Extended Behavior Networks for Behavior Selection in Dynamic and Continuous Domains

Klaus Dorer[1]

Abstract. In this paper we present how behavior networks can be extended to model behavior selection of agents in dynamic and continuous domains. More precisely, the focus is on a mechanism for selection of concurrent behaviors by explicit representation of resources a behavior makes use of. Further it describes how the behavior selection process can be coupled with behavior execution in continuous domains. Behaviors may be influenced by the decidedness of the behavior selection as is the case in biological systems. Empirical results in the RoboCup domain show that both extensions improve the performance of soccer playing agents significantly.

1 INTRODUCTION

Behavior selection in dynamic domains is complicated by the fact that the deciding agent has limited amount of time for its decision before the situation has changed. This is usually addressed by improving the speed of the decision mechanism for dynamic domains. However, this does not take into account the possibility to improve the agent's performance by conducting multiple actions concurrently. Moreover, in some domains, concurrent actions are not simply a way of improving agents' performance, but they can become a necessary condition to perform tasks. For driving a car, for example, it is at least necessary to turn the steering wheel and accelerate or break concurrently. Most action selection mechanisms result in a single action to be performed. To overcome this limitation two possibilities exist: (1) either the decision mechanism is provided with a (usually huge) set of combined actions like 'turnLeft', 'turnLeftAndBreak', 'turnLeftAndAccelerate', etc., or (2) the decision mechanism decides on more complex behaviors like 'drive' that combine actions appropriately leaving the detailed decision to the execution module of the agent. The later is usually the preferred option accepting the disadvantage of more complex behaviors.

This situation is even complicated in continuous domains, where actions may be performed with variable strength, degree, duration. For example, the 'turnLeft' action of the above example would have to be split into 'turnLeft5Degrees', 'turnLeft10Degrees', ... Again it is usually preferred to put the decision of the degree with which an action is performed into the low level behavior execution module. Behavior selection and behavior execution is usually strictly separated.

Extended behavior networks [2, 3] (EBNs) are a means to carry out behavior selection in dynamic and continuous domains. They extend original behavior networks [5, 6, 7] by explicit representation of goals with dynamic, i.e. situation-dependent, utility function and by the introduction of continuous state-propositions to represent attributes of continuous domains. In this paper we describe how EBNs are able to select multiple behaviors in a single decision cycle to be performed concurrently. We also show how the decidedness of behavior selection can be used to control the intensity with which behaviors are performed as is the case in biological systems [4].

The remainder of this paper is organized as follows: Section 2 describes the basic concept of behavior selection using extended behavior networks. In section 3 this concept is extended by introducing concurrent behavior selection. Section 4 explains how behaviors can be parametrized by the decidedness of the behavior selection. In section 5 we summarize empirical results gained in the RoboCup simulated soccer domain. Finally, in section 6 we discuss possible future work directions before concluding.

2 EXTENDED BEHAVIOR NETWORKS

Extended behavior networks [5, 2, 3] have been introduced to combine reactive and goal-directed behavior selection in dynamic and continuous domains. This section gives a short overview on the structure of extended behavior networks and the behavior selection mechanism using activation spreading. The next two sections will then describe two further extensions of EBNs, selection of concurrent behaviors and behavior parametrization, to improve action selection in dynamic and continuous domains.

2.1 Network Definition

Extended behavior networks consist of goals and so called competence modules that are linked into a network.

