed multi-agent interaction.

MASDA was applied on the RoboCup Simulation League [8]. RoboCup is an international scientific initiative aiming to advance the state-of-the-art intelligent robots. One of its oldest leagues is the RoboCup Simulation League, where soccer teams comprised of software players (agents) play on a virtual field. Our goal was to extract team strategies when attacking the goal. MASDA discovered strategic behaviour patterns that experts confirmed as relevant. We also used the discovered strategic patterns to detect strategic activity on-line. Tests showed reliable performance.

The paper is organised as follows. Section 2 presents related work in agent modelling and behaviour analysis. Section 3 describes the MASDA algorithm. The presentation is accompanied by examples from the RoboCup domain. Section 4 evaluates the algorithm in RoboCup. Section 5 concludes the paper and presents future work.

2 RELATED WORK

2.1 Agent Modelling

Lettmann et al. [9] present a basic, formal model of agents as a universal description of their properties, unifying existing work on the topic [10, 11, 12]. Agents act in an environment abstracted as a state transition system. Based on sensor input, they determine environment state using a vision function that considers sensor noise. The central concept of the model is the agent's mental state. The mental state encompasses all concepts relevant to the agent's decision making: the agent's internal state, its sensed environment state, cognition function (defines the agent's internal state based on its previous internal state and the sensed environment state), policy function (defines the action to be executed according to the agent's internal state) and internal state transition function (defines the agent's successive internal state based on its current internal state and executed action). We accept the proposed agent model with two extensions. First, the agent role is added as a parameter that influences the agent's behaviour. Second, we extend the definition of actions.

In the agent model, Lettmann et al. specify the policy function as an interface to the agent's behaviour model without specifying its implementation. Several agent behaviour model implementations can be found in the literature, including the BDI architecture [13], the PESC reference model [14], and layered architectures [15]. Because the MASDA algorithm is based on layered architecture, we describe only that architecture in detail below.

Stone [15] introduces layered behaviour architectures, i.e., decomposing the problem on several hierarchical behavioural levels. He argues that the activity patterns of agents in a limited communication, real-time, noisy environment with both teammates and adversaries, as in the RoboCup domain, are too complex to be

addressed by a simple mapping from agent's sensors to their actuators. Research in human cognition also argues for the existence of several hierarchical behaviour levels. Hollnagel [16] developed the Contextual Control Model (COCOM), which decomposes human behaviour in four control modes: strategic (based on long-term planning), tactical (based on procedures), opportunistic (based on present context), and scrambled (random). Based on experience with imitation learning in First-Shooter-Person games and the widely accepted COCOM model, Thurau et al. [7] propose a three-level model for imitating human player behaviour using strategic (long-term goals), tactical (smart localised behaviours, anticipation) and reactive levels (basic movement, aiming and similar). The MASDA algorithm extracts strategic behaviour from low-level game traces using layered behaviour architecture with four hierarchical levels.

2.2 Extracting Behaviour Patterns from Low-Level Agent Behaviour Observations

ISAAC [4] is one of the first systems for analysing teams participating in RoboCup. It enables automated post-hoc, off-line agent-team analysis based on game traces. ISAAC analyses team behaviour at three different granularity levels: individual agent actions, inter-agent interactions and global team behaviour. Analysing an individual agent's actions uses the C5.0 decision tree inductive learning algorithm, an extension to C4.5 [17], to create rules for successful shots on goal. To analyse inter-agent interactions, pre-defined patterns are matched to find prevalent patterns of shots on goal. Finally, to develop rules for team successes or failures, game-level statistics are mined from all available previous games, and inductive learning is again used to determine the causes of a team's success or failure. MASDA operates at the granularity level of inter-agent interactions, but it extracts strategic interactions from observations of low-level agent activity without using libraries with predefined behaviour patterns.

Ramos et al. [18] present an approach for discovering tactical plays (e.g., offensive plays) adopted by soccer-agents during a match within the context of formations. Formations are represented by planar topological graphs. Tactical plays are presented by the path of the ball occurring in a particular context. The planar topological graph enriches this information with information about the players participating in a particular tactical play and the zones in which the play has occurred. MASDA extracts a team's characteristic plays as sequences of performed agent actions (e.g., pass, dribble, attack support, intercept), not as the typical path of the ball.

Kaminka et al. [3] and Horman et al. [2] present an approach for learning sequential coordinated team behaviours. Their method translates observations of a dynamic environment into a time series of recognised basic actions. A trie (a prefix tree), which compactly represents the time series of recognised basic actions, is created. The trie is then analysed to find repeating subsequences characterising each team. Iglesias et al. [1] also effectively use tries to store team behaviour sequences.

MASDA represents multi-agent activity as graphs. This representation allows it to extract coordinated team behaviour, considering both the sequence of performed actions and the situation in which they were performed (e.g., the role of the agent that performed the action, the place where it was performed).

Aler et al. [5] present an approach for determining reactive behaviour in robotic soccer. They aimed to program RoboCup agents by imitating the actions of humans playing soccer. Humans were allowed to control RoboCup agents in a virtual soccer environment, thus collecting data about how humans act in soccer based on the current situation. In particular, they examined the conditions under which human players perform the following five actions: turn, run slow, run fast, kick ball and kick to goal. The conditions were represented by information concerning the agent's position, the agent's relative position to the ball, the two closest opponents and the opponents' goal. Reactive behaviour was represented by rules induced on the data obtained from humans controlling soccer agents. MASDA learns not only the conditions under which individual actions are performed but also strategic behaviour patterns and the conditions when strategic patterns are started.

Thurau et al. [19] present an approach for learning strategic behaviour in 3D gaming worlds, which contains two steps. First, a topological representation of the 3D gaming world is learned by applying the Neural Gas Algorithm to find the positions that a human player holds during a match. Artificial potential fields are then placed into the topological map to guide the game-bots. Their work focuses on behaviour cloning, not the understandability of the created model. MASDA creates human-understandable models that allow studying observed multi-agent interactions at different abstraction levels.

MASDA was published at the AAMAS conference [20]. This paper extends the work presented in the AAMAS publication, formalising agent modelling and presenting both a detailed description of the algorithm with symbolic description-generation supplements and test results on a new dataset from RoboCup.

3 MULTI-AGENT STRATEGY DISCOVERING ALGORITHM

Section 3.1 presents a formal definition of agents and agent behaviour models. Section 3.2 contains a detailed description of the Multi-Agent Strategy Discovering Algorithm (MASDA), accompanied by RoboCup examples.

3.1 Formal Definitions

As described above, the formal definition of agents is based on the model of Lettmann et al. [9], with two extensions. First, the agent role concept is added to the model representing the agent's responsibilities in the multi-agent system. In dynamic environments, agents change roles to fulfil their goal most effectively given the current environment state. The policy function depends on the agent's role. Second, we extend the action definition by associating the triple (preconditions, parameters,

effects) to each action. To execute an action, its preconditions must be met. The way the actions are performed depends on their parameters. Effects define the environmental state when the action is terminated. This gives greater flexibility for action definition.

Definition 1. *Agent* A is a tuple $(S, S_A, I_A, R_A, M_A, A_A, v_A, adapt_A)$ where

- S is a countable set of environmental states.
- S_A is a countable set of internal representations of the environment's states.
- I_A is a countable set of A's internal states.
- R_A is a countable set of A's roles.
- M_A is a countable set of A's mental states. The mental state of the agent, i.e., its "mind", contains all information relevant to the agent's decision making.
- A_A is a countable set of A's possible low-level actions, where each $a \in A_A$ is defined as $a = a(\text{preconditions}, \text{parameters}, \text{effects})$ containing at least one special action representing no action $a_0 = a_0$ ("always", "", "no change").
- $v_A : S \longrightarrow \Pi(S_A)$ is a probabilistic vision function that maps the current environmental state to a probability distribution over all possible internal representations of the environmental states.
- $adapt_A : M_A \longrightarrow M_A$ is an adaptation mechanism that translates the current mental state into another mental state.

A single mental state $m_A \in M_A$ is defined as a tuple $m_A = (s_A, i_A, r_A, \varrho_A, \pi_A, o_A, \tau_A)$ where

- $s_A \in S_A$ is the internal representation of the environmental state of agent A.
- $i_A \in I_A$ is the current internal state of agent A.
- $r_A \in R_A$ is the current role of agent A.
- $\rho_A : S_A \times I_A \longrightarrow I_A$ is a cognition function that calculates the successive internal state of the agent based on the internal representation of environmental state s_A and the current internal state i_A.
- $\pi_A : I_A \times R_A \longrightarrow \Pi(A_A)$ is the agent's probabilistic policy function. It defines the probability of executing a low-level action $a \in A_A$ if the agent is in the internal state $i_A \in I_A$ and has role $r_A \in R_A$.
- o_A is an action selector mechanism (e.g., Roulette wheel selector) that selects an action for the agent based on the probability distribution over the possible actions $\Pi(A_A)$.
- $\tau_A : I_A \times A_A \longrightarrow I_A$ is a state transition function. It defines the successive internal state $i'_A \in I_A$ if the agent performs action $a \in A_A$ in the internal state $i_A \in I_A$.

