Neuromorphically Inspired Appraisal-Based Decision Making in a Cognitive Robot

Stephen M. Gordon, *Member, IEEE*, Kazuhiko Kawamura, *Life Fellow, IEEE*, and D. Mitchell Wilkes, *Member, IEEE*

***Abstract*—Real-time search techniques have been used extensively in the areas of task planning and decision making. In order to be effective, however, these techniques require task-specific domain knowledge in the form of heuristic or utility functions. These functions can either be embedded by the programmer, or learned by the system over time. Unfortunately, many of the reinforcement learning techniques that might be used to acquire this knowledge generally demand static feature vector representations defined *a priori*. Current neurobiological research offers key insights into how the cognitive processing of experience may be used to alleviate dependence on preprogrammed heuristic functions, as well as on static feature representations. Research also suggests that internal appraisals are influenced by such processing and that these appraisals integrate with the cognitive decision-making process, providing a range of useful and adaptive control signals that focus, inform, and mediate deliberation. This paper describes a neuromorphically inspired approach for cognitively processing experience in order to: 1) abstract state information; 2) learn utility functions over this state abstraction; and 3) learn to tradeoff between performance and deliberation time.**

***Index Terms*—Cognitive processing, cognitive system and development, decision making, self-organization.**

I. Introduction

DECISION making in autonomous systems requires purposive deliberation that uses knowledge of the goal specific value of particular states as well as an understanding of what is relevant in the current situation. In addition, decision makers embedded in real-world environments require the ability to balance fast commitment against deliberation to ensure that system operating frequency keeps pace with that of the surrounding environment. It is therefore necessary that these systems be able to quickly identify sufficient solutions to the problem at hand, however, this ability is contingent upon a number of other factors. First, these systems must be able to identify which aspects of the present situation are most relevant to the current goals. This ability enables feature extraction and concept formation, and focuses the deliberative processes. Second, such systems must be able to mine utility functions from experience in order to prioritize and rank potential responses. This involves associating the goal-relevant information with evaluation signals that indicate goal benefit/harm. Third, such systems must have an appreciation of their own ability to tradeoff between solution quality and deliberation time, in order to appropriately portion deliberation given their own resources and the demands imposed on them by the current situation. This last ability is rooted in knowledge, mined from experience and related to goal accomplishment, and enables comparisons of solution quality under various conditions. It is believed that for autonomous systems (e.g., cognitive robots) to make decisions that reflect an understanding of goal relevance, goal specific situational value, and urgency pressures, these systems must mine from their own experience the knowledge that will ultimately be used to inform each of the multidimensional evaluation criteria just described.

In robotics, balanced decision making is critical. Robots operate in environments that are often characterized by complex stochastics and large, typically continuous, state spaces. Real-time decision-making techniques [1]–[4], as well as anytime algorithms [5]–[7], have been successfully applied to robotic systems. However, these methods often require maintaining tabular state-value or heuristic functions ($V(s)$ and $h(s)$, respectively), which can be prohibitively expensive in large state spaces. Function approximators have been used, with some success, to reduce the representational cost of maintaining such functions, but many approximators require preset, static-sized feature vectors that can omit relevant state information. Furthermore, most real-time systems do not adjust performance on a situation-dependent basis, which prohibits learning adaptations that may improve solution quality.

Rather than use static feature vectors, tabular state value and heuristic functions, or preset performance parameters, it would be preferable to have a system "bootstrap" itself through the cognitive processing of its own experience. The result should be the derivation of experience, and task-based dynamic feature mappings that associate situations with task-dependent appraisals, which can then be used to internally signal the deliberation process and adaptively tune control parameters. Such signals should identify the following.

1) What is *relevant* in the current situation, given the current goals?
2) What *utility* should be attached to response options in order to achieve those goals?
3) How *urgently* must the robot perform actions in pursuit of those goals?

In complex biological systems, specific neural circuits have evolved to help creatures cope with and answer each of these

Manuscript received August 05, 2009; revised December 03, 2009. Date of publication February 23, 2010; date of current version March 12, 2010. This work was supported in part by NSF Grant EIA0325641.

S. M. Gordon is with the DCS Corporation, Alexandria, VA 22310 USA (e-mail: stephenmgordon@comcast.net).

K. Kawamura and D. M. Wilkes are with the Department of Electrical Engineering, Vanderbilt University, Nashville, TN 37235 USA (e-mail: kawamura@vuse.vanderbilt.edu; wilkes@vuse.vanderbilt.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAMD.2010.2043530

questions. In such systems these processes may be viewed as operating as innate reinforcement mechanisms, but are also believed to integrate with many aspects of purposeful, cognitive decision making [8]–[11]. Furthermore, the resultant cognitive processes may influence future emotional states by directing attention, associating utilities with actions, and appreciating situational complexity and the subsequent demands for immediate (or nonimmediate) action [12]–[15].

This cycle is often both automatic and controlled: automatic operations are used to quickly appraise relevance, urgency, or utility, while controlled operation measure error, and perform *post hoc* evaluations and reflections [16], [17]. In addition, the multidimensional associations that arise from these processes can be used to provide a means for collapsing complex potential outcomes onto a common currency scale represented by specific emotional states, which can be used in future deliberations (e.g., predicting and planning) [12], [18], [19].

This paper discusses how theories of cognitive processing and the emotional appraisals that underlie such processing can be applied to cognitive control in a robotic platform for improved task performance. Here, and throughout this paper, the term *cognitive control* refers to the type of executive control defined by many psychologists and neuroscientists in which top–down executive processes use attention and working memory, planning and internal rehearsal, along with error correction and novelty detection to purposefully respond to complex situations [20], [21]. The focus of this paper is on the realization of a control system to perform this functionality, and how such a system may be neuromorphically inspired from current neuroscientific research. The discussions of, and references to, emotion within this paper will be from the perspective that emotions provide goal-contingent and situation-based evaluations of *functional* importance to the decision-making process [14], [15], [22]. This involves processing both the current situation and past experience with respect to: what is *relevant* and *urgent*, as well as how much *utility* should be attached to specific responses. There will be three simultaneous aspects to this approach:

1) processing and mining experience, stored as episodic memory, for relational information that can be used to derive situation-based appraisals;
2) representing the mined relations and appraisals for use in online decision making;
3) integrating these appraisals within the control process.

II. Background

A. Internal Appraisals for Decision Making

To investigate how a robotic system can make complex decisions that reflect an appreciation of both the current situation and the robot's past experience, the current researched looked at neuroscientific and psychological theories of emotion and information processing. Critical to this work is the idea that, as a part of the *functional* emotional process, multiple appraisals exist to inform and guide the organism with respect to the various dimensions of the decision-making process. This paper, however, is concerned with the concepts of *relevance*, *utility*, and *urgency*, as these are believed to be integral concepts for decision making in artificial systems.

Zeelenberg and Pieters [13] argue that emotion is a powerful force in decision making, and that emotion has evolved precisely for the production of behavior. They argue that the utility of specific emotions, and the appraisals that underlie them, is found in their ability to affect future behavior, and that each appraisal provides a particular function with its own adaptive value. The primary role of emotion, as they see it, is to serve as motivator in order to *urge* action. This form of appraisal is rooted in experience and extracts information related to "how the individual is doing" (with respect to the current goals) and "what should be done next" [13]. A similar motivation-based appraisal has also been proposed by Peters [23], who argues that such a process is derived from low-level affect [24], [25], and influences the speed at which information is processed. From a functional perspective, the process proposed by Peters is critical for situated agents: it signals when to start and when to stop, and ensures that system needs are met in a timely and appropriate fashion. A similar function, *speed*, has been described in research by Pfister and Böhm [11].

While appraisals for speed and motivation urge the individual to decide and act quickly, often before the deliberative processes can examine all of the possible alternatives, Frijda [14] proposes that appraisals for urgency also establish changes in control precedence and action readiness. Scherer [15], however, suggests that urgency is informed by checks of goal-significance and coping potential. This is compatible with the process of automatic affect proposed by Baumeister *et al.* [16], and is also reflected in separate research by Bechara [26], and Loewenstein [27]. Likewise, Panksepp's [28] evolutionarily primitive affect programs, aspects of the proto-affect signals proposed by Ortony [24], and the interrupts proposed by Sloman [25] are also compatible with this notion of urgency-based appraisals proposed by Fridja and Scherer.

In addition to speed and urgency, Frijda [14], Scherer [15], Peters [23], and Pfister and Böhm [11] all propose *relevance* detection as another emotion-based appraisal. In Peters' theory, low-level affective signals are used to identify pertinent events (i.e., stimuli) that should occupy an individual's focus of attention and influence decision making. Scherer [15] includes relevance within the goal-significance check and describes this signal as a mediator for other cognitive processes.

Other researchers, such as Pfister and Böhm [11], propose that certain emotional states have the power to grab one's attention and focus it intently on specific causes, events, or possible outcomes. In terms of Frijda's [14] theory, these states possess this power by comparing situations and detecting errors. The notion of relevance is also integral to many theories of working memory [29], [30], in which select chunks (quantifiable sets) of information are actively maintained by the organism's cognitive processes for the purposes of prediction and deliberation. It is, potentially, no small coincidence that the neural circuits that are believed to be primarily responsible for the proper functioning of working memory (dorsolateral and anterior cingulated cortices) are located in close proximity to, and receive projections from, those neural regions believed to be responsible for many emotion-based appraisals (orbitofrontal cortex, amygdala, and hippocampus) [31], [10], [12]. However, it is not the goal of this paper to validate or refute any neuroscientific theories, but

rather to suggest to the roboticist possible linkages that may architecturally be important.

The identification and appraisal of relevance is closely related to the ability to associate and assign *utility*. It has been proposed that utility-based signals enable problems to be identified and behavioral responses to be prioritized [13]. Research by Slovic *et al.* [18] proposes that low-level affect acts as a heuristic signal within the decision-making process. With respect to the individual's goals, affect aids comparison between different response options and assigns weights for noting relative importance. There have been several theories that specifically propose utility-based appraisals for "emotional" decision making or integrate emotion within a theory of cognitive control [32]–[35]. However, unlike standard utility values, which provide specific monetary-type gains, affect-based utilities measure anticipated pleasure and pain, as well as other such hedonic factors and may not operate on linear scales. It has even been proposed by Kahneman and Tversky [36] that the mapping function for hedonic value is nonlinear: concave for gains, convex for losses, and has a steeper slope for losses than for gains. Cacioppo and Bernston [37], however, argue that the best representation for such affect-based signals may be to define separate functions over both losses and gains. Their research indicates a slight bias towards positive affect at the zero point, but that negative affect has a steeper slope.

The theory that affect provides an information-laden signal for use as utility in the appraisal process has also been proposed by Schwarz and Clore [25], Peters [23], Slovic *et al.* [18], Pfister and Böhm [11], and Frijda [22]. Schwarz and Clore [25] propose the *affect-as-information-mechanism* (AIM), and contend, like Peters [23] and Slovic [18], that affect provides information related to how an individual "feels" about a situation and that this information guides decision making. In addition, predictions and projections of future feelings are used to adjust responses toward, or away from, particular situations.

In order to use affect for information and utility, it is necessary to collapse numerous, goal-relevant evaluations onto a common scale, or currency [38]. By providing a common currency, utility signals are able to alleviate much of the need for costly and tedious logical evaluations. This same functionality has been noted by Peters [23], but can also be found in the neurobiological research of Montague and Berns [39], and Rolls [9]. Interestingly, the role of "affect as information" appears to be commonly associated with the idea of involuntary feelings, while the notion of "affect as common currency" seems to treat affect as a high-level signal that enables conscious comparisons [23], [18], [26], [38].

The appraisals that have been discussed here are viewed as part of the bottom-up processes necessary for the development and proper interaction of those cognitive tools that enable appropriate, adaptive, and timely decision making. *Relevance* detection enables the individual to identify which goals, events, and appraisals should be focused on and considered. *Utility* enables construction of the decision-space and prioritization of the various response options. Navigation of this decision-space is then achieved by incorporating predictions of future hedonic and utility-based signals (e.g., goal-relevant utility). Finally, *urgency* appraisals, based on the individual's current goals and needs, impose real-time constraints on the decision process by influencing the manner in which the decision-space is searched, as well as specifically signaling events that demand immediate action.

In Section III, this paper will describe how the offline cognitive processing of experience can be used to develop and train the cognitive processes that ultimately appraise *relevance*, *utility*, and *urgency*. These appraisals are based on aspects of various psychological theories of emotion and emotional states, and each appraisal will then be integrated within the decision-making process, while the decision process as a whole must be integrated into a larger cognitive architecture. However, before describing the system implementation, it is necessary to describe what is meant by the phrase "cognitive processing of experience," and to outline how such processing may be accomplished. This requires the introduction of one more concept: *episodic memory*.

B. Cognitive Processing of Experience and Episodic Memory

A second critical aspect of this work is episodic memory: the memory system devoted to the retention and retrieval of an individual's unique subjective [40]. The father of episodic memory, Endel Tulving, argues that this type of memory is unique to humans [40], [41]. Other researchers, such as Clayton *et al.* [42], and Morris [43], however, argue for the existence of "episodic-like" systems in animals. Their arguments are based on research that indicates certain animals are able to process situations with respect to remembered contextual cues and that these animals appear to have the ability to retrieve previously encountered information related to "what," "when," and "where," and to use this information to impact future behavior [42], [44], [45].

Specifically, rats and certain birds have shown the ability to process visual scenes with respect to spatial context and research indicates that this processing exhibits its own episodic-like characteristics [44], [45]. In rats, this processing appears to be mediated by current goals and needs, which agrees with human experiments that suggest that not only are the contents of episodic memory dependent upon the individual's current focus of attention and emotional states, but also that such cues are useful during recall [46]. Therefore, if an individual's current appraisals and emotional states mediate which aspects of a situation become encoded into episodic memory and that subsequent recollection of that episode informs future deliberation, then the process of forming episodes and learning situation-based appraisals should be considered architecturally intertwined.

In this paper, a basic episodic memory system is used to provide the experiential database from which the cognitive processing of experience will train the necessary appraisals. In this context, the phrase "cognitive processing of experience" is used to denote that the specific algorithms applied are inspired and based on theories of cognitive processing in humans and animals, in particular the myriad processes that are believed to be required to form, abstract, associate, and retrieve episodic information that is either stored or indexed by complex relational patterns within the brain.

During training, the episodic memory database is used to create an autoassociative network in which individual episodes

have been generalized, abstracted, and linked with learned appraisals. Partial pattern matching is then used to retrieve this information. Finally, the retrieved information is used to impact deliberation.

III. System Implementation

The current system was implemented through the ISAC cognitive architecture [47], [48]. This architecture has been an ongoing project at the Center for Intelligent Systems (CIS) at Vanderbilt University and has been only, in part, informed by the current research. However, it is believed and will be argued here that this version of the architecture along with many of the general algorithms employed therein can be applied to multiple, specific problems. Throughout the experiments and results sections only a single, general, problem is presented, but the discussion include what classes of problems this approach is currently best suited for as well as what sort of specific algorithmic changes may be required for generalization to other classes of problems.

Functionally, the ISAC architecture specified the layout of components, as well as the interconnections and flow of information between components. The implementation details for each component were then derived from the types of cognitive processing required at each stage and from the psychological and neuroscientific research into similar types of processing within biological systems. Because of the general reusability of most cognitive architectures only the implementation of a few components needs to be described in detail here, as they represent the novel aspects of this work while the remaining components are briefly introduced. For detailed descriptions of the remaining components, the interested reader is directed to [48].