Definition 1 *A* goal *consists of a tuple (*GCon, ι, RCon*) with*

- GCon *the* goal condition *(conjunction of propositions, i.e. possibly negated atoms), the situation in which the goal is satisfied,*
- $\iota \in [0..1]$ *the (static)* importance *of the goal,*
- RCon *the* relevance condition *(conjunction and disjunction of propositions), i.e. the situation-dependent (dynamic) importance of the goal.*

Definition 2 *A* competence module *consists of a tuple (*Pre, b, Post, a*) with*

- Pre *the* precondition *and* $e = \tau_P(Pre, s)$ *the* executability *of the competence module in situation* s *where* $\tau_P(Pre, s)$ *is the (fuzzy) truth value of the precondition in situation s;*
- b *the* behavior *that is performed once the module is selected for execution;*

[1] Living Systems GmbH, Humboldtstrasse 11, D-78168 Donaueschingen Germany, email: kdorer@living-systems.com

- Post *a set of tuples (Eff, ex), where Eff is an expected effect (a proposition) and $ex = P(Eff|Pre)$ is the* probability *of Eff getting true after execution of behavior b,*
- a *the* activation $\in \mathbb{R}$*, representing a notion of the expected utility of the behavior (see below).*

Definition 3 *An* extended behavior network *(EBN) consists of a tuple $(\mathcal{G}, \mathcal{M}, \Pi)$, where $\mathcal{G}$ is a set of goals, $\mathcal{M}$ is a set of competence modules and Π is a set of* parameters *that control activation spreading (see below)*

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

2.2 Behavior Selection

The decision of which behavior to adopt should be based on the the expected utility out of executing such behavior. In EBNs, the expected utility of a behavior is approximated by a mechanism called *activation spreading*. The competence modules are connected to the goals and other competence modules of the network. Across those links activation is spread from the goals to the competence modules and among competence modules.

A competence module receives *activation* directly from a goal if the module has an effect that is equal to a proposition of the goal condition of that goal. The amount of activation depends on the probability ex of the effect to come true and the utility of the proposition in the goal condition. Activation from a goal represents the expected utility of the behavior to reach that goal. The utility of propositions that are part of a goal condition can be directly derived from the importance and relevance of the goal [2].

A competence module is *inhibited* by a goal if it has an effect proposition that is equal to a proposition of the goal condition and one of the two propositions is negated. Inhibition represents negative expected utility and is used to avoid the execution of behaviors that would lead to undesired effects.

A competence module x is linked to another competence module y if x has an effect that is equal to a proposition of the precondition of y. y is called a *successor* module of x. Module x gets activation from the successor the amount of which depends on the utility of the precondition and the probability of the effect to come true. The utility of propositions that are not part of a goal condition is not available directly. It can be determined indirectly using the activation of the containing module and the truth value of the proposition [2]. In this way, unsatisfied preconditions get implicit sub-goals of the network. Their utility directly depends on the utility of the competence module itself.

Finally a competence module x is linked to another competence module y if it has an effect that is equal to a proposition of the precondition of y and one of the two propositions is negated. y is called a *conflictor* of x, because it has an effect that destroys an already satisfied precondition of x. Again, a conflictor link from x to y is inhibiting (negative activation) to avoid undesired effects.

The activation of a module k at time t is then the sum of all incoming activation and the previous activation of the module decayed by β (defined in the set of parameters Π):

$$a_k^t = \beta a_k^{t-1} + \sum_i a_{kg_i}^t, \tag{1}$$

where $a_{kg_i}^t$ is the maximal activation module k receives at time t from goal g_i to which the module is linked directly or indirectly across incoming successor and conflictor links of other competence modules. For more details on activation spreading see [2, 3].

Behavior selection is done locally in each competence module in a cycle containing the following steps:

1. Calculate the activation a of the module.
2. Calculate the executability e of the module.
3. Calculate the execution-value $h(a, e)$, which is a monotonically increasing function of the activation and executability of a module (calculated e.g. by multiplication) [2].
4. If the highest value $h(a, e)$ of all competence modules lies above a threshold θ (defined in the set of parameters Π), execute the corresponding competence module's behavior b, reset θ to its original value in Π and go to 1.
5. Otherwise reduce θ by $\Delta\theta$ (also defined in Π) and go to 1.