Figure 1. Agent behaviour model

The formal agent definition defines the policy function π_A as an interface to the agent behaviour model, while the concrete application determines its implementation. To model multi-agent systems in adversary settings, MASDA uses the agent behaviour model presented in Figure 1.

The agent behaviour model has a four-level architecture. At the highest level, agents execute strategies to achieve their goals. Strategies (e.g., offensive, defensive) provide general guidelines for distributing and applying available resources to achieve a desired goal. The strategies decompose into macroactions, which represent a sequence of actions taken to achieve a higher-level purpose (e.g., shot on goal). Macroaction components are high-level actions, i.e., individual agents' actions that are meaningful from the perspective of the analysed domain (e.g., dribble, shot or intercept). Finally, high-level actions contain low-level actions – elementary agent actions like a movement or kick.

Definition 2. *Agent behaviour model* BM_A, an implementation of agent policy π_A, is a tuple $(i_A, r_A, MA_A, HA_A, A_A, MG_A, HG_A, LG_A)$ where

- $i_A \in I_A$ is the current internal state of agent A.
- $r_A \in R_A$ is the current role of agent A.
- MA_A is a countable set of A's possible macroactions, where each $ma \in MA_A$ is defined as $ma = ma(\text{preconditions}, \text{parameters}, \text{effects})$, containing at least one special action representing no macroaction $ma_0 = ma_0$ ("always", "", "no change").
- HA_A is a countable set of A's possible high-level actions, where each $ha \in HA_A$ is defined as $ha = ha(\text{preconditions}, \text{parameters}, \text{effects})$, containing at least one special action representing no high-level action $ha_0 = ha_0$ ("always", "", "no change").
- A_A is a countable set of A's possible low-level actions, where each $a \in A_A$ is defined as $a = a(\text{preconditions}, \text{parameters}, \text{effects})$, containing at least one special action representing no action $a_0 = a_0$ ("always", "", "no change").
- $MG_A : I_A \times R_A \longrightarrow \Pi(MA_A)$ is a set of macro-level agent decision functions that reflect an agent's strategy. A macro-level decision function $mg \in MG_A$

defines the probability of executing macroaction ma for each $ma \in MA_A$ if the internal state of the agent is $i_A \in I_A$ and its role is $r_A \in R_A$.

- $HG_A : I_A \times R_A \times MA_A \longrightarrow \Pi(HA_A)$ is a set of high-level decision functions that define the probability of executing high-level action ha for each $ha \in HA_A$ according to the inner state of the agent $i_A \in I_A$, its role $r_A \in R_A$ and the executed macroaction $ma \in MA_A$.
- $LG_A : I_A \times R_A \times HA_A \longrightarrow \Pi(A_A)$ is a set of low-level decision functions that define the probability of executing low-level action a for each $a \in A_A$ according to the inner state of the agent $i_A \in I_A$, its role $r_A \in R_A$ and the executed high-level action $ha \in HA_A$.

Definition 3. *Agent strategy* is a set of macro-level agent decision functions MG_A.

3.2 Algorithm

As input, MASDA is given as follows:

- Raw data present in game traces
- Domain knowledge in the form of feature taxonomies (hierarchically organised domain concepts).

MASDA consists of three steps:

1. Data pre-processing
   (a) Detecting low-level agent actions from raw data present in game traces
   (b) Detecting high-level agent actions from the extracted low-level agent actions.
2. Generating a visual strategic behaviour model
   (a) Action graph creation
   (b) Merging actions to create abstract action graphs
   (c) Strategy selection and visual model generation.
3. Generating a symbolic strategic behaviour model
   (a) Symbolic description of discovered strategy
   (b) Generating a dataset for rule induction
   (c) Symbolic model generation by rule induction.

MASDA outputs the multi-agent strategy:

- Visually as graph paths
- Symbolically using rules describing characteristic agent interactions.

The first MASDA step (data pre-processing) transforms the raw multi-agent action data present in game traces to a sequence of high-level agent actions (HA_A). RoboCup game traces [21] contain data about agent and ball movements at discrete time points. There are approximately 6 000 equidistant time frames and 512 attributes in one game trace. The first step of the data pre-processing step is to transform the data in such game traces into low-level actions performed by agents (A_A), e.g., move, kick or dash. Heuristic rules are used, as in [22]. An example of a rule is "An agent performs dash if it increases its speed". Low-level agent actions are further processed to discover high-level actions meaningful in soccer (HA_A), including dribble, pass and intercept. Heuristic rules are also used to discover high-level actions [3, 22]: "A pass between two members of a team, p and q, occurred in period $t-k$ to t if the ball was kicked by one player p at time $t-k$ and the ball came into control of another player q at time t and, in interval $[t-k+1, t-1]$, the ball was not under the control of any player and the game was not interrupted". Each high-level action is described by a set of features (e.g., the role of the agent who performed the action, field area). Domain knowledge in the form of feature taxonomies (i.e., a hierarchical representation of domain concepts) is used for this purpose. Section 3.2.1 discusses representing data using taxonomies. The discovered high-level actions and their descriptions, along with the corresponding feature taxonomies, serve as input to the visual-model generation step.

In the second step (generating a visual strategic behaviour model), high-level actions are processed to extract strategic behaviour, which is represented in the form of graph paths. Section 3.2.2 describes this step in detail.

In the third step, the graph paths representing strategic behaviour are described symbolically. Rules describing characteristics of the multi-agent interaction in the game are also generated. Section 3.2.3 describes this step in detail.

3.2.1 Taxonomies

MASDA exploits domain knowledge using taxonomies. *Taxonomy* T is a hierarchical representation of domain concepts. A concept $x \in T$ is an ancestor of concept $y \in T$, $x \leftarrow y$, if x denotes a more general concept than y. Hierarchically ordered domain knowledge allows data aggregation and generating data descriptions at different abstraction levels.

MASDA uses taxonomies for agent roles (R_A) and high-level agent actions (HA_A). It also uses three taxonomies to describe features of the perceived environment state (S_A): the position-, speed- and ball-related taxonomies. Figure 2 presents the high-level action taxonomy. At the highest level, high-level agent actions are grouped as movements or kicks. Movement can be offensive or defensive, whereas kick can be a pass, miss or shot on goal. Offensive movement on the ball encompasses high-level actions, including control dribble, long dribble and speed dribble, while offensive movement off the ball is attack support. Defensive movement encompasses intercept. A pass can be done to a player or space. A miss represents a kick that ends with the ball passing the goal line or side-

line, being intercepted by the opponent, going to corner or an unsuccessful shot on goal.

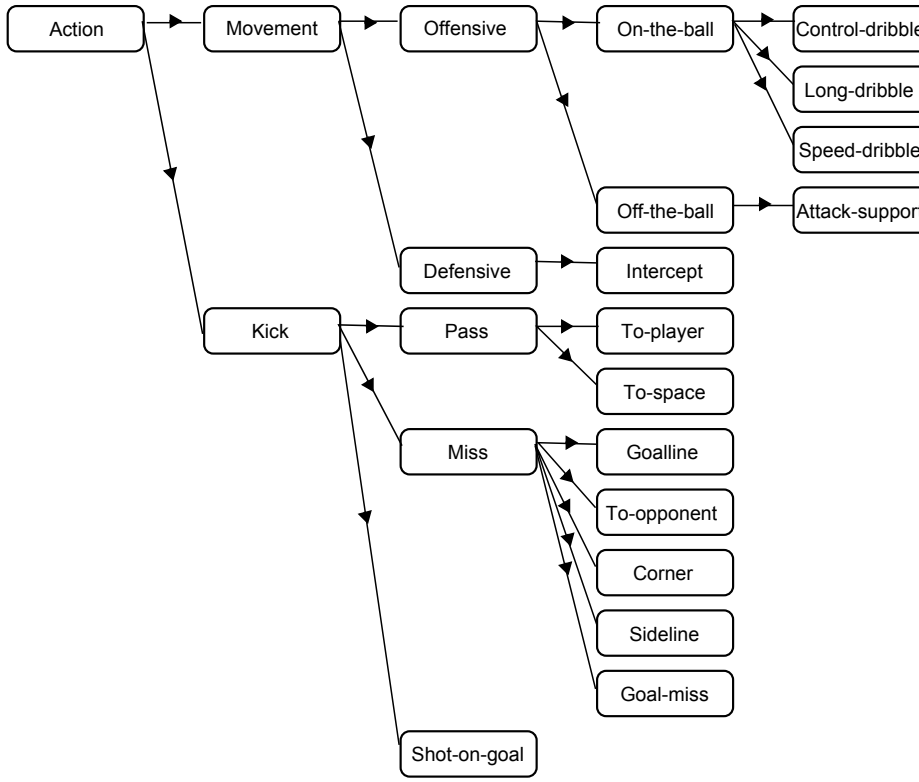


Figure 2. High-level action taxonomy

3.2.2 Generating a Visual Strategic Behaviour Model

This section describes how strategic behaviour is discovered from the extracted high-level multi-agent action sequence.