A. ISAC Cognitive Architecture

The field of robotics that attempts to use cognitive control processes to create robotic systems that can cope with the challenges presented by complex, dynamic environments has been referred to as cognitive robotics [48]. Research in cognitive robotics focuses on organizing and understanding how different subcomponents can be used to create a functioning whole, while utilizing psychological and neuroscientific research on biological cognitive systems to inform system design. In order to specify the organizational scheme in cognitive robotics, researchers employ cognitive architectures (first popularized by the production systems of Anderson [49] and Newell [50]). The current research uses the ISAC cognitive architecture [48], shown in Fig. 1. This architecture is used as the conceptual framework in which the appraisal mechanisms introduced earlier must be integrated, and cognitive control must be achieved. This architecture has three distinct control loops similar to those described by Shrobe *et al.* [51], Sloman [52], and Ortony *et al.* [24]. These loops provide reactive, routine, and deliberative control. In addition, there are multiple memory systems, such as short-term, long-term, and working memory. Of these systems, long-term memory is further subdivided into the three categories: procedural, episodic, and semantic. Finally, there is a complex, higher order executive control agent

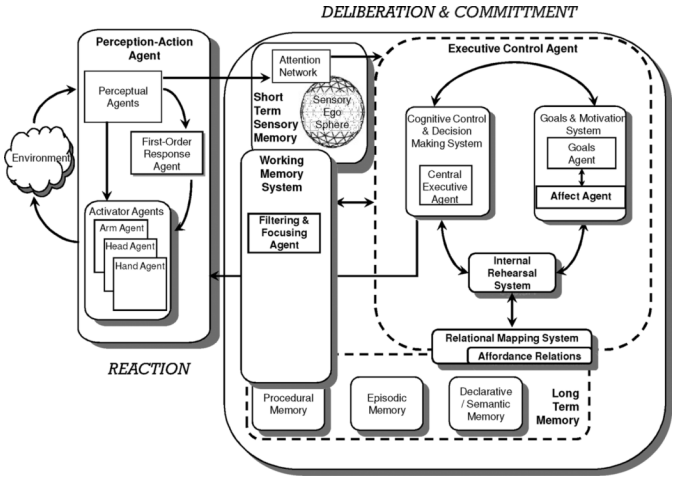


Fig. 1. The ISAC cognitive architecture.

that (among other things) assigns goals, generates plans, and selects responses.

1) Component Descriptions: The *sensory egosphere* (SES) [53] is a short-term memory system that integrates multimodal sensory information. The underlying data structure is designed as a complex geodesic dome consisting of a set of sparsely interconnected vertices. Within the ISAC architecture, 1962 are used, however, this number is merely a parameter of the system. The structure of the SES enables the use of spreading activation networks [54] that perform spatiotemporal coincidence detection and mediate the salience of each percept. Salience values are used as attentional markers, but also facilitate perceptual binding [53], [48].

The *first-order response agent* (FRA) [55] initiates routine behaviors without reliance on more complex, high-level control structures. Routine responses are either preset or learned over time. Currently, routine responses are represented as stored percept/behavior combinations.

The *working memory system* (WMS) implements working memory and is designed to integrate and filter perceptual and procedural information using knowledge of the current goals and situation-based appraisals. Additionally, the WMS provides the gateway through which the high-level cognitive processes interact with the low-level perception and action processes. Previous work [56] with the WMS has used a specially designed working memory toolkit (WMtk) written in ANSI C++ that implements a multilayer neural network function approximator. This toolkit was developed to mimic some of the functionality argued for by [57], [58].

The *long-term memory* (LTM) system is composed of three distinct parts: procedural, semantic, and episodic memory. Procedural LTM retains information related to the performance of behaviors, while semantic LTM retains facts and beliefs about the world. For example, attributes belonging to specific percepts (described in the next section) are stored in semantic LTM, and behavioral control laws are stored in procedural LTM. Episodic LTM retains linked episodes consisting of state-action-outcome sets. States are based on the information stored in the WMS, actions represent the selected (and then performed) behaviors, and

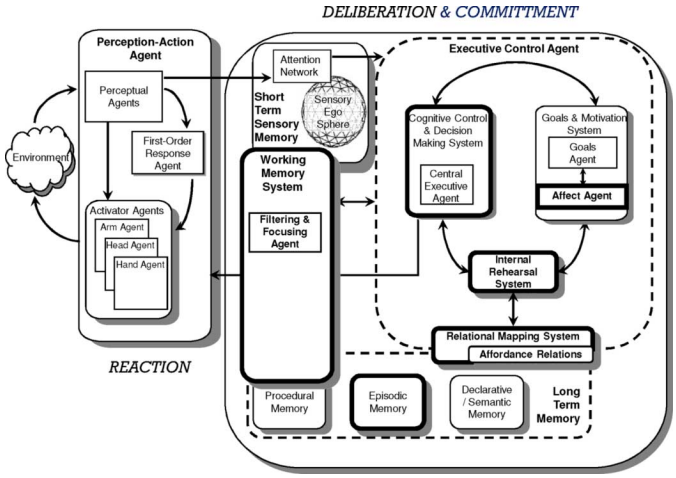


Fig. 2. Primary components of the ISAC cognitive architecture used for this work.

outcomes are the sensed perceptual events, along with resultant appraisals.

The *relational mapping system* (RMS) maintains abstracted representations of states and state features and associates these representations with statistically determined evaluation information (e.g., utility). The contents of the individual relational maps within the RMS are mined from experience (i.e., episodes) and each map enables autoassociative access and retrieval.

The *affect agent* uses knowledge of the current goals to interpret the evaluations retrieved from the RMS and adjusts the system's decision-making strategy accordingly. Adjustment includes adaptively tuning parameters in the cognitive cycle, and interrupting this cycle when necessary.

The *goals agent* assigns goals to the system. Goals determine the current task and evaluations. Goal setting is critical for both the cognitive control components and the low-level perception/action components.

The *central executive agent* (CEA) [59] initiates cognitive control, plans, and selects responses by utilizing knowledge of the current state (from WMS) and the current goals (from the goals agent). The CEA searches the decision-space and prioritizes response options. The current "best" plan is maintained within the WMS, so that it may be rapidly deployed if needed. During planning, interrupt signals are used to halt the planning process and trigger the activation of the best plan. Interrupts are provided by the affect agent, or by the low-level perception systems.

The internal rehearsal system (IRS) [60], [61] internally simulates actions, predicts outcomes, and retrieves evaluations. The IRS uses knowledge of the current goals to retrieve evaluative information from the relational maps. In addition, offline simulation by the IRS is critical in the development of the relational maps.

The portions of the ISAC architecture that are critical for this work and described in Section III-B are shown in Fig. 2. Though these components are consistent with the functional purpose described in earlier work (e.g., [48], [55], and [59]) their implementations differ slightly as a result of having been tailored for the specific types of cognitive processing described previously. These components include ISAC's WMS, RMS, IRS, and portions of the CEA.

B. Dynamic Situation Representation

One of the first steps in evaluating and performing a complex task is to abstract goal-relevant information from the environment. For many tasks, the most readily available types of information are the detectable percepts in the environment (including any known properties of those percepts), as well as the emergent relationships amongst those percepts.

In humans this process of abstract goal-relevant information is often attributed to the WMS, which is believed to be primarily "instantiated" in the prefrontal regions of the neocortex [29], [31], [44]. However, this type of functionality is not specific to humans and research indicates that many animals, particularly vertebrates, are also capable of creating object categories that are mediated, to some extent, by the organism's current goals [43].

Various computational methods have been employed by researchers in an attempt to emulate this ability in artificial systems. In particular, the production rule system ACT-R models this functionality through the use of a complex associative network in which chunks of information are interconnected by weights that mediate association and retrieval [49], [62].

Research by Phillips and Noelle [57] implements a feedforward neural network in which specific chunks of information are selectively attended to and used to complete an orienting task. In their work, a TD learning algorithm is used to modify the network weights based on task performance. Training continues until the system learns which information should be given attention in each of the different situations that may be encountered. It should be noted that the network designed by Phillips and Noelle [57] learns "what to focus on" and not "what to do." Therefore, though network training is based on task performance, the network itself cannot affect task performance except by altering what information is presented to the planning system.

While the model of Phillips and Noelle [57] reflects aspects of the neurological functioning of working memory, their approach, as well as the approach taken in ACT-R, assumes that the network is provided *a priori* with a set of information chunks and that the only task for the network is to choose from these chunks. Yet, neither approach fully addresses how these chunks may be acquired by the system rather than preset by the designer (i.e., engineer).

Assumptions and Applicability Requirements: The process of abstracting chunks, as well as creating and maintaining dynamic representations within the ISAC architecture, is assigned to the WMS. The abstracted chunks are maintained within feature vectors that are designed to capture some of the basic relational properties between percepts. In the current work, learning must be performed to determine which attributes are the most important for creating goal-relevant chunks. It is assumed that the system has knowledge of the possible relationships between percepts and is capable of using the feature vectors to represent these relationships. It is beyond the scope of the current work to provide a similar method for abstracting, identifying, and representing the most relevant interpercept relationships (i.e., "on top of," "in front of"). Sections III-B2 and III-B3 describe the process used for goal-relevant information for chunk formation.

Section III-C then describes how the abstracted chunks and the known relationships are analyzed and evaluated.

For this task, it is assumed that percepts have multiple attributes, each of which can have multiple, distinct values that can be represented symbolically within a declarative knowledge-base, such as semantic memory. For example, a computer desk may be large and dark brown, while a loaf of bread (percept) may be soft and inexpensive. The authors acknowledge, however, that not all attributes are naturally symbolic and in such cases appropriate measures must be taken. One possible solution that can be applied as a preprocessing step, and that the authors have found particularly useful, is to generate symbolic labels through k-means clustering. Another possible solution, which is described in Section III-B2 involves using a modified form of the chunk abstraction algorithm.

As previously noted, it is also assumed that the relational properties between percepts in the environment can be represented by some vector notation, which is referred to as a feature vector in the sections that follow. For example, percepts stacked on top of each other could be represented with the bottom percept appearing first in the feature vector and the top percept appearing last; percept proximity to the robot, or relative locations such as "to the left of" and "to the right of" could also be represented in this manner.

1) Weight Learning in the WMS: The input to this component is the current state, s_i, which is composed of: 1) a list of detected percepts along with positions, orientations, and attribute-value pairs; 2) a set of first order logic statements that describe the environment, specifically the relationships between percepts, i.e., $InFront(x, y)$, $OnTop(u, v)$, etc.; and 3) any numeric rewards that have been given to the system and associated with the current state. The function of the WMS is to append to the current state a constructed set of feature vectors, $\boldsymbol{f}_i$, in which the individual elements represent specific percepts that have been replaced by goal-relevant abstractions. The structure of the feature vector itself represents a fundamental relationship (for the current task) between these percepts. This structure is currently extracted from the first order logic statements.

A simple example is as follows: the current environment consists of a table (percept) with three boxes (percepts) on top. Each box has a distinct size and weight, represented numerically; Box_A and Box_C are the same color ($color_1$) while Box_B is a different color ($color_2$). For this example, these are assumed to be the only known box attributes. The table may also have attributes, but these are not relevant for the current discussion. There is one dimension along which reward is received, and the reward for the current state is zero. For the sake of brevity, the individual attributes of each percept are represented by the terms $\{position, orientation, known\ attributes(\ldots)\}$.

Percepts:
$Table : \{position, orientation, known\ attributes(\ldots)\}$
$Box\_A : \{position, orientation, known\ attributes(\ldots)\}$
$Box\_B : \{position, orientation, known\ attributes(\ldots)\}$
$Box\_C : \{position, orientation, known\ attributes(\ldots)\}$
First Order Logic:
$Table(w)$
$Box(x)$
$Box(y)$
$Box(z)$
$OnTop(w, y)$
$OnTop(w, z)$
$NearEachOther(x, y)$
$FarFromEachOther(x, z)$
$FarFromEachOther(y, z)$
$w = Table$
$x = Box\_A$
$y = Box\_B$
$z = Box\_C$
Reward:
0

One can imagine that this simple state representation could be the representative backbone for multiple problem spaces including, but not limited to: 1) stacking boxes on top of each other or; 2) separating boxes based on color. Furthermore, let's assume that the system does not explicitly know which goal should be pursued, but rather performs reinforcement learning based on its current and past experience. Given this simple example, it would be useful if the system could quickly identify which box attributes are relevant for the current problem and which are not. In other words, for the problem of separating boxes it would be beneficial to know that box color is an important attribute, while for the problem of box stacking the attributes size and weight are important.

Using this knowledge, the system could then ignore attributes that are unimportant and, focusing only on important attributes, re-represent each state using a simpler notation that would is more conducive for subsequent learning, planning, and decision making. For example, by ignoring all attributes except for color, or by compressing size and weight into a single attribute with values $\{GoodBase, GoodMiddle, GoodTop\}$, the system could then pass a more manageable state representation off to additional learning algorithms that look for successful state patterns such as:

Goal 1 – Box Stacking
$OnTop(GoodBase, GoodMiddle)$
$OnTop(GoodMiddle, GoodMiddle)$
$OnTop(GoodMiddle, GoodTop)$
… etc.
Goal 2 – Color Separation
$NearEachOther(color_1, color_1)$
$NearEachOther(color_2, color_2)$
$FarFromEachOther(color_1, color_2)$
… etc.

It is argued here that the foundation for identifying such information should be the system's experience, and thus the processes in the WMS look to the information encoded in long-term episodic memory to perform their function. Once abstracted chunks have been created, the state representation is modified to incorporate those chunks. As previously mentioned, the ultimate outcome of this process is the formation of a set of feature vectors $\boldsymbol{f}_i$ in which each $f_{ij} \in \boldsymbol{f}_i$ is a variably sized representation of the different first order logic statements from s_i that has been collapsed to an indexed vector notation and in which percepts have been replaced by the chunks that represent them. For example, $OnTop(Box\_A, Box\_B) \rightarrow \{GoodBase, GoodMiddle\}$).

These feature vectors will later be used during planning and decision making; however, the original state information is kept, rather than overwritten, so that no information is lost. This also helps future learning.

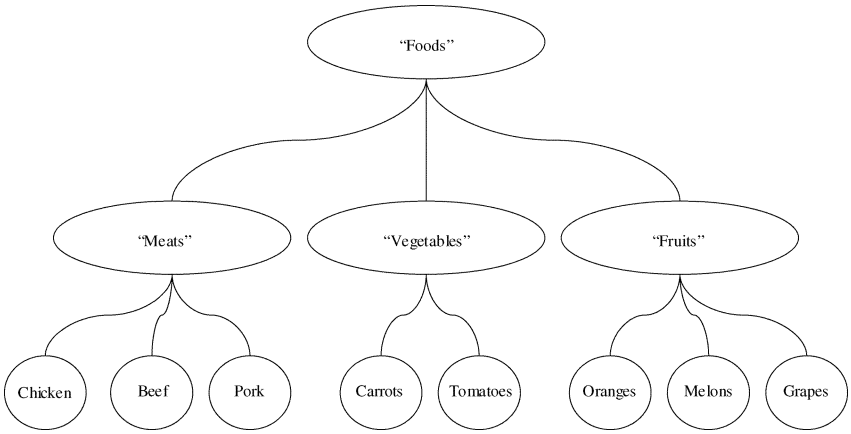


Fig. 3. Example concept hierarchy for "Foods."

To develop goal-relevant chunks from the perceptual information in the environment, it is necessary to learn which attributes are the most relevant for the current goal, and to what extent those attributes should contribute to chunk formation. Here, the system employs a weight-learning algorithm that looks for statistical patterns in individual training vectors (i.e., feature vectors in which percepts have not been reclassified) and uses the evaluations for those vectors to increment specific weights associated with different attributes. Only training vectors that contain multiple percepts are used during training.

For each training vector B_k, in which percepts are represented without abstraction, the system determines the probability for each attribute, A_i, having specific value, V_{ij}, (i.e., $P(A_i = V_{ij}|B_k)$) and the value $\max_i^{Bk} = \max\{P(A_i = V_{ij}|B_k)\}\ \forall j$. It is assumed for this work that attribute predictability is a desirable property in the chunk formation process.

