Behavior Networks for Continuous Domains using Situation-Dependent Motivations

Klaus Dorer*
Centre for Cognitive Science
Institute for Computer Science and Social Research
Albert-Ludwigs-University Freiburg, Germany
klaus@cognition.iig.uni-freiburg.de

Abstract

The problem of action selection by autonomous agents becomes increasingly difficult when acting in continuous, non-deterministic and dynamic environments pursuing multiple and possibly conflicting goals. We propose a method that exploits additional information gained from continuous states, is able to deal with unexpected situations, and takes multiple and conflicting goals into account including additional motivational aspects such as dynamic goals, which allow for situation-dependent motivational influence on the agent. Further we show some domain independent properties of this algorithm along with empirical results gained using the RoboCup simulated soccer environment.

1 Introduction

Agents in a complex dynamic domain need to take multiple goals into account, which may be of different type (as exemplified in the RoboCup soccer environment):

- *maintenance goals*, which should be less demanding the more the goal is satisfied (e.g. 'have stamina').
- *achievement goals*, which should be more demanding the closer the agent is to the goal (e.g. 'score a goal').

Maes [1989; 1990; 1992] suggested a mechanism for action selection (MASM - Maes Action Selection Mechanism [Tyrrell, 1994]) in dynamic and unpredictable domains based on so-called behavior networks. Although MASM-networks do work in continuous domains, they do not exploit the additional information provided by continuous states. Similarly, though there are mechanisms to distinguish different types of goals in MASM, there are no means to support goals with a continuous truth state (like 'battery charged') to become increasingly demanding the less they are satisfied.

We propose a revised and extended version (REASM) of Maes' action selection mechanism, that takes the step from discrete to continuous domains by introducing real-valued propositions. It also allows for advanced motivational control by situation-dependent goals, and retains the advantages of MASM, such as reactivity, planning capabilities, robustness, accountance of multiple goals and the cheap and distributed calculation.

We give a formal definition of extended behavior networks and describe the activation spreading and action selection algorithm in section 2. In section 3 we show some domain independent properties such as activation spreading always reaching a stable state. Among other empirical results, we show in section 4 that the extensions proposed show significantly better success in the RoboCup domain.

2 REASM Formalism

In this section we give a formal description of the revised and extended behavior networks followed by the algorithms for activation spreading used to calculate the utility of a behavior and for action selection, which decides on the behavior to execute.

2.1 Behavior Network Description

Let $\mathcal{S}$ be a set of worldstates, $\mathcal{P}^+$ a set of atoms, and $\tau : \mathcal{P}^+ \times \mathcal{S} \to [0..1]$ a function assigning a *truth value* to each atom in each worldstate. $\mathcal{P}$ is a set of atoms and negated atoms where $\tau(\neg p, s) := 1 - \tau(p, s)$. $\mathcal{L}^\wedge$ is a propositional language over $\mathcal{P}$ and the logical connective $\wedge$, where $\tau(p \wedge q, s) := \tau(p, s) \otimes \tau(q, s)$ and $\otimes$ is any continuous triangular norm (e.g. $min(p, q)$, pq). $\mathcal{L}$ is a propositional language over $\mathcal{P}$ and the logical connectives $\wedge$ and $\vee$, where $\tau(p \vee q, s) := \tau(p, s) \oplus \tau(q, s)$ and $\oplus$ is any continuous triangular conorm (e.g. $max(p, q)$, $x + y - xy$) [Saffiotti *et al.*, 1995].

REASM behavior networks $\mathbf{B}$ are described by a tuple $(\mathcal{G}, \mathcal{M}, \Pi)$, where

- $\mathcal{G}$ denotes the set of *goals* characterized as tuples ($GCon$, ι, $RCon$) with

*The work reported here has been funded by the German Research Association (DFG, Graduiertenkolleg Menschliche und maschinelle Intelligenz). I would like to thank Gerhard Strube and Bernhard Nebel for important comments and suggestions during the preparation of this research.

- $GCon \in \mathcal{L}^\wedge$ the *goal condition*, i.e. the situation in which the goal is satisfied,
- ι the *importance* of the goal $\in [0..1]$,
- $RCon \in \mathcal{L}$ the *relevance condition*, i.e. the situation-dependent importance of the goal with $r = \tau(RCon, s)$ the *relevance* of the goal.

