Situation-dependent Utility in Extended Behavior Networks

Matthias Hofmann[1] and Thorben Seeland[1]

Robotics Research Institute, TU Dortmund University, 44221 Dortmund, Germany

Abstract. In this paper, we present a modification of extended behavior networks that enables an agent to learn the relationship between world states and undertaken actions. To this end, we introduce a situation-dependent utility value that is based on the observation of effects after the execution of an action. The utility values serve as bases of multi-dimensional interpolation functions, and supports the revised and extended action selection mechanism to take better actions over time. The evaluation shows that our approach improves action selection. We assess the performance of our system in the RoboCup domain using simulation.

1 Introduction and Related Work

Action selection is an important and well-addressed topic in autonomous robotics, esp. in the RoboCup domain. The simulation leagues are a suitable playground to investigate the performance of high-level behavior. Extended behavior networks provide a powerful and flexible framework for organizing and managing behaviors [1],[2],[3]. Our approach addresses two outstanding weaknesses of extended behavior networks: First, we provide a notion of learning to extended behavior networks, and second, we add knowledge of truth values of propositions over time to the network so that it is able to adapt itself to changing environmental conditions.

There is plethora of other methods that have been used for action selection in the robot soccer domain. For instance, a case-based approach has been applied to robot soccer for coordinated action selection [4]. A vision-based approach in combination with fuzzy reasoning has been investigated in [5]. Decision-tree learning was employed in the simulation league [6].

The remainder of the paper is structured as follows: We outline the main concepts of Extended Behavior Networks along with our changes in section 2. We assess the performance and capabilities of the system in section 3. We conclude the paper in section 4.

2 Extended Behavior Networks

This section briefly describes the main concepts of prior versions of extended behavior networks. Therefore, subsections 2.1 and 2.2 are based on the work of Dorer [1]. Subsection 2.3 covers our changes to the existing system, introducing the concept of a situation-dependent utility value for effects of actions.

2.1 Notations and Conventions

For further reading of the paper, we define the following sets, relations, and operations.

1. $\mathcal{S}$ is the set of world states, $s \in \mathcal{S}$.
2. $\mathcal{P}^+$ is a set of atoms.
3. $\mathcal{P}$ is the set of atoms, and negated atoms $\mathcal{P}^+$.
4. $\mathcal{L}^\wedge$ is a language over $\mathcal{P}$ with logical $\wedge$.
5. $\mathcal{L}$ is a language over $\mathcal{P}$ with logical operations $\wedge$ and $\vee$.
6. Let $\tau : \mathcal{P}^+ \times \mathcal{S} \to [0 \dots 1]$ the fuzzy value of an atom in relation to the world state, with $\tau(\neg p, s) = 1 - \tau(p, s)$, $\tau(p \wedge q, s) = \tau(p, s) \otimes \tau(q, s)$, $\tau(p \vee q, s) = \tau(p, s) \oplus \tau(q, s)$. $\otimes$ is a continuous t-Norm, and $\oplus$ a continuous t-Conorm [7, 8]. Furthermore, $p, q \in \mathcal{P}^+$.
7. Let $\mathrm{def} : \mathcal{L} \to Pot(\mathcal{P}^+)$, and for $l \in \mathcal{L}$, $\mathrm{def}(l)$ the set of all in l used atoms.

