Affective Action Selection and Behavior Arbitration for Autonomous Robots

Matthias Scheutz
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
mscheutz@cse.nd.edu

Abstract: *In this paper we suggest an action selection and behavior arbitration scheme for autonomous robots, called AASBA (for "affective action selection and behavior arbitration"), which uses affective states to select the robot's behavior at any given time and pass control from the currently active behavior to the newly selected one. AASBA views action selection and behavior arbitration as an integral part of the agent architecture. The major strength of the proposed scheme is that it can be employed in a variety of agent architectures and that it allows for modifications and extensions of the agent architecture without the need of restructuring the overall control system. Preliminary evaluations of the AASBA scheme are performed with a sample implementation of a two-layered architecture (based on the AASBA scheme) on an autonomous robot.*

Keywords: *Behavior arbitration, action selection, agent architectures, affective states, autonomous robots*

1. Introduction

Action selection, the process of deciding which action to execute next, and behavior arbitration, the process of taking control away from the component of an agent architecture executing the current action, have been investigated extensively in autonomous agent research (e.g., [2], [3], [5], [10], [12]). Many action selection schemes have been compared with respect to various properties (e.g., [9], [10]). But typically, they are not compared with respect to (1) how behavior arbitration and action selection are integrated into the agent architecture, and (2) how agent architectures with particular behavior arbitration and action selection mechanisms can be extended.

In this paper we propose a scheme, which essentially uses affective states of an autonomous robot (e.g., motivational and emotional states) for action selection and behavior arbitration— for easy reference called *AASBA* ("affective action selection and behavior arbitration"). It views action selection and behavior arbitration not as separate from an agent architecture, but rather integrates both into the agent architecture in a unified way. The major strength of the proposed scheme is that it can be employed in a variety of agent architectures and that it allows for modifications and extensions of the agent architecture without the need of restructuring the overall control system.

2. Distributed Action Selection and Behavior Arbitration

Motivated by the classical ethological view of behavior arbitration (e.g., [6]), researchers in artificial intelligence and robotics have pursued various ways of implementing *distributed* action selection schemes (e.g., [5], [12]), where the decision of which action to perform next is reached in various places in parallel (i.e., in many components in the agent architecture) as opposed to one place (i.e., in a single component).

In action selection schemes, the overall behavioral capacity of an agent is typically decomposed into a fixed set of individual "behaviors", each of which has associated a numeric activation value and an action that can be performed by the agent. All behaviors are implemented in the same kind of components, call them "behavior nodes". Behavior nodes are arranged in a graph connected by various kinds of links (at the least excitatory and inhibitory links), which allow them to compete for activation. Actions are then selected according the activation level of each behavior node: only the node with the highest activation gets to execute its action. Often, such behavior nodes are arranged in "competitive clusters". This is to allow actions that use different effectors to be performed in parallel, while ensuring that mutually incompatible behaviors cannot access the agent's effectors at the same time. In addition, competitive clusters are often arranged in hierarchies, where higher level clusters are concerned with more "abstract" actions or action sequences.

Behavior arbitration is then achieved by using an algorithm, which for each competitive cluster interrupts the action of the current node if its activation level has dropped below that of another node, and starts the action of the node with the higher activation. This algorithm does not have a representation within in the agent architecture (i.e., there is no component that implements this algorithm).

While the action selection mechanism implicit in the distributed competition for activation has several advantages over local action selection in a single component (see, for example, [5]), there is at least one major disadvantage to an "external" arbitration mechanism that does not have a representation within the architecture: it cannot be modified by the agent—this is a problem for any mechanism that is not part of the architecture *per se*. Yet, there are many circumstances where a flexible behavior arbitration mechanism that can be modified (altered, extended, or even learned) by the agent would be advantageous.

A common problem with distributed action selection as described, for example, is that nodes with similar activation levels may frequently cause the arbitration mechanism to switch among their actions. This may lead to a state of the system, where actions can never complete, because they are interrupted too early. Yet, all actions could have completed, had the respective nodes only been active for a little longer, say. With an explicit architectural representation of the arbitration scheme and a learning module that notices that actions do not complete, the agent could have learned to "block" behavior arbitration for this particular competitive cluster for a short time whenever it is triggered instead of performing it right away. Furthermore, if the above method was not successful in a few exceptional cases, the agent could have learned to "block the blocking", and hence apply the original arbitration mechanism in those cases.

