

# REACTIVE LAYER IN AGI AGENT

## *Implementation of Adaptive Reactive Behavior and Beyond*

Vilem Benes

Computer Science Department, University of West Bohemia, Univerzitni 8, Pilsen, Czech Republic  
shodan@kiv.zcu.cz

Keywords: AI, AGI, agent architecture, reactive agent, situation discrimination

Abstract: Basic mechanisms of cognition working in AGI agent are presented. I argue that reactive behavior is the baseline of intelligence – it is the base component working and it can be further extended to produce more intelligent agents. Mechanisms employed at reactive level enable the agent to develop behavior which both explores and exploits the environment with the purpose of receiving highest reward possible. Three fundamental mechanisms are intertwined – action selection, action value estimation and situation discrimination. Whole process of adaptation is completely unsupervised and depends only on reward received from environment. Some technical details of implementation of given mechanisms (BAGIB agent) are described together with implications to other planned parts of “AGI-compliant” architecture. Discussed are several challenges we encounter in AGI, which are not present in usually narrow and domain-limited approach to AI.

## 1 INTRODUCTION

In this section I shortly introduce used notions of intelligence, artificial intelligence, reinforcement learning and outline design of my agent called BAGIB. In next sections, we will go through more detailed description of BAGIB implementation. Final part of the article holds discussion about BAGIB performance and about reactive behavior in AGI realm.

### 1.1 AI

Intelligence is naturally (and also usefully for us) defined as ability to reach goals (receive high reward (Benes, 2004)) in *wide range* of environments (Legg and Hutter, 2007).

Artificial general intelligence (AGI) is relatively new term for creating of intelligent artefacts. It emphasizes the fact that the phenomenon called intelligence is general. Till now, most AI solutions are “narrow”, limited only to some given domain. Designing AI without the aim for generality often encourages using domain-dependent tricks and

knowledge (see (Goertzel, 2008)) and leads to solutions that are not robust and adaptive – and dumb.

Reactive layer we are dealing with here is characterized by not having any inner states – all actions of the agent directly depend on observation. Reactive component that is adaptive is fundamental for intelligent agent. Because it is tightly bound with agent’s body (its actuators and sensors) and sufficient for considerable degree of intelligence.

The underlying framework I use is the one of Reinforcement Learning (RL) (Sutton and Barto, 1998). It is compatible with the notion of situatedness (embodiment; structural coupling of agent and environment).

### 1.2 BAGIB Agent

In this article, we are going to explore the inner working of AGI agent called BAGIB<sup>1</sup>. As AGI agent, BAGIB is meant to be able to cope with any environment. BAGIB agent adapts – it changes its responses and inner structure according to reward

---

<sup>1</sup> BAGIB – Brain of Artificial General Intelligence agent (by Benes).

received. Agent derives and maintains reward estimates (Q values) for primitive actions. Moreover, these values are conditioned by perceived situations which are adaptively defined. This allows more specific and hopefully better reacting. Discrimination between situations<sup>2</sup> is being used and refined at the same time. Agent is able to adapt and is increasing complexity of inner structures “on-fly” to be able to exploit more the environment.

If the situation does not clearly solicit a single response or if the response does not produce a satisfactory result, the agent is led to further refine his discriminations, which, in turn, solicit more refined responses (interpretation of work of Merleau-Ponty in (Dreyfus, 2007) fits exactly as description here).

To be more specific, BAGIB features adaptation and creation of situation discrimination by a variant of iteratively built regression tree that is taking observation directly or preprocessed by detection of clusters of frequent data-points. Creating these structures (clusters, regression tree) can be viewed as creating a kind of pre-processing mechanism that transforms sensory data and then outputs reward estimates for different primitive actions. Current version of BAGIB agent maintains no inner states for use in succeeding steps (not counting stored data that are used for adaptation of reactive layer), therefore this agent is said to be reactive.

Current version of BAGIB is limited to use of primitive actions. The agent tries to find best perception of circumstances in the world (i.e., best situation discrimination), to be able to reach highest possible reward by performing only (the best) primitive action in each perceived situation.

Inner representations (symbols) are grounded in the experience – in the structures derived from data acquired from the interaction with environment. This should be the correct way to deal with the symbol grounding problem. Furthermore, effects of actions on reward are continually stored. Thanks to observing many different combinations, the agent is to some extent able to infer what actions led to what effects.