2.2 Extended Behavior Networks

Extended Behavior Networks consists of three basic components: Literals, (competence) modules, and goals. A literal is a proposition that receives a (fuzzy) truth value by a function t with respect to the current world state s of the robot. Extended behavior networks utilize literals to formulate conditions for goals and modules, e.g. whether a specific goal has reached. A module contains preconditions Pre, a specific behavior b, and postconditions $Post$. In the course of this paper, b is also called an action. $Post$ consists of effects Eff_j, and a probability ex_j for the effect to become true. A goal comprises a target condition $GCon$. Moreover, there are a static and a situation-dependent importance (ι, r) of the goal, and a relevance condition ($RCon$). Additionally, there is a set of parameters for extended behavior networks:

1. $\gamma \in [0..1]$ controls the influence of activation of modules.
2. $\delta \in [0..1]$ controls the influence of inhibition of modules.
3. $\beta \in [0..1]$ the inertia of activation across activation cycles.
4. $\theta \in [0..a]$ the activation threshold that a module has to exceed to be selected for execution, with a as the upper bound for a module's activation.
5. $\Delta\theta \in [0..\theta]$ the threshold decay.

The action selection mechanism distributes activation over the network. It is called revised and extended action selection mechanism (REASM) based on Maes [9], and consists of the following steps:

1. Calculate the executability e for each module.
2. Calculate the activation a^n for each module in the current execution cycle $n \in \{0, 1, \dots\}$.
3. Combine activation and executability by a nondecreasing function $h : \mathbb{R} \times [0, 1] \to \mathbb{R}$. The value $h(a^n, e)$ is called execution value.

4. If the maximum value $h(a_k^n, e_k)$ of a module k is greater than a threshold value $\theta > 0$, the behavior of module k is executed.
5. If the maximum value of $h(a_k^n, e_k)$ is smaller than θ, θ is decreased by $\Delta\theta$, and n incremented. The mechanism continues with the second step.

For activation spreading, the following rules apply:

1. A module k receives activation ${a_{kg_i}^n}'$ from goal g_i by the effect Eff_j, iff $Eff_j \in \mathrm{def}(GCon_{g_i})$.
2. A module k receives negative activation ${a_{kg_i}^n}''$, iff $\neg Eff_j \in \mathrm{def}(GCon_{g_i})$.
3. A module k receives activation ${a_{kg_i}^n}'''$ from goal g_i by a successor $succ$, iff $Eff_j \in \mathrm{def}(Pre_{succ})$.
4. A module k receives negative activation ${a_{kg_i}^n}''''$ by the goal g_i by a successor $conf$, iff $\neg Eff_j \in \mathrm{def}(Pre_{conf})$.
5. The activation of a module from goals is

$$a_{kg_i}^n = abs\max\left({a_{kg_i}^n}', {a_{kg_i}^n}'', {a_{kg_i}^n}''', {a_{kg_i}^n}''''\right),$$

6. and the overall activation of the module is

$$a_k^n = \beta a_k^{n-1} + \sum_i a_{kg_i}^n.$$

2.3 Introducing Situation-Dependent Utility

Prior versions of extended behavior networks [1, 3] utilize fixed expectation (probability) values ex_j for each effect Eff_j in module k. However, determining ex_j is a difficult task, and is either done by optimization, or manually set by the behavior designer. This requires domain knowledge and experience. Additionally, ex_j is a generalization. Thus, modeling situation-dependency in prior versions of extended behavior networks would require the definition of additional competence modules that model relations between preconditions that are not part of the executability, and effects for each situation. Therefore, the number of competence modules could greatly increase. Although the executability of a module k depends on the world state, the effect of an action that is taken by the agent may greatly rely on the game situation as well. The original version of extended behavior networks does not take this relevant aspect into account.

From a technical perspective, we replace the current probability of an effect to become true, ex_j by a situation-dependent rating value function $ex_j^*(s)$ that express the utility of an action at time t with respect to the world state s. Formally, we define for each effect Eff_j^* a function such that $ex_j^*(s) = \tau^*(Eff_j^*, s)$. In contrast to $\tau(Eff_j, s)$, $\tau^*(Eff_j^*, s)$ estimates the expectation value for each literal of the effects of each module with respect to $\tau(Eff_j, s)$ for all p in Pre. This means that only the subset of the world state is deemed relevant for the effect according to Pre, is used. Figure 1 summarizes the changes in the competence modules of the extended behavior networks.