The central concept of the MASDA algorithm is the action graph, which is created using the reconstructed high-level agent actions. An action graph is a directed graph, where nodes represent the state space at the start of an agent's high-level action by means of action features and connections correspond to agent high-level actions. In other words, a reconstructed high-level action instance is fully represented by a graph node-connection pair. Terminal actions (i.e., the last actions in an action sequence) are represented as connections to terminal nodes, which are nodes with no outgoing connections that define the state space at the end of a high-level action sequence. The node positions of the action graph correspond to agent

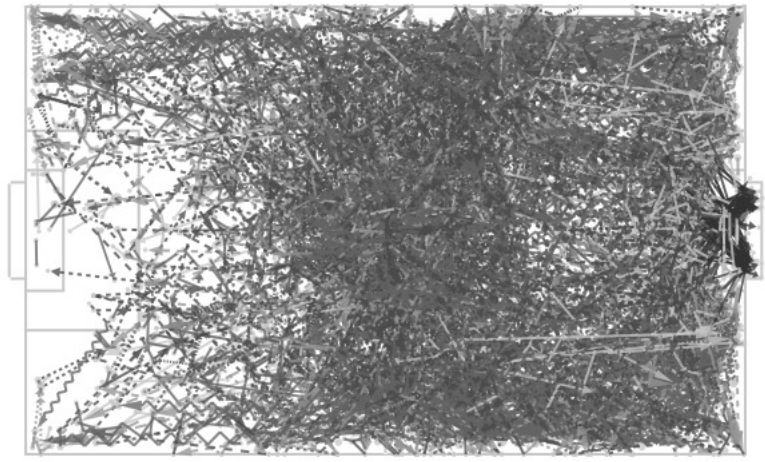


Figure 3. Action graph

positions in the domain space at high-level action start. Figure 3 presents an example of an action graph. It is constructed using data from 10 matches of a reference RoboCup team. The graph nodes are represented as white circles. The graph connection patterns represent the type of performed high-level action: a dashed line represents a pass, a wavy line represents dribbling and a solid line represents a shot on goal.

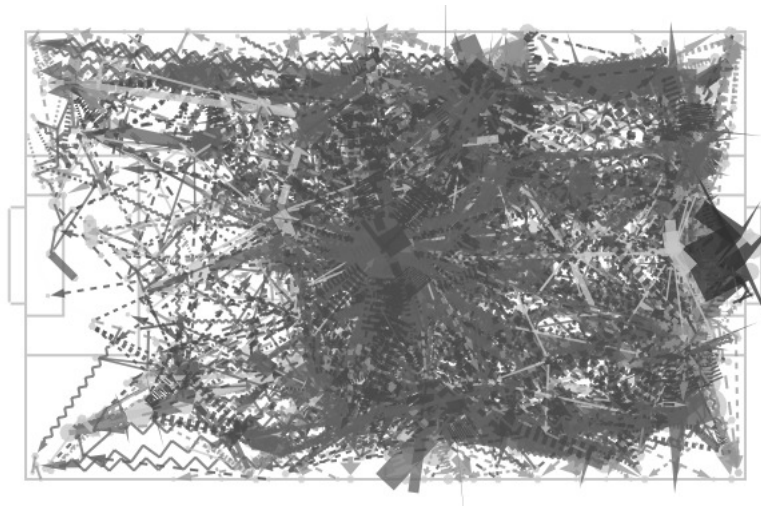


Figure 4. Abstract action graph AAG_{15}

Strategic behaviour cannot be easily discovered from the action graph because it contains a lot of high-level action instances, many of which reflect reactive behaviour. Its level of detail must therefore be reduced, while preserving the characteristics of the observed multi-agent behaviour. MASDA accomplishes this by hierarchically clustering high-level action instances to form abstract action graphs (AAG). High-level action instances similar in the domain space (i.e. started under similar circumstances as represented by the action features of their starting graph nodes and indicating the execution of a similar high-level action as represented by their underlying graph connections) are merged together and marked as a single abstract action instance. An abstract action instance is a single graph node-connection pair in an AAG assigned with the common high-level action- and role-parent of the merged high-level action instances. Each AAG is characterized by an abstraction level. The distance between two (abstract) high-level action instances in an AAG, computed by the instance distance metric defined below, is greater than its abstraction level. In other words, all high-level action instances which are closer to each other than the abstraction level of an AAG are merged together and represented by one abstract action instance (one graph node-connection pair) in the AAG. The abstraction level of an AAG is a number greater than 0, with smaller numbers indicating low abstraction and higher numbers indicating greater abstraction. Figure 4 presents an example abstract action graph at abstraction level 15 (AAG_{15}). All high-level action instances with distance smaller than or equal to 15.0 are merged together and represented as one abstract action instance in AAG_{15}. Therefore, the distance between two (abstract) high-level action instances (represented by two node-connection pairs) in this AAG computed using the instance distance metric is greater than 15.0. AAG_{15} contains graph nodes and connections that represent more than one similar high-level action; the number of nodes and connections in Figure 4 is thus smaller than that in Figure 3. This is graphically depicted by larger nodes and connections. A larger node/connection indicates that it contains more high-level action instances from the initial action graph. The appropriate abstraction level must be carefully determined because too little abstraction leads to models reflecting agents reactions to local environment changes and too much abstraction may neglect important strategic behaviour details.

In hierarchical clustering, (abstract) high-level action instances are iteratively merged together according to an instance distance-metric. The distance between two (abstract) high-level action instances, i_1 and i_2, $\mathrm{dist}(i_1, i_2)$, considers the average distance between their starting positions $\overline{\mathrm{dist}_{pos}(i_1, i_2)}$, the average distance of their roles in the role taxonomy $\overline{\mathrm{dist}_{role}(i_1, i_2)}$ and the average distance of their actions in the high-level action taxonomy $\overline{\mathrm{dist}_{action}(i_1, i_2)}$. The instance distance-metric is defined as the weighted sum of these three distances:

$$\mathrm{dist}(i_1, i_2) = w_{pos} \cdot \overline{\mathrm{dist}_{pos}(i_1, i_2)} + w_{role} \cdot \overline{\mathrm{dist}_{role}(i_1, i_2)} + w_{action} \cdot \overline{\mathrm{dist}_{action}(i_1, i_2)} \quad (1)$$

where w_{pos} represents the weight on position distance, w_{role} the weight on role distance and w_{action} the weight on action distance.

Distance $\overline{\text{dist}_{pos}(i_1, i_2)}$ is domain-dependent. For RoboCup, it is calculated as the Euclidean distance between the position of the starting graph nodes of the high-level action instances i_1 and i_2.

$$\overline{\text{dist}_{pos}(i_1, i_2)} = \sqrt{(i_1.\text{pos}.x - i_2.\text{pos}.x)^2 + (i_1.\text{pos}.y - i_2.\text{pos}.y)^2} \tag{2}$$

Distance $\overline{\text{dist}_{role}(i_1, i_2)}$ is domain-independent and defined as follows:

$$\overline{\text{dist}_{role}(i_1, i_2)} = \sum_{\forall(x,y): x \in i_1.roles, y \in i_2.roles} \frac{\text{dist}_{role}(x, y)}{|i_1.roles| \cdot |i_2.roles|} \tag{3}$$

where $\text{dist}_{role}(x, y)$ is the role-taxonomy distance of elements x and y, while $|i_1.roles|$ and $|i_2.roles|$ are the number of distinct role instances in the starting graph nodes of the high-level action instances i_1 and i_2, respectively.

Similarly, $\overline{\text{dist}_{action}(i_1, i_2)}$ is defined as follows:

$$\overline{\text{dist}_{action}(i_1, i_2)} = \sum_{\forall(x,y): x \in i_1.actions, y \in i_2.actions} \frac{\text{dist}_{action}(x, y)}{|i_1.actions| \cdot |i_2.actions|} \tag{4}$$

where $\text{dist}_{action}(x, y)$ is the high-level action-taxonomy distance of elements x and y and $|i_1.actions|$ and $|i_2.actions|$ are the number of distinct action instances in the graph connections of the high-level action instances i_1 and i_2, respectively.