- $\mathcal{M}$ is a finite set of *competence modules*, where m $\in \mathcal{M}$ is a tuple (*Pre*, *b*, *Post*, *a*) with
  - *Pre* $\in \mathcal{L}^\wedge$ denoting the *precondition* with $e = \tau(Pre, s)$ the degree of *executability*;
  - *b* the *behavior*, which is executed once the module is selected for execution;
  - *Post* is a tuple (*Eff*, *ex*), where *Eff* $\in \mathcal{L}^\wedge$ are the *effects* of the behavior and ex_j denotes the *expectation* $\in [0..1]$ of effect proposition Eff_j to become true after executing this module;
  - *a* the *activity* $\in \mathbb{R}$ indicating the utility of the module with a_g the vector of activations a_{g_i} received (directly or indirectly) by goal g_i.
- Π is a set of (domain-dependent) *parameters* used to control activation spreading;
  - $\gamma \in [0..1[$ activation of modules,
  - $\delta \in [0..1[$ inhibition of modules,
  - $\beta \in [0..1[$ inertia of activation,
  - $\theta \in [0..\hat{a}]$ activation threshold, with $\hat{a}$ the upper bound for a module's activation,
  - $\Delta\theta \in]0..\theta]$ threshold decay.

The revised activation spreading algorithm of REASM (see next section) made it possible to reduce the number of parameters and to restrict them to the ranges printed above. This simplifies the process of finding best performing parameter values for a domain (see section 4.1).

2.2 Activation Spreading

The competence modules are connected in a network [Maes, 1989] to receive activation from goals and other modules. A competence module k receives activation from a goal g_ii at timestep t

$$a^t_{kg_i}{}' = \gamma \cdot f(\iota_{g_i}, r^t_{g_i}) \cdot ex_j \qquad (1)$$

if the module has an effect (with expectation ex_j) that is part of the goal condition and both are atoms or both are negated atoms, i.e. the behavior satisfies the goal. f is any continuous triangular norm, that combines the static importance of the goal ι_{g_i} and the dynamic relevance r_{g_i} (e.g. $\iota_{g_i} \cdot r_{g_i}$).

A module k is inhibited by a goal i by

$$a^t_{kg_i}{}'' = -\delta \cdot f(\iota_{g_i}, r^t_{g_i}) \cdot ex_j, \qquad (2)$$

if the module has an effect that is part of the goal condition and exactly one of them is negated, i.e. the behavior would undo an already satisfied goal.

A module receives activation by a so called successor module if it has an effect (with expectation ex_j) that is part of the preconditions of the successor module (p_{succ}) and both are atoms or negated atoms. The activations are calculated separately for each goal activation $a_{succ\,g_i}$ of the successor module and are

$$a^t_{kg_i}{}''' = \gamma \cdot \sigma(a^{t-1}_{succ\,g_i}) \cdot ex_j \cdot (1 - \tau(p_{succ}, s)), \qquad (3)$$

where σ: $\mathbb{R} \to [0..1[$ is the transfer function of the modules activation for which we used $\sigma(x) = (1 + e^{\kappa(\mu - x)})^{-1}$ [Goetz, 1997]. The term $(1 - \tau(p_{succ}, s))$ states that the less the precondition of the successor module is satisfied in situation s the more activation is spread to modules making this precondition true, i.e. the false precondition becomes an increasingly demanding subgoal of the network.

Finally a module is inhibited by conflictor modules by

$$a^t_{kg_i}{}'''' = -\delta \cdot \sigma(a^{t-1}_{conf\,g_i}) \cdot ex_j \cdot \tau(p_{conf}, s), \qquad (4)$$

if it has an effect that is part of the preconditions of the conflictor module (p_{conf}) and exactly one of them is negated. $a^{t-1}_{conf\,g_i}$ is the activation the conflictor module received directly or indirectly by goal g_i at timestep $t-1$.

The activation a_{kg_i} of each goal is kept separately by each module and is set to the activation of that link with the highest absolute maximum activation

$$a^t_{kg_i} = abs\max(a^t_{kg_i}{}', a^t_{kg_i}{}'', a^t_{kg_i}{}''', a^t_{kg_i}{}''''). \qquad (5)$$

In other words, only the strongest path from each goal to a module is taken into account. Any confluence of activation within a module from the same goal is prohibited. This leads to some important new properties of the algorithm shown in section 3.