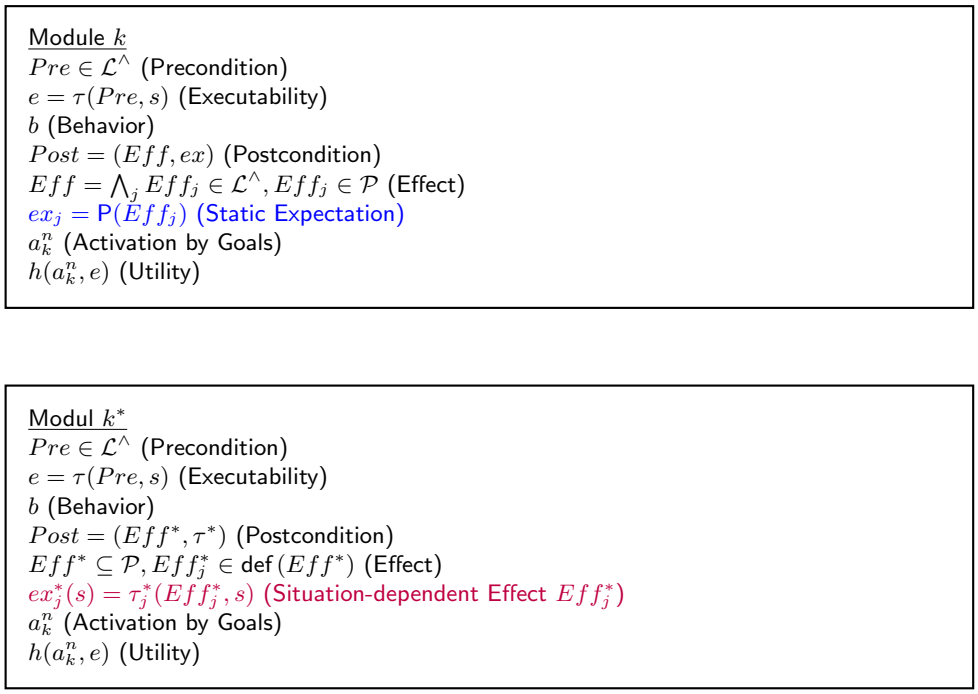


Fig. 1. We compare module k in its prior version shown in [1] with the new module k^*. Here, we replace ex_j with the function τ^*.

The definition of τ^* depends on the one hand on the observation of the effect itself, and on a relevance function r which takes the influence of the time t into account. This way, we observe the effect of the action over a period and calculate the value. Figure 2 exemplifies an observation function while Figure 3 shows an example of a relevance function r. In this particular example, we assume that a soccer agent passes the ball to a team member. We observe that the ball rolls too far. Altogether, the utility value $\tau^*_{Pass}(BallNearTeammate, s)$ is calculated by using $\tau_B = \tau(BallNearTeammate, s)$:

$$\tau^*_{Pass}(BallNearTeammate, s) = \frac{\sum \tau_B(t) \cdot r(t)}{\sum r(t)} \tag{1}$$

Since we know the preconditions that are relevant for the current action, and the corresponding utility values, we need a mathematical method to keep this knowledge for reuse and update. Hence, we choose interpolation functions to save the relationship between preconditions (the game situation), actions, and the corresponding utility. This way, we are able to improve the knowledge of an agent over time by recording and rating the effects on each action in many situations. Moreover, after a training phase, unseen game situations can

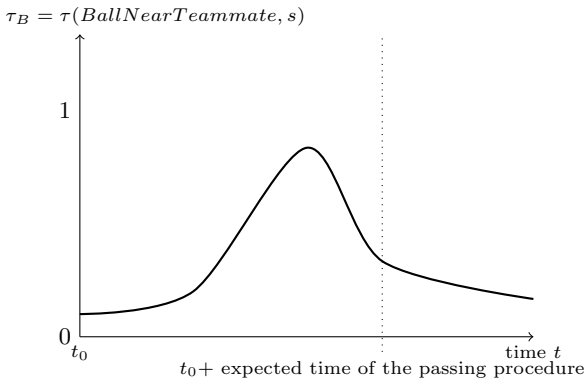


Fig. 2. We calculate $\tau^*_{Pass}(BallNearTeammate, s)$ by observing the effect of the action over time.