The taxonomy distance metric evaluates the dissimilarity between two taxonomy elements n_1 and n_2. In MASDA, it is defined as follows:

$$\text{dist}_T(n_1, n_2) = \frac{\left(\frac{\text{distance}_T(n_1, n_2)}{2 \cdot \text{depth}_{T_{\max}}}\right) + (\text{depth}_{T_{\max}} - \text{depth}_T(\text{common\_ancestor}(n_1, n_2)))}{\text{depth}_{T_{\max}} + 1}. \tag{5}$$

This metric evaluates the dissimilarity of two elements based on the level of their common ancestor. A common ancestor higher in the taxonomy indicates that the elements are more dissimilar and vice versa. This factor, whose values range between 0 and $\text{depth}_{T_{\max}}$, is represented by the following element in the hierarchy distance metric:

$$\text{depth}_{T_{\max}} - \text{depth}_T(\text{common\_ancestor}(n_1, n_2))$$

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 in the taxonomy is longer. This factor, whose values range between 0 and 1, is represented as follows:

$$\frac{\text{distance}_T(n_1, n_2)}{2 \cdot \text{depth}_{T_{\max}}}.$$

The taxonomy distance metric is normalised in the range [0, 1] by dividing it by $(depth_{T_{max}} + 1)$.

The merging process reduces the number of high-level action instances by representing similar high-level action instances with one graph node-connection pair. Connections in the abstract action graph with high numbers of high-level action instances represent strategic behaviour, whereas those with fewer high-level action instances represent reactive behaviour (i.e., agent reactions to dynamic changes in the environment).

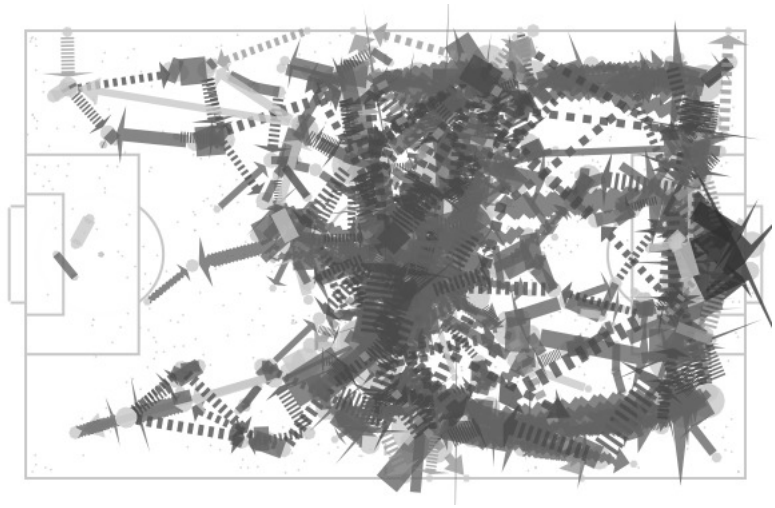


Figure 5. AAG_{15} with weak connection 3

Reactive behaviour is filtered using a measure called weak connection. Weak connection of a path *path* in the abstract action graph is the minimum number of high-level action instances encompassed by a single connection in *path*. Figure 5 presents a subgraph of the abstract action graph presented in Figure 4 in which the weak connection of each path is at least 3. Compared to Figure 4, repeated agent behaviour is clearer in Figure 5.

Strategic agent behaviour is extracted using a measure called strong connection. Strong connection of a path *path* in the abstract action graph is the number of paths of length length(*path*) from the starting action graph entirely encompassed by *path*. The graph paths with the highest strong connections are macroactions MA_A representing strategic behaviour. Figure 6 represents a strategic macroaction extracted by MASDA. It depicts a successful attack on the goal, which starts with a pass from the area near opponent's left corner to the penalty area where, after dribbling, a successful shot on the right side of the goal is performed. The graph path representing the discovered macroaction is the visual macroaction model.

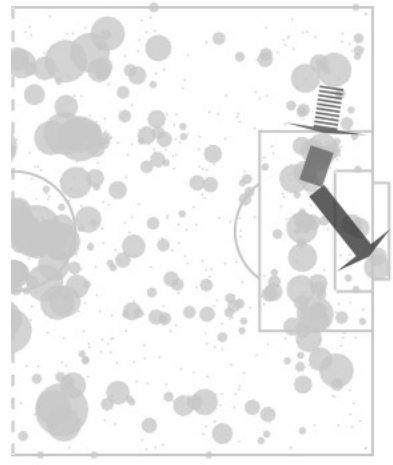


Figure 6. Visual model of a macroaction

ROLE	LTeam.L-FW	LTeam.C-FW	LTeam.C-FW	LTeam.C-FW
ACTION	Pass-to-player	Control-dribble	Successful shot-on-goal	Successful shot-on-goal (end)
POSITION	(x: 46.90, y: -24.49)	(x: 45.15, y: -12.47)	(x: 43.46, y: -7.27)	(x: 53.15, y: 5.47)
ABSTRACTION	abs:15	abs:15	abs:15	abs:15

Figure 7. Symbolic description of a macroaction's visual model

3.3 Generating a Symbolic Strategic Behaviour Model

This section describes how the visual strategic behaviour representation is enriched with symbolic description in the form of rules.

The visual macroaction model is first enriched with symbolic description that contains all high-level actions comprising the macroaction, the agent roles, the positions where the high-level actions were performed and the abstraction level of the abstract action graph from which the macroaction was extracted. Figure 7 presents a symbolic description of the visual macroaction model in Figure 6. The macroaction was extracted from the abstract action graph at abstraction level 15. The macroaction starts with a pass performed by the left forward of the reference team (LTeam.L-FW) in the area near the opponent's left corner (coordinate values x: 46.90, y: -24.49). The centre forward of the reference team (LTeam.C-FW) re-

ceives the ball in the penalty area (coordinate values x: 45.15, y: −12.47), where he performs a control dribble. After the control dribble, he shoots on the opponent's goal from the penalty area (coordinate values x: 43.46, y: −7.27). The ball ends in the right side of the opponent's goal (coordinate values x: 53.15, y: 5.47). The symbolic description of visual macroaction model can be used to anticipate an opponent's future high-level actions. If an opponent's activity matches a starting part of a discovered macroaction, it is reasonable to conclude that the agent will continue to execute that macroaction.

In the next step, rule induction determines the situation in which the macroaction occurs. One dataset example corresponds to one performed high-level action. Example features represent the situation at the start of the high-level action (e.g., the players' positions on the field, their position relative to the ball, movement speed and direction, ball position and speed, ball distance to opponent's goal), and the class value is the pair agent_role:performed_high_level_action. Figure 8 shows four possibilities for selecting positive and negative training examples for rule induction. Each circle in the figure represents one example (start of high-level action instance). The circles annotated with '+' represent positive examples. The circles annotated with '−' represent negative examples. The examples represented with empty circles are not included in the training dataset. The connection pattern represents the type of performed high-level action. A dashed line represents a pass, whereas a wavy line represents a dribble. Positive and negative examples can be selected as follows.

1. Positive examples represent all high-level action instances that correspond to a connection of interest, whereas negative examples correspond to all high-level action instances that start in the same node as the connection of interest but continue along the node's remaining connections. Figure 8 a) presents an example in which the pattern 'dribble from middle-field to penalty area' is learnt. The training examples represent the difference between high-level action instances representing dribble to penalty area and pass to player, such that all high-level action instances start in the middle-field.

2. Positive examples represent all high-level action instances that correspond to a connection of interest, whereas negative examples are all other high-level action instances. Figure 8 b) presents an example in which the pattern 'dribble from middle-field' is learnt. The training examples represent the difference between the high-level action instances representing dribble starting in the middle-field and all other high-level action instances.

3. Positive examples represent all high-level action instances contained in a node of interest, whereas negative examples represent all other high-level action instances. Figure 8 c) presents an example in which the pattern 'start of offensive action from middle-field' is learnt. The training examples represent the difference between high-level action instances representing the start of offensive-actions from the middle-field and all other high-level action instances.

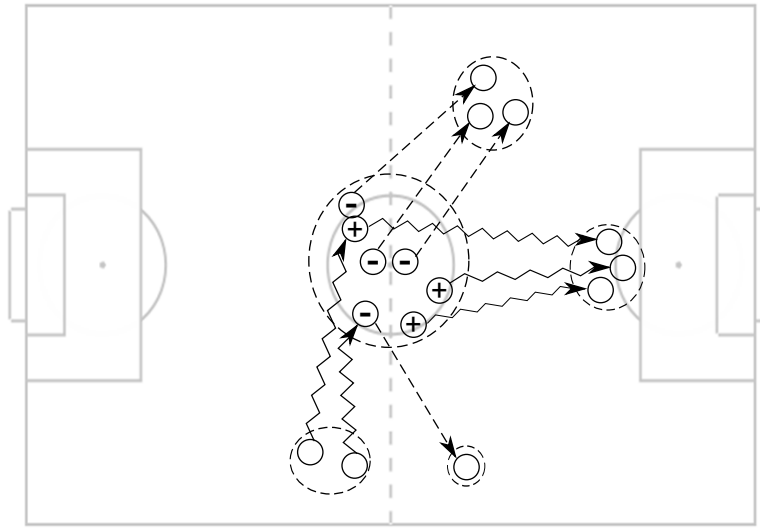
4. Positive examples represent all high-level action instances that correspond to a connection of interest, whereas negative examples represent high-level action instances that do not start in the node in which the connection of interest starts. Figure 8 d) presents an example in which the pattern 'dribble from middle-field to penalty area' is learnt. The training examples represent the difference between high-level action instances representing dribble from the middle-field to penalty area and all other high-level action instances that did not start in the middle-field.