In the first cycle of activation spreading, only competence modules that directly have links to goals get activation. Activation by successor and conflictor links is zero at that time, because no module has activation initially. So only behaviors that directly satisfy a goal will be taken into account for selection. In the second cycle also competence modules get activation that may reach the goal within two actions. They got activation through successor and conflictor links to modules that got activation in the first cycle. The more cycles activation is spread the longer is the (timely) horizon of action sequences taken into account that lead to goals. This cyclic approximation of expected utility of a behavior in EBNs is somewhat similar to a growing horizon when solving a Markov Decision Process (see e.g. [1]). For behavior selection a good trade-off is therefore necessary between running enough activation spreading cycles to look far enough into the future and acting fast enough.

3 CONCURRENT BEHAVIOR SELECTION

A shortcoming of the above described mechanism for behavior selection is that behavior selection results in a single behavior to be performed at any time. Humans on the other side are able to performed well trained behaviors concurrently if they do not use the same resources [10, 8]. A typist, for example, is able to type a text she is reading and speak aloud a text she is listening to at the same time [10]. Performing behaviors that use the same resources usually ends with no behavior performed successfully. For instance, when a human is undecided between the words 'close' and 'shut' it may end up pronouncing a non existing word 'clut' [9]. The common resource 'language processing' may not be used by multiple behaviors. It may, however, be influenced by multiple goals.

Sequential behavior selection of Maes networks [5] avoids the problem of resource conflicts. The disadvantage is on the one side a reduced performance in domains where multiple behaviors may be performed in parallel. On the other side it may prevent the completion of tasks completely for which concurrent behavior execution is essential (like car driving).

To perform multiple behaviors in parallel the agent needs knowledge about the resources used by the behaviors. The definition of competence modules and extended behavior networks has therefore to be extended with the notion of resources.

Let $\mathcal{R}$ be the set of all resources and $\tau_R : \mathcal{R} \times \mathcal{S} \rightarrow \mathbb{R}^+$ a function that assigns to each element of $\mathcal{R}$ an amount of available resources in the domain in state s. The function $\tau_U : \mathcal{M} \times \mathcal{R} \times \mathcal{S} \rightarrow \mathbb{R}^+$, with $\mathcal{M}$ the set of all competence modules, defines the expected amount of resource units used by the corresponding competence module in state s to reach its effects.

Definition 4 *A* resource node *is a tuple (res, g, θ_{Res}) with*

- $res \in \mathcal{R}$ *the* resource *represented by the node,*
- $g \in \mathbb{R}^+$ *the amount of* bound resource units, *i.e. units that are bound by a currently executing competence module and*
- $\theta_{Res} \in]0..\theta]$ *the resource specific* activation threshold *(where θ is the global activation threshold of the network).*

The definition of a competence module can then be extended to:

Definition 5 *A* competence module k *consists of a tuple (*Pre, b, Post, Res, a*) with* Pre, b, Post *and a as defined above and* Res *is a set of* resources $res \in \mathcal{R}$ *used by behavior b. $\tau_U(k, res, s)$ is the situation-dependent amount of resource units expected to be used by behavior b.*

Definition 6 *An* extended behavior network *EBN consists of a tuple ($\mathcal{G}, \mathcal{M}, \mathcal{U}, \Pi$), where $\mathcal{G}$ is a set of goals, $\mathcal{M}$ a set of competence modules, $\mathcal{U}$ a set of* resource nodes *and* Π *a set of* parameters *(see section 2).*

To coordinate concurrent behaviors the competence modules of $\mathcal{M}$ are connected with resource nodes in $\mathcal{U}$. A competence module has for each resource $res \in Res$ a link to the corresponding resource node. This link enables the competence module to check the availability of the resource. Concurrent behavior selection may therefore be calculated locally in each competence module. It is done in a cycle containing the following steps:

1. Calculate the execution-value h of the module as described above.
2. For each resource res used by competence module k, starting with the previously unavailable resource
   (a) Check if h exceeds the activation threshold θ_{Res_i} of the corresponding resource node.
   (b) Check if enough resource units are available in the current situation, i.e. check if $\tau_U \leq \tau_R(res, s)$. If so, bind the resource-units, i.e. increase the number of used resource-units of the resource node by the number of expected units the behavior will use.
3. If all tests in 2 succeeded
   (a) Execute the corresponding behavior.
   (b) Reset the activation thresholds of all resources used.
4. Release all bound resource-units, i.e. reduce the number of bound resource units of the resource node by the number of previously bound units.
5. Repeat from 1.