Finally the activation of a module k is

$$a^t_k = \beta a^{t-1}_k + \sum_i a^t_{kg_i} \qquad (6)$$

where β controls the inertia of the activation and therefore the inertia of the agent's behavior.

Activation and inhibition as well as the introduction of relevance conditions allow the modelling of different types of goals. Increasingly demanding maintenance goals (e.g. 'have stamina' in the RoboCup soccer environment) can be achieved by adding a relevance condition ('stamina low'). This increases the relevance of the goal and therefore the activation of satisfying behaviors by the goal as the situation diverges from the state to be maintained. Achievement goals (e.g. 'score a goal') can be realized by adding a relevance condition ('close to goal') whose truth value increases on nearing the goal. Modules achieving the goal are increasingly activated, modules conflicting with the goal are inhibited.

2.3 Action Selection

Action selection is done in a cycle containing the following steps [Maes, 1990]

1. Calculate activation of each module a^t_j (Eq. 6).
2. Combine activation and executability of a module by a non-decreasing function $h : \mathbb{R} \times [0..1] \to \mathbb{R}$. To prevent non-executable modules from being executed, $h(a, 0)$ should be zero.

3. If the highest value $h(a, e)$ lies above θ, execute the corresponding module's behavior, reset θ to its original value and go to 1.
4. Otherwise reduce θ by $\Delta\theta$ and go to 1.

Step 2 is necessary because modules have a continous executability e and can therefore not be divided into executable and non-executable as opposed to MASM. All modules have to be considered for execution prefering modules with higher executability, although modules with high activation may be executed even if their executability is low.

3 Domain Independent Properties

In this section we show properties of the algorithm for activation spreading that are domain independent.

3.1 Stability

An important property of activation spreading networks is to reach a stable state of activation. Although this seems to be the case for Maes' behavior networks, to our knowledge it has never been proven. However, variations of Maes' networks (variation four of [Tyrrell, 1994]) oscillate under some circumstances. The algorithm proposed here can be proven to reach a stable state.

Lemma 1 *The above described algorithm for activation spreading does not allow feedback of activation for $\sigma(a) \leq a$ and $t \to \infty$.*

Proof. Although there possibly are cycles within REASM behavior networks, the activation a module gets from itself $a^t_{self\, g_i}$ drops to zero for $t \to \infty$. This holds because $|a^t_{self\, g_i}| < \gamma^{n\zeta}\delta^{m\zeta}|a^{t-(n+m)}_{self\, g_i}|$, $\zeta = \frac{t}{n+m}$ (Eq. 3,4 and $\sigma(a) \leq a$, $\gamma, \delta < 1$ and $\tau(p, s), ex_j \leq 1$). n is the no of excitation links, m the number of inhibition links within the cycle. For $t \to \infty$ either $\gamma^{n\zeta}$ (for $n > 0$) or $\delta^{m\zeta}$ (for $m > 0$) and therefore $|a_{self} g^t_i|$ approaches zero. For $m = n = 0$ the module is not part of a cycle. □

Theorem 2 *Activation in REASM networks always reaches a stable state (unless the situation changes).*

Proof. Activation for each goal is calculated separately by the competence modules. Therefore we can treat each goal separately and look at the connected subgraphs containing one goal. We split the vertices of this subgraph into two sets: V^0 contains the vertices which have reached a stable state of activation, V contains all the other nodes of the subgraph. Initially, V^0 only contains the goal. After one step of activation spreading the node with the strongest link to the goal (in terms of maximum absolute activation (Eq. 1, 2)) will receive constant activation a_{g_i} and can be removed from V and put into V^0. This holds because activation decreases along activation paths (proof of lemma1) and because each module receives activation across a single incoming link (Eq. 5). This can be repeated for all nodes in V, although activation of these nodes may take more than one step of activation spreading to reach a stable state, because of previous activation caught in feedback cycles. This activation feedback, however, drops to zero for $t \to \infty$ (lemma 1). The main activation of a module m received by all goals a_m then equals $a_m = (1-\beta)^{-1}\sum_n a_{mg_n}$ (Eq. 6) and $\Delta a_m = 0$. □

3.2 Problems mentioned by Tyrrell

Tyrrell [1994] pointed out some problems of behavior networks proposed by Maes. We show that none of these problems hold within REASM.