MASDA uses the second approach to select positive and negative examples. When the second approach is used, the number of negative examples is much higher than that of the positive. Standard induction algorithms do not perform well when the training dataset has strongly skewed class distribution. MASDA thus selects a subset of negative examples using two approaches:

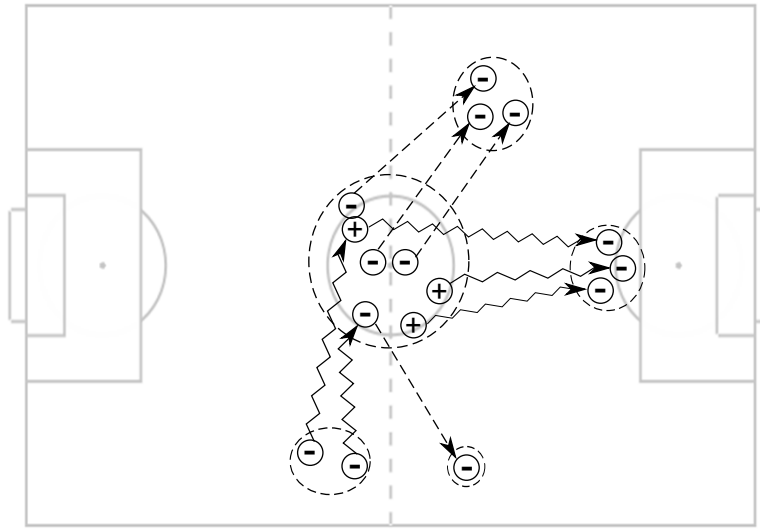
- Select examples that represent near misses, i.e., similar action instances to the action concept of interest with respect to the previously described instance distance metric. In this case, the resulting rules represent local differences between similar high-level action concepts.
- Select examples that represent far misses, i.e., action instances that represent much different action concepts than the action concept of interest with respect to the previously described instance distance metric. In this case, the resulting rules represent a global description of the high-level action concept of interest.

MASDA uses the SLIPPER algorithm [23] for rule induction. SLIPPER is suitable for this domain because it is a set-valued rule inducer, meaning that the value of a feature can be a set. First, it enables a compact representation of the environment when high-level actions start. The attribute 'approaching the ball' is used as an example. To capture which of the 22 players on the field approach the ball without using set-valued attributes, 22 Boolean features of the type 'player id:approaching the ball' are needed. If set-valued attributes are allowed, a single attribute captures this information. Secondly, it allows creating rule conditions at different abstraction levels. During one attack on the goal, the left fullback of the defending team approaches the ball, i.e., tries to stop the attack; in another, the right fullback approaches the ball. Due to the combination of set-valued attributes and feature taxonomies, such situations can be represented as a single rule condition: opponent's fullback approaches the ball.

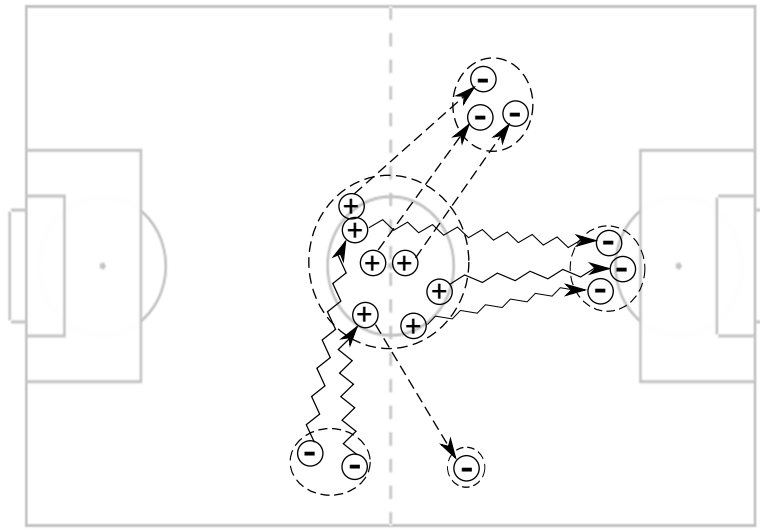
Finally, Figure 9 describes situations in which components of the example macro-action (Figure 6) are executed, i.e., the symbolic macroaction model. Rule a) in Figure 9 presents a situation in which the left forward player performs a pass. The ball is in the left wing (Ball:Left-wing). An opponent's fullback is left behind (RTeam.FB:Back). An opponent's middle-fielder, centre fullback and goalkeeper move towards the player with the ball (RTeam.MF:Incoming $\bigwedge$ RTeam. C-FB:Incoming $\bigwedge$ RTeam.GK:Incoming) to stop the attack. The opponent's right middle-fielder is back in the team's defending third (RTeam.R-MF:Attacking-half.



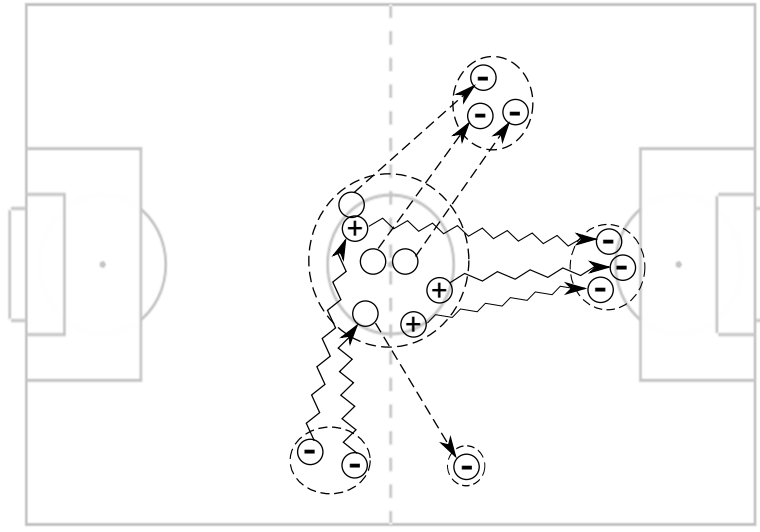
a)



b)

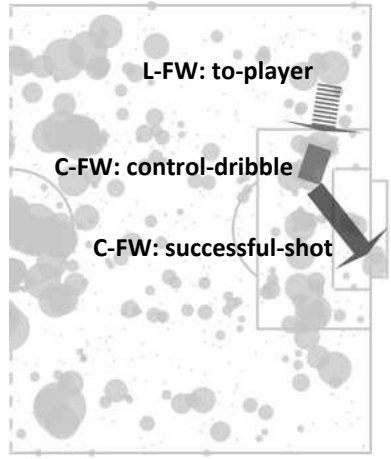


c)



d)

Figure 8. Selection of positive and negative examples for rule induction



a)	IF	Ball:Left-wing ⋀ RTeam.FB:Back ⋀ RTeam.MF:Incoming ⋀ RTeam.C-FB:Incoming ⋀ RTeam.GK:Incoming ⋀ RTeam.R-MF:Attacking-half.Attacking-third ⋀ LTeam.C-FW:Incoming ⋀ LTeam.MF:Incoming	THEN	LTeam.L-FW: Pass-to-player
b)	IF	LTeam.C-FW:Has-ball ⋀ LTeam.L-FW:Near ⋀ RTeam.GK:Short-distance ⋀ RTeam.FB:Medium-distance ⋀ RTeam.L-MF:Back	THEN	LTeam.C-FW: Control-dribble
c)	IF	LTeam.C-FW:Attacking-half.Attacking-third.Penalty-box ⋀ LTeam.R-FW:Attacking-half.Attacking-third.Penalty-box ⋀ LTeam.L-FW:Medium-distance ⋀ RTeam.GK:Short-distance	THEN	LTeam.C-FW: Successful shot-on-goal

Figure 9. Symbolic macroaction model

*Attacking-third) to help in the defence. The centre forward and a middle-fielder of the attacking team move towards the ball (C-FW:Incoming ⋀ LTeam.MF:Incoming) to assist the left forward player in the attack. Rule b) in Figure 9 presents a situation in which the centre forward player starts a control dribble. The center forward player of the attacking team has the ball (LTeam.C-FW:Has-ball) and its left forward is near him (LTeam.L-FW:Near). The opponent's goalkeeper is also near the centre forward of the attacking team (RTeam.GK:Short-distance), an opponent's fullback is

at medium distance from the ball (RTeam.FB:Medium-distance), and its left middle-fielder is left behind (RTeam.L-MF:Back). Except for the opponent's goalkeeper, there is no direct attack on the ball from the opponent's side. Finally, Rule c) in Figure 9 presents a situation in which the centre forward player shoots on the goal. The centre forward of the attacking team and its right forward are in the penalty box of the opponent's team (LTeam.C-FW:Attacking-half.Attacking-third.Penalty-box $\bigwedge$ LTeam.R-FW:Attacking-half.Attacking-third.Penalty-box). The left-forward of the attacking team is at medium distance from the ball (LTeam.L-FW:Medium-distance). The opponent's goalkeeper is near the centre forward (RTeam.GK:Short-distance).