Equation (1) is then used to update the weight (or relevance) for each attribute appearing in B_k. In this equation, the value E_c is the evaluation for the cth reward (assuming multidimensional reward) and for this work can have value $[-1, 1]$. The term $F(E_c)$ is used to map reward to an appropriate weight-priority interval. For example, in the work that follows the function $F(E_c) = (1 + E_c)/2.0$ is used to drive up the weight for highly predictable attributes that are associated with positive reward and to drive down the weights associated with negative reward. However, one can imagine that for different problems this function may take different forms. The term α_ω is the learning rate. $P(A_i = V_{il})$ is the unconditional probability over all known percepts for the given attribute-value pair associated with $\max_{Bk}$ where $l = \text{argmax}\{P(A_i = V_{il}|B_k)\}\ \forall j$. Thus the term $1.0 - P(A_i = V_{il})$ is used to discount the importance given to attribute-value pairs that naturally occur with high probability. Finally, since it is assumed that reward may be available along multiple dimensions, it is desired for relevance detection to be uniformly associated across each reward dimension. In other words, attributes that help predict rewards along more dimensions should be preferred to attribute that predict rewards along fewer dimensions and thus should receive higher weight. Equation (1) must, therefore, loop through each reward dimension, c

$$\omega_i \leftarrow \omega_i + \alpha\omega * (1 - P(A_i = V_{il})) * \max_{Bk} * \{F(E_c) - \omega_i\}. \quad (1)$$

The goal of this update equation is to suppress the relevance for any attributes that appear with probability near chance and to accentuate the relevance assigned to attributes that are highly predictive of positive situations.

The rationale for implementing a one-sided update equation (in this case one that focuses on positive situations rather than negative situations) is that it is often the case in reinforcement learning that the appearance of one type of reward (positive/negative) is a highly selective process, while the remaining type of reward (negative/positive) is then reserved for "all other cases". As an example, consider a robot traversing a maze in which it receives -0.05 in all states except for the goal state, and in the goal state it receives $+1$. The intent of the current approach, in this case, is to focus the relevance detection process on the reward-selective states as these are believed to provide the most information for identifying goal-relevant features.

Once the final weights have been obtained, they are normalized by dividing by $\max(\omega_i)$. This is for presentation and comparison purposes only and does not affect the performance of the clustering algorithm described in the next section, as it is merely multiplication by a constant scalar.

2) Conceptual Clustering: After the weights have been learned, the percepts are partitioned using a variation of the conceptual clustering algorithm COBWEB [63]. The overall role of the weight learning and COBWEB algorithms are to identify the most critical dimensions (i.e., attributes) for goal-relevant percept classification and then to create conceptual classes that reflect this knowledge. Later, during online performance, potentially new percepts can be given goal-relevant labels and related to previous experience based on this classification technique.

The COBWEB algorithm creates a hierarchical class partition, in which individual clusters are created that simultaneously maximize attribute predictability per cluster, while minimizing the total number of clusters. The output of the COBWEB algorithm is a hierarchical concept tree, such as the one shown in Fig. 3 for the general category "Foods". In the partitions created by COBWEB more general concepts are located higher in the tree.

The COBWEB algorithm begins with a single root node at the top of the tree. As new observations are incrementally added to the tree, these observations are filtered down from the root node to the leaf nodes by determining which nodes best predict the attributes for the new observation. At each node, the basic

COBWEB algorithm considers four different operations: place the observation in the best child node (if there are child nodes below the current node), create a new child node, split the current node in two, or merge the two best nodes. To determine which operation to perform, each operation is assigned a utility value, known as *Category Utility (CU_k)* [64], that indicates the "usefulness" of performing that operation at node k. The standard equation for calculating CU_k is shown in (2)

$$CU_k = P(C_k) \cdot \left(\sum_i \sum_j P(A_i = V_{ij}|C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2 \right). \quad (2)$$

Here, $P(C_k)$ is the probability that an observation is a member of class C_k, $P(A_i = V_{ij})$ is the probability that attribute A_i has value V_{ij} over the set of observations, and $P(A_i = V_{ij}|C_k)$ is the conditional probability for that attribute-value pair over just the observations in class C_k. When the COBWEB algorithm considers each operation, it calculates that CU_k as if that operation had been performed, and then selects the operation that results in the highest CU_k.

The modification of the COBWEB algorithm for this research involves modifying the CU_k equation to include the specific attribute weights previously determined. This ensures that the concepts created by COBWEB reflect goal-relevance. The modified CU_k equation is shown in (3), where ω_i is the learned weight for attribute A_i

$$CU_k = P(C_k) \times \left(\sum_i \omega_i \sum_j P(A_i = V_{ij}|C_k)^2 - \sum_i \omega_i \sum_j P(A_i = V_{ij})^2 \right). \quad (3)$$

Because the COBWEB algorithm creates a full concept hierarchy, which begins at a root node (representing all concepts) and eventually branches into leaf nodes (representing the most specific concepts) it is important to have a means of pruning the tree to identify those concepts that are the most significant. One technique to do this is to use the CU_k value at each level of the hierarchy (descending from root to leaves) and prune branches below a certain threshold. However, Biswas *et al.* [65] note that as one descends from the root node to the leaf nodes, the values of CU_k tend to initially increase before eventually decreasing towards the leaves. Therefore, Biswas, [65] suggests that rather than using a preset threshold, branches should be pruned at the level in which CU_k begins to decrease. This is the method used in this work to prune branches and create the final percept-based, goal-relevant abstractions.

The COBWEB algorithm as described here is specifically developed for nominal data. If, however, the perceptual attributes are numeric and it is desired to not preprocess the data to generate symbolic labels, there is another version of the COBWEB algorithm, termed COBWEB/95, that has been developed to include numeric data. This algorithm replaces the probability of attribute-value pairs, $P(A_i = V_{ij})$, with the probability that the attribute value is above or below a specific value [66].

C. Relational Mapping

Once feature vectors have been created from the current state, these vectors must be used to retrieve the utility evaluations that guide the deliberation process. These evaluations are determined by matching each feature vector to internally generated and trained relational maps that abstract and associate these vectors with appropriate evaluations (i.e., expected reward). The relational maps described here are implemented as a set of self-organized neural networks that average and retain the individual feature vectors that have been acquired through experience along with the rewards received in those states in which the feature vectors were encountered.

The rational for using the self-organized map technique [67], [68] is based on the need to combine individual symbolically represented relations (i.e., feature vectors) into a map structure that facilitates organization and association, while linking the generalized relations to specific appraisals. Additionally, such an auto-associative relational technique is also supported by the psychology and neuroscience literature. Research indicates that process of encoding experience, typically expressed as human episodic memory, is based on the ability to autoassociate different representations of salient, relevant features in the environment as well as the structural relations composed of those features [69], [49]. This research indicates that the hippocampus is a critical structure for memory formation and retrieval, and that this structure is highly implicated in performing relational pattern matching [70], [44].

Neural research on rats suggests that the ability to perform allocentric spatial processing is strongly tied to the ability to deploy prior experience for future tasks (often maze navigation) [44], [71]. Such processing combines incoming perceptual information into spatial arrays (i.e., feature vectors) that reflect the important arrangements and patterns of information as it exists in the environment. In humans, however, this representational ability is believed to extend beyond spatial patterns and include abstracted temporal and sequential patterns of stimuli [45]. These representations act as "event codes," or indexing schemes, to retrieve specific memories or activate specific appraisals [69], [72]. Therefore, as the current situation is unfolding, relational representations are formed that both facilitate the encoding of that situation in long-term memory, as well as activate previously encoded representations [69].

Previous machine learning research that has focused on similar relational structures and self-organizing maps (SOMs) has been conducted by Provost *et al.* [73], who used self-organizing distinctive state abstractions (SODA) to learn high-level perceptual features that define distinctive states. As high-level actions are learned, policies are generated using these distinctive states. In addition, research by Martinez [74], Sehad and Touzet [75], and Smith [76] also use SOMs to abstract state representations and learn general policies through various reinforcement and Q-learning techniques.

There has been research by groups interested in the concept of creating relational maps to abstract and represent experience [77], [78]. One such example uses self-organizing incremental neural networks (SOINN) [79] to create basic common patterns and then, hierarchically, form associations between patterns in order to create more abstract pattern representations. Research by Strosslin *et al.* [80] uses recurrent networks to represent navigation information related to location and action. These

networks are trained from experience using Hebbian learning [81]. Another method, described by Kuipers *et al.* [82], uses the well-known, statistical-based SLAM technique [83] to derive local maps, while simultaneously developing hierarchical and topological representations between local map features and actions.

Many of the techniques just described, however, primarily focus on techniques to discretize and represent a continuous (or near continuous) state space for the purpose of then applying standard machine learning algorithms. The self-organizing, relational maps described here, are somewhat similar to the incremental neural networks of [79], but focus on abstracting and autoassociating the goal-relevant information stored in long-term, episodic memory.

Assumptions and Applicability Requirements: The process of deriving relational maps is implemented within the RMS, and is architecturally located between the long-term memory systems and the executive control processes. The approach used here is based on the assumptions that the goal-relevant features and relationships between the robot and environment can be represented by the system and that the system possesses an appropriate metric for determining the *difference* between any two representations.

In this work the representations are called feature vectors and are comprised of individual chunks that have been given symbolic labels, and whose order reflects order dependent relationships between percepts in the environment. However, the representations need not be limited by this vector approach, but may also be extended to other complex methods such as multidimensional matrices, other maps, or directed/undirected graphs. In addition, the individual components do not necessarily have to be symbolic, but could also be traditional numeric values. The take home point, though, is that the system must possess a distance function that provides some quantification of the differences between separate representations.

This, of course, can be viewed as a limiting factor in the approach presented here, but it could also be argued that one of the mysteries related to the neural circuitry underlying emotion is how multiple disparate situations are ultimately represented by an affect-based common currency [9], [23], [38], [39].

1) Self Organization: The specific inputs necessary for the RMS are the symbolic feature vectors $\boldsymbol{f}_i$ associated with s_i. The function of the RMS is to append a utility vector $\boldsymbol{u}_i$ and a confidence vector $\boldsymbol{\chi}_i$ to s_i. The utility vector associates a positive or negative reward with each of the situational components captured by the feature vectors (i.e., goal benefit or goal harm) and the confidence vector indicates how well each individual feature vector matches the system's previous experience. The relational maps are implemented as self-organizing maps as described in [67] and [68].

A SOMs is a multidimensional neural network that uses unsupervised learning to generate generalized and associative representations of the input space [67]. SOMs are composed of an interconnected set of vertices, v_k, and each vertex has an associated weight vector $\boldsymbol{w}_k$ that represents a complete instance, in this case feature vector. During training, the weight vectors are collectively modified by individual training instances, and over time regions of the map self-organize into basic patterns that reflect the trends in the training data.

To train a SOM, a distance function is used to match each training instance $\boldsymbol{x}_k$ to the nearest vertex, v_k. Once v_k has been determined all weight vectors in the map are updated using the update rule shown in (4)

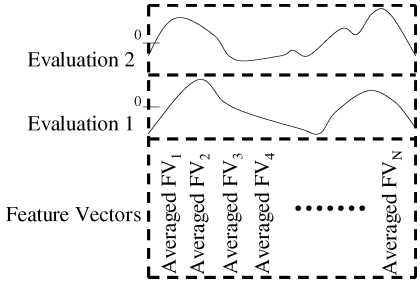


Fig. 4. Combined symbolic and numeric SOM.

$$\boldsymbol{w}_j \leftarrow \boldsymbol{w}_j + \theta(v_k, v_j, t) * \alpha(t) * (\boldsymbol{x}_k - \boldsymbol{w}_j). \tag{4}$$

Here, $\theta(v_k, v_j, t)$ is a neighborhood function that determines the amount of update performed at node v_j based on the distance between v_j and v_k (note: $\theta(v_k, v_j, t) = 1.0 \; \forall k = j$). The function $\alpha(t)$ is the learning rate and both $\alpha(t)$ and $\theta(v_k, v_j, t)$ are designed to decay over time; a measure used to ensure convergence.

Though frequently used for numeric data, SOMs are not restricted to such domains. The key to training a SOM is to have: 1) an appropriate distance function that is defined over the range and types of inputs; and 2) an update function that can modify the desired representation of $\boldsymbol{w}_i$. However, it is possible to define update functions solely through the use of the distance function [68]. Research in Kohonen and Somervuo [68] proposed the use of SOMs for symbol strings and detailed a method for averaging string representations (i.e., symbolic weight vectors). Their method used the Levenshtein (or edit) distance [84] to determine the minimum number of insertions, deletions, or substitutions required to transform one string into another. Dynamic programming was used to find compute an "average" string using all nearest neighbor training instances defined by a discretized neighborhood function, $\theta_d(v_k, v_j, t)$.

Self-organizing symbolic feature vectors are only one of the critical aspects of the required relational maps. It is also necessary to associate evaluative information (utility appraisals), with each individual relational instance, v_k. Therefore, the method for training symbolic SOMs proposed by Kohonen and Somervuo [68] has been extended to include additional numeric dimensions. The technique involves overlaying two SOMs (one symbolic and one numeric), but treating them (and training them) as a whole. A cross-sectional example is shown in Fig. 4. Here, the numeric weight vector has two dimensions. The chosen topology for this work is a standard two dimensional grid where the nodes represent a concatenation of the feature vectors and utility values. This topology was chosen to simplify the representation, future work should investigate additional topologies.