In the following, we will introduce the *AASBA* scheme to show (1) how behavior arbitration can be integrated into an agent architecture using behavior nodes and (2) how such "blocking" or "delaying" of behavior arbitration can be achieved using simple "affective states". The first point is particularly important since we believe that arbitration schemes should not enforce architectural designs (e.g., such as the hierarchical arrangements of behavioral modules in subsumption architectures [3], which are required for the behavior arbitration mechanism implicit in the subsumption design methodology).

3. Affective Behavior Arbitration in the *AASBA* Scheme

In the above described distributed action selection scheme, affective states are typically implemented as behavior nodes that do not have an action associated with them, but can modify and trigger the actions of other nodes. In *AASBA*, however, affective states can have an "internal action" associated with them: a *behavior arbitration operation* (in accordance with our construal of affective states as initiators and modulators of agent behavior). Consider, for example, the following simple agent (Figure 1), which can exhibit two overall behaviors: foraging and playing.

The agent also has a way of monitoring its hunger level, which it will use to decide which action to perform: if the agent is not hungry, it

will engage in playing, but if it is hungry, it will look for food. The architecture of the agent consists of three behavior nodes, one for the foraging action, one for the playing action, and one for the affective "hunger" state. While the first two nodes have the obvious actions associated with them, the hunger node has an arbitration operation associated with it, which constantly monitors the activation level of the first two states and effectively switches control to whichever state has the highest activation. Note that "hunger" is technically speaking its "own competitive" group, hence it is active all the time.

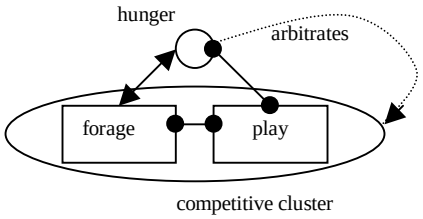

Figure 1. A competitive cluster consisting of the two complex actions "forage" and "play", which are arbitrated by the affective "hunger" state. Arrows indicate excitatory links, circles inhibitory connections. Neither sensory inputs nor motor outputs are depicted.

The details of the process of taking control away from one node and passing it to another node depends on the particular agent architecture as well as the nature of the associated action (e.g., whether it needs to finish first or whether it can be interrupted). Yet, common to all implementations of this process is that they perform operations on the architecture as opposed to the effectors of the agent. Specifically, there are two parts to the *AASBA* scheme: (1) the *implementation-independent* triggering of the arbitration process, which occurs by way of the functioning of behavioral networks (i.e., the "action selection" part) and (2) the *implementation-dependent* switching of behaviors (i.e., the arbitration operation).

While behavior arbitration in the above example works exactly the same as in other distributed action selection mechanisms—the node with the highest activation gets to execute its action—the difference between *AASBA* and other schemes like contention scheduling ([4], [8]) can been seen in a slight modification of the above example. Consider an extended version of the above agent with a second affective state, call it "persistence", which is located in a competitive cluster together with the "hunger state" (arbitration in this cluster follows the rule that the state with the highest activation will always be active).[1] "Persistence" has no behavior associated with it and is always reset to a high activation value when behaviors are switched, after which it decays slowly. Although there is no arbitration mechanism associated with "persistence", it can effectively block the arbitration mechanism associated with "hunger" as long as its activation is higher than that of "hunger". Hence, it is possible for "forage" to have a higher activation than "play", while the agent remains in play mode until the activation of "hunger" surpasses that of "persistence".

The second example shows that it is possible in *AASBA* for a behavior node to have the highest activation in a competitive cluster without actually being active (i.e., without having its behavior take control). It also hints at the flexibility of the *AASBA* scheme: it can accommodate hierarchical behavior arbitration (such as voting schemas or the arbitration implicit in the subsumption architecture) as well as non-hierarchical combination schemes (such as schema-based arbitration). Furthermore, it is possible to modify behavior arbitration without having to alter the arbitration operation itself, e.g., by changing the weights of incoming and outgoing connections of affective states that have associated arbitration operations. These weight changes, in turn, could be the result of a learning/adaptation process, which effectively allows the agent to "reorganize" its priorities without having to go through major architectural

[1] Note that arbitration operations for competitive clusters of affective states, whose associated actions are arbitration mechanisms, amounts to "meta-arbitration" among those mechanisms (as they arbitrate arbitration operations). Meta-arbitration in the above case can be easily implemented by another node without any connections to any other node. This new node will be active all the time, hence execute its action continuously, which simply is the standard arbitration mechanism for competitive clusters.

changes or without the need for higher level explicit goal representations in a deliberative layer, say. Furthermore, the *AASBA* scheme allows for different arbitration strategies (e.g., hierarchical and non-hierarchical) within the same architecture, all of which can be performed in parallel.