The activation threshold θ_{Res_i} ensures that the competence module with highest execution-value will be performed. θ_{Res_i} linearly decreases over time so that eventually a module exceeds the threshold and may be performed. If modules have equal execution-values in a range of $\Delta\theta$, the threshold reduction, the module that first binds the resource is performed. If the execution of the module with highest activation value is prevented by a missing resource, another module with less activation not using the missing resource may be performed. Modules with a disjunct set of resources Res may be performed concurrently.

Besides allowing concurrent behavior selection, this algorithm overcomes another limitation of original behavior networks. Behavior selection has previously been done by selecting the most active executable competence module for execution. Unfortunately, this information can not be calculated locally in a competence module. Therefore, the process of action selection could not be calculated distributively in each competence module. By introducing resource nodes, a competence module is now able to perform action selection locally. All information is available within the node or within linked nodes. The information a competence module gets across a link to a resource node is the current activation threshold and the number of bound resource units. Information a resource node gets from a competence module using the resource includes the number of resource units to bind and release and when to reset activation threshold.

4 BEHAVIOR PARAMETRIZATION

Most decision mechanisms for agents only have influence on the decision which behavior the agent should perform, but not on the behavior execution itself. In biological systems, however, the determinedness of a decision has influence on the execution of a behavior. "Intensity and endurance of an activity is determined by the volition strength of the goal intention"[4]. Of course different intensities (i.e. strength/degree of execution) of the same basic behavior could also be modeled by distinguishing these as different behaviors and let the decision mechanism decide between those. Obviously, at least in continuous domains, this would increase the number of behaviors considerably making the decision process much more complex. Therefore it would be desirable if the determinedness of the agent's decision would directly influence the execution of the behavior itself. The behavior 'run to ball' of a soccer agent, for example, could be more or less intens depending on the determinedness of the agent to run. The higher the expected utility and the executability of the behavior the more effective it should be to spend resources (stamina) on this behavior. An adequate measure for determinedness in extended behavior networks is the execution-value h of a competence module (see section 2.2). It reflects the expected utility for reaching the goals of the agent as well as the executability of the behavior with respect to the situation.

The problem of using the execution-value is that its absolute value depends on the goals defined in the behavior network. This is because h is a function of the sum of all activation received by the goals it is contributing to directly or indirectly. In an extreme case all effects of a behavior might be defined as goals resulting in a high execution-value. In another network, none of the effects might be defined as goals and the module only receives activation indirectly through other modules. A parametrized behavior on the other side should be independent on the specific network architecture. It is therefore necessary to normalize the execution-value adequately. Following we describe three approaches to map execution-values to the codomain of $[0..1]$.

One obvious approach to normalize the execution-value is to divide it by the number of goals $|\mathcal{G}|$ of the behavior network. However, $|\mathcal{G}|$ is not available within a competence module. A competence module only knows the number of goals it (directly or indirectly) receives activity from. Normalization by using division by the number of goals violates the locality principle and is therefore inappropriate.

Another approach is to use the maximal ($\hat{h}$) and minimal ($\check{h}$) execution-value. It can be calculated locally within a competence module. The influence parameter p of a module can then be calculated as

$$p = \frac{h - \check{h}}{\hat{h} - \check{h}} \tag{2}$$

where h is the current execution-value of the competence module. This approach, however, is vulnerable to extremely high or low execution-values.