BAGIB agent gives us insight into more complicated things. First, we can see origins of symbols in situation discrimination. BAGIB also presents simple mechanism for approaching credit assignment problem.

Second, BAGIB is example of general principle of reusing same mechanisms at different levels of inner hierarchy – selection mechanism is same for primitive actions, regression tree condition candid-

<sup>2</sup> See also *associative search* in (Sutton and Barto, 1998).

ates (features) and also whole behaviors. Whole described reactive behavior could be taken as structural component and reused inside the brain of the agent after definition of inner actions and sensors – to achieve metacognition. This may enable self-monitoring, self-reflection, control of inner mechanisms and also gathering needed environment-independent knowledge.

Third, we can think more about AGI theory using BAGIB example – we see what reactive architecture like this is capable of and we can guess what extensions to this model can result into higher intelligence. These extensions may include specialization of behaviors, keeping and using inner states<sup>3</sup>; using model of causal relations for expectations and planning; experimenting; preparing and conducting more complicated actions and others.

In the following sections, all fundamental parts of reactive component of BAGIB AGI agent are described – action selection, value estimation (reward assignment) and situation discrimination. Additionally, more specific features and implementation details of this agent are presented.

## 2 MECHANISMS

### 2.1 Selection

AGI agent deals with the exploitation vs. exploration problem. Primitive actions that were identified as good should be repeated (exploitation) – so the agent can collect reward, but also actions that seemed bad at first sight should be tried from time to time (exploration) – to get the chance to reveal better actions/behaviors<sup>4</sup>/situations and avoiding being stuck in local optima. The agent can be never certain that whole environment was properly explored and that reward estimates are correct. This is because the environment can be either non-stationary (i.e., changing with time) or there can always be regions in state-space of the environment which were never reached. Usually, intricate structures that imply complex behavior need to be developed before the agent can reach (and “maintain”) highly rewarded state-space regions of the environment.

<sup>3</sup> Manipulating inner structures is quite similar in principle to interacting with (outer) environment. See A-Brain B-Brain C-Brain idea in (Minsky, 1981).

<sup>4</sup> “Behavior” is for us an action-selection policy (possibly depending on perceived situations) or its realization – a sequence of actions which were already performed.

BAGIB agent uses  $\epsilon$ -greedy strategy – best action is selected with 0.99 probability. With probability of 0.01, one of remaining actions is selected at random. This strategy aims at revealing the one action that is best for given situation. Unfortunately, often ideal conditions are not reached. Usually, more actions seem to be equal in likelihood to be best, or more actions are alternating as best in one situation or their combination into sequence is needed. This leads to non-trivial dynamics in selecting.

BAGIB agent uses same selection mechanism (with  $\epsilon$ -greedy) strategy not only for primitive action selection, but also for using of sub-tree candidates during their evaluation in situation discrimination and same selection mechanism is also used for whole behaviors.

## 2.2 Value Estimation

BAGIB agent uses a general principle for evaluating of all primitive actions. It is variation of the method known in RL as  $Q(\lambda)$  learning with incremental update rule, step-size parameter and eligibility traces. By this method, the agent is assigning values to actions – reward taken as input is distributed to actions that are supposed to have influence on that particular reward. Value is also assigned to adaptively created states (see situation discrimination in next sub-section).

The basic form of Q learning rule is:

$$Q_{k+1} = Q_k + \alpha [r_{k+1} - Q_k], \quad (1)$$

where  $r_{k+1}$  is reward in this step and  $Q_k$  are Q values from previous step.

If step-size parameter is set to be small enough, for stationary environments, the convergence to correct values is assured. Often, stationary environment seems to be non-stationary to agent, because the agent has not reached the correct situation discrimination or environments are really non-stationary. We trade ability to drift with Q values (if we are learning or in case of non-stationary environment) with the ability to converge to correct values (for stationary environment). The former is preferred in BAGIB with usage of fixed (i.e., non-decreasing) step-size parameter. However, adaptive control of step-size parameters may prove to be useful.

Besides estimating values of primitive actions in given states, BAGIB uses Q-learning also for candidates for state-space bisections (i.e., sub-tree candidates – see next section) and also for whole behaviors. This means that the same observed reward is used to evaluate more things at once –

things which are standing at different places in inner structures. Selection based on value estimation is at the hearth of intelligence – it ensures adaptation and robust behavior.