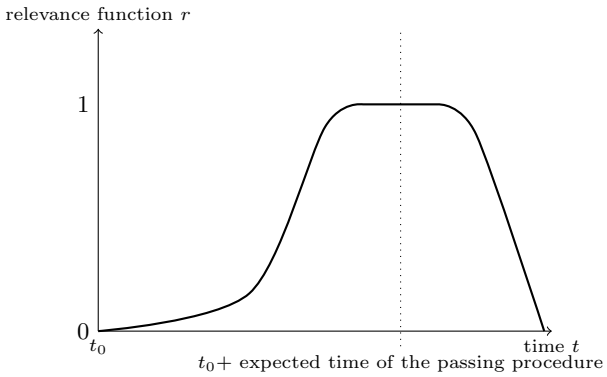


Fig. 3. We utilize the relevance function r for weighting the influence of t.

be assessed based on experience of previously seen game situations, and their utilities of actions. Finally, we are able to revise the knowledge of an agent by updating the system if something in the environment of the agent changes. To this end, outdated bases[1] can be removed from the agent's knowledge, and bases that are close to each other can be merged to a single base. The usage of the aforementioned aspects depend on the application, and are part of the design choices of the behavior engineer.

In this paper, we utilize radial basis functions based on Lowe [10]:

$$s_{RBF}(\underline{x}) = \sum_{j=1}^{m} \lambda_j \phi(||\underline{x} - \underline{y_j}||), \underline{x} \in \mathbb{R}^n. \tag{2}$$

[1] Base means a supporting point of the function.

The radial basis functions determine the weights λ_i by combining conditions of the interpolation with the interpolation function s_{RBF}. We solve the following equation for the application of radial basis functions:

$$\begin{pmatrix} f_1 \\ \vdots \\ f_m \end{pmatrix} = \begin{pmatrix} A_{11} & \cdots & A_{1m} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mm} \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_m \end{pmatrix} \tag{3}$$

with

$$A_{ij} = \phi(||\underline{x_i} - \underline{y_j}||). \tag{4}$$

Radial basis functions offer different types of interpolation functions ϕ. The most popular ones are the Gaussian ($\phi(r) = e^{-0.5\frac{r^2}{r_0^2}}$), multi-square ($\phi(r) = \sqrt{r^2 + r_0^2}$), and the thin-splate ($\phi(r) = r^2 \cdot \log(\frac{r}{r_0})$) method by Dunchon [11].

We implemented the radial basis interpolar based on Burkardt [12][2]. Equation 3 is solved by singular-value decomposition according to [13].

3 Evaluation

The evaluation exclusively addresses our changes to extended behavior networks. For evaluation, it is required to repeat the shown experiments many times in order to acquire knowledge about the utility of actions in the extended behavior network. Moreover, robotics has to deal with many uncertainties that may influence the results of the experiments. Therefore, it is in our case beneficial to use ground truth data in order to validate the concepts of our work. Due to the aforementioned reasons, it is very difficult to evaluate the system on robotic hardware. Therefore, we opted for an evaluation of the system with the SimRobot simulator based on the 2016 version of the B-Human framework [14][3].

This section is structured as follows: First, we describe the different types of extended behavior networks that are used in evaluation (see subsection 3.1). Second, we illustrate the initialization process, and two experimental setups (see subsection 3.2). Finally, the results are presented in 3.3.