This does not matter if instead of extreme values the distribution of execution-values is taken into account. Assuming that execution-values are normally distributed it is enough to calculate mean and standard deviation of the execution-values. Mapping execution-values to an influence parameter p is then done by

$$p = \begin{cases} 0 & : \quad h < \mu - k \cdot s \\ \frac{h-(\mu - k \cdot s)}{2k \cdot s} & : \quad \mu - k \cdot s \leq h \leq \mu + k \cdot s \\ 1 & : \quad \mu + k \cdot s < h \end{cases} \tag{3}$$

where k defines the range of the normal distribution that is mapped to the interval $[0..1]$. The calculation of μ and s can be done incrementally:

$$\mu_{n+1} = \mu_n + \frac{h - \mu_n}{n+1} \quad and \tag{4}$$

$$s^2_{n+1} = (n+1) \cdot (\mu_{n+1} - \mu_n)^2 + \frac{(n-1) \cdot s^2}{n} \tag{5}$$

Section 5.2 presents empirical results of behavior parametrization gained in the RoboCup domain.

5 EMPIRICAL RESULTS

Empirical tests have been conducted in the RoboCup simulated soccer environment. In this domain agents represent soccer players. Two Teams of eleven soccer players each play against each other in a simulated dynamic and continuous soccer domain.

The domain is dynamic from the perspective of a single agent, because 21 other agents change the domain without this agent doing anything. Also the decision cycle within which an agent has to decide is quite short (100ms). Within one decision cycle an agent may decide for concurrent actions. Dashing, kicking or turning the agent's body may be done concurrently with turning the agent's head and talking to other agents. The RoboCup domain is therefore quite well suited for testing concurrent behavior selection.

The domain is continuous in most of the underlying attributes. Examples are the position and velocity of players and the ball and the view and body direction of the agents. Also most actions of the agents are continuous. Dashing is done with variable strength, turning with continuous momentum and kicking with continuous strength and direction. This makes the RoboCup domain an ideal testbed for behavior parametrization.

5.1 Concurrent Behavior Selection

Section 3 explained how extended behavior networks are able to decide on multiple concurrent behaviors. This enables the agent to reach a goal faster or to pursue multiple goals at once. This should lead to improved behavior control of the agent especially in dynamic domains where success also depends on the time an agent needs to decide and act.

Since version 5 of the RoboCup-soccerserver, commands can be executed concurrently, if they do not use the same resources. A `say`-command, for example, can be executed concurrently with a `kick`-, `dash`-, or `turn`-command and a `turn_neck`-command. The concurrent execution of such actions should improve the speed and reactivity of an agent.

This has been examined in a series of 30 soccer-games. Two identical teams of 11 agents played against each other. The only difference was that one team used concurrent behavior selection, while the other team used serial action selection. For the serial team only the action with highest execution-value within a cycle was executed. The concurrent team's agents were able to execute communication, head turning and running or kicking actions concurrently. An example for competence modules the behaviors of which may be performed concurrently is shown in figure 1. τ_R has been defined independent of the situation as $\tau_{legs} = 2$, $\tau_{neck} = 1$ and $\tau_{mouth} = 1$. Since no commands using legs may be performed concurently, τ_U was set to 2 for all behaviors using legs.

The soccer agents turned their head in direction of the ball in case the ball left the visible area of the agent (mindBall). This way the agent can run in an angle of up to 135° relative to the ball and keeping it in the visible area. Without turning the head this would only be 45°. This is especially useful for all positioning behaviors. An agent is only able to run forward and backward in body direction. If, for example, an offender positions itself in the middle of the field while the ball is on the wing it can run towards the goal while keeping the head turned to the ball. An agent that runs and turns the head in consecutive cycles is much slower than an agent that is dashing each cycle and turns its head concurrently. Separate turning of the head relative to the body was performed in about 8% of all cycles. This is not surprising since turning the head is only necessary once the ball is close to leave the visible area.

Also the agents communicated to each other their position and positions of some other players (sayPosition). The number of cycles an agent can communicate is restricted to 4% of all cycles by the soccerserver to restrict the bandwidth of communication. Only one agent is allowed to say something every second cycle in the server version 7 used for the experiments. The agents used a simple round robin scheduling that effectively allowed an agent to talk each 22 cycles. Again the agents of the concurrent team were able to talk while running or kicking. The agents of the serial team only talked if that behavior had higher activation as all other behaviors.