At present time, BAGIB uses ad hoc solution for assigning reward – circular buffer is maintained that stores last 100 observations (reward + sensors) together with taken actions. Stored are also pointers to all kinds of other evaluated elements (sub-trees, behaviors). With correspondence to eligibility traces in  $Q(\lambda)$  method, factor of influence of every one element stored in circular buffer is derived (using discount factor) and share of newly received reward (together with difference of expected value between initial and end situation) is assigned to given influencing elements.

### 2.2.1 Double Q values

I investigate in BAGIB usage of two Q values for one primitive action. Purpose of this is having better reactions to changing conditions – agent maintains long time policy, but also reacts quickly to sudden big changes. In detail – each Q value is calculated as sum of two Q values:  $Q_{\text{long}}$  and  $Q_{\text{short}}$ . Both of these are calculated in the same way, but using different step-size parameters in formula (1) – eg.  $\alpha_{\text{long}}$  is 0.00001 and  $\alpha_{\text{short}}$  is 0.001.

Agent is able to use long-time averaged Q values, but it also gets the ability to actually avoid using action that is generally good, but bad recently. Quick reaction to recent changes due to  $Q_{\text{short}}$  fraction in Q allows building of bigger complexity. Besides random selection in  $\epsilon$ -greedy strategy, this gives another possibility to stop being stuck while maintaining long-term structure in Q values. This is useful especially when the agent has only limited options to recognize situations in the environment (eg. before detailed situation discrimination was developed).

### 2.2.2 Advanced Value Estimation

Exact estimation of the value of actions (and also components of the inner structure of the agent) is limited by many factors. Effect (of using) of each of them can also be only temporal or reaching far to the future. Accurate assessment of value of many different elements by described naïve method may require more steps that are available (we have possibly many combinations of evaluated elements). I feel that improving value estimation will require

using more advanced (higher-level) methods in combination with present naïve approach. For example – rules that tie effects of actions to their value may improve reward assignment. These rules (possibly conditioned by situation and derived in simpler observed cases) may be used in distributing reward<sup>5</sup>. Generalization provided by rules may significantly decrease the complexity of the reward assignment.

### 2.3 Situation Discrimination

Because state-space (observation-space) for many environments is either huge or infinite, we need some sort of abstraction. That means some kind of observation-space partition mechanism, that will give us only limited number of useful states that would be used in learning of according behaviors (action policies). Not enough states means that we are far from Markov property<sup>6</sup>, too much states means learning is too slow.

Strategy for continual incremental<sup>7</sup> observation state-space partitioning was adopted – it is done by growing the regression tree. Each leaf node in incrementally induced tree defines one situation. In each of them, policy learning takes place and best further branching is searched for. Situation here is a general term, it corresponds to the “state” term that is prevalently used in reinforcement learning theory.

Regression tree holds Q-values for all primitive actions for each of the situations defined by tree branching. Used mechanism generally allows learning good overall behaviors and then evolving them into better, more specialized solutions “on fly”.

At the beginning, regression tree consists of only one leaf node. The only state represented by this node is encompassing whole observation-space, using Q-learning in this state, the agent learns best overall behavior. Then, regression tree is extended to

enable better overall behavior by splitting observation-space into two states (situations).

In more detail, this regression tree induction process (see Table 1) runs as follows. The agent performs Q-learning for the first leaf node (the root node). During it, observations are stored. When number of stored cases reaches threshold, candidate sub-trees are created. These “leaf trees” are held in candidate list and consist of a decision node with condition and two leaf nodes with primitive action lists. The decision node of the candidate sub-tree creates the (next) observation-space bisection.

Table 1: **Situation discrimination done by iterative regression tree induction algorithm.** ‘Store observation’ saves all sensory data in given step – possibly conditioned (by step count, reward etc.). ‘Select’ stands for applying  $\epsilon$ -greedy selection in given list. ‘Tree extension criterion’ is based on difference of reward obtained by using leaf-node and by using candidate sub-tree and also on candidate use count. ‘Alternate’ is between leaf node and sub-tree candidate list with period of 1000 steps.

```

wait for observation
reach leaf node (descent tree)
if candidate sub-tree list empty then
    store observation
    select and output action
else
    alternate leaf node and cand. list
    if using reached leaf node then
        select and output action
    else if using candidate list then
        select sub-tree from cand. list
        evaluate sub-tree condition
        descent to sub-tree leaf
        select and output action
        check if chosen sub-tree
        ...candidate meets tree extension
        ...criterion
        if candidate good enough then
            replace reached leaf node
            ...with candidate sub-tree
            copy rest of candidate list
            ...to both sub-tree leaf nodes
repeat forever