Preference for appetitive behaviors[1]

The action selection mechanism proposed by Maes shows some undesirable preference for appetitive nodes over consummatory nodes (see Fig. 1) independent of parameter settings [Tyrrell, 1994].

In MASM activation of an appetitive node (action2) is $a^t_2 = r(a^{t-1}_2) + \gamma + \phi + \phi/\gamma a^{t-2}_2 + ...$ while a consummatory node (action3) receives less activation $a^t_3 = r(a^{t-1}_3) + \gamma + \phi)$. This is undesirable, because action3 directly satisfies the goal while action2 only satisfies a precondition of action1 which reaches the same goal.

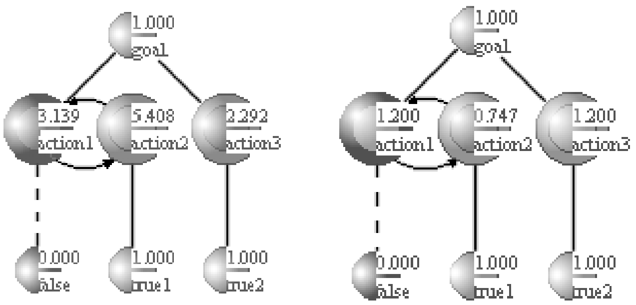

Figure 1: Preference for appetitive nodes
At the top level are the goals with their importance, below are the competence modules and their activation and at the bottom level the situation propositions (perceptions) with their τ-values. Only the relations of activation values matter. MASM (left) undesirably prefers appetitive action2, REASM (right) correctly prefers action3 (action1 is non-executable).

In REASM, activation from a goal always decreases with the distance to that goal (proof of lemma 1) preferring modules that directly satisfy it. This prevents activation feedback from occurring (lemma 1) and therefore the preference for appetitive behaviors.

Activation fan

MASM divides activation spread by goals by the number of links connected to the goal (activation fan). Similarly, activation received by a node is divided by the number of incoming links. This penalizes goals with more behaviors satisfying it and does not prefer modules satisfying multiple goals at once (see Fig. 2).

Tyrrell shows however that leaving out division by the number of links causes a different problem, namely the

[1] In contrast to consummatory behaviors that try to satisfy a goal, appetitive behaviors try to prepare consummatory.

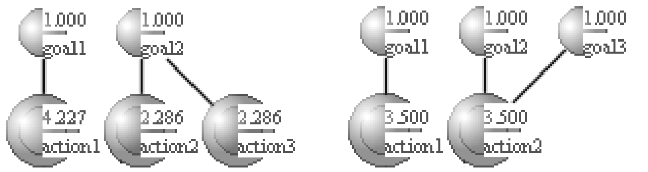

Figure 2: Problems with activation fan
Goals with multiple satisfying competence modules are penalized in MASM (left) due to the division of activation by the number of leaving links. Modules satisfying multiple goals are not preferred (right) due to the division of activation by the number of incoming links.

confluence of activation in nodes with many successors which may all be alternatives of one goal (see Fig. 3).

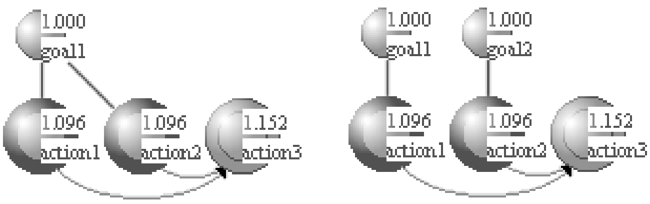

Figure 3: Problems without activation fan
An appetitive behavior (action3) cannot distinguish between getting activation originally spread by one goal (left) or by multiple goals (right).

Neither problem holds for REASM. Activation is not divided by the number of incoming or outgoing links. Therefore behaviors satisfying multiple goals are prefered (Eq. 6), goals with alternative behaviors satisfying them are not penalized. And there is no confluence of activation from the same goal due to the fact that only the strongest path of activation from each goal to a module is taken into account (Eq. 5).