3.1 Types and Instances of the Extended Behavior Network

In this section, we describe the instance of the extended behavior network that we use for evaluation. First, we define the following literals:

- *angleToBall*: Describes the relative angle of the robot to the ball. The fuzzy value is 0 when the agent is heading the opposite diretion of the ball,a nd vice versa.
- *ballValidity*: Confidence of the ball position on the field.

[2] Source code: `https://people.sc.fsu.edu/~jburkardt/cpp_src/rbf_interp_nd/rbf_interp_nd.html`

[3] https://www.b-human.de/downloads/publications/2016/coderelease2016.pdf

- *distanceToBall*: Distance of the robot to the ball. To define the fuzzy variable, we use the Euclidian, and a maximum distance that results into 0.
- *ballInsideGoal*: Becomes true if a goal was scored.
- *ballPosition*: Models whether the ball is in favorable position on the field and depends on relative positions to the own and opposite goal.
- *ballPossession*: Models whether a team is in ball possession.
- *danger*: Models whether an opponent blocks the ball.
- *passFreeTo#* Models whether the way to the team member is free for passing.
- *goodPassDistanceTo#*: Expresses whether the team member is in a good position for receiving a pass depending on the distance to the passing agent.
- *matePositionOf#*: Expresses whether the team member is in a favorable position on the field. This corresponds to the rating of the ball position on the field.
- *shotFree*: Models whether the way to the goal is free for kicking.
- *goodShotDistance*: Models whether the distance to the opponent goal is suitable for kicking.

The goals of the extended behavior networks are outlined in Table 1 while the modules are shown in Table 2. The parameters of the radial basis functions, and extended behavior networks are listed in Table 3. We choose the settings according to our initial experience with the system.

Goal 1	
GCon:	$\neg danger \wedge ballPossession$
RCon:	$\neg ballPossession$
ι:	0.8
Goal 2	
GCon:	$\neg danger \wedge ballPossession$
RCon:	$\neg ballPosition \vee danger \vee \neg ballPossession$
ι:	0.8
Goal 3	
GCon:	$ballInsideGoal$
RCon:	$trueValue$
ι:	1

Table 1. Goals of the extended behavior network.

While the structure of the extended behavior network is the same for all experiments, we define three different types of extended behavior networks: The first type is an instance of an *extended behavior network without our modifications*, i.e. with fixed ex_j. The second type is an extended behavior network with *situation-dependent utility values and offline training*. The third type is an *extended behavior network with situation-dependent utilities, and online adaptation*. This is simply done by constantly adding and revising the knowledge that has previously been acquired with the interpolation function.

shoot
Pre: $shotFree \wedge angleToBall \wedge ballValidity \wedge distanceToBall \wedge goodShotDistance$
Post
Eff: $ballInsideGoal \wedge ballPosition \wedge danger \wedge \neg distanceToBall$
(ex_j): $(0.8, 0.8, 0.5, 0.6)$
dribble
Pre: $angleToBall \wedge ballValidity \wedge distanceToBall$
Post
Eff: $angleToBall \wedge ballValidity \wedge distanceToBall \wedge \neg danger \wedge ballPosition$
(ex_j): $(0.6, 0.6, 0.6, 0.6, 0.7)$
passTo#
Pre: $passFreeTo\# \wedge goodPassDistanceTo\# \wedge matePositionOf\#$
Post
Eff: $ballPossession \wedge \neg danger \wedge ballPosition$
(ex_j): $(0.8, 0.5, 0.7)$

Table 2. Modules of the extended behavior network.

radial basis function interpolation	
$maxSupportPoints$	50
$minDistance$	0.05
Extended behavior network	
$h(a, e)$	$a \cdot e$
γ	0.7
δ	0.6
β	0.0
θ	0.3
$\Delta\theta$	0.1