```

In following steps, evaluation takes place. Agent alternates between using original leaf node and sub-trees from candidate list. If using candidate list,  $\epsilon$ -greedy strategy is used to select the candidate from list, its condition (decision node) is evaluated according to current observation and respective leaf node primitive action list is used to select ( $\epsilon$ -greedy) the output action. If candidate sub-tree from meets tree-extension criterion (used enough times and bringing higher reward than parent leaf node), it is used in regression tree as replacement of the original

<sup>5</sup> For example: (we learned before that) pushing into the wall has no positive influence – and if we detect positive change in reward while doing so, we know that it probably needs to be caused by something else.

<sup>6</sup> If the state (perceived situation) has Markov property, it means that the influence of next action taken only depends on this state and not on the history of previous states. Being close to Markov property also implies better chance to converging of the state value (expected reward in this state).

<sup>7</sup> Agent needs to be able to learn rapidly (to exploit sufficiently as soon as possible), but it needs also to be improving in larger time scales.

parent leaf node. After this replacement, regression tree now consists of one decision node and two leaf nodes. The agent recognizes two states (situations) and develops two action selection policies – one for each of the them. Distinct situations given by the position in decision tree form the basis of symbol grounding (assigning symbols to perceived phenomena). Symbol represents important situation.

Process described above is repeated and situation discrimination becomes finer. The agent searches for best way to do situation discrimination and for best action policy for it at the same time.

Described creation and usage of regression tree represents in fact developing a behavior. BAGIB agent employs not one behavior, but there is a list with many of them – as I said before, selection and value estimation mechanisms are used for list of whole behaviors in the same way as it is used for lists of primitive actions. This is a simple attempt to address problems with changing conditions and environments.

## 2.4 Feature definition

Mechanism used to create the observation-state bisection – *feature definition* was presented in (Benes, 2005). Situation is generally defined as combination of defined features. Feature here is taken as abstraction created from data cases – a circumstance that can either be present or not.

Features used as conditions are currently defined as high density clusters in observations (with various dimensionality). Currently all observations are sent to feature definition procedure.

Potentially useful may be more guided approach – sending only interesting observations to feature definition procedure (eg. observations that were followed by high/low reward or other interesting effect).

## 2.5 Problems

Finding out the optimal observation-space partitioning and corresponding action policies for different environments is very hard for more reasons.

Agent needs to be able to cope with any number of dimensions and all possible ranges of values. The agent also needs to be able to cope with changing conditions.

How we perceive the world also depends strongly on what we do in the world – with learning

of more elaborate ways to act or ways to perceive things, the agent needs to adjust its situation discrimination (and policies for situations) – the regression tree in our case. Often, at the beginning, the agent is unable to reach all important regions of environment state space, because of its initial simple behavior.

Failure in exploration may lead to development of lowly rewarded behavior. If adapting agent misses general rules at the beginning of learning, adaptation may be significantly slowed down or it may fail completely. Adoption of bisections of little usefulness in the regression tree at the beginning of adaptation may happen if regression tree creation is too fast. On the other hand, slow situation discrimination and policy learning leads to sub-optimal performance also.

List of more competing behaviors enables BAGIB agent to create one behavior, reveal important things<sup>8</sup> during performing it and then switch to newly developed behavior that uses acquired knowledge and is better.

Competition between similar structures and mechanisms seems like an fundamental principle for intelligent mind operation. In BAGIB, primitive actions compete with each other and same holds for features (candidates for observation-space bisections) and whole regression trees (i.e., behaviors).

Further research should tell whether it is more beneficial to create more complex structures and mechanisms for reactive component or whether it will be better to create and use more advanced higher-level methods that will control reactive parts.

## 3 AGENT PERFORMANCE

At present time, BAGIB is being evaluated in 6 different environments: classic RL environments *Mountain car* and *Pole balancing*, three variations of *Capture region*<sup>9</sup> environment and *Q2 deathmatch*.