Training the hybrid SOM is accomplished by concatenating the feature vectors (from adjacent vertices) in order to compute an aggregate distance function to identify the nearest vertices for each training example. Once the appropriate vertices are found, the individual weight vectors are modified based on their respective counterparts in the current training instance. This allows

both the symbolic map and the numeric map to be trained simultaneously. The aggregate distance function is shown in (5), where $Eucl(\ )$ indicates the standard Euclidian distance. The vector $\boldsymbol{w}$ is now the concatenated weight vector with components $\boldsymbol{w}_k^s$ and $\boldsymbol{w}_k^n$, the symbolic and numeric portions, respectively. Likewise, $\boldsymbol{x}_k$ represents the individual training instances with symbolic and numeric portions, $\boldsymbol{x}_k^s$ and $\boldsymbol{x}_k^n$. The values $\gamma_{\rm s}$ and $\gamma_{\rm n}$ are additional weights that allow preferential status to be assigned to either one of the individual distance functions. Finally, retrieval is based on matching an input vector to one, or both, of the maps. This is achieved by setting $\gamma_{\rm s}$ and $\gamma_{\rm n}$ appropriately (e.g., $\gamma_{\rm n} = 0$ to match only using the symbolic map)

$$Dist(\boldsymbol{w}_k, \boldsymbol{x}_k) = \gamma_{\rm s} * EditDist(\boldsymbol{w}_k^s, \boldsymbol{x}_k^s) + \gamma_{\rm n} * Eucl(\boldsymbol{w}_k^n, \boldsymbol{x}_k^n). \tag{5}$$

The trained SOM is used to provide the appraisal vector $\boldsymbol{u}_i$ by matching the elements $f_{ik} \in \boldsymbol{f}_i$ to the symbolic SOM, and setting u_{ik} equal to the retrieved numeric set $\boldsymbol{w}_k^n$. However, because the $EditDist(\ )$ function only returns discrete values it is often the case that multiple vertices are "similarly different." As a simple example, using the edit distance function the string "car" is equidistance from both "cat" and "bar." Depending on the situation, either match may be acceptable but to eliminate random behavior and to ensure that the best matches are found, a distance matrix is calculated using the distance values for each $v_j \in V$ and then that matrix is smoothed by averaging across the N vertices nearest each v_i. The distance matrix not only indicates those regions of the map that best match the input string, but also which specific vertices within those regions are closest to the input string. The distance to the nearest vertex, using the smoothed distance matrix, is returned along with the numeric evaluations stored at that vertex. The distance values created from matching f_{ik} to v_k are used to create the confidence vector $\boldsymbol{\chi}_i$ as shown in (6), where $d_{ik} = d(f_{ik}, v_k)$ is the individual distance measure and D is the maximum allowable distance (i.e., the size of the largest stored feature vector)

$$\chi_{ik} = 1 - \frac{d_{ik}}{D}. \tag{6}$$

D. Planning and Executive Control

The planning algorithm is recursively implemented within the executive control agent and performs a depth-limited search through the current decision space. The first order logic statements in the current state s_i are used to determine which actions are possible. To further support planning a set of expected postconditions are assigned to each action. For the current research, this knowledge is provided *a priori*; however, it could be learned from experience using any of a number of techniques. A decision space is created from the known, or more appropriately expected, consequences of each action. The retrieved evaluations from the relational maps for each new state in the decision space are used to order potential actions from best to worst. For each state an overall priority value is calculated as shown in (7)

$$\rho_j = \sum_k ||eval_{jk} - \psi_{jk}|| \tag{7}$$

where

$$eval_{jk} = \chi_{jk} * u_{jk}. \tag{8}$$

The vector $\boldsymbol{\psi}$ represents the highest values obtainable for each $eval_{jk}$ and must either be fed to the system or derived from experience. As each state in the decision space is expanded, only the percentage of b best branches are kept, the rest are pruned. This is done to limit the search to only those responses that are expected to produce the best outcome and while the pruning process is based on expected reward and not value, the assumption used in this work is that in many problems, negative reward states must often be simply avoided. This assumption, though, requires that both negative and positive states be possible in the state space or that the expected reward be an approximate function (i.e., uncertainty in some reward values). Regardless, if only negative reward states are possible, the system will ultimately have to choose one of these states. Once the decision space is pruned, the current best response is chosen and expanded. The process of expanding, pruning, and selecting is repeated until the specified depth limit is reached, or there are no more possible actions, at which point the search backs up and expands the next best state. The current best plan is maintained in the form of a policy over the current search window. If an interrupt signal is generated, or planning must be stopped, the best plan is returned and used to execute actions.

E. Internal Rehearsal

A function that is critical to planning is the ability to appreciate situations and the time constraints, or urgency, imposed by different situations. On the one hand, urgency appraisals are a form of utility, or interrupts, that signal when actions must be performed; however, on the other hand urgency appraisals represent an optimization process in which the system must tradeoff solution quality for deliberation time.

In this work, urgency appraisals are used in two respects: 1) to tune parameters of the decision-making process, thus impacting deliberation time; 2) interrupt decision-making when necessary actions are required. Both of these processes used a technique known as *internal rehearsal*, which (in humans) is a mental process that occurs and enables the simulation and practice of specific behaviors without the need for physical action [85].

Internal rehearsal is a process that proceeds "as if" the person was actually performing the behavior and is a critical method by which humans learn. Examples of how this ability may be implemented in artificial systems is shown by the work of Jirenhed, *et al.* [86] and Erdemir *et al.* [61], as well as the architectural research of Shanahan [87]. In the work of Jirenhed, *et al.* [86] and Erdemirb *et al.* [61] a robot uses an "internal world" to rehearse actions and to investigate the consequences of action. Within this internal world, the robot may either possess a model of the physical environment *a priori*, or be required to learn either the model or features of the model through training. The latter is the case in the research conducted by Erdemir *et al.* [61], in which the robot is required to develop its own understanding of the physical world as well as its ability within the world; where the robot's ability is dependent on the robot's *unknown* physical morphology.

Research by Shanahan [87] takes an architectural approach to internal rehearsal and models a dual loop process in which routine behaviors are constantly produced in response to situations, and these behaviors proceed unabated unless interrupted by higher order cognitive processes. These higher order cognitive processes run in parallel to the routine behaviors, but perform mental simulation of each routine behavioral response before that response is executed [87]. Shanahan's approach, however, assumes that the higher order processes operate at a *higher* frequency than the routine behaviors; an assumption that provides a great deal of difficulty to developers of physical systems (i.e., robots). This type of limitation has also been encountered and noted in research by both Hall [60] and Ratanaswasd [59]; in the latter case rehearsal specifically had to be performed in the interval between receiving a command and executing the reactive/routine behavior.

In an effort to avoid placing such high resource demands on the system's cognitive processes, the rehearsal process in this research is designed as an offline processing tool. This is similar to the approach used by Erdemir *et al.* [61] in which offline rehearsal is used to develop relational knowledge that can later be deployed online in a time-efficient manner. In this work, the system is first allowed to sample its past experience in order to develop a basic state transition model. This model is used when the system needs to predict specific changes in the external environment. Then, using this transition model, the system "mentally" simulates its performance under a variety of conditions and then self-evaluates its performance using the most recently learned utility appraisals. The self-evaluation enables the system, over time, to develop a "sense" of how its performance should be expected to vary by situation. While it must be acknowledged that the learned relevance and utility appraisals do not necessarily reflect the true goal-relevant and utility information, it is argued here that regardless of their accuracy, this information is still useful in that it is ultimately what guides the system's decision making. Thus this type of internal rehearsal, at the very least, operates as a preprocessing tool designed to help the system appreciate how its performance varies under different circumstances. Ultimately, this information can then be used to reduce deliberation time when no further improvements are expected, or to signal instances that require a more thorough search of the decision space.

As previously noted, the appraisals for urgency determine the amount of time allowed for deliberation as well as whether or not the current deliberation process should be interrupted. Interrupt signals are generated in response to actual and expected external conditions. Urgency appraisals made before deliberation begins are used to adaptively preset the decision-making parameters *depth* and *breadth*, and are inspired from the notion of contract and anytime algorithms [88]–[90], [5]. Appraisals made after deliberation begins (i.e., interrupts) are used to halt deliberation in favor of rapid resource deployment and are inspired by the alarm mechanisms described by Sloman [52]. The cognitive processes that enable these urgency appraisals are trained using offline simulation and rehearsal in order to form relations between input states, deliberation time, search depth and breadth, and expected solution quality.

Assumptions and Applicability Requirements: It was not considered integral to the current research to demonstrate that the system was capable of self-generating the transition model that would be used during rehearsal. Therefore, to ease the learning requirements placed on the system, it is assumed here that a transition model is provided. It is also assumed that the system is given regular breaks during which offline training can be performed. Finally, it is assumed that the system possesses a basic understanding of those states in which no further action, or more appropriately *corrective* action, can be performed

TABLE I
STORING PERFORMANCE PROFILES

Feature Vectors ($\boldsymbol{f}_i$)	Depth (d)	Breadth (b)	Deliberation Time (t_i^e)	Solution Quality (q_i)
$\boldsymbol{f}_0$	d_0	b_0	t_0^e	q_0
$\boldsymbol{f}_1$	d_1	b_1	t_1^e	q_1
.	.	.	.	.
.	.	.	.	.
$\boldsymbol{f}_n$	d_n	b_n	t_n^e	q_n

1) Bayesian Networks and Performance Appraisals: Given knowledge of those states in which no further corrective action can be performed the first process of appraising urgency used a known Bayesian model to predict the amount of time, given a specific set of state transitions, until a decisive action *must* be taken (i.e., failure to take action will result in a state from which there is no return). For this work, states of no return were simply defined as states in which potential action possibilities have been removed. The specific significant changes used in this research are described in Section IV.

The offline process of internal rehearsal was used to generate relationships between feature vectors $\boldsymbol{f}_i$ deliberation time t_i^a expected solution quality q_i^e and the search parameters used during deliberation. This is based on the concept of learning performance profiles [5] to model the relationship between specific decision-making parameters, deliberation time, and solution quality and uses the latest domain knowledge to generate these profiles. As the domain knowledge improves, the performance profiles become more useful in reducing deliberation time while preserving solution quality. The learned profiles enable the system to estimate how "good" it can expect to do, given an input set of feature vectors and specific decision-making parameters. The format used to store performance profiles is shown in Table I. In this table, the decision-making parameters shown are the depth and breadth of the search through the decision space.

At regular intervals during the learning process, the system is allowed to randomly select episodes and states and to perform offline rehearsal through these states using the latest learned appraisals for relevance and urgency In other words, the system reevaluates its past experience, in light of the new knowledge, and forms new plans "as if" the previous situations were actually occurring (again). Because the system has already encountered each of these situations, it is acknowledged that this selection process can produce bias towards optimistic appraisals, To counter this, random permutations can be performed on the selected states to create situations that produce slightly new feature vectors. The result is internally-generated experience that can be stored and can be matched as templates in the future to enable extraction of the appropriate performance profiles.

During online task performance each set of feature vectors is matched to the data stored in the performance profiles, and the N best matches are returned. The matches are first pruned by

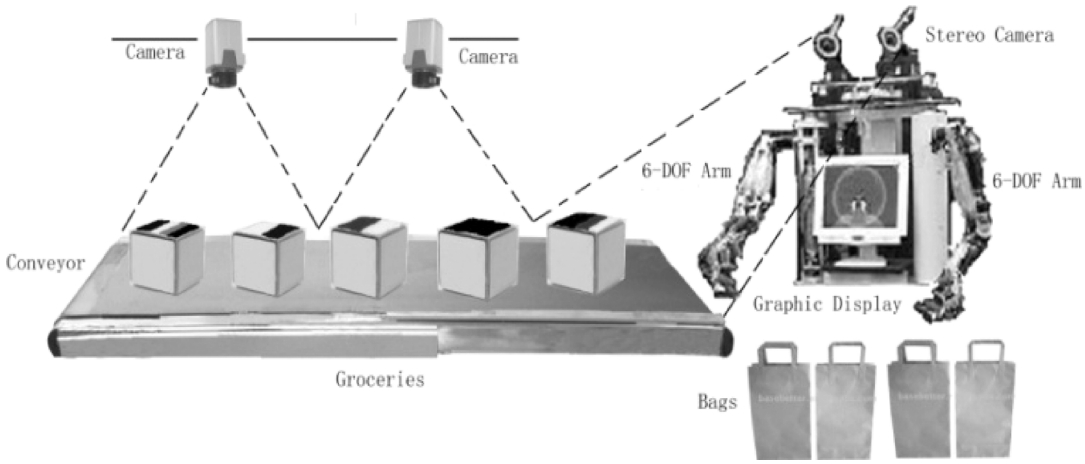


Fig. 5. Experimental layout for grocery bagging.

removing all instances in which the expected deliberation time exceeds the allowable deliberation time. Next, the "best" match is selected from the remaining set by maximizing the tradeoff between solution quality and deliberation time.

F. System Integration

The appraisals described here are designed to facilitate appropriate, adaptive behavior. Urgency appraisals can only mediate deliberation if the system has basic knowledge of its own performance abilities, and such knowledge is intricately related to situation-based expectations and predictions of utility. Interpreting utility requires that relevant features of the situation be detected. Therefore in new situations, an adaptable system must (at minimum) be able to appraise what is relevant in that situation, determine the utility of individual responses, and be able to adjust its own deliberation time as dictated by its goals and the situation.

Proper investigation of how these appraisals may be derived and used to impact performance requires an understanding of each appraisal's architectural role within the cognitive control process. Within the ISAC architecture, the short-term and working memory systems provide the initial buffers for perceptual and state information. In addition, the creation of goal-relevant dynamic representations (i.e., feature vectors) is fundamentally linked to relevance appraisals and is assigned to ISAC's WMS. In this context, working memory must interact closely with the current goals to appropriately filter and process incoming stimuli. Utility signals must be used to prioritize responses, make predictions, and recursively generate plans. In the ISAC architecture, this functionality is placed within the executive control agent, which houses the internal rehearsal system, the goals and motivation system, and the cognitive control and decision making system.

IV. Experiments and Results

To test and evaluate this system, experiments are needed that require the system to identify what is *relevant* in current given situation, assign *utility* as appropriate, and manage deliberation time when *urgent* action is required. In addition, the experiments should require the satisfaction of multiple, simultaneous concerns, in order to stress the system's ability to use these appraisals to perform balanced decision making.

During each experiment, it must be necessary for the system to learn to correctly appraise each situation along multiple dimensions in order to appropriately balance behavior selection. No single constraint will be given *a priori* the status "more important"; therefore while the system attempts to balance behavior selection, there will be no preset constraints on how such balance is achieved. This is intended to cause the system to select those constraints it deems to be more important. Furthermore, because many of the algorithms used to implement the system are unsupervised, system performance may exhibit oscillatory behavior while learning. Due to the fact that behaviors are preferentially ordered using a one-step lookahead, as the accuracy of the relational maps increases the system's initial reactions will become more accurate. This increase in accuracy should ultimately result in less time being required to obtain appropriate solutions.

Finally, because the system is, by design, an experiential learner, aspects of the behavior may demonstrate sensitivity to the type of experience encountered as well as the underlying distributions from which that experience is drawn. In other words, increasing experience may cause the system to become temporarily trapped in a local niche. Within the niche, performance should be very accurate, but near the fringes of the niche performance should decrease. The location of this niche determines, in part, the type of new experiences acquired by the system (i.e., it will begin to take different paths through the decision space). The new experience should expand the niche as well as shift the location of the niche. The system should improve performance while simultaneously choosing more difficult situations.

A. Experimental Layout: Bagging Groceries

The hardware system used in this work is the ISAC humanoid robot, shown in Fig. 5 and described in [47], [48]. The designed experiment is based on the "everyday" task of bagging groceries. In this experiment, the robot is situated at the end of a conveyor belt and must successfully bag all groceries that appear on the belt. To the left and right (and within the robot's

workspace) are tables upon which boxes (grocery bags) are placed. As groceries are deposited in front of the robot, the robot must place each grocery in a bag, and in so doing, observe certain constraints. The experimental layout is shown in Fig. 5.

While bagging groceries, reward was provided along three dimensions, which were used for evaluating system performance. These dimensions reflect three constraints that must be observed to ensure success and are as follows.

1) *Do not destroy any groceries* (e.g., "crush bread with potatoes," "break eggs with milk," "mix hot and cold items").
2) *Do not overload a bag* (e.g., "20 lbs of groceries in a single bag").
3) *Do not use too many bags* (e.g., "10 groceries and 10 bags").

An external critic was used to determine whether or not the robot had successfully adhered to these constraints, and provided reward along each of the three dimensions. The critic was based on a set of predefined rules designed to reflect the nature of the grocery bagging task. For the sake of brevity, these rules are not presented here. However, for constraints (1) and (2) variable placeholders and preset thresholds were used to determine if each constraint had been violated. For example, "if the total weight of groceries placed on top of a *soft* item also exceeded the weight of the *soft* item, the *soft* item was crushed and constraint (1) violated," or "if the total volume of groceries in a bag was more than x (where x was a preset threshold), the bag was overload and constraint (2) was violated." Unlike the rules for constraints (1) and (2), though, the rules for constraint (3) were implemented as fuzzy rules that defined varying amounts of reward based on the number of total groceries bagged and the total ratio of groceries to bags for a given episode.

The groceries used for this experiment were represented by a number of colored cardboard squares. Each grocery had a total of nine attributes, which were known to ISAC *a priori*. A total of 25 groceries were used, as shown in Table II. Because the conceptual clustering algorithm requires symbolic attributes, all nonsymbolic grocery attributes were passed through a k-means algorithm to create symbolic representations, as described in Section III-B1.