4. A Implementation of the *AASBA* Scheme on a Robot

To test the proposed *AASBA* scheme in practice, we defined a particular architecture based on the *AASBA* scheme for an autonomous robot, which had to perform the following three tasks (at different times): explore the environment, find a target, and play with the human observer.[2] "Explore" is a set of behaviors that allow the robot to navigate without collisions through its environment—an office room—making a map of yet unexplored territory. "Find" is a related collection of behaviors, which allow the robot to look for certain target objects in its environment—soda cans in our setting—possibly using its internal map to orient itself and move to a place where it has found a target object previously. "Play", finally, is a set of behaviors, which permit the robot to play "hide-and-seek" with a human partner. Each individual behavioral module provides furthermore a "progress value", which reflects how much progress has been made towards accomplishing that module's major task. This value can be used as input to other states.

The architecture consists of two layers: a lower layer competitive cluster comprising all behavior nodes needed to implement the three complex actions (i.e., "explore", "find", and "play") and an upper layer of seven affective states (to be described below). Similar to other distributed action selection architectures ([5], [12]) and general interactive activation and competition networks ([7]), the activation of each "behavior node" depends on various factors such as the activation of other nodes, sensory input (from internal and external sensors), and the previous activation. In addition, there are also special feedback mechanisms that monitor the execution of each action (e.g., to indicate the progress made, or whether the action was interrupted). The activation level $a(t)$ of a behavior node at time t is given by

$$a(t) = a(t-1) + minmax(t,a(t-1),netinput(t)) - decay(t,a(t-1),goal(t))$$

where

$$netinput(t) = \Sigma(ext(t) + prop(t) - intra(t-1) + extra(t-1) + progress(t))$$

The $netinput(t)$ to a behavioral node is computed by adding the weighted sums of all connected external sensory inputs $ext(t)$, proprioceptive sensory inputs $prop(t)$, extra-cluster behavioral states $extra(t)$, subtracting the weighted sum of the respective activations of intra-cluster behavioral nodes $intra(t)$. If a behavioral state has behaviors associated with it that provide progress feedback, then these progress values are added as well. The function $minmax(t,x,y)$ determines the interval of possible activation levels of behavioral nodes dependent on time t, inputs x and the previous activation level y. $Goal(t)$ is the activation level on which the behavioral state tends to settle over time if it did not receive any input (e.g., a state like "hunger" would have a goal value close to the upper limit as hunger increases over time without any supply of food). $Decay(t,x,y)$ determines how fast the activation level will move towards the goal state as a function of time t, the previous activation level x, and some goal state y (e.g., the activation level of a node could increase in a non-linear fashion over time if a certain threshold for the activation level is surpassed). $Progress(t)$ is a numeric value that reflects the progress made of the action associated with the behavior node with respect to the implicit goal of the action (if the behavior node's action is object recognition, the $progress(t)$ could be a linear function of the time it takes to perform edge detection, for example).

[2] Each of these complex actions consists of various simpler actions that contribute to overall behavior. Because of space limitations, however, we will not be able to describe them and their implementations here.

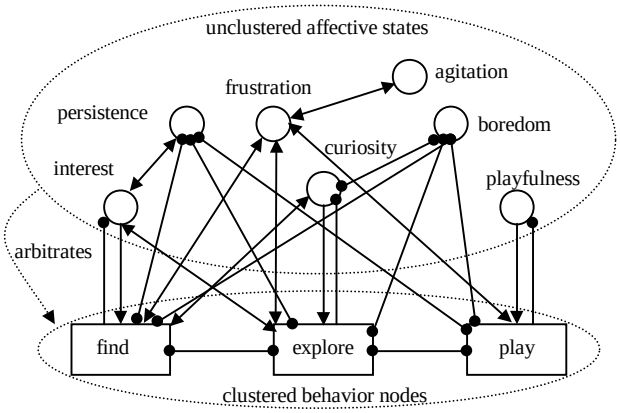

Figure 2. The two-layered architecture based on the *AASBA* scheme for an autonomous robot: the upper oval shows the affective states, whose associated actions arbitrate the complex actions (indicated by boxes) in the competitive cluster (indicated by the lower oval). Arrows indicate excitatory, circles inhibitory influences. Not all connections are depicted here, in particular sensory inputs and motor outputs are missing.