Although we present only one rule per macroaction component in Figure 9, the symbolic macroaction model contains a set of rules for each macroaction component. Each rule is associated with a confidence level outputted by SLIPPER. The probability of executing a concrete high-level action in a given environment state is proportional to the confidence levels of the rules of the high-level action that cover the particular environment state.

4 EVALUATION

The MASDA evaluation was performed on the reference team Helios [24], currently one of the most successful teams in the RoboCup 2D Simulation League, having won the tournament in 2010. We collected data from 10 games in which this team scored many goals. Because the analysis focused on attacks on the goal, repetitions of patterns that ended with successful shots on goal were needed for MASDA to be able to discover strategic offensive behaviour. The games encompass competitions of Helios against 9 opponents. The average number of scored goals per game by Helios is 7.4 (ranging from 4 to 15).The merged data of the 10 games are denoted as a reference game.

4.1 Human Analysis

Figure 10 presents the team's successful attacks on the goal. This is an abstract action graph of the reference game, where the abstraction level is 10, the weak connection is 3 and the strong connection is 2. The figure shows that the attacks on the goal are mainly conducted from the right side of the field, but attacks also come from the middle and left sides of the field. The attacks along the right side are mainly conducted by the right forward player, who dribbles to the area near the right opponent's corner. When he reaches this area, he passes the ball to the centre forward in the penalty area. The centre forward performs a control dribble and shoots on the goal, passes the ball to the left forward player, also in the penalty area, or returns the ball to the right forward player. When the attack is performed from the left side, the left forward dribbles to the area near the opponent's left corner, where he passes the ball to the centre forward in the penalty area, who

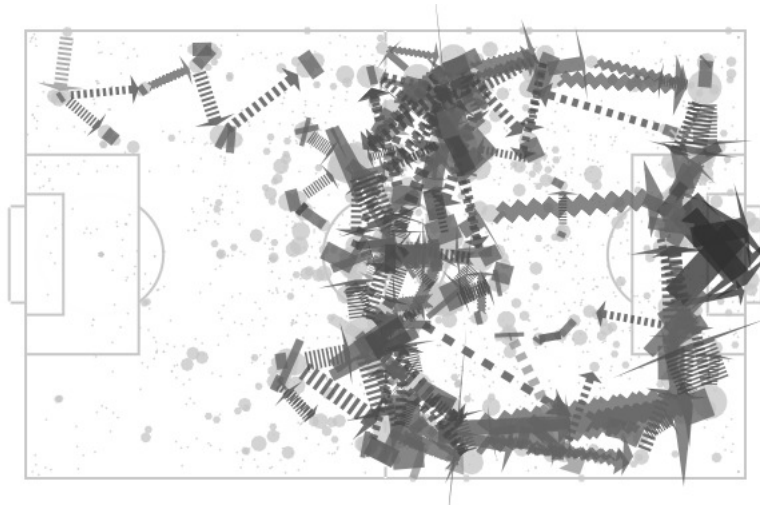


Figure 10. Helios' offensive play – AAG_{10} with weak connection 3 and strong connection 2 of the reference game

then shoots on goal. The attack in the centre of the field represents a situation in which a dribbling player manages to reach the penalty area, after which he shoots on the goal. Figure 11 presents all macroactions in the reference AAG10 that represent successful attacks on the goal. Figure 11 e) shows the example macroaction described in Section 3.

Two soccer experts evaluated the relevance of the discovered strategic patterns. They assessed the adequateness of the rules in the symbolic model of strategic macroaction situations. Each rule condition was labelled as very important, meaningful or irrelevant for the analysed situation. The experts designated 34 % of the rule conditions as very important, 34 % as meaningful and 32 % as irrelevant. They confirmed that the symbolic description captured the main advantages and disadvantages of the analysed situations.

4.2 Quantitative Evaluation

The quantitative evaluation measures to what extent strategic macroactions discovered in a training dataset are also present in a separate test dataset.

Tests were performed for three cases. In the first case, the visual model, i.e., the graph path representing a discovered macroaction, was used for classification. The visual model classifies an example as strategic if the minimum distance between the example and a node in the graph path representing the discovered macroaction (defined in Section 3.2.2) is less than the abstraction level of the abstract action graph from which the macroaction was extracted. The second case uses the symbolic model, i.e., rules representing the situations in which a discovered macroaction

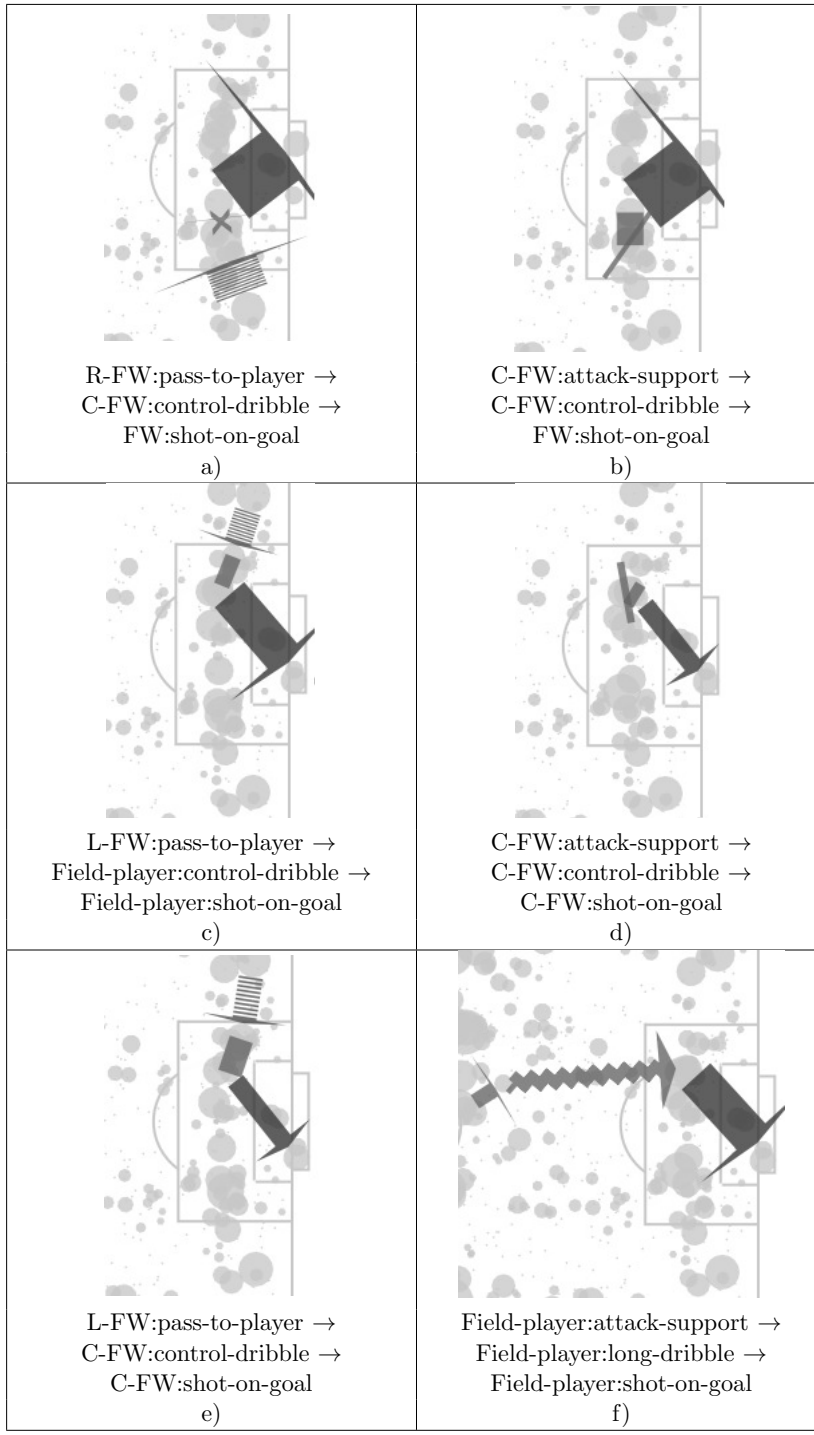


Figure 11. Macroactions representing successful attacks on the goal

occurs. The symbolic model classifies an example as strategic if it is covered by at least one rule of the model. The third case concerns both the visual and symbolic models. An example is classified as strategic in this case when both the visual and symbolic models classify it as such.