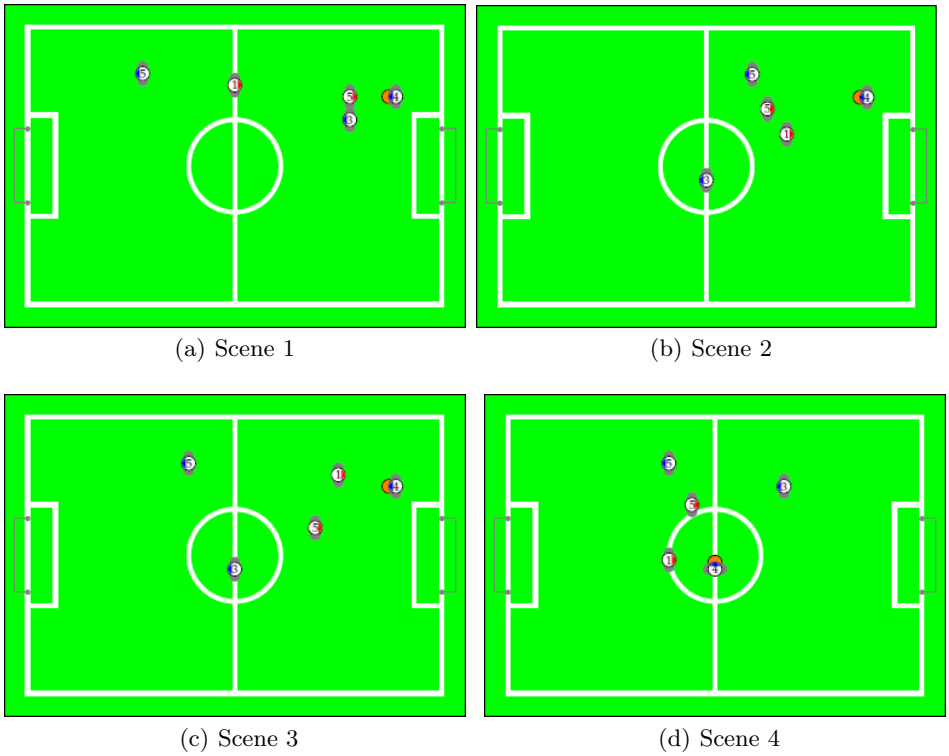
Table 3. Strategy parameters of the extended behavior networks and the radial basis interpolation method.

3.2 Experiments

Before we start with evaluation, it is required to initialize the extended behavior networks with situation-dependent utilities. This is done as follows: We create 20 scenes, where 5 robots and the ball is being placed on predefined, but varying positions per scene onto the simulated SPL[4] field. The setup consists of one robot which is using an extended behavior network, and taking actions. Moreover, there are two team members, and two opponent players, which are fixed in their positions not doing anything. The agent executes ten times each of the actions on each scene. The observed utility and precondition values are recorded for later validation in the experiments.

Experiment 1 validates the system with respect to the initial conditions. To this end, we use scenes number 1 to 3 (see Figure 4). In a first step, we determine the optimal action for each scene. This way, each possible action

[4] Standard Platform League



(a) Scene 1 (b) Scene 2

(c) Scene 3 (d) Scene 4

Fig. 4. Scenes used for evaluation. One team composes of the robots red 1 and red 5, the other team consists of robots blue 3, blue 4, and blue 5.

(*dribble*, *passTo3*, *passTo5* and *shoot*) is executed ten times on scenes 1 to 3. We record the fuzzy values of each literal of the postconditions, and calculate a and h. We use the average of these values for assessment. The best actions of each scene serve as a reference for comparison for the various extended behavior network types that have been previously trained. This way, we count how often the extended behavior network is able to conduct the optimal action. By using the aforementioned procedure, we find the best actions for scene 1 is *passTo3*, for scene 2 *passTo5*, and for scene 3 *shoot*.