For this experiment a set of specific actions were provided to ISAC as a form of procedural knowledge. These actions were abstracted considerably to reduce the amount of computational effort required to search the decision space. The actions were stored in procedural memory as complex behaviors. During action execution an independent controller was responsible for performing each of the many individual steps constituting an action. For the purposes of this research, the highly abstracted actions that were chosen were $BagGroceryLeftArm(x, b)$ and $BagGroceryRightArm(x, b)$, where x and b represented specific groceries or bags, respectively. An independent controller was used to oversee the performance of each high-level action.

TABLE II
GROCERY SET

Name	Color	oz.	in^3	Firm	F^0	$	Type	Healthy
granola	red	10	48	hard	75	2.29	T_1	Y
tissue	white	8	252	soft	75	1.45	T_7	X
soda	red	64	294	hard	75	1.50	T_6	N
ice cream	blue	23	70	hard	32	2.99	T_4	Y
frozen pizza	red	11	81	hard	32	1.25	T_1	N
chicken	pink	24	108	hard	32	3.05	T_5	Y
milk	white	69	160	hard	45	1.67	T_4	Y
cereal	blue	17	227	hard	75	3.15	T_1	N
oranges	orange	64	504	hard	75	4.29	T_3	Y
potatoes	brown	80	504	hard	75	3.99	T_2	Y
strawberry	magenta	16	140	soft	55	2.30	T_3	Y
bread	brown	16	325	soft	75	0.89	T_1	Y
rotisserie	brown	32	160	hard	150	7.99	T_5	Y
eggs	yellow	24	144	soft	45	1.59	T_5	Y
hot soup	white	12	48	hard	175	3.50	T_2	Y
chips	yellow	12	378	soft	75	2.68	T_6	N
yogurt	blue	24	75	hard	45	1.59	T_4	Y
frozen fruit	blue	48	180	hard	32	7.00	T_3	Y
ziploc bags	blue	10	99	hard	75	2.99	T_7	X
spaghetti	brown	12	22	hard	75	1.19	T_1	Y
green beans	green	15	25	hard	75	0.50	T_2	Y
cucumber	green	28	72	hard	50	2.40	T_2	Y
teriyaki bowl	black	17	105	hard	150	5.29	T_5	N
fruit juice	magenta	64	198	hard	75	2.18	T_3	N
tuna	blue	20	86	hard	75	7.39	T_5	Y

B. Experiments

Three experiments were performed to evaluate and validate the system's design. The first two experiments involved the use of a large number of episodes for training and testing. These experiments were performed in simulation as it was impractical to use the physical hardware when performing such comprehensive testing. However, the simulation did model the physical limitations of the robot, with respect to the workspace of each arm. This was done so that the simulation could not "skip ahead" and choose the order in which groceries were bagged. It is noted that the simulation also provided benefits with respect to the noise and nondeterminism associated with sensing and acting in the real-world.

For the simulated experiments, the same control architecture was used; however, the perception and action agents (Fig. 1) were modified in order to interact with a simulated robot rather than the real environment. In each of these first two experiments, the system was essentially faced with a continuous sorting problem in which there were p known items that had already been sorted (i.e., "groceries in bags"), n known items that currently required sorting ("groceries on the conveyor belt"), and an unknown number (m) of unknown items that had yet to be presented (i.e., "groceries still in the shopping cart"). The purpose of these experiments was to provide the system with a large number of experiences that could be dealt with and then evaluated quickly.

The third experiment involved the use of the ISAC robot, and thus included many of the various time constraints involved with acting in the real world, as well as some of the noise constraints associated with robot perception and action. Whereas the first experiment was designed to evaluate the long-term trends in system performance, this experiment was designed to test how well the system could perform when faced with a "less than ideal" environment. Due to time and resource constraints, fewer

episodes were used and, therefore, task constraints were modified slightly to reflect the fact that the robot had fewer opportunities to learn.

The general process for each experiment was to present the system with a set of groceries that need to be bagged. As these groceries were bagged, additional groceries were presented until a prespecified number M had been reached. Once all groceries had been bagged, the episode was over and the system was provided with a reward for each of the reward dimensions described earlier. The first two dimensions were evaluated on a *per bag basis*. Thus, two rewards were provided for each bag present at the end of an episode. The third dimension, however, was evaluated on a *per episode basis*.

The groceries used for these experiments are listed in Table II. Each grocery has nine different attributes

$$\{name, color, weight, size, firmness, \\ storage\ temperature, price, food\ type, healthy\}$$

where the value for each attribute was determined empirically from groceries observed at a local grocery store. The name used for each grocery referred to common grocery names and not specific product types. The color was determined from the predominant color seen on that grocery, or on its container, at a local grocery store. Weight and size were determined from physical measurements. Two categories were used for firmness $\{hard, soft\}$, which were based on a particular item could be bruised or smashed. Storage temperature was based on whether the grocery should be stored at room temperature, in a standard refrigerator, or in the freezer. Price was determined directly. Food type was based on where the grocery should be placed on a standard food pyramid [91]. The final attribute, healthy, was determined somewhat subjectively based on the nutritional content for each item. All none symbolic attributes (i.e., *weight*, *size*, *temperature*, *price*) were passed through the k-means clustering algorithm to determine the symbolic labels. For the attributes *weight*, *size*, and *temperature*, the value $\{k = 3\}$ was used. For the attribute, *price* the value $\{k = 4\}$ was used.

In each experiment, the system was trained and evaluated concurrently using its own acquired experience. This involved randomly selecting groceries to be placed on the conveyor belt, allowing the system to bag groceries using its current knowledge, and randomly selecting new groceries to be placed on the conveyor belt. Therefore, at each step of the planning process, the system was unaware of how many and what type of groceries may appear at the next time step. After a predefined number of groceries had been selected (and subsequently bagged), the episode was concluded. Retraining was performed every 10 episodes using all episodes encountered up to that point. For each episode, the number of groceries presented was randomly selected from the range {10, 15}. This range was chosen simply to keep the episodes reasonably short by limiting the size of the decision space for the worst case scenario (i.e., p groceries all bagged individually, with n groceries waiting on the conveyor to be bagged, and m groceries waiting in the queue).

During experiments 1 and 2, the size of the relational map for constraints (1) and (2) was set to 40×40 and the size of the relational map for constraint (3) was set to 25×25. The difference in size of these two maps is based on the fact that much more training experience was generated for the first map. These values were empirically determined during initial tests to allow the system to evaluate a range of possible responses, while simultaneously enabling experiments to be performed timely and efficiently.

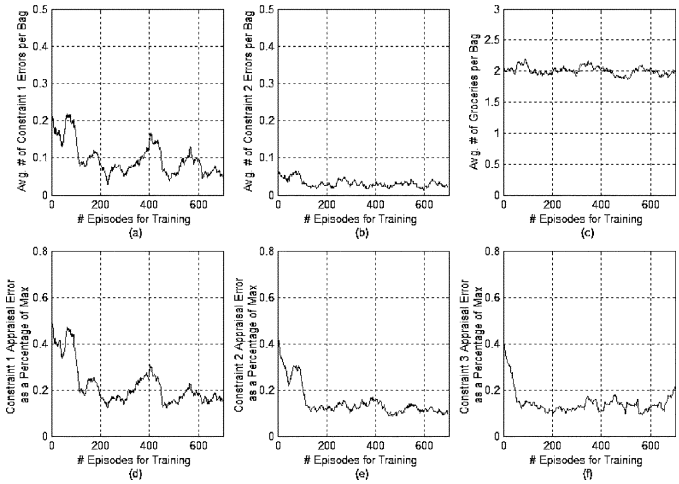


Fig. 6. Evaluation graphs for episodes generated during experiment 1.

Experiment 1 was performed to only evaluate how well the system learned the appraisals *relevance* and *utility*. For this experiment a search depth of 3.0 and a breadth of 100% (i.e., keep 100% of the branches and prune 0%) were used.

Experiment 2 was performed to evaluate whether or not the performance demonstrated in experiment 1 would transfer to a situation in which *urgency* appraisals must also be learned. For the first 100 episodes in experiment 2, the same search parameters were used. After these 100 episodes (and each subsequent 100 episodes), the *urgency* appraisals were trained. Therefore, after the first 100 episodes the system was allowed to dynamically reset its search parameters.

Experiment 3 was performed to investigate system performance under more real-world constraints, in which interrupts must be identified and used to halt lengthy deliberation in favor of rapid action. Experiment 3 used knowledge acquired during experiments 1 and 2 to guide deliberation, but used a (predefined) Bayesian model of the environment to generate interrupts signals and force action before the deliberation was complete.

Finally, it should be noted that the system is being required to learn multiple concepts in parallel and to apply the learned knowledge to ever-increasing corpus of experience. It is, therefore, difficult to extract a true baseline for comparison of performance. Rather, the system is evaluated with respect to whether its performance "gets better," "gets worse," or "stagnates" over time. It will be argued, however, that the results indicate that neuromorphically inspired decision making, in which multiple appraisals are ultimately learned in parallel, is a feasible approach to generalizable, autonomous intelligence.

1) Experiment 1: Fig. 6 presents the results for experiment 1. Fig. 6(a) shows that the number of constraint (1) errors per bag decreases with training. This decrease is achieved while the system maintains a consistent ratio of *groceries per bag*,

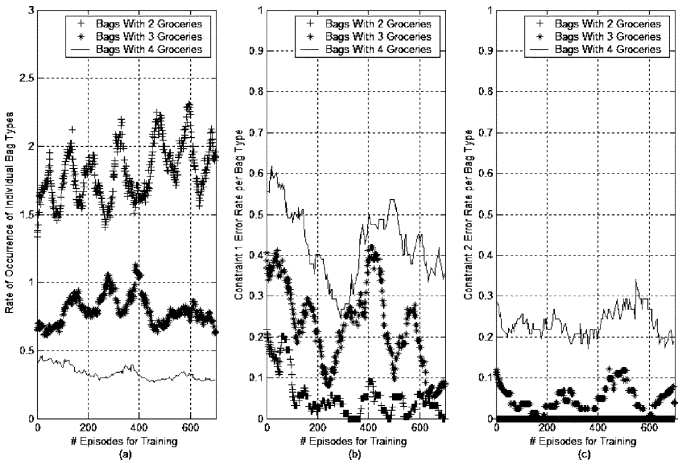


Fig. 7. Breakdown of error rate and rate of ocurrence for bags with different amounts of groceries.

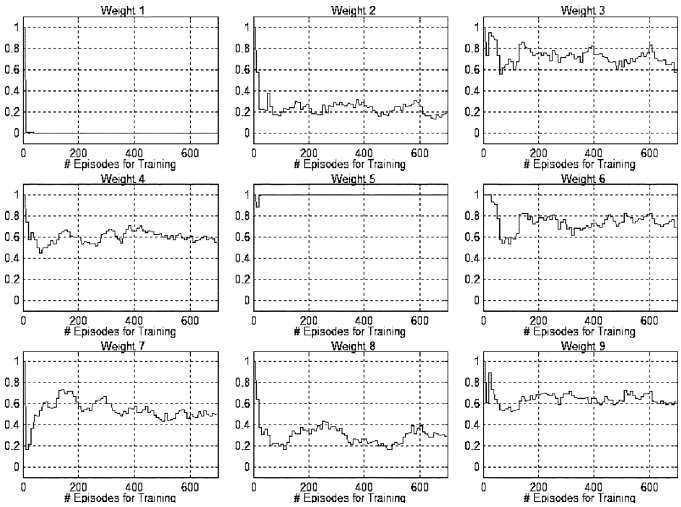


Fig. 8. Learned weights with increased training.

which is shown in Fig. 6(c). In addition, the number of constraint (2) errors per bag [see Fig. 6(b)] shows a very slight decrease. Fig. 6(d)–(f) show that with increased training the system's utility appraisals become more accurate.

The results indicate that the system learns to decrease the error rate per bag while not sacrificing the number of groceries per bag. In other words, the system is not simply learning the rule "place one item in one bag." While having only two groceries in a bag may seem like a low number it must be noted that only 10–15 groceries were chosen for each episode, which could be split into two groups (one for each arm). The system did have the option of reaching across its body to grasp a grocery on the opposite side and place it in a bag on the near side; however, because the simulation was modeled after the physical robot, grasping a grocery was only allowable when that grocery was within the designated arm's workspace. Since groceries on the left (right) side of the robot reach the left (right) arm's workspace first, this caused the effect that the groceries on the left (right) tended to be handled by the arm on the left (right) an overwhelming percentage of the time.

Effectively, this cut the decision space in half and on average 5–7 groceries were being bagged per side, Since the entire grocery set was chosen in order to have several incompatible subsets (i.e., *bread/potatoes*, *milk/eggs*), it is not unreasonable to expect that each side would have at least two bags so that hot/cold, heavy/soft, big/small groceries could be separated.

In addition, it was observed that the SOMs tended to learn iteratively and that simple pairs (2) were picked up quickly, more complicated triplets next, and occasionally quadruple. This effect can, somewhat, be observed from Fig. 7(a), which plots the number of bags with {2,3,4} groceries per episode. Throughout the first 400 episodes, the system increasingly creates bags with 2 and 3 groceries per bag, while simultaneously improving the error rate per bag [seeFig. 7(b)]. However, at 400 episodes an event happens that causes the system to dramatically backoff from its exploration and take a much more conservative approach. From inspections on Figs. 6 and 7, it appears as if the system encountered local minima in which the relational map that it has learned (or the grocery classifications) is no longer

TABLE III
FINAL LEARNED PARTITION USING 700 EPISODES (30% OCCURRENCE)

Class	Grocery
C_0	ice cream, chicken, yogurt, cucumbers
C_1	granola, ziploc bags, frozen pizza, cereal, spaghetti, green beans, teriyaki bowl, hot soup
C_2	rotisserie
C_3	tuna
C_4	milk, frozen fruit
C_5	oranges, potatoes
C_6	soda, fruit juice
C_7	tissue, bread
C_8	chips
C_9	eggs
C_{10}	strawberry

applicable to the new experiences. Since the original grocery set was not expanded, it suggested that the system was trying to identify a pattern that did not truly hold in the real world. After 400 episodes, the error of this pattern was exposed and the system was forced to reduce exploration.

This intriguingly suggests that the system is highly sensitive to its own experience, and can become "pessimistic" in response to harsh criticism (i.e., the high probability of negative rewards between episodes 400–500).

Further examination of Fig. 7 reveals jitter within the specific performance levels. Some of this jitter may be explained by noting that the system is required to simultaneously learn both *relevance* appraisals and *utility* appraisals. Fig. 8 plots the individual weight values as a function of the number of episodes used for training. Here, the weight associated with the grocery's *name* immediately drops to 0.0, while the weight associated with *firmness* of the grocery stays at 1.0. This result is not surprising because the weight learning algorithm is essentially looking for attributes that predict the bag success/failure. Since the *name* attribute is unique for each grocery, it stands to reason that this attribute provides zero predictability. Likewise, stacking hard groceries on top of soft groceries is an easy way

TABLE IV
LEARNED PARTITION (25% OCCURRENCE)

Class	Grocery
C_0	ice cream, chicken, yogurt, cucumbers
C_1	granola, ziploc bags, frozen pizza, cereal, spaghetti, green beans, teriyaki bowl, hot soup
C_2	rotisserie
C_3	tuna
C_4	milk, frozen fruit
C_5	oranges, potatoes
C_6	soda, fruit juice
C_7	tissue, bread
C_8	chips
C_9	eggs, strawberries

TABLE V
LEARNED PARTITION (25% OCCURRENCE)

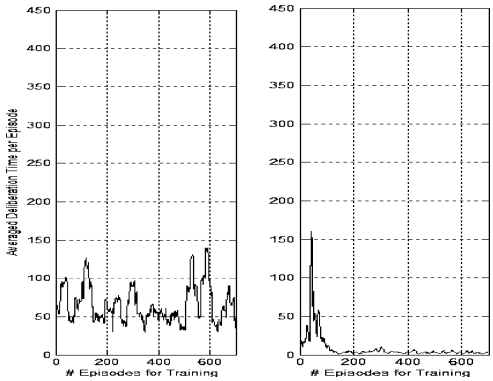
Class	Grocery
C_0	ice cream, chicken, yogurt, cucumbers, eggs, tuna
C_1	granola, ziploc bags, frozen pizza, cereal, spaghetti, green beans
C_2	rotisserie
C_3	teriyaki bowl
C_4	milk, frozen fruit
C_5	oranges, potatoes
C_6	soda, fruit juice
C_7	tissue, bread
C_8	chips
C_9	strawberries
C_{10}	hot soup

to violate a constraint and the quickly learns that *firmness* is a highly relevant attribute for bagging groceries.

