IEEE Access
The journal for rapid open access publishing

Received May 13, 2014, accepted June 9, 2014, published June 25, 2014, date of current version July 12, 2014.

Digital Object Identifier 10.1109/ACCESS.2014.2332333

Cognitive Control: Theory and Application

MEHDI FATEMI[1], (Member, IEEE), AND SIMON HAYKIN[2], (Life Fellow, IEEE)
[1]School of Computational Science and Engineering, McMaster University, Hamilton, ON L8S 4L8, Canada
[2]Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON L8S 4L8, Canada

Corresponding author: M. Fatemi (mehdi.fatemi@ieee.org)

ABSTRACT From an engineering point-of-view, cognitive control is inspired by the prefrontal cortex of the human brain; cognitive control may therefore be viewed as the overarching function of a cognitive dynamic system. In this paper, we describe a new way of thinking about cognitive control that embodies two basic components: learning and planning, both of which are based on two notions: 1) two-state model of the environment and the perceptor and 2) perception-action cycle, which is a distinctive characteristic of the cognitive dynamic system. Most importantly, it is shown that the cognitive control learning algorithm is a special form of Bellman's dynamic programming. Distinctive properties of the new algorithm include the following: 1) optimality of performance; 2) algorithmic convergence to optimal policy; and 3) linear law of complexity measured in terms of the number of actions taken by the cognitive controller on the environment. To validate these intrinsic properties of the algorithm, a computational experiment is presented, which involves a cognitive tracking radar that is known to closely mimic the visual brain. The experiment illustrates two different scenarios: 1) the impact of planning on learning curves of the new cognitive controller and 2) comparison of the learning curves of three different controllers, based on dynamic optimization, traditional Q-learning, and the new algorithm. The latter two algorithms are based on the two-state model, and they both involve the use of planning.

INDEX TERMS Cognitive dyanamic systems, cognitive control, dynamic programming, two-state model, entropic state, Shannon's entropy, explore/exploit tradeoff, learning, planning, Bayesian filtering.

I. INTRODUCTION

Cognition is a distinctive characteristic of the human brain, which distinguishes itself from all other mammalian species. It is therefore not surprising that when we speak of cognitive control, we naturally think of cognitive control in the brain [1]. Most importantly, cognitive control resides in the executive part of the brain, reciprocally coupled to its perceptual part via the working memory [2]. The net result of this three-fold combination is the perception-action cycle that embodies the environment, thereby constituting a closed-loop feedback system of a global kind.

In a point-of-view article published in the Proceedings of the IEEE on the integrative field of Cognitive Dynamic Systems viewed from an engineering perspective, it was first described in the literature [3]. This new way of thinking was motivated by two classic papers: ''Cognitive Radio: Brain-empowered Wireless Communications'' [4], and ''Cognitive Radar: A Way of the Future'' [5]. However, it was a few years later that the second author became aware of Fuster's basic principles of cognition, namely, perception-action cycle, memory, attention, and intelligence. It was that particular awareness that prompted the engineering need for bringing cognitive control into the specific formalism of cognitive dynamic systems.

During the past few years, cognitive control viewed from an engineering perspective, has featured in two journal papers, as summarized here:

1) In [6], a control-theoretic approach was described using *dynamic optimization*, representing a simplified version of Bellman's dynamic programming. It was in this paper that for the first time, we faced the imperfect state information problem, so called due to the fact that the controller does not have the provision to sense the environment in a direct manner. Although it is feasible to mitigate this problem algorithmically as formulated in [7], the incurred cost of computational complexity is so expensive that we had to limit the dynamic programming algorithm with no provision in looking into the future; thereby the name dynamic optimization.
2) The *two-state model*, proposed in [8], provides the most effective notion to bypass the imperfect state information problem; more will be said on this notion later in the paper. For the present, it suffices to say that practical validity of this new way of thinking about

cognitive control was demonstrated in [9] through the use of Q-learning that represents an approximate form of dynamic programming.

It was these two early contributions to cognitive control that set the stage for a novel cognitive controller presented in the current paper. Unlike the two previous procedures for implementing cognitive control, the new cognitive controller is optimal, in that it is well and truly a special case of Bellman's dynamic programming. Most importantly, unlike dynamic programming, the new cognitive controller follows a *linear law* of computational complexity measured in terms of actions taken on the environment. The other desirable attribute of this cognitive controller is the use of planning.