In *experiment 2*, we change the environment by significantly increasing the ground friction in the simulation. As in the first experiment, we use scenes 1 and 2 again, but use scene 4 instead of scene 3 (see Figure 4). Due to the higher friction, the ball is not able to reach the goal after kicking it. The further steps are the same as in experiment 1. The best action for scene 1 is *passTo3* and for scene 2 and scene 4 *passTo5*.

The adaptive, situation-dependent version of the extended behavior network is executed 30 times in each experiment in order to check convergence. The other types of the extended behavior networks are executed ten times.

3.3 Results

The following Table 4 lists the performance of each interpolation method in comparison to each type of extended behavior networks in both experimental setups with different initial parameters. Each interpolation method is by default initialized pessimistically (0.0), neutral (0.5), or optimistically (1.0) before the knowledge about actions and their effects are being introduced.

It is cognizable that the type and initialization of the interpolation methods play a key role in the performance of selecting the best action in the particular scenes. By trend, interpolation method works better if they are initialized neutral or optimistically. Moreover, the interpolation methods should ideally respect the interval [0..1]. Naturally, the performance of the approach depends on the number of scenes that are being used for training the situation-dependent extended behavior networks.

	Experiment 1			Experiment 2		
	Scene 1	Scene 2	Scene 3	Scene 1	Scene 2	Scene 4
unchanged EBN (10 attempts per scene)						
fixed	0	0	10	0	0	0
situation-dependent estimation with interpolation (10 attempts per scene)						
Gaussian (0.0)	5	10	2	5	9	9
Gaussian (0.5)	0	6	10	0	9	3
Gaussian (1.0)	0	0	10	0	0	0
multiquad. (0.0)	1	0	10	0	6	7
multiquad. (0.5)	0	1	10	0	2	2
multiquad. (1.0)	0	0	10	0	0	0
thin-plate-spline (0.0)	2	8	1	3	0	1
thin-plate-spline (0.5)	1	0	10	0	0	1
thin-plate-spline (1.0)	0	1	10	0	2	2
situation-dependent, adaptive estimation with interpolation (30 attempts per scene)						
Gaussian (0.0)	29	30	0	27	30	30
Gaussian (0.5)	0	11	30	0	30	0
Gaussian (1.0)	0	0	30	0	0	0
multiquad. (0.0)	0	0	30	0	0	0
multiquad. (0.5)	0	0	30	0	30	30
multiquad. (1.0)	0	0	30	0	0	1
thin-plate-spline (0.0)	14	30	0	24	28	28
thin-plate-spline (0.5)	0	0	28	3	1	0
thin-plate-spline (1.0)	0	0	30	0	30	30

Table 4. Number of taking the best action grouped by experiment and type of the extended behavior network.

It can be seen that the best performance is achieved with the Gaussian interpolation method. With respect to the situation-dependent, adaptive version of the extended behavior network, the system was able to select the best actions

despite scene 3 in experiment 1. It has to be mentioned that the performance of the extended behavior network with fixed expectation values depends on chosen values. The parameters used in Table 2 are based on own experiences and trials. Finding the best action for most of the scenes with fixed expectation values would require a parameter optimization procedure.

Figure 5 shows an example of the convergence capability of the situation-dependent, adaptive extended behavior network. The graph shows very fast convergence towards the optimal action in experiment 2 in scene 1. The result is consistent with the fact that the system was able to select the best action in this scene 24 times by using thin-splate interpolation and pessimistic initialization.

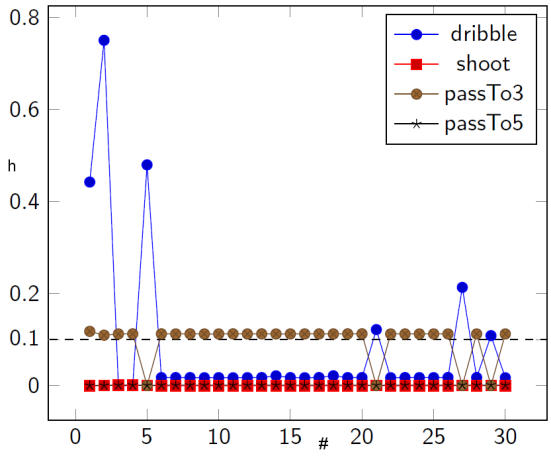


Fig. 5. Exemplification of the convergence of the adaptive, situation-dependent utility extended behavior network. The graph refers to exeperiment 2, scene 1. The interpolation method is thin-splate spline, and pessimistically. The optimal action is passTo3.