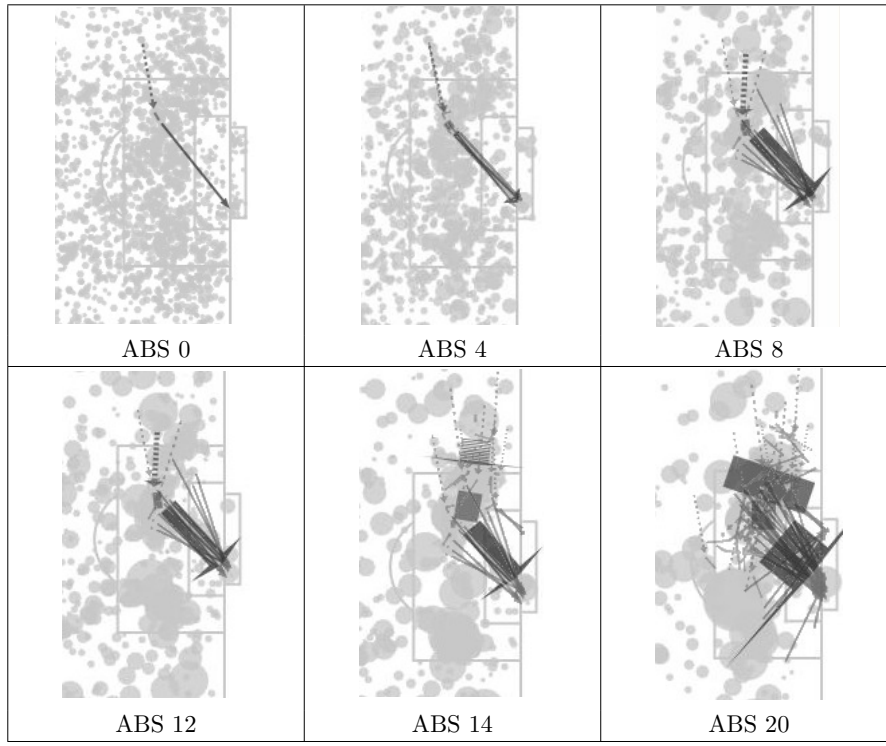
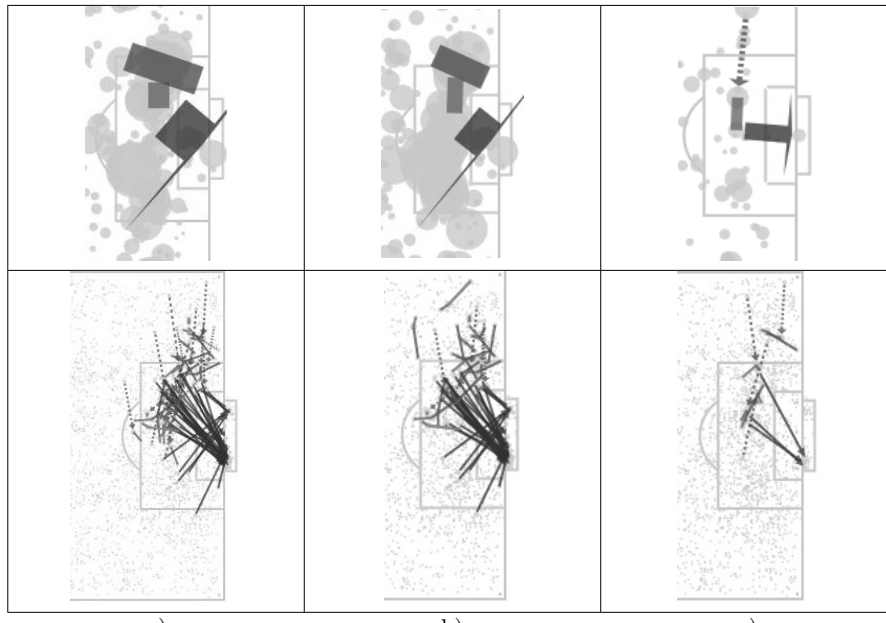


Figure 12. Reference path at different abstraction levels

The evaluation was executed on the example macroaction presented in Figure 6 as a reference path. The reference path is extracted from data from 10 team games merged together. Figure 12 shows the reference path at several abstraction levels. The thicker lines represent the reference path, whereas the thin lines represent performed high-level actions contained in the reference path.

The evaluation was performed using the leave-one-game-out scheme. We performed 10 evaluation runs. In each of the 10 runs, 9 of the games were merged together to create a training game AG_{learn}, while the remaining 1 game represented a test game AG_{test}. Each test run encompassed evaluation of the performance of the visual and symbolic models at abstraction levels 1 through 25. Therefore, abstract action graphs of the training game $AAG_{abs-learn}$ and the test game $AAG_{abs-test}$ were created for abstraction levels 1 through 25. At each abstraction level, the reference path was used to determine a learning path out of the training game according to



a) b) c)

Figure 13. Macroaction at abstraction level 20 in a) the reference game (10 matches), b) a training game (9 matches) and c) a test game (1 match)

which the visual and symbolic macroaction models were created. In addition to this, the reference path was used to select the test set out of the test game. The presentation of one evaluation run at one abstraction level that follows is accompanied with an example shown in Figure 13. The upper row of Figure 13 a) contains the example macroaction from Figure 6 (i.e. the reference path) at abstraction level 20, while the upper rows of Figure 13 b) and Figure 13 c) present the example macroaction at abstraction level 20 in the training game $AAG_{20-learn}$ and the test game $AAG_{20-test}$. The lower row of Figure 13 shows the constituent high-level actions of the example macroaction in the three cases. A learning path in $AAG_{abs-learn}$ is determined as the path with the same length as the reference path that contains the most examples (i.e., high-level action instances) along the reference path. The example learning path in $AAG_{20-learn}$ presented in Figure 13 b) contained 79 % of the high-level actions of the reference path. The visual model is the graph path of $AAG_{abs-learn}$ that represents the learning path. The symbolic model is a set of rules generated for the learning path, as in Section 3.2.3. After creating the visual and symbolic models, positive and negative examples in the test game must be determined. Positive examples in $AAG_{abs-test}$ are all performed high-level actions in $AAG_{abs-test}$ contained in the reference path. Figure 13 c) presents the positive test examples in $AAG_{20-test}$. Although all other examples in the test game are negative, we restricted the set of negative examples to those within a predefined maximum

negative distance from the reference path. The focus is thus set on negative (non-strategic) examples, which can easily be misclassified as strategic. In the performed evaluation, the maximum negative distance was set to the maximum distance between each consecutive pair of nodes of the reference and learning paths (i.e., the distance between the first nodes of the reference and learning paths, the distance between their second nodes, and so on) incremented by the abstraction level. Having determined the test dataset, we calculate the performance of the visual model and the symbolic model separately, as well as the merged prediction of the both models using the F-measure. This procedure was repeated for abstraction levels 1 through 25 in each of the 10 evaluation runs. The average F-measure values per abstraction level were calculated.

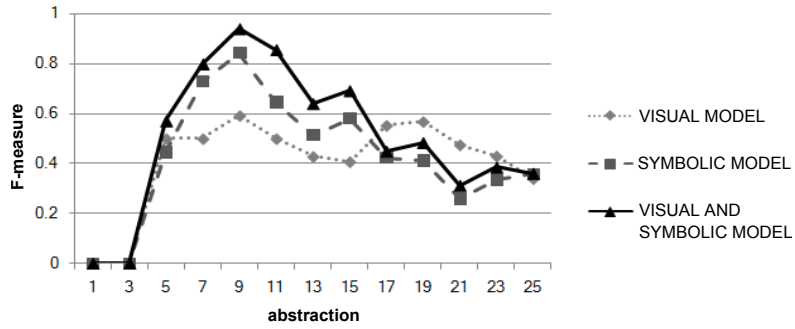


Figure 14. Evaluation of models' performance

Figure 14 presents to what extent the strategic behaviour learned on a training dataset is also found in a separate test dataset. Model performance increases as it approaches abstraction level 9, where it reaches the highest value (94 % with respect to F-measure); it then starts to fall. At lower abstraction levels, the number of examples used to build the models is small. Lower abstraction levels thus produce more specific models. Higher abstraction levels indicate highly abstract but less accurate models because the graph nodes and connections contain many examples, some of which may represent different action concepts. The best model performance is achieved at the abstraction level that best balances specificity and abstraction, which is abstraction level 9 on the analysed data.

5 CONCLUSION

MASDA is a general domain-independent algorithm for discovering strategic behaviour in multi-agent systems. It presents two novel features. First, it constructs strategic patterns from low-level actions by means of abstraction and complex algorithms using only basic domain knowledge in the form of feature taxonomies. Second, MASDA constructs strategic patterns in visual and symbolic rule form and combines both mechanisms to improve comprehensibility and classification accuracy.