4 Empirical Results

Empirical analysis of behavior networks has been conducted using the RoboCup soccer server program [Noda, 1995]. Agents in this domain are simulated soccer players getting their (relative) perceptions from the server across a network and sending executed actions to the server which changes its state accordingly. Perception and action are non-deterministic, i.e. perceptions as well as actions are perturbed by some noise, and may be lost in the network. The state of the soccer field is dynamic: It changes whenever any of the agents performs some action. It is continuous: State, perceptions and also actions are described by continuous values. In short, the RoboCup domain is non-deterministic, dynamic and continuous and is therefore a demanding environment for any algorithm for action selection.

The network used contained three goals and eight competence modules (see Fig. 5). The corresponding behaviors were implemented using C++ methods that were called once the competence module was selected. τ-functions for perception-propositions were similarly calculated from the agent's perceptions and state information using C++ methods.

4.1 Parameter Setting

As stated above, activation spreading and action selection depend on a set of parameters. These parameters are domain dependent and have to be tuned to obtain best performance from the agents. This was done by playing a series of games with equal teams of two players[2] except for varying one parameter of one team along its definition area. 25 games for each of eleven variations per parameter were conducted to obtain statistically reliable results. The quality of the varied team was measured by the difference between scored goals and those scored by the other team. For the variation of the first parameter, the other parameters were set to 'sensible' values. The following parameter variations used previously found values, which performed best. Because parameters are not independent of each other, this process was repeated for all parameters, until changes of best values became small (< 0.1). This led to curves as shown for the activation by goals γ in Figure 4.

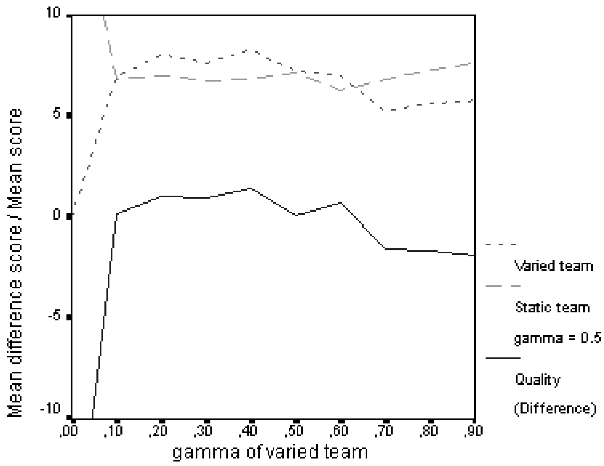

Figure 4: Quality of a team as a function of the activation by goals γ.

Two things should be noted: First, without any motivation by goals ($\gamma = 0$) the agents perform very poorly, because all modules have same activation (zero). Hence, always the first executable module is executed. Second, although differences like $\gamma = 0.4$ to $\gamma = 0.5$ are significant, the score level is high for a wide range of parameter values indicating that finding a 'functional' parameter setting is not too difficult.

4.2 Real-valued Propositions

To evaluate the usage of real-valued propositions we conducted a series of 30 games where one team used real-

[2] Offside rule was switched off, stamina recovery was increased w.r.t. eleven player games.