In simple environments (Capture region) it is able to learn better behavior than hardwired agents tuned for best performance. In classic RL environments it

<sup>8</sup> Observation space regions (defined features), potentially also relations between situations and actions (model).

<sup>9</sup> This environment developed by author consists of a 2-d rectangular space divided into roughly 100 smaller regions which can be captured by moving agents. Variations of this environment have different number of actions and sensors. Reward is directly derived from the fraction of regions captured by given agent. This environment was designed to investigate reward assignment under non-friendly conditions (another agent, human-created, is maximizing the same reward function) and basic situation discrimination.

is comparable to other general RL learning agents. In *Pole balance* (Barto, 1993) environment, BAGIB was tested against learning neuron-network agent tailor-made for this environment – learning took approx. 10 times longer, 20 percents of runs were successful. BAGIB learned fast to keep the pole up, but failures were occurring due to moving out of the allowed area. Either improving reactive layer or adding other, non-reactive structural components (model, planning) may help to achieve better results.

In *Quake 2 deathmatch* environment, there are no other RL agents available for comparison, yet. In comparison with existing complex hard-wired solutions (bots) or experienced human players, BAGIB performs poorly<sup>10</sup>. This environment provides many useful insights, though. One example is this – in *Q2*, the agent receives a list of perceived entities as part of observation. In environments closer to real-world, the agent would be required to build this list from only partial information received. If our agent learns to use entity list in *Q2*, we understand better how this agent would process its inner representations in more “difficult” environments.

As preliminary results show, BAGIB is able to learn more slowly and little less successfully than adaptive solution custom-tailored and tuned for given environment, but is fully comparable to other general RL agents. BAGIB shows potential to outperform human coded agents, which are often fragile and receive great penalties in situations unforeseen by the designer. BAGIB agent trades performance in particular single tasks with generality. Unlike most other agents developed, it is able to cope with many different environments – this is what we aim at in designing of AGI agents.

## 4 CONCLUSIONS

Development of BAGIB agent was an success so far – described reactive part of BAGIB architecture is general and able to perform reasonably in tested scenarios. It helps us understand what is adaptive reactive layer capable of and what will be needed in additional higher-level mechanisms and layers (planning, memory, model, attention, etc.).

---

<sup>10</sup> This also depends on how complex are the given “primitive” actions and sensors. If primitive actions available to agent are in fact quite sophisticated in the environment, BAGIB may perform fairly well. Especially if these primitive actions are already well-designed pieces of behavior, so the learning agent is merely “tuning” their use in different situations.

## REFERENCES

- Barto, A. G.; Sutton, R. S. and Anderson C. W., 1983. *Neuronlike elements that can solve difficult learning control problems*, IEEE Transactions on Systems, Man, and Cybernetics, 13: 835-846,.
- Benes, V., 2004. *Intelligence means to be rewarded*, Proceedings of the 5th International PhD. Workshop on Systems and Control. A Young Generation Viewpoint. Balatonfüred, Hungary. (online: <http://home.zcu.cz/~shodan/>)
- Benes, V., 2005. *Defining Situations - Fundamental Level of Perception*, Proceedings of the 6th International PhD Workshop on Systems and Control Young Generation Viewpoint. Ljubljana: Jozef Stefan Institute. (online: <http://home.zcu.cz/~shodan/>)
- Dreyfus, H. L., 2007. *Why Heideggerian AI Failed and How Fixing it Would Require Making it More Heideggerian*, Philosophical Psychology, Volume 20, Issue 2 (April)..
- Goertzel, B., 2008. *AI and AGI: Past, Present and Future*, AGI-08 Conference. University of Memphis. (online: <http://www.acceleratingfuture.com/people-blog/?p=1814>)
- Legg, S. and Hutter, M., 2007. *A collection of definitions of intelligence*, Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms. NL: IOS Press. (online: <http://www.vetta.org/definitions-of-intelligence/>)
- Minsky, M., 1981. *Jokes and the Cognitive Unconscious*, In Cognitive Constraints on Communication, Vaina and Hintikka (eds.). Reidel. (online: <http://web.media.mit.edu/~minsky/papers/jokes.cognitive.txt>)
- Sutton, R. S. and Barto, A. G., 1998. *Reinforcement Learning: An Introduction*, MIT Press. (online: <http://webdocs.cs.ualberta.ca/~sutton/book/ebook/th-e-book.html>)