Since separate turning of the head was done in 8% and communication in 4% of the simulation cycles, concurrent behavior selection effectively only took place in 2% of the cycles. Despite this, the team using concurrent behavior selection scored significantly[2] more goals than the team using serial behavior selection (see table 1).

	serial	parallel	p ($n = 30$)
Mean no of goals	2.4	4.3	< 0.001

Table 1. Comparison of serial and parallel behavior selection of EBNs in the RoboCup domain.

[2] two samples t-test with $\alpha = 0.01$.

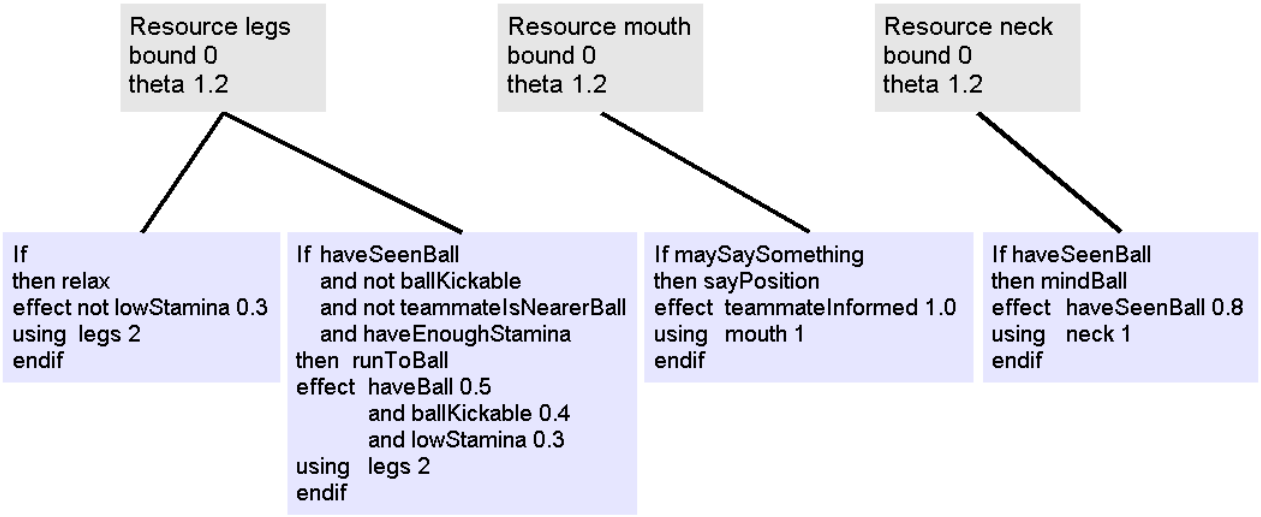

Figure 1. Parts of the network used for concurrent behavior selection in the RoboCup domain. Modules runToBall, sayPosition and mindBall may be performed concurrently. Modules relax and runToBall use the same resource legs and may not be performed concurrently.

5.2 Behavior Parametrization

In section 4 we described how the execution of behaviors may be influenced by the decidedness of the action selection. This can ensure that the execution of a behavior is more appropriate to the current situation. The intensity of behavior execution can be adjusted to the importance of the current situation. The usage of resources is focused to these situations.

These effects can be shown by experiments in the RoboCup domain. Agents have limited stamina for running on the soccer field. They have to make pauses in order to recover from running. If an agent runs out of stamina it gets very slow. The faster an agent runs the more stamina is consumed. For the experiments the 'run to ball' behavior has been parametrized. A normalized execution-value of 0.0 was translated to 60% dash power a value of 1.0 to a dash power of 100% with linear interpolation. Relevance conditions in the goals (see [2]) ensure that the decidedness in important situations like being close to one of the goals is high. This should ensure that the agent consumes less stamina in less important situations and has more stamina available in important situations.