Interestingly, the weight values that exhibit the most oscillation are those associated with the attributes *color*, *price*, and *type*, which are three attributes that are not required for grocery bagging. It appears that while these attributes are believed to be the least relevant, the system has a very difficult time pushing these weights to zero. This is explained by the fact that spurious patterns may arise within the data set and thus attributes that have little-to-no bearing on the outcome may (through association) become positive or negative based on the systems current experience.

Table III presents the final grocery clusters after 700 episodes of training. However, over the final 30 training epochs (episodes 400:700) these clusters were only identified $\sim 30\%$ of the time. The remaining training epochs are nearly uniformly distributed across three additional partitions. Two of these partitions are presented in Tables IV and V. Each differ from the partition presented in Table III by just a small amount, and each occur $\sim 25\%$ of the time. The third partition appears 20% (6/30) of the time and is an inconsistent partitioning in which class C_0 and C_1 (Table III) have been arbitrarily split into 3–4 new classes, however, of these arbitrary partitions only occur once in the final 15 training epochs and not at all in the final 10 epochs. The take home point regarding these learned partitions is that for the final 300 episodes (30 training epochs) as the learned weights fluctuate the system alternates between the three similar classification schemes presented in Tables III–Table V, approximately 80% of the time, with the classification scheme of Table III occurring most frequently (30%).

2) Experiment 2: Next, the system's ability to intelligently modify its search through the decision space was evaluated. This included the urgency appraisals used to adaptively preset the search parameters *depth* and *breadth*. The generation and evaluation of *interrupt* signals was postponed until experiment 3, the experiment using the physical ISAC system. Experiment 2, therefore, tested the system's ability to generate useful performance profiles and to employ these profiles in future grocery-bagging situations.



(a) (b)

Fig. 9. Comparisons of deliberation time in seconds.

The experimental procedure used here was the same as that used for experiment 1, with the exception that a new step "process experience to learn urgency appraisals" was included. Whereas the appraisals for *relevance* and *urgency* were trained after every 10 episodes, in this experiment the appraisal for urgency was trained every 100 episodes. The rationale for this discrepancy was the computational costs associated with internally rehearsing past experience to derive the performance profiles. For every 100 episodes the system selected 10 episodes at random from episodic memory for internal rehearsal. This number was chosen from previous tests, which indicated this amount provided a good balance between analyzing performance and deliberation time.

It is important to recall that during internal rehearsal that for each situation a variety of depth and breadth values were used. During training, the system repeatedly analyzed it's experience for the following search depths {1, 2, 3} and breadths {10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%}, where the minimum number of branches kept was forced to be one.

Fig. 9 presents the averaged deliberation time for both the previous experiment [Fig. 9(a)] and the current experiment [Fig. 9(b)] per episode. In both cases averaging was performed using a square filter of width 30 episodes. The large oscillations in deliberation time for the first experiment were the result of

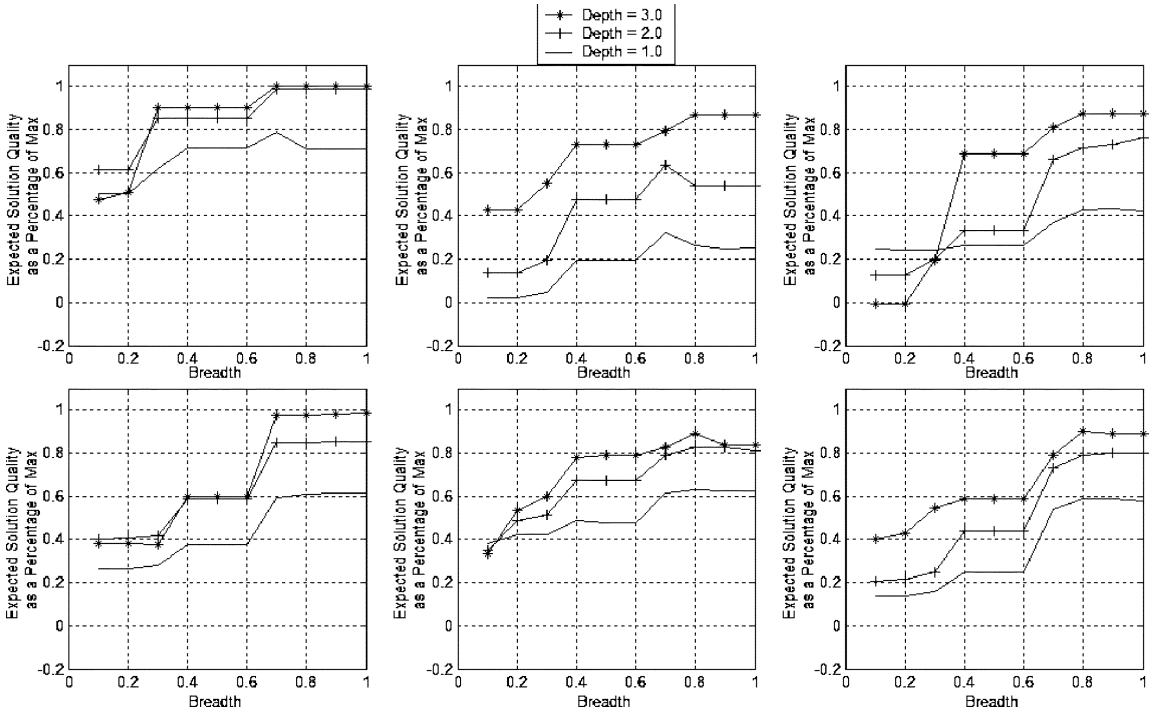


Fig. 10. Learned performance profiles for appraising urgency and adjusting the search parameters depth and breadth.

the random nature of episode generation and the static nature of the planning algorithm.

In the current experiment, however, there was a significant decrease in the mean deliberation time. After the first 100 episodes, the average deliberation time shows a dramatic improvement. This corresponds to when the system begins learning the performance profiles, and thus to maximize solution quality versus deliberation time by adjusting *depth* and breadth.

While the trend in Fig. 9(b) shows improvement in the mean deliberation time, it also appears as if the improvement was more of a global change than a set of specific responses to individual situations. Further analysis revealed that the system overwhelmingly was choosing to limit its branching factor to 10% or 20%, and was oscillating the search depth between 1 and 2. Given the current experimental setup, and such low branching factors, the system was often only required to search 1–2 branches at each search depth. Obviously, this resulted in the drastic reduction in deliberation time. It is important to note that the system is still learning to reduce deliberation time, but the current experimental setup left the system with an "all or none" choice. It is an area of current research to design experiments that are less sensitive to changes in the branching factor, but these results are shown to illustrate that including appraisals that are based on previous experience and attempt to optimize performance against deliberation time is a potentially useful and functional aspect of cognitive decision making.

Fig. 10 shows how the solution quality changes as a result of changes in the search parameters. The claim that the search algorithm approximates the anytime property is supported experimentally by the results presented in Fig. 10, which plots the learned performance profiles for each of the six internal rehearsal epochs. While there are some exceptions, Fig. 10 shows that, in general, the solution quality with respect to the system's internal appraisals increases with both search depth and search breadth. Furthermore, the variance across these results can be explained by noting the following.

1) The system does not rehearse the same states at each training epoch
2) The system can only use the most current knowledge, and that this knowledge is derived through unsupervised learning which necessarily injects some variability.
3) These plots, by design, only reflect the percentage of maximum solution quality achievable and not the actual maximum solution quality.

Fig. 11 presents the performance evaluations for experiment 2. Fig. 11(a) and (b) show that the final error rate for both constraints (1) and (2) were not as good as in experiment 1. This is attributed to the fact that less search was performed. In addition, the system had more difficulty learning the appropriate utility appraisals, though its improvement is more dramatic than in experiment 1.

Fig. 12 shows a more detailed analysis of the error rates per bag type. Similar to experiment 1, the error rates for bags with two and three groceries per bag show both oscillation and a net decrease over time. However, the error rates for bags with four groceries actually get worse. The first explanation for this behavior is that the reduction in the branching factor keeps the system from identifying better alternatives by pruning all but a single branch: the unsatisfactory, four-groceries-in-a-bag branch. A second explanation, however, notes that the system creates bags with four groceries at a much lower rate than in the previous experiment. This leads to fewer opportunities to learn which, in part, explains the poorer performance.

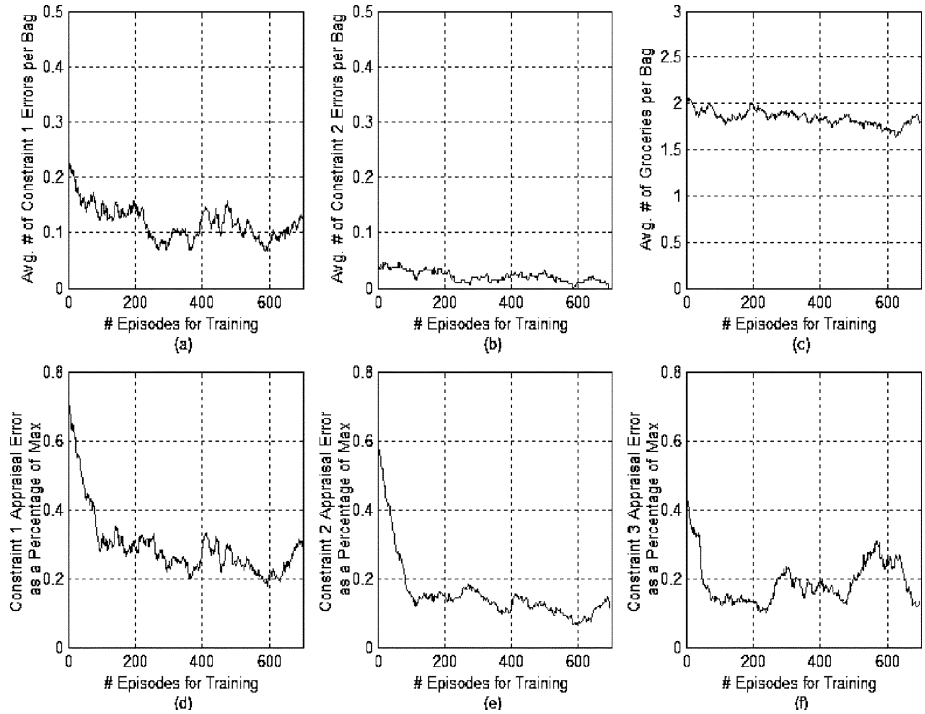


Fig. 11. Evaluation graphs for episodes generated during the urgency experiment.

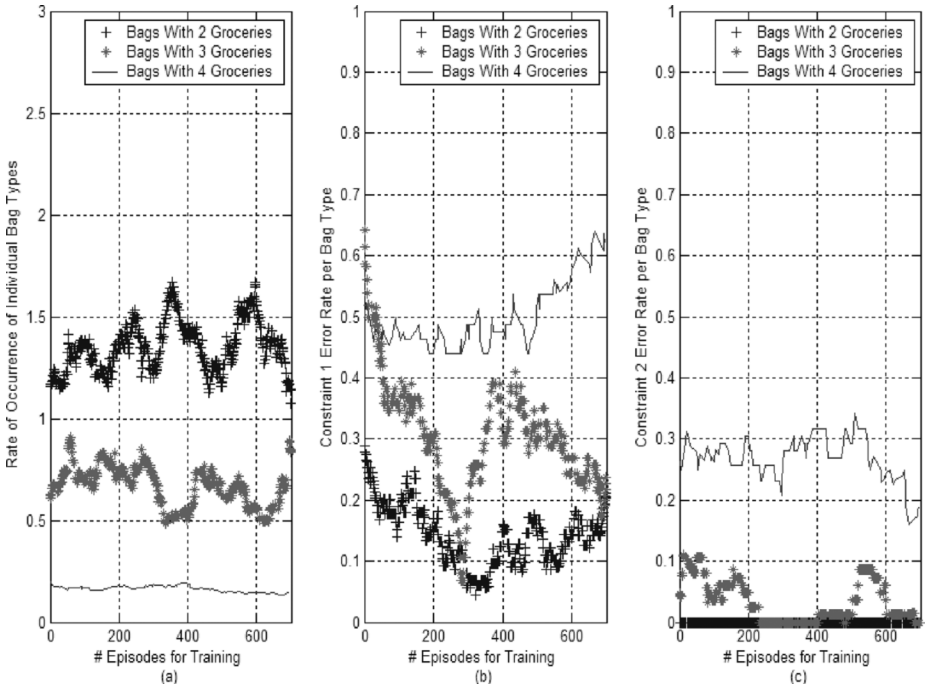


Fig. 12. Breakdown of the error rate and rate of occurrence for bags with different amounts of groceries.

3) Experiment 3: ISAC Integrated Experiments: The previous simulation-based experiments focused on running multiple trials with large amounts of experience and numerous conditions. Through these experiments, the system has shown the ability to appraise relevance and identify useful concepts for goal accomplishment. The system has also learned to appraise utility, and has demonstrated an initial ability to use internal rehearsal for self-evaluation.

The interaction of a humanoid robot with the real world critically depends on the robot's morphology and on its environment. Therefore, simulation is only one aspect of system validation. The current experiment aims to evaluate the integration of the cognitive control system described earlier and conceptually validated in the previous experiments, with the ISAC hardware system and peripheral software components. In particular, the objective is to integrate the designed control system with ISAC's perceptual agents and activator agents in order to complete the cognitive control process.