As already mentioned, the architecture has seven behavior nodes that implement affective states labeled "frustration", "boredom", "agitation", "curiosity", "interest", "playfulness", and "persistence".[3] Except for the agitation node, all other nodes are connected to nodes implementing the three complex actions and can influence their activation level in complex ways (Figure 2). The lower-level nodes, in turn, can also exert influence on the affective states, which can be roughly divided into two groups: "agitation", "boredom", "persistence", and "frustration", whose actions together perform the arbitration operation to be described below, and "interest", "curiosity", and "playfulness", which have no associated actions. While the first group of affective states effectively triggers the arbitration operation, affective states in the second group represent the overall activation level of the three sets of nodes implementing the three complex actions, and thus which complex action should become active once arbitration is triggered. We shall briefly describe the functional role of each affective state.

"Boredom" reflects the level of activity of the robot. It increases over time when the robot is inactive, but is lowered every time the robot makes use of its effectors (depending on the magnitude of the activation of the effector). "Persistence" reflects the robot's tendency to stick with the behavior at hand (i.e., to keep the currently active module active), while "frustration" reflects (to some degree) the robot's inability to complete the current task (as implicitly determined by the currently performed action). All actions of lower-level nodes constantly decrease "persistence" and increase "frustration" values by virtue of being simply active. Whenever the robot completes part of its current goal, "frustration" is slightly sated, yet any time actions fail to achieve their implicitly associated goals "frustration" is slightly fueled. Repeated failures cause the robot to get more and more "frustrated", but will also increase "persistence". "Agitation" reflects an overall level of "frustration" of the robot, which depends on past failures to complete a task. Whenever a task is completed successfully, "agitation" goes down. The activation levels of the remaining affective states increase over time and are reduced through the behavior nodes that are connected to them via inhibitory links. "Find" and "playfulness" can in addition be increased by sensory input, i.e., by buttons pressed by users who want the robot to find target objects or who want to play hide-and-seek.

Behavior arbitration is triggered, whenever "boredom" is greater than the difference between "frustration" and "persistence" (i.e., the robot is not performing useful actions) or "frustration" exceeds "persistence" (i.e., the robot repeatedly fails at achieving its goal). If the current complex action does not have the highest activation value as measured in terms of the activation level of its corresponding affective state, control is passed to the set of actions with the highest activation. "Persistence" and "boredom" are reset to their resting values and

[3] Note that these labels were chosen, because the implemented states vaguely resemble ones usually denoted by the label terms. However, there is no claim made that they *are* the same as states (by the same labels) in humans; in fact, we would argue that they are *not the same* (e.g., see also Scheutz [11] for a critical view of the common pratice to label states of an artifact using terms from human psychology).

"frustration" is reset to the current "agitation" value. If, however, the node performing the current action has the highest activation level (as represented by its affective state), then "agitation" is increased by the difference between "persistence" and "frustration" before resetting "boredom", "persistence" and "frustration" as before. Hence, the higher the agitation level, the more likely that behavior arbitration will be triggered (as "frustration" will exceed "persistence"). This is to ensure that if a set of actions repeatedly failed in the past to achieve a task, the robot will very frequently attempt to switch to another complex action. As soon as the activation of the affective state representing another action exceeds the activation of the affective state representing the repeatedly failing action, the robot will switch to this other task.

5. Discussion

So far, we have only conducted a few unsystematic, yet nevertheless very promising experiments with this architecture to see if the arbitration mechanism works according to our expectations. For example, we verified the following prediction about switching between the "explore" and "play" actions: if the robot is placed in a new environment, it should explore the environment until it repeatedly encounters already visited locations (as marked in its internal map). During that time it should be difficult to interrupt the behavior, e.g., by requesting to play "hide-and-seek". Many repeated requests would be necessary to make the activation of "playfulness" exceed that of "curiosity". And even if the activation is higher, it should take a while for the arbitration mechanism to get triggered, which then will cause the robot to switch to "hide-and-seek" behavior (e.g., if frustration is low). If the robot, however, has repeatedly failed at finding new locations, its frustration level should be high, hence behavior arbitration will occur faster. If, in addition, the robot repeatedly failed to map out a room in the past, it should switch to "hide-and-seek" right away.