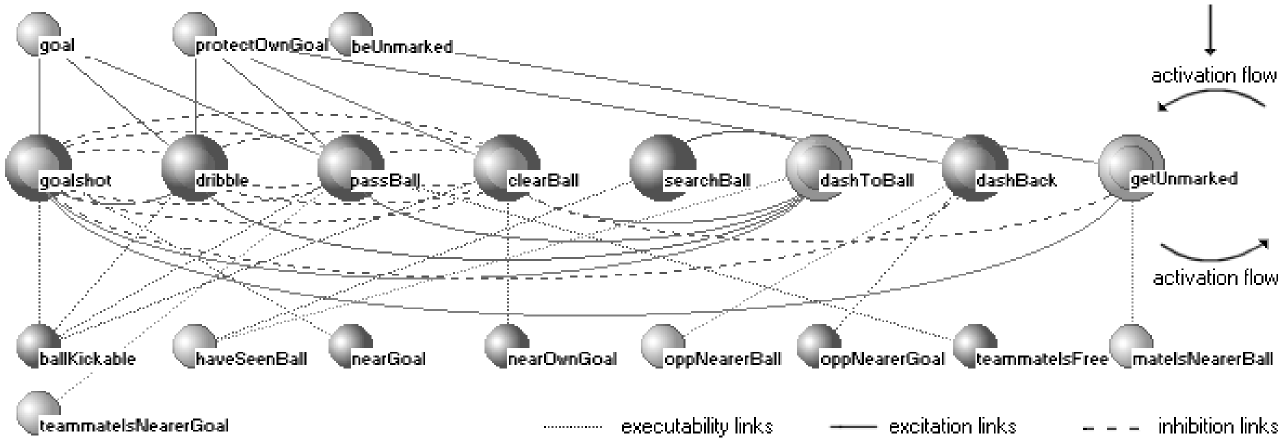

Figure 5: Behavior network used for empirical evaluation in the RoboCup domain. Top level are the goals of the agent, below the competence modules and at the bottom the perceptions. Links to perceptions are used to calculate the executability of a module. Links to goals and other modules are used to calculate the utility of a module.

valued propositions while the other team used discrete propositions (τ-values were truncated to 0 for $0 \leq \tau < 0.5$ and rounded to 1 for $0.5 \leq \tau \leq 1$). No goal relevance was used for these games. The team using real-valued propositions scored significantly higher[3] (Table 6).

	discrete	real-valued	p
mean score	6.6	9.0	0.001
mean no of shots	12.4	24.8	< 0.001
mean possession	368	372	0.85

Table 6: Comparison of discrete and real-valued propositions

One reason for this is that using continuous propositions, modules with a high utility may be executed even if their executability is low. So even if the likelihood of a successful execution of the behavior is small, the high utility of one or more of its effects makes it worthwhile to try. This is reflected for example by a significantly higher number of shots at the goal by the team using, real-valued propositions, although both teams had almost equal ball possessions.

4.3 Situation-Dependent Goal Relevance

In another experiment we introduced situation dependent goal relevances (dynamic goals). The relevance condition for the goal 'score goal' was to be in the opponent's half (true, 10m behind the midline and false 10m before with linear interpolation), for 'protect own goal' it was not to be in the opponent's half and for 'be unmarked' it was for the teammate to be the nearest player to the ball. Two teams, one team using situation-dependent goals, played a series of 30 games. Although relevance conditions were fairly primitive, the team using dynamic goals scored significantly higher (Table 7).

	static	dynamic	p
mean score	6.8	9.0	0.003

Table 7: Comparison of static and dynamic goals

4.4 Comparison to MASM

We also implemented the original algorithm proposed by Maes [Maes, 1990] to be able to compare both algorithms. After parameter optimization described above for both networks, we played a series of 30 games. The agents of one team were controlled by MASM, the other team's action selection was conducted by REASM. Both teams' agents used the same perceptions[4] and identical behaviors, so the only difference was in action selection.

	MASM	R(E)ASM	p
mean score	6.3	7.8	0.03
behavior switches	1105	546	< 0.001

Table 8: Comparison of MASM and REASM without real-valued propositions and situation-dependent goals

REASM scored considerably higher than agents using MASM even without the usage of real-valued propositions and situation-dependent goals (Table 8). This can be explained by the significantly higher rate in behavior switches conducted by MASM. It is caused by resetting the activation of executed competence modules to zero in MASM and by using sigmoidal transfer functions in REASM making behaviors attractors for activation resulting in fewer behavior changes [Goetz, 1997].

[3] All statistical test are paired samples T-tests for differences of means with $\alpha = 0.01$

[4] Using discrete propositions for MASM.

When equipping REASM with real-valued propositions and situation-dependent goals it scores significantly higher than MASM (Table 9).

	MASM	REASM	p
mean score	4.2	10.9	< 0.001

Table 9: Comparison of MASM and REASM using real-valued propositions and situation-dependent goals

5 Limitations

REASM does not allow for multiple behaviors executed concurrently. However, humans are able to perform well trained behaviors in parallel, unless they use the same resources [Gopher and Donchin, 1986]. Assuming knowledge about the resources used by a behavior, the action selection algorithm could be changed to build sets of executable behaviors with disjunct resources and execute all behaviors in that set, with the highest sum of utilities.