MASDA can be used to study and understand the behaviour of agents in multi-agent systems. It was extensively tested on RoboCup. Experts confirmed the relevance of the strategic behaviour patterns discovered by the algorithm. Quantitative evaluation of the models developed by MASDA in terms of F-measure showed reliable performance, with the F-measure reaching 94 %. MASDA can also be used for behaviour cloning. This was successfully accomplished in robotic Keepaway [25].

The main task for the future is testing the algorithm in a domain substantially different from RoboCup. Currently, the algorithm is being used to analyse human behaviour during asymmetric conflicts in urban environments [26], where initial results are encouraging. Perhaps even more importantly, MASDA's cloning capabilities need to be tested in a more complex, real-world domain.

Acknowledgements

This work was partially financed by the European Union, European Social Fund. The authors also acknowledge the financial support from the Slovenian Research Agency.

REFERENCES

[1] Iglesias, J. A.—Ledezma, A.—Sanchis, A.: CAOS Coach 2006 Simulation Team: An Opponent Modelling Approach. Computing and Informatics, Vol. 28, 2009, No. 1, pp. 57–80.

[2] Horman, Y.—Kaminka, G.: Removing Statistical Biases in Unsupervised Sequence Learning. In: Proceedings of Intelligent Data Analysis, IDA 2005.

[3] Kaminka, G.—Fidanboylu, M.—Chang, A.—Veloso, M. M.: Learning the Sequential Coordinated Behaviour of Teams from Observations. RoboCup 2002: Robot Soccer World Cup VI, Lecture Notes in Artificial Intelligence, 2003, No. 2752, pp. 111–125.

[4] Raines, T.—Tambe, M.—Marsella, S.: Automated Assistants to Aid Humans in Understanding Team Behaviours. In: Proceedings of the Fourth International Conference on Autonomous Agents, AGENTS '00, pp. 419–426.

[5] Aler, R.—Valls, J. M.—Camacho, D.—Lopez, A.: Programming Robosoccer Agents by Modeling Human Behaviour. Expert Systems with Applications, Vol. 36, 2009, No. 2, pp. 1850–1859.

[6] Abbott, R. G.: Behavioral Cloning for Simulator Validation. RoboCup2007: Robot Soccer World Cup XI. Lecture Notes in Computer Science, Vol. 5001, 2008, pp. 329–336.

[7] Thurau, C.—Bauckhage, C.—Sagerer, G.: Imitation Learning at All Levels of Game-AI. In: Proceedings of the International Conference on Computer Games, Artificial Intelligence, Design and Education, 2004, pp. 402–408.

[8] RoboCup Soccer Simulation League. Availaible on: `http://www.robocup.org/robocup-soccer/simulation/`, 22.03.2012.

[9] Lettmann, T.—Baumann, M.—Eberling, M.—Kemmerich, T.: Modeling Agents and Agent Systems. Transactions on Computational Collective Intelligence V, Vol. 6910, 2011, pp. 157–181.

[10] Russell, S.—Norvig, P.: Artificial intelligence: A Modern Approach. 3rd Edition. Prentice Hall, 2010.

[11] Wooldridge, M.: An Introduction to Multi-Agent Systems. 2nd Edition. Wiley, 2009.

[12] Ferber, J.: Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley, Longman, 1999.

[13] Rao, A. S.—Georgeff, M. P.: BDI Agents: From Theory to Practice. In: V. R. Lesser and L. Gasser (Eds.): Principles of Knowledge Representation and Reasoning, Proceedings of the First International Conference on Multiagent Systems, ICMAS '95, San Francisco, June 1995, pp. 312–319.

[14] Schmidt, B.: Modelling of Human Behavior: The PECS Reference Model. In: Proceedings of the Fourteenth European Simulation Symposium, 2001.

[15] Stone, P.: Layered Learning in Multi-Agent Systems. Ph. D. Thesis, 1998.

[16] Hollnagel, E.: Human Reliability Analysis: Context & Control. Academic Press, London, UK, 1993.

[17] Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufman, 1994.

[18] Ramos, F.—Ayanegui, H.: Discovering Tactical Behaviour Patterns Supported by Topological Structures in Soccer-Agent Domains. In: L. Padgham, D. C. Parkes, J. P. Müller and S. Parsons (Eds.): Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2008, Estoril, May 2008, pp. 1421–1424.

[19] Thurau, C.—Bauckhage, C.—Sagerer, G.: Learning Human-Like Movement Behavior for Computer Games. In: Proceedings of the Eighth International Conference on the Simulation and Adaptive Behaviour, SAB 2004, 2004, pp. 315–323.

[20] Bežek, A.—Gams, M.—Bratko, I.: Multi-Agent Strategic Modeling in a Robotic Soccer Domain. In: Proceedings of the Fifth International Confefence on Autonomous Agents and Multi-Agent Systems, AAMAS 2006, Hakodate, Japan, pp. 457–464.

[21] Chen, M.—Foroughi, E.—Heintz, F.—Kapetanakis, S.—Kostiadis, K.—Kummeneje, J. et al.: Users Manual: RoboCup Soccer Server Manual for Soccer Server Version 7.07 and Later. 2003.

[22] Nair, R.—Tambe, M.—Marsella, S.: Role Allocation and Reallocation in Multiagent Teams: Towards a Practical Analysis. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2003, Melbourne, July 2003, pp. 552–559.

[23] Cohen, W. W.—Singer, Y.: A Simple, Fast, and Effective Rule Learner. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence, AAAI '99, pp. 335–342.

[24] Akiyam, H.—Shimora, H.: HELIOS 2010 Team Description. Available on: `http://julia.ist.tugraz.at/robocup2010/tdps/2D_TDP_HELIOS.pdf`, 12.03.2012.

[25] Bežek, A.—Gams, M.—Bratko, I.: Imitating Observed Multi-Agent System in the 3vs2 Keepaway Domain using MASDA (Posnemanje opazovanega veagentnega

sistema z uporabo MASDA v domeni 3vs2 Keepaway). In: M. Bohanec and M. Gams (Eds.): Intelligent Systems, Proceedings of the Ninth International Multiconference Information Society, IS 2006, Ljubljana, Slovenia, pp. 101–104.

[26] Tavčar, A.—Mirchevska, V.—Gams, M.: EUSAS: European Urban Simulation for Asymmetric Scenarios. In: Intelligent Systems, Proceedings of the Thirteenth International Multiconference Information Society, IS 2010, Ljubljana, October 2010, pp. 90–93.

Violeta Mirchevska is a researcher at Result d.o.o., cooperating closely with the Department of Intelligent Systems at the Jožef Stefan Institute. She completed her Bachelor's in computer science and automation at the Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and Methodius University, Macedonia in 2007 and is currently a Ph. D. candidate at Jožef Stefan International Postgraduate School, Slovenia. Her research focuses on modelling agent behaviour based on observing low-level action sequences by leveraging both domain knowledge and machine learning. The main application areas of her research are user profiling, remote health monitoring and security.

Mitja Luštrek is a researcher at the Jožef Stefan Institute in Slovenia. He is the head of the ambient intelligence group at the Department of Intelligent Systems. His main research area is ambient intelligence, with a focus on human behaviour analysis. He has experience with wearable inertial sensors and real-time locating systems and has used artificial intelligence techniques for activity recognition and detecting anomalous behaviours. He has also worked in several other artificial intelligence areas, including game playing, heuristic search, and machine learning in bioinformatics. He is an editor of the Informatica journal and a member of the executive board of the Slovenian Artificial Intelligence Society.

Andraž Bežek received his Ph. D. in computer science from the University of Ljubljana in 2007. He is currently a Chief Marketing Officer at Marg d.o.o. where he is responsible for applying Enterprise 2.0 concepts into Marg's flagship product BusinessConnect Social. He is an international award-winning software engineer and AI specialist, specialising in machine learning, data mining and autonomous agent strategy learning, with academic and business backgrounds. His professional experience includes load-balancing server design, game development, data mining and intelligent agents design. He is an exciting presenter with great stage presence and insightful ideas. His academic background and business experience with various industry and government institutions make him a profound speaker.

Matjaž Gams is a senior researcher at the Jožef Stefan Institute, Ljubljana, Slovenia. His research interests include artificial intelligence, intelligent systems, intelligent agents, machine learning, cognitive sciences, and information society. His publication list includes 500 items, 70 in scientific journals. He is heading the Department of Intelligent Systems at the Jožef Stefan Institute. He is currently the President of ACM Slovenia and the cofounder of the Engineering Academy, Artificial Intelligence Society and Cognitive Sciences Society in Slovenia. He is an executive contact editor of the journal Informatica and member of the editorial board of several international journals. He headed several major applications in Slovenia, including a virtual agent for the Slovenian employment agency, an expert system controlling the quality of nearly all national steel production and a text-to-speech system in Slovenian, donated to several thousands of users. He also teaches several courses in computer sciences at the graduate and postgraduate levels.