5.2.1 Normalization of the Execution-Value

Section 4 explained the need for normalization of the execution-value. Two approaches have been mentioned that can be used for normalization without violating the principle of locality. One possibility is to store the minimal and maximal execution-values of a competence module and map it to the interval $[0..1]$ (MinMax). Another possibility is to calculate the mean execution-value μ and its standard deviation s (incrementally). Then a range of values from $\mu - k \cdot s$ to $\mu + k \cdot s$ can be mapped to normalized execution-values in the interval $[0..1]$ (distribution).

Since MinMax normalization is vulnerable to extreme values one would expect to get worse results with this approach. This was empirically evaluated in 30 games of 2 Robocup soccer agent teams. One team played with MinMax normalization the other team played with distribution normalization. Besides that both teams have been exactly identical. For the distribution normalization we chose $k = 1$. As shown in table 2, the team with distribution normalization scored significantly more goals than the team with MinMax normalization.

	MinMax	distribution	p ($n = 30$)
mean number of goals	4.2	6.0	0.008

Table 2. Comparison of the MinMax normalization and normalization using the distribution of values.

5.2.2 Comparison of Parametrized and Static Behavior

As mentioned above, parametrized behavior execution should improve the utilization of resource 'stamina' in the Robocup domain. This should improve the overall performance of a team measured by the number of goals scored. This can be verified by experiments running Robocup games where one team uses parametrized behaviors and the other does not (static). Normalization of execution-values was done using the distribution method. The parameter for the execution-value of the static team was constant during one game. It was varied in the interval $[0..1]$, however, for different series of games. In this way parametrized behaviors can be compared with growing static parameter values. The hypothesis is that for low static values the disadvantage of being too slow (e.g. to reach a ball) outweighs the advantage of being less tired. For high static values the disadvantage of fast exhaustion should outweigh the advantage of being faster at the ball.

First it is interesting to look at the number of pauses an agent takes during a game. This is a measure for the consumption of stamina of the agent. As expected the number of pauses of the static team grows with increasing parameter values (Fig. 2).

It is interesting to compare the two teams at the intersection of both curves at value 0.7. Although both teams' agents have to make