For the empirical studies, expectations of effects were set manually. Although Maes proposed an algorithm to learn the links of a network and their expectations [Maes, 1992], this work does not extend to continuous domains with delayed effects. Adaptive behavior networks are however inevitable once domains get increasingly complex. Work in the area of reinforcement learning with delayed reward could help to extend the algorithm for learning behavior networks from experience.

6 Discussion

Maes' algorithm contains two further kinds of links spreading activation from perceptions p to competence modules with precondition p (situation links) and from competence modules with effect p to other modules with precondition p (predecessor links). These links account for the reactivity of the system because they insert activation from perceptions into the network. However, when dropping the division of activation by the number of links that use a perception, as we proposed, situation activation of all executable modules equals $a_s = \phi$ (ϕ is a parameter that controls the amount of situation activation). In that case there is no direct influence of these links to the selection of a behavior. The indirect influence of having more activation in non-executable modules with some preconditions satisfied did not turn out to improve action selection as Goetz [1997] and our own studies demonstrated, where parameter variations of ϕ showed best performance for $\phi = 0$.

Decugis and Ferber [1998] introduced a different variation of Maes' algorithm for which they proved convergence of activation. However, their algorithm does not include inhibition and therefore the ability to take unwanted effects into account. Another shortcoming is that goals do not depend on the current situation. Besides showing better success, situation-dependent goals simplify the creation of behavior networks, especially when domains get increasingly large and complex. This is because relevance conditions of goals can be used to divide up the domain into different contexts. The soccer-playing agents for example can use goal relevance to easily incorporate different strategies for play-on phases, and for phases when the game has been interrupted by the referee. When using the behaviors' preconditions to produce the same set of strategies, precondition lists of all behaviors grow rapidly, complicating the introduction of new behaviors and new strategies.

In this paper, we have argued that real-valued propositions can be integrated into an action control algorithm using behavior networks to improve the performance of agents in continous domains. Further the introduction of situation-dependent goals simplifies the creation of large behavior networks and improves agents' performance by focussing on relevant goals as exemplified by studies using the Robocup domain.

References

[Decugis and Ferber, 1998] Decugis, V. and Ferber, J. (1998). Action selection in an autonmous agent with a hierarchical distributed reactive planning architecture. In Sycara, K. and Wooldridge, M. (eds.) *Proceedings of the 2^{nd} Int. Conference on Autonomous Agents*, pages 354-361, New York, ACM Press.

[Goetz, 1997] Goetz, Ph. (1997). *Attractors in Recurrent Behavior Networks.* PhD thesis, State University of New York, Buffalo.

[Gopher and Donchin, 1986] Gopher, D., and Donchin, E. (1986). Workload: An examination of the concept. In K. R. Boff, L. Kaufman and J. P. Thomas (eds.) *Handbook of perception and human performance* (vol. 2, ch. 41). Wiley, New York.

[Maes, 1989] Maes, P. (1989). The Dynamics of Action Selection. In *Proceedings of the International Joint Conference on Artificial Intelligence-'89* , Morgan Kaufmann, Detroit.

[Maes, 1990] Maes, P. (1990). Situated Agents Can Have Goals. In *Journal for Robotics and Autonomous Systems*, Vol. 6, No 1, pages 49-70, North-Holland.

[Maes, 1992] Maes, P. (1992). Learning Behavior Networks from Experience. In Varela, F. and Bourgine, P. *Proceedings of the First European Conference on Artificial Life*, pages 48-57, MIT-Press, Paris.

[Noda, 1995] Noda, I. (1995). Soccer server: a simulator of robocup. In *Proceedings of AI symposium '95*, pages 29-34, Japanese Society for Artificial Intelligence.

[Saffiotti *et al.*, 1995] Saffiotti, A., Konolige, K. and Ruspini, E. (1995). A Multivalued Logic Approach to Integrating Planning and Control. In *Artificial Intelligence*, Vol 76, No 1-2, pages 481-526.

[Tyrrell, 1994] Tyrrell, T. (1994). An evaluation of Maes' bottom-up mechanism for behavior selection. In *Adaptive Behavior 2 (4)*, pages 307-348.