The operation of the individual components within the ISAC architecture and used within this experiment can be visualized

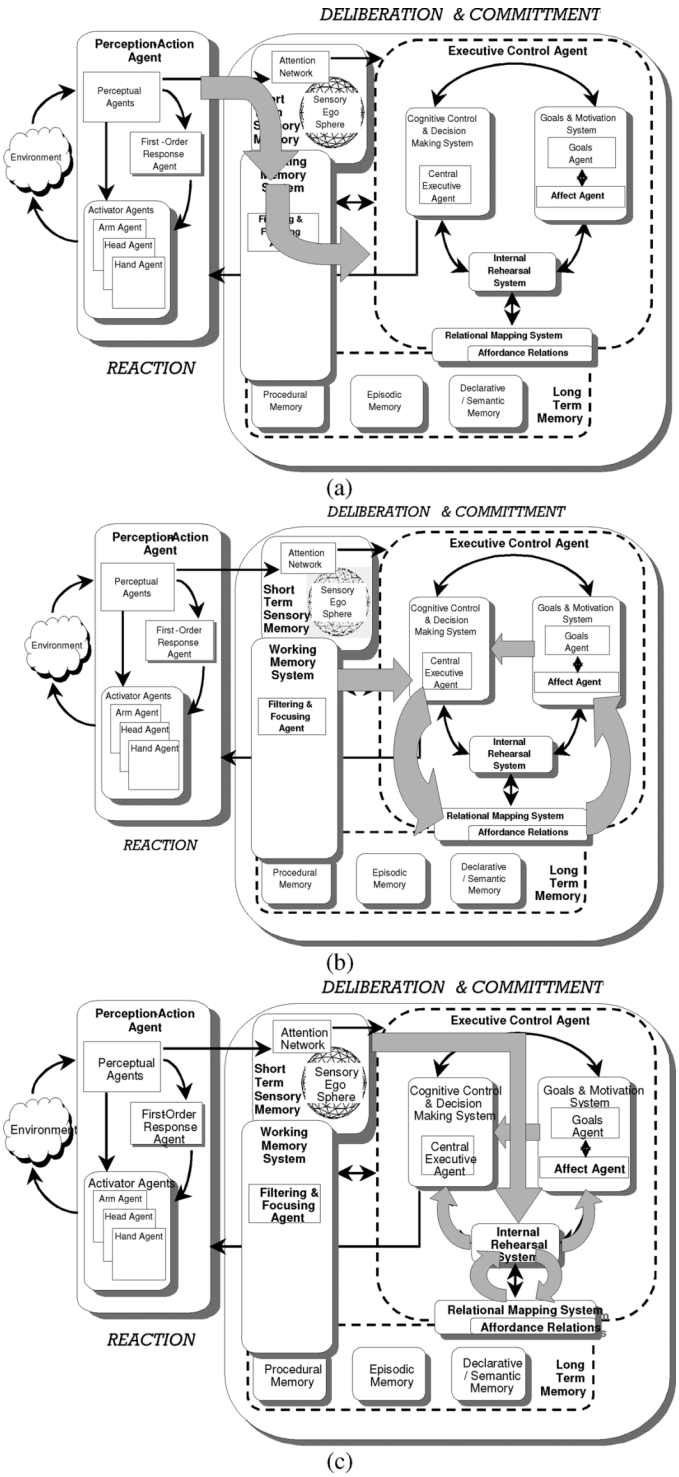


Fig. 13. Information paths through the ISAC architecture for (a) Relevance, (b) Utility, and (c) Urgency appraisals. (a) Relevance path. (b) Utility path. (c) Urgency path.

as a set of control paths through the cognitive architecture, as shown in Fig. 13. Within the ISAC architecture, appraising relevance occurs along the path shown in Fig. 13(a). This path should be considered a "preprocessing" step that filters and focuses incoming stimuli into a set of task-relevant categories that can then be passed to the deliberative control loop. Appraising utility occurs along the path shown in Fig. 13(b), and ends in the CEA which is in charge of comparing options and making the final decision. Urgency appraisals occur along the path shown in Fig. 13(c) and are used to inform both the CEA and the affect agent.

Integrating the designed system with the perceptual agents requires replacing the simulated perceptual input with input from the SES. Once the perceptual information has been received from the SES, it can be sent in two directions. The first direction involves sending that information to the WMS where it is filtered and used to create the necessary feature vectors that will later be used to access the relational maps. As described in Section III, it is in the WMS that the learned weights are applied for chunk formation.

The second direction involves sending the perceptual information directly to the IRS and goals and motivation system. Specifically, this enables the identification of interrupts by using the current state of the world to make prediction about future states (as described in Section III-D2). In both cases, only specific perceptual information is monitored. The IRS monitors grocery positions, and the goals and motivation system monitors external feedback.

The feature vectors from the WMS are used to retrieve internal utility appraisals and to facilitate planning within the CEA. This process involves accessing the relational maps and passing the retrieved information through the Affect Agent back to the CEA. Once this has been accomplished the CEA can continue to search through the decision space, or can return the current best policy.

Once the policy has been developed, a behavior must be selected and executed. The selection process requires choosing the behavior for the current state, but the execution process requires integration with the activator agents. This integration requires two steps: 1) parsing the desired behaviors into the properly sequenced atomic actions; and 2) sending the parsed action commands to the agents in charge of the physical hardware rather than to the simulated agents.

Behavior execution employs knowledge stored in procedural memory, specifically the pre-and post-conditions for selecting each behavior and the required control laws for performing each behavior.

The objective of this experiment was to evaluate how well ISAC could deploy the knowledge learned in simulation, and related to the appraisals for *relevance*, *utility*, and *urgency* to the grocery-bagging task. Because this experiment required the use of the same external critic that was used in the simulated experiments, all of the experience acquired in simulation was applied to this task as initial knowledge. However, because this experiment used the actual conveyor belt, it was necessary for ISAC to have an appreciation for *how long* it could deliberate before a decision must be made. This is a complex problem that was beyond the scope of the current research, therefore, the system was provided with a Bayesian probability model that could be used to predict the time between state transitions. For this experiment, this model was used to predict the amount of time remaining until groceries began falling off of the conveyor belt.

The conveyor belt was implemented using a modified treadmill in which the control box had been replaced a computer

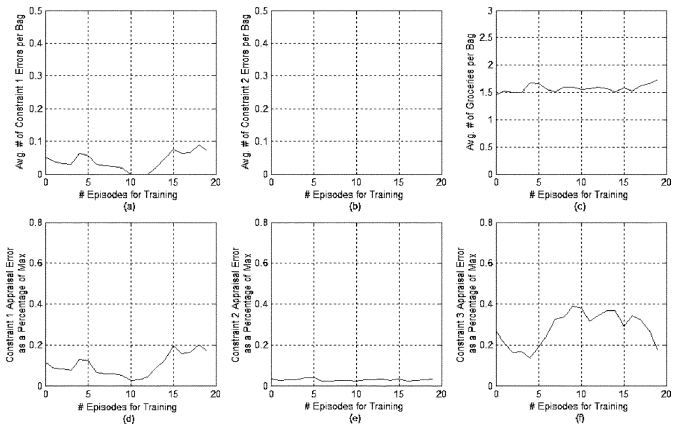


Fig. 14. Performance results for the ISAC experiment using knowledge derived from simulation and trained Bayesian networks.

control software. Using the computerized control system, along with a transformer-based power reduction method, the treadmill was able to operate at speeds as slow as 0.1–0.15 mph.

ISAC was tested and evaluated for 20 episodes using derived knowledge of each appraisal acquired and used during simulation as well as the interrupts generated from the Bayesian model. For each episode, a different trained knowledge set was selected from experiment 2 so that error rates were not a function of a single faulty component. Knowledge selection was performed sequentially from the final 20 training epochs. Experiment 2 was chosen as the source of the trained knowledge because it incorporated all of the appraisals investigated in the current research (with the exception of the interrupts provided by the Bayesian model). The results for these 20 episodes are presented in Fig. 14. The same smoothing window that was used in the simulation experiments (three training epochs) has also been used here.

The results shown here are demonstrate that the knowledge learned in simulation transfers well to the physical robot. The error rates shown in Fig. 14(a) and (b) are equal to, or better than, the error rates shown in Fig. 12(a) and (b) (for the final 20 training epochs). In addition, the appraisal error rates for constraints (1) and (2) are also better.

Therefore, the timing constraints introduced by performance on the real robot, and controlled via interrupts, does not drastically increase the error rate. However, it is apparent from Fig. 14(c) that forcing timing constraints on the system drastically reduces the amount of risk taken, i.e., the number of groceries per bag. This also partly explains the improvement in performance with respect to constraints (1) and (2). It also highlights the tradeoffs that must be met as robots are required to deliberate and make intelligent decisions in rapidly changing, dynamic environments. It has been argued throughout this paper that the ability to: 1) make state abstractions that reflect the goal-relevance of specific state components; 2) auto-associate situations with one another and with utility evaluations; and 3) appreciate the need to act in a timely manner are key ingredients to the success of true cognitive, robotic decision making. This paper presents a first step towards realizing such a goal.

V. Conclusion

The goal of this research was to investigate how multiple appraisals could be designed and integrated to enable decision making in a cognitive robot. The appraisals that were the focus of this work were *relevance*, *utility*, and *urgency* and correspond to the ideas of abstraction, learning reward, and trading between deliberation time and task performance. Learning these appraisals required that the robot cognitively process its own unique experience, which was stored in episodic, long-term memory. This research has found that a cognitive robot which possessed these tools could learn a task, by developing its own appreciations of what was relevant for that task, how utility should be associated with different situations, as well as how much urgency should be attached to each situation.

In robotics research there are a vast multitude of challenges that exist today and must eventually be solved before any autonomous robotic system will ever truly succeed outside of the laboratory. Within in the myriad challenges are the decision-making issues related to generalizability, real-time responsiveness, and adaptability. These issues require that the robot possess the ability to filter and focus on what is relevant in the current situation, develop adaptive, flexible representations of the environment, evaluate these representations quickly and accurately, and then adjust its responses based on internal measures of resource and time demand. In many cases, these problems are often simplified in order to focus on optimality and to make tasks more tractable; however, this simplification risks ignoring the interdependencies between each problem and the decision-making process as a whole. Robots will eventually need to develop their own conceptual notions of task relevance that do not require preset mappings. It is argued that the most natural way for such concepts to develop is through the robot's own experience and that the architectural approach and learning algorithms implemented be designed to flexibly identify concepts in as much of an unsupervised manner as possible.

Along with the identification of goal-relevant concepts, truly adaptable systems must be able to associate and assign utility to whatever representations have been identified as relevant. The current research enabled the system to develop goal-relevant concepts concerning percept classification, but stopped short of allowing the system to develop an understanding of which relationships within the environment were the most important for goal accomplishment (i.e., feature vector structure was fixed). Future work must address this issue so that the system can truly autonomously and dynamically represent both the concepts and the relationships within the current state in a manner that aligns with both the current goal and past experience. Within this paradigm utility assignment becomes more complex; however, it is believed that the relational structure used in the current work (along with the neuromorphic rationale behind it), points towards one possible solution.

But, even as a system learns to abstract the necessary information from the environment and develops appropriate utility associations with those abstractions, it is still critical that the system appreciate time critical nature of specific tasks and the value of increasing/decreasing deliberation in favor of rapid response. Ultimately, this final appraisal process will likely need to be based on a similar structure as the relevance and utility appraisals (in which the system bootstraps itself and learns what

are the time critical aspects of the current situation). Rather than requiring the current system to learn a complex state transition model, a Bayesian model was provided *a priori*, and the system used internal rehearsal to make predictions about the future and to reflect on situations from the past. Future work should perform more learning and evaluation for this component in order to tease apart the relationships between learning, performance, and deliberation time.

The research presented here is the first step towards a cognitive robot that can truly bootstrap itself to learn a complex task. The designed system has only begun to scratch the surface of appraisals that can be used to inform cognitive control; therefore, in addition to the points made previous paragraphs, there are many other possible directions for future work.

The appraisal for relevance could be expanded to include fuzzy notions of class membership in which individual percepts may belong to multiple classes to varying extents. This would enable the system to better track multiple conflicting goals where each goal dictates a different internal classification of an object.

Additional work could investigate whether the learned relationships stored within the Relational Mapping System could be extended to tasks for which they were not originally defined. This is an extremely complex problem that would also require highly flexible and reusable identification of goal-relevant classes. Consider, for example, an autonomous robotic vehicle driving along a crowded city street; it would certainly be advantageous for this system to understand that "large heavy objects do *not* go on top of smaller softer objects". The interesting question is whether this important relationship could be learned merely by bagging groceries and then noting the attribute and relationship similarity across tasks? Researchers have begun to ask such questions (for an excellent review see [92]); however, few employ the type of complex architectural approach that ultimately may be required.

The process of internal rehearsal to develop appreciations between situations and the behavioral and cognitive constraints that each affords is also an interesting problem that has recently seen some very intriguing results [61], [93]. As the environments in which robots are deployed become increasingly more complex, the need for offline preprocessing to mine a reusable set of easily deployable relations will become increasingly more important. This necessitates that a full investigation of internal rehearsal, including its biases and drawbacks, be performed.

Ultimately, as robot technology advances the control systems that enable these robots to adaptively function in complex environments will require increasingly more complex systems, components, and appraisals. The continuous and recursive interaction of these signals with the cognitive processes of the system will define its own set of control states in a manner that is fundamentally different than the way in which current robot control is often preset and preprogrammed.

REFERENCES

[1] R. E. Korf, "Real-time heuristic search," *Artif. Intell.*, vol. 42, pp. 189–211, 1990.

[2] A. G. Barto, S. J. Bradtke, and S. P. Singh, "Learning to act using real-time dynamic programming," *Artif. Intell.*, vol. 72, no. 1, pp. 81–138, 1995.

[3] A. Stentz, "The Focussed D* algorithm for real-time replanning," in *Proc. Int. Joint Conf. Robot. Autom.*, 1995, pp. 3310–3317.

[4] S. Koenig and M. Likhachev, "D* lite," in *Proc. Int. Conf. Autonom. Agents Multi-Agent Syst.*, Bologna, Italy, 2002, pp. 476–483.

[5] S. Zilberstein, "Using anytime algorithms in intelligent systems," in *Proc. Nat. Conf. Artif. Intell.*, Portland, OR, 1996, pp. 73–83.

[6] T. Dean, L. P. Kaelbling, J. Kirman, and A. Nicholson, "Planning with deadlines in stochastic domains," in *Proc. 11th Nat. Conf. Artif. Intell.*, Washington, D.C., 1993, pp. 574–579.

[7] S. Paquet, L. Tobin, and B. Chaib-draa, "An online POMDP algorithm for complex multiagent environments," in *Proc. 4th Int. Joint Conf. Autonom. Agents Multiagent Syst.*, 2005, pp. 970–977.

[8] M. A. Arbib and J.-M. Fellous, "Emotions: From brain to robot," *Trends Cogn. Sci.*, vol. 8, no. 12, 2004.

[9] E. T. Rolls, "The functions of the orbitofrontal cortex," *Brain Cogn.*, vol. 55, pp. 11–29, 2004.

[10] A. Damasio, *Descartes' Error: Emotion, Reason, and the Human Brain*. New York: Grosset/Putnam, 1994.

[11] H. Pfister and G. Böhm, "The multiplicity of emotions: A framework of emotional functions in decision making," *Judgment Decision Making*, vol. 3, no. 1, pp. 5–17, 2008.

[12] E. T. Rolls, *The Brain and Emotion*. Oxford, U.K.: Oxford Univ. Press, 1999.

[13] M. Zeelenberg and R. Pieters, "Feeling is for doing: A pragmatic approach to the study of emotions in economic behavior," in *Social Psychology and Economics*, D. De Cremer, M. Zeelenberg, and K. Murnighan, Eds. Mahwah, NJ: Erlbaum, 2006, pp. 117–137.

[14] N. H. Frijda, *The Emotions*. Cambridge, U.K.: Cambridge Univ. Press, 1986.

[15] K. R. Scherer, "Profiles of emotion-antecedent appraisal," *Cogn. Emotion*, vol. 11, no. 2, pp. 113–150, 1997.

[16] R. F. Baumeister, K. D. Vohs, C. N. DeWall, and L. Zhang, "How emotion shapes behavior: Feedback, anticipation, and reflection, rather than direct causation," *Soc. Personality Soc. Psychol.*, vol. 11, no. 2, pp. 167–203, 2007.

[17] G. Richter-Levin, "The amygdala, the hippocampus, and emotional modulation of memory," *Neuroscientist*, vol. 9, no. 7, pp. 1–9, 2004.

[18] P. Slovic, M. Finucane, E. Peters, and D. G. MacGregor, "The affect heuristic," in *Intuitive Judgment: Heuristics and Biases*, T. Gilovich, D. Griffin, and D. Kahneman, Eds. Cambridge, MA: Cambridge Univ. Press, 2003, pp. 1333–1352.

[19] A. Ortony and T. J. Turner, "What's basic about basic emotions?," *Psychol. Rev.*, vol. 97, pp. 315–331, 1990.

[20] M. I. Posner and C. R. Snyder, R. L. Solso, Ed., "Attention and cognitive control," in *Information Processing and Cognition: The Loyola Symposium*, Hillsdale, NJ, 1975.

[21] M. M. Botvinick, T. S. Braver, D. M. Barch, C. S. Carter, and J. D. Cohen, "Conflict monitoring and cognitive control," *Psychol. Rev.*, vol. 108, no. 3, pp. 624–652, 2001.