4 Conclusion and Future Work

In this paper, we have shown a modification to extended behavior networks by Dorer. We exchanged static expectation values by situation-dependent utility functions for the effects of actions. Our evaluation shows first promising results, and suggests the application of the adaptive, situation-dependent version of extended behavior networks.

For future work, we will integrate other kinds of interpolation methods. Interesting approaches are Shephard's interpolation [15], and the nearest-neighbor interpolation. Another topic is the application of our work to the entire decision making of a simulated soccer robot. This would make it possible to observe the effects of the behavior over entire games, and consequently, makes it possible to

compare the performance of various methods on a game result basis. Finally, a lot of data can be recorded to serve as an input for training networks. We will also work towards an application of the proposed method on robotic hardware.

References

1. Dorer, K.: Behavior networks for continuous domains using situation-dependent motivations. In: IJCAI. Volume 99., Citeseer (1999) 1233–1238
2. Dorer, K.: Motivation, Handlungskontrolle und Zielmanagement in autonomen Agenten. PhD thesis, PhD thesis, Albert-Ludwigs-Universität Freiburg, Freiburg (1999)
3. Dorer, K.: Extended behavior networks for behavior selection in dynamic and continuous domains. In: Proceedings of the ECAI workshop Agents in dynamic domains, Valencia, Spain, Citeseer (2004)
4. Ros, R., Arcos, J.L., de Mantaras, R.L., Veloso, M.: A case-based approach for coordinated action selection in robot soccer. Artificial Intelligence **173**(9) (2009) 1014 – 1039
5. Wu, C., Lee, T.: A fuzzy mechanism for action selection of soccer robots. Journal of Intelligent and Robotic Systems **39**(1) (2004) 57–70
6. Konur, S., Ferrein, A., Ferrein, E., Lakemeyer, G.: Learning decision trees for action selection in soccer agents. In: In Proc. of Workshop on Agents in dynamic and real-time environments. (2004)
7. Zadeh, L.A.: Fuzzy logic and approximate reasoning. Synthese **30**(3-4) (1975) 407–428
8. Gerla, G.: Fuzzy logic: mathematical tools for approximate reasoning. Volume 11. Springer Science & Business Media (2013)
9. Maes, P.: The dynamics of action selection. In: Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 2. IJCAI'89, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1989) 991–997
10. Lowe, D., Broomhead, D.: Multivariable functional interpolation and adaptive networks. Complex syst **2** (1988) 321–355
11. Duchon, J.: Splines minimizing rotation-invariant semi-norms in sobolev spaces. In: Constructive theory of functions of several variables. Springer (1977) 85–100
12. Press, W.H.: Numerical recipes 3rd edition: The art of scientific computing. Cambridge university press (2007)
13. Liesen, J., Mehrmann, V.: Die singulärwertzerlegung. In: Lineare Algebra. Springer (2015) 313–321
14. Laue, T., Spiess, K., Röfer, T.: Simrobot - a general physical robot simulator and its application in robocup. In Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y., eds.: RoboCup 2005: Robot Soccer World Cup IX. Number 4020 in Lecture Notes in Artificial Intelligence, Springer; http://www.springer.de/ (2006) 173–183
15. Gordon, W.J., Wixom, J.A.: Shepard's method of "Metric Interpolation" to bivariate and multivariate interpolation. Mathematics of Computation **32** (1978) 253–264