We are currently in the process of conducting more systematic tests, which will also allow us to fine-tune various parameters of the nodes implementing the affective states so as to give the robot an overall different appearance for different purposes (e.g., "more focussed" as opposed to "more distracted"). Our predications are that this relatively simple architecture—depending on the exact internal configuration—can exhibit a vast variety of different behaviors and reactions, which will be largely unpredictable for outside observers. For this reason, the above architecture using the *AASBA* scheme may be of interest to the entertainment industry, where unpredictable, yet believable agents are needed for the ever-increasing demands of today's computer games.

Our test implementation based on the *AASBA* scheme shows that integrating an arbitration mechanism in an agent architecture can be advantageous, e.g., the possibility of changing the arbitration mechanism depending on internal states (such as the activation level of "agitation"), which then gives rise to different behavioral sequences. In the above scenario, for example, it is futile to attempt to find new locations, if such attempts have repeatedly failed in the past. Hence, modifications of the arbitration mechanisms can make an agent more adaptive to its current environment.

The *AASBA* scheme also demonstrates that complex action control is possible without the need for a deliberative system (e.g., the supervisory system in contention scheduling that is needed to generate top-down excitations on action schemas, [1], [4], [8]). Furthermore, *AASBA* is not restricted to software agents, but can be used on robots under real-time constraints. It is compatible with a great variety of architectures and can be incorporated into existing architectures for distributed action selection.

Finally, a property of *AASBA* that seems worth while exploring, is the possibility of hierarchies of arbitration methods, where one level in the hierarchy determines which arbitration scheme to use at the lower level (the lowest level being the level of actions). Such hierarchies might be extremely helpful in designing versatile agents that learn to adapt to their environments by changing the way they select their actions.

6. Acknowledgments

I would like to thank the students of my robotics course (in Spring 2000) for their various contributions to the individual behavioral modules of the architecture and in particular Tim Brick for his subsequent help with the implementation of the affective model.

7. References

[1] Andronache, V. and Scheutz, M. Contention Scheduling: A Viable Action-Selection Mechanism for Robotics? In Proceedings of MAICS 2002, AAAI Press, 2002.

[2] Arkin, R.C. Behavior-Based Robotics. MIT Press, Cambridge, MA, 1998.

[3] Brooks, R.A. A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation, Vol. RA-2, No.1, March 1986, 14-23.

[4] Cooper, R. and Shallice, T. Contention Scheduling and the Control of Routine Activities. Cognitive Neuropsychology, 17, 4, 298-338, 2000.

[5] Maes, P. A Bottom-Up Mechanism for Behavior Selection in an Artificial Creature. In From Animals to Animats, MIT Press, 1991.

[6] McFarland, D. The Oxford Companion to Animal Behavior, Oxford University Press, 1981.

[7] McClelland, J. L. and Rumelhart, D. E. Parallel Distributed Processing, Vol. 1 and 2, MIT Press, Cambridge, 1986.

[8] Norman, D. A. and Shallice, T. Attention to action: Willed and automatic control of behaviour." Reprinted in revised form in R. J. Davidson, G. E. Schwartz, & D. Shapiro (Eds.) Consciousness and self-regulation, Vol. 4 (pp. 1-18). New York, Plenum Press, 1986.

[9] Snaith, M. and Holland, O. An Investigation of Two Mediation Strategies for Behavioral Control in Animals and Animats. In From Animals to Animats, MIT Press, 1991.

[10] Tyrrell, T. Computational Mechanisms for Action Selection. Dissertation. University of Edinburgh, 1993.

[11] Scheutz, M. Agents with or without Emotions? In Proceedings of FLAIRS 2002, AAAI Press, 2002.

[12] Velàsquez, J. Modeling Emotion-Based Decision-Making. In Proceedings of the 1998 AAAI Fall Symposium Emotional and Intelligent: The Tangled Knot of Cognition (Technical Report FS-98-03). Orlando, FL: AAAI Press, 1998.