[22] N. H. Frijda, "Emotions in robots," in *Comparative Approaches to Cognitive Science*, H. L. Roitblat and J.-A. Meyer, Eds. Cambridge, MA: MIT Press, 1995.

[23] E. Peters, "The functions of affect in the construction of preferences," in *The Construction of Preference*, S. Lichtenstein and P. Slovic, Eds. Cambridge, U.K.: Cambridge Univ. Press, 2006, pp. 454–463.

[24] A. Ortony, D. A. Norman, and W. Revelle, "Affect and proto-affect in effective functioning," in *Who Needs Emotions? The Brain Meets the Robot*, J.-M. Fellous and M. Arbib, Eds. Oxford, U.K.: Oxford Univ. Press, 2004.

[25] N. Schwarz and G. L. Core, "How do I feel about it? The informative function of affective states," in *Affect, Cognition, and Social Behavior*, K. Fiedler and I. Forgas, Eds. Gottingen: Hogrefe, 1988, pp. 44–62.

[26] A. Bechara, H. Damasio, D. Tranel, and A. R. Damasio, "Deciding advantageously before knowing the advantageous strategy," *Science*, vol. 275, pp. 1293–1295, 1997.

[27] G. Loewenstein, "Out of control: Visceral influences on behavior," *Organizational Behav. Human Decision Process.*, vol. 65, no. 3, pp. 272–292, 1996.

[28] J. Panksepp, *Affective Neuroscience*. Oxford, U.K.: Oxford Univ. Press, 1998.

[29] A. Miyake and P. Shah, *Models of: Mechanisms of Active Maintenance and Executive Control*. Cambridge, MA: Cambridge Univ. Press, 1999.

[30] A. D. Baddeley, *Human Memory: Theory and Practice*, Rev. ed. Boston, MA: Allyn and Bacon, 1998.

[31] M. S. Gazzaniga, R. B. Irvy, and G. R. Mangun, *Cognitive Neuroscience: The Biology of the Mind*, 2nd ed. New York: Norton, 2002.

[32] A. Sloman, "Varieties of affect and the CogAff architecture schema," in *Proc. Symp. Emotion, Cogn., Affective Comput.*, York, U.K., 2001.

[33] A. Tversky and D. Kahneman, "Rational choice and the framing of decisions," *J. Bus.*, vol. 59, no. 4, pp. S251–S278, 1986.

[34] B. Mellers, A. Schwartz, and I. Ritov, "Emotion-based choice," *J. Exp. Psychol.: General*, vol. 128, no. 3, pp. 332–345, 1999.

[35] B. Mellers, "Choice and the relative pleasure of consequences," *Psych. Bul.*, vol. 126, no. 6, pp. 910–924, 2000.

[36] D. Kahneman and A. Tversky, "Prospect theory: An analysis of decision under risk," *Econometrica*, vol. 47, pp. 263–291, 1979.

[37] J. T. Cacioppo and G. G. Bernston, "The affect system: Architecture and operating characteristics," *Curr. Directions Psych. Sci.*, 1999.

[38] M. Cabanac, "Pleasure: The common currency," *J. Theor. Biol.*, vol. 155, pp. 173–200, 1992.

[39] R. P. Montague and G. S. Berns, "Neural economics and the biological substrate of valuation," *Neuron*, vol. 36, pp. 265–284, 2002.

[40] E. Tulving, *Elements of Episodic Memory*. Oxford, U.K.: Oxford Univ. Press, 1983.

[41] E. Tulving, "Episodic memory: From mind to brain," *Annu. Rev. Psychol.*, vol. 53, pp. 1–25, 2002.

[42] N. S. Clayton, D. P. Griffiths, N. J. Emery, and A. Dickenson, "Elements of episodic memory in animals," in *Episodic Memory: New Directions in Research*, A. Baddeley, M. Conway, and J. Aggleton, Eds. Oxford, U.K.: Oxford Univ. Press, 2002.

[43] R. G. M. Morris, "Episodic-like memory in animals: Psychological criteria, neural mechanisms and the value of episodic-like tasks to investigate animal models of neurodegenerative disease," in *Episodic Memory: New Directions in Research*, A. Baddeley, M. Conway, and J. Aggleton, Eds. Oxford, U.K.: Oxford Univ. Press, 2002.

[44] R. G. M. Morris, F. Schenk, F. Tweedie, and L. E. Jarrard, "Ibotenate lesions of hippocampus and/or subiculum: Dissociating components of allocentric spatial learning," *Euro. J. Neurosci.*, vol. 2, pp. 1016–1028, 1990.

[45] J. P. Aggleton and J. M. Pearce, "Neural systems underlying episodic memory: Insights from animal research," in *Episodic Memory: New Directions in Research*, A. Baddeley, M. Conway, and J. Aggleton, Eds. Oxford, U.K.: Oxford Univ. Press, 2002.

[46] N. Kapur, "Syndromes of retrograde amnesia: A conceptual and empirical synthesis," *Psych. Bul.*, vol. 125, pp. 800–825, 1999.

[47] K. Kawamura, R. A. Peters, II, R. Bodenheimer, N. Sarkar, J. Park, A. Spratley, and K. A. Hambuchen, "Multiagent-based cognitive robot architecture and its realization," *Int. J. Humanoid Robots*, vol. 1, no. 1, pp. 65–93, 2004.

[48] K. Kawamura, S. M. Gordon, P. Ratanaswasd, E. Erdemir, and J. Hall, "Implementation of cognitive control for a humanoid robot," *Int. J. Humanoid Robot.*, pp. 547–586, 2008.

[49] J. R. Anderson, *The Architecture of Cognition*. Cambridge, MA: Harvard Univ. Press, 1983.

[50] A. Newell, *Unified Theories of Cognition*. Cambridge, MA: Harvard Univ. Press, 1990.

[51] H. Shrobe, P. Winston, J. Tennenbaum, P. Shaftoe, S. Massaquoi, P. Robertson, B. Williams, I. Eslick, S. Rao, M. Coen, and R. Bobrow, CHIP: A Cognitive Architecture for Comprehensive Human Intelligence and Performance 2006 [Online]. Available: http://www.darpa.mil/ipto/programs/bica/phase1.htm

[52] A. Sloman, "Beyond shallow models of emotion," *Cogn. Process.*, vol. 2, no. 1, pp. 177–188, 2001.

[53] R. A. Peters, II, K. A. Hambuchen, K. Kawamura, and D. M. Wilkes, "The sensory ego-sphere as a short-term memory for humanoids," in *Proc. IEEE-RAS Int. Conf. Humanoid Robot.*, 2001, pp. 451–459.

[54] S. Pinker and J. Mehler, *Connections and Symbols*. Cambridge, MA: MIT Press, 1988.

[55] P. Ratanaswasd, "Implementation of Cognitive Control in a Humanoid Robot," Ph.D. dissertation, Vanderbilt Univ., Nashville, TN, 2007.

[56] S. M. Gordon and J. F. Hall, "System integration with working memory management for robotic behavior learning," in *Proc. Int. Conf. Develop. Learn.*, Bloomington, IN, 2006.

[57] J. L. Phillips and D. Noelle, "A biologically inspired working memory framework for robots," in *Proc. 27th Annu. Conf. Cogn. Sci. Soc.*, 2005, pp. 1750–1755.

[58] T. S. Braver and J. D. Cohen, "On the control of control: The role of dopamine in regulating prefrontal function and working memory," in *Control of Cognitive Processes*, S. Monsell and J. Driver, Eds. Cambridge, MA: MIT Press, 2000, vol. 18, ch. 31, pp. 713–737.

[59] P. Ratanaswasd, C. Garber, and A. Lauf, "Situation-based stimuli response in a humanoid robot," in *Proc. Int. Conf. Develop. Learn.*, Bloomington, IN, 2006.

[60] J. Hall, "Internal Rehearsal for a Cognitive Robot Using Collision Detection," M.Sc. thesis, Vanderbilt Univ., Nashville, TN, 2007.

[61] E. Erdemir, C. B. Frankel, K. Kawamura, S. M. Gordon, S. Thorton, and B. Ulutas, "Towards a cognitive robot that uses internal rehearsal to learn affordance relations," in *Proc. Int. Conf. Intell. Robots Syst.*, Nice, France, 2008.

[62] J. R. Anderson and C. Lebiere, *The Atomic Components of Thought*. New York: Erlbaum, 1998.

[63] D. H. Fisher, "Knowledge acquistion via incremental conceptual clustering," *Mach. Learn.*, vol. 2, pp. 139–172, 1987.

[64] M. A. Gluck and J. E. Corter, "Information, uncertainty, and the utility of categories," in *Proc. 7th Annu. Conf. Cogn. Sci. Soc.*, 1985, pp. 283–287.

[65] G. Biswas, J. Weinberg, and C. Li, "ITERATE: A conceptual clustering scheme for knowledge discovery in databases," in *Artificial Intelligence in the Petroleum Industry*, B. Braunschweig and R. Day, Eds. Paris, France: Editions Technip, 1995, pp. 111–139.

[66] J. P. Yoo, C. C. Pettey, and S. Yoo, "A hybrid conceptual clustering system," in *Proc. ACM 24th Annu. Conf. Comput. Sci.*, 1996, pp. 105–114.

[67] T. Kohonen, *Self-Organization and Associative Memory*. New York: Springer-Verlag, 1988.

[68] T. Kohonen and P. Somervuo, "Self-organizing maps of symbol strings," *Neurocomputing*, vol. 21, pp. 19–30, 1998.

[69] N. Burgess, S. Becker, J. A. King, and J. O'Keefe, "Memory for events and their spatial context: Models and experiments," in *Episodic Memory: New Directions in Research*, A. Baddeley, M. Conway, and J. Aggleton, Eds. Oxford, U.K.: Oxford Univ. Press, 2002.

[70] L. Nadel and M. Moscovitch, "Memory consolidation, retrograde amnesia and the hippocampal complex," *Curr. Opinions Neurobiol.*, vol. 7, pp. 217–227, 1997.

[71] M. B. Moser and E. I. Moser, "Functional differentiation in the hippocampus," *Hippocampus*, vol. 8, pp. 608–619, 1998.

[72] D. Marr, "Simple memory: A theory of archicortex," *Phil. Trans. Roy. Soc. London*, vol. B 262, pp. 23–81, 1971.

[73] J. Provost, B. J. Kuipers, and R. Miikkulainen, "Developing navigation behavior through self-organizing distinctive state abstraction," *Connect. Sci.*, vol. 18, no. 2, 2006.

[74] T. M. Martinez, H. Ritter, and K. J. Schulten, "Three-dimensional neural net for learning visuomotor coordination of a robot arm," *IEEE Trans. Neural Netw.*, vol. 1, pp. 131–136, Jan. 1990.

[75] S. Sehad and C. Touzet, "Reinforcement learning and neural reinforcement learning," in *Euro. Symp. Artif. Neural Netw.*, Brussels, Belgium, 1994.

[76] A. J. Smith, "Applications of the self-organizing map to reinforcement learning," *Neural Netw.*, vol. 15, pp. 1107–1124.

[77] A. Kawewong, Y. Honda, M. Tsuboyama, and O. Hasegawa, "A common-neural-pattern based reasoning for mobile robot cognitive mapping," in *INNS-NNN Symp.*, Auckland, The Netherlands, 2008.

[78] A. Sudo, M. Tsuboyama, C. Zhang, A. Sato, and O. Hasegawa, "Pattern-based reasoning system using self-incremental neural network for propositional logic," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2008.

[79] F. Shen and O. Hasegawa, "An incremental network for online unsupervised classification and topology learning," *Neural Netw.*, vol. 19, no. 1, pp. 90–106, 2005.

[80] T. Strosslin, D. Sheynikhovich, R. Chavariaga, and W. Gerstner, "Robust self-localization and navigation based on hippocampal place cells," *Neural Netw.*, vol. 18, no. 9, pp. 1125–1140, 2005.

[81] D. O. Hebb, *The Organization of Behavior*. New York: Wiley, 1949.

[82] B. Kuipers, J. Modayil, P. Beeson, M. Macmahon, and F. Savelli, "Local metrical and global topological maps in the hybrid spatial semantic hierarchy," in *ICRA*, 2004.

[83] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[84] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, 1966.

[85] G. Hesslow, "Conscious thought as simulation of behavior and perception," *Trends Cogn. Sci.*, vol. 6, pp. 242–247, 2002.

[86] D. A. Jirenhed, G. Hesslow, and T. Ziemke, "Exploring internal simulation of perception in mobile robots," *Lund Univ. Cogn. Stud.*, vol. 86, pp. 107–113, 2001.

[87] M. P. Shanahan, "A cognitive architecture that combines internal simulation with a global workspace," *Consciousness Cogn.*, vol. 15, pp. 433–449, 2006.

[88] P. Haddawy, "Focusing attention in anytime decision-theoretic planning," in *Proc. AAAI Symp. Planning Incomplete Info. Robot Problems*, Palo Alto, CA, 1996.

[89] T. L. Dean and M. Boddy, "An analysis of time-dependent planning," in *Proc. 7th Nat. Conf. Artif. Intell.*, St. Paul, MN, 1988, pp. 49–54.
[90] E. J. Horvitz, "Reasoning about beliefs and actions under computational resource constraints," in *Proc. Workshop Uncertainty Artif. Intell.*, Seattle, WA, 1987.
[91] Food Pyramid MIT [Online]. Available: http://web.mit.edu/athletics/sportsmedicine/wcrfoodpyr.html
[92] S. Thrun and L. Pratt, Eds., Learning to Learn Kluwer, 1998.
[93] E. Sahin, M. Cakmak, M. R. Dogar, E. Ugur, and G. Ucoluk, "To afford or not to afford: A new formalization of affordace-based robot control," *Adapt. Behav.*, vol. 15, no. 4, pp. 447–472, 2007.

Stephen M. Gordon received the B.Sc. degree in mechanical and electrical engineering from Tennessee Technological University, Cookeville, in 2003, and the M.Sc. degree in electrical engineering from Vanderbilt University, Nashville, TN, in 2005. He received the Ph.D. degree from Vanderbilt University in 2009.

He is currently an Electrical Engineer at DCS Corporation in Alexandria, VA. His research interests include association-based learning, artificial cognition, and human–robot teaming.

Kazuhiko Kawamura (F'00–LF'04) received the B.Eng. from Waseda University, Tokyo, Japan, the M.Sc. degree from the University of California, Berkeley, and Ph.D. degree in electrical engineering from the University of Michigan, Ann Arbor.

He is currently a Professor of Electrical and Computer Engineering and Engineering Management and the Director of the Center for Intelligent Systems (CIS), Vanderbilt University, Nashville, TN. He has published over 180 research papers, a book, several book chapters in the fields of intelligent systems, intelligent robotics, human–robot interfaces, cognitive robotics, and control. He directs research projects at the CIS in humanoid robots, cognitive robotics and architecture, working memory systems, and affordances learning.

Dr. Kawamura was Founding Chair of the Technical Committee on Service Robots for the IEEE Robotics and Automation Society. He is a member of the editorial board of *International Journal of Humanoid Robotics*, and *International Journal of Assistive Robotics and Systems*.

D. Mitchell Wilkes received the B.Sc. degree in electrical engineering from Florida Atlantic University, Boca Raton, and the M.Sc. and Ph.D. degrees in electrical engineering from Georgia Institute of Technology, Atlanta, in 1987.

He is currently an Associate Professor of Electrical and Computer Engineering at Vanderbilt University, Nashville, TN, and the Director of the Intelligent Robotics Laboratory at the Center for Intelligent Systems. He has more than 20 years experience in digital signal processing, image processing, and robot vision, and has published more than 100 publications on these subjects. His recent research interests have been in robot vision in autonomous mental development and the use of working memory models.