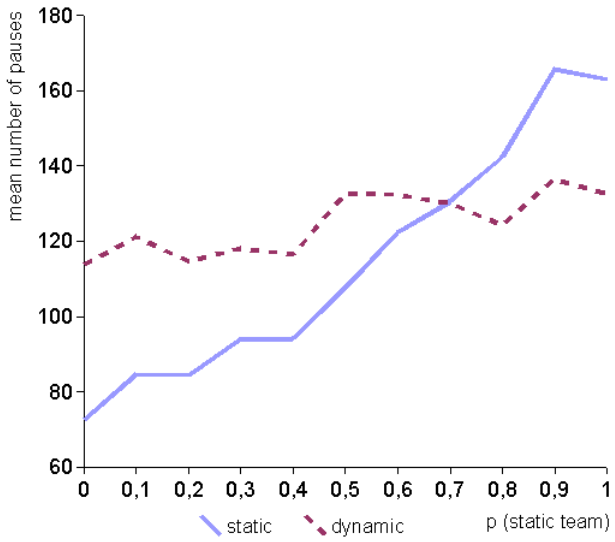

Figure 2. Mean number of pauses of the static and parametrized team

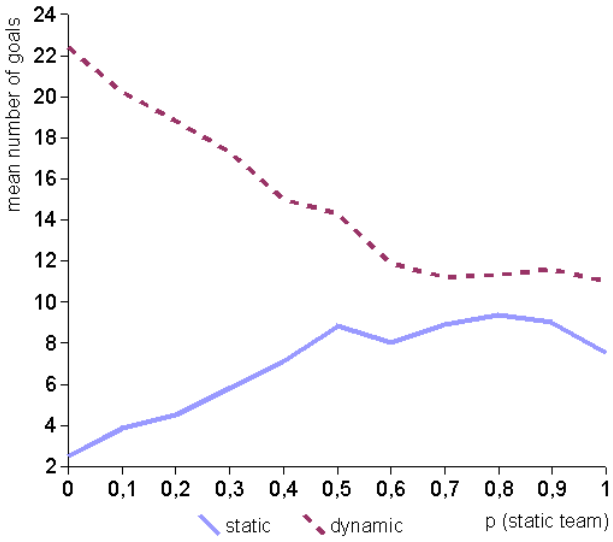

Figure 3. Mean number of goals of the static and parametrized team

the same number of pauses on average, the team with parametrized behaviors scored significantly more goals (Tab. 3). Although the average usage of resources of both teams is equal the team with parametrized behaviors makes more use out of it. It uses the resources in situations in which the goals of the agent are more relevant. In such situations the execution-values of behaviors directed towards such goals are higher.

$p_{statickteam} = 0.7$	static	parametrized	p ($n = 45$)
mean scored goals	8.9	11.2	0.003
mean number of pauses	130.6	130.2	0.950

Table 3. Comparison of the mean number of goals and pauses of players of static (parameter $p = 0.7$) and parametrized behavior execution.

The comparison of scored goals for the static and parametrized team shows significantly better results for the parametrized team for all parameter values used for the static team (Fig. 3).

6 CONCLUSION

In this paper, we describe a mechanism that can be used for an agent to select multiple actions to be performed concurrently using extended behavior networks. The concurrent action selection mechanism is calculated distributively in the competence modules (nodes) of the EBN. Conflicts between actions are moderated by resource nodes that are explicitly represented in the EBNs. In addition, we introduce a mechanism for EBNs to influence behavior execution using the execution-value of a competence module as a measure of the decidedness of the agent to perform the action. Both extensions improved the performance of agents in the RoboCup simulated soccer domain significantly.

Future work will mainly have to examine if these results generalize to other dynamic and continuous domains. Especially domains will be interesting, where the amount of available resources depends on the current situation. The stamina resource in the RoboCup domain that resembles how much 'energy' is left for dashing can not be used in this sense, because although enough stamina would be available for different behaviors the server does not allow concurrent dashing behaviors.

Also it would be interesting to examine the stability of the proposed concurrent behavior selection in cases where the estimated amount of resources used by a competence module's behavior may differ from the effectively used resources. The behavior selection itself should still work in such occasions, the performance of the agent, however, is expected to decrease.

References

[1] C. Boutilier, T. Dean, and S. Hanks, 'Decision–theoretic planning: Structural assumptions and computational leverage', *Journal of Artificial Intelligence Research*, **11**, 1–94, (1999).
[2] K. Dorer, 'Behavior networks for continuous domains using situation–dependent motivations', *Proceedings of the Sixteenth International Conference of Artificial Intelligence*, 1233–1238, (1999).
[3] K. Dorer, *Motivation, Handlungskontrolle und Zielmanagement in autonomen Agenten*, PhD thesis, Albert-Ludwigs University, Freiburg, 2000.
[4] H. Heckhausen, *Motivation und Handeln*, Springer, Berlin, 1989.
[5] P. Maes, 'The dynamics of action selection', *Proceedings of the International Joint Conference on Artificial Intelligence*, 991–997, (1989).
[6] P. Maes, 'How to do the right thing', *Connection Science Journal*, **1**(3), (1990).
[7] P. Maes, 'Situated agents can have goals', *Journal for Robotics and Autonomous Systems*, **6**(1), 49–70, (1990).
[8] D. Navon and D. Gopher, 'On the economy of the human processing system', *Psychological Review*, **86**(3), 214–255, (1979).
[9] D. A. Norman, 'Categorization of action slips', *Psychological Review*, **88**(1), 1–15, (1981).
[10] L. H. Shaffer, 'Multiple attention in continuous verbal tasks', in *Attention and Performance V*, eds., P. M. A. Rabbit and S. Dornic, Academic Press, New York, (1975).