The rest of the paper is organized as follows:

- With cognitive control being the primary objective of the paper, Section II discusses two underpinnings of cognitive control, namely, learning and planning, each of which is based on two notions:
  1) The two-state model, which embodies target state of the environment and entropic state of the perceptor.
  2) The cyclic directed information flow, which follows from the global perception-action cycle: the first principle of cognition.
- Next, mathematical formalism of the learning process in cognitive control is presented in Section III, resulting in a state-free cognitive control learning algorithm, where computational complexity follows the linear law.
- Section IV goes one step further: the cognitive control learning algorithm is shown to be a special case of the celebrated Bellman's dynamic programming; hence, convergence and optimality of the new algorithm.
- Section V briefly discusses how to balance optimality of the learning process versus the convergence rate of the cognitive control learning algorithm, thereby setting the stage for both planning and the explore/exploit tradeoff, which are discussed in Sections VI and VII, respectively.
- At this point in the paper, we are ready to address structural composition of the cognitive controller in Section VIII.
- Then, Section IX validates an engineering application of the cognitive controller by presenting a computational experiment involving a cognitive tracking radar.
- Finally, Section X concludes the paper.

II. COGNITIVE CONTROL

From a cognitive neuroscience perspective, cognitive control plays a key role in the prefrontal cortex in the brain; most importantly, cognitive control involves two important processes: learning, and planning. And, so it is in a cognitive dynamic system, inspired by the brain. The learning process is discussed in Section III, followed by the planning process, which is discussed in Section VI. Both processes are dependant on the two-state model as well as the cyclic directed information flow, which are discussed in what follows.

A. THE TWO-STATE MODEL

As mentioned in the introduction, the two-state model is an essential element in deriving the cognitive control algorithm. By definition, the two-state model embodies two distinct states, one of which is called the *target state*, pertaining to a target of interest in the environment. The second one is called the *entropic state* of the perceptor,[1] the source of which is attributed to the unavoidable presence of uncertainties in the environment as well as imperfections in the perceptor itself.

Insofar as cognitive control is concerned, the two-state model is described in two steps as follows:

1) State-space model of the environment, which embodies the following pair of equations:

$$\begin{cases} \text{Process equation:} & \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k) + \mathbf{v}_k \\ \text{Measurement equation:} & \mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{w}_k \end{cases} \tag{1}$$

where $\mathbf{x}_k \in \mathbb{R}^n$, $\mathbf{z}_k \in \mathbb{R}^m$ are the state and measurement (observable) vectors at cycle k, respectively; $\mathbf{f}$ is a vector-valued transition function, and $\mathbf{h}$ is another vector-valued function that maps the target state-space to the measurement space[2]; $\mathbf{v}_k$ denotes an additive process noise that acts as the driving force, evolving state $\mathbf{x}_k$ at cycle k to the updated state $\mathbf{x}_{k+1}$ at cycle $k+1$; finally $\mathbf{w}_k$ is the additive measurement noise.

2) Entropic state model of the perceptor, which is formally defined by the following equation:

$$\text{Entropic-state equation:} \quad H_k = \phi(p(\mathbf{x}_k|\mathbf{z}_k)) \tag{2}$$

The H_k is the entropic state at cycle k in accordance with the state posterior $p(\mathbf{x}_k|\mathbf{z}_k)$ in the Bayesian sense, which is computed in the perceptor.[3] As such, H_k is the state of the perceptor, and ϕ is a quantitative measure such as Shannon's entropy.[4]

It is important to note here that, in general, Shannon's entropy could assume the value zero; however, in cognitive control, the entropic state H_k will always have a non-zero, positive value due to the fact that the environment always involves uncertainty and we can never reach perfect target-state reconstruction with 100% accuracy.

[1] The terms *cognitive perceptor* and *perceptor* are used interchangeably in the paper.

[2] In order to guarantee the existence and uniqueness of the solution to (1), both $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ are assumed to be Lipschitz continuous [10]; i.e., there exists $\lambda > 0$ such that $||\mathbf{f}(x_2) - \mathbf{f}(x_1)|| \leq \lambda ||x_2 - x_1||$, for all x_1 and x_2, with $||.||$ denoting the Euclidian norm and likewise for $\mathbf{h}(\cdot)$.

[3] To emphasize the cycles, in which the state and the measurement are taken, in this paper, we may also use the notation $H_{k|k}$, in accordance with the subscripts in the posterior, $p(\mathbf{x}_k|\mathbf{z}_k)$.

[4] Shannon's entropy for a random variable X, having the probability density function $p_X(x)$ in the sample space Ω, it is defined as [11]:

$$H = \int_{\Omega} p_X(x) \log \frac{1}{p_X(x)} dx$$

Correspondingly, Shannon's entropy of target state $\mathbf{x}_k$ with the posterior $p(\mathbf{x}_k|\mathbf{z}_k)$ is defined as:

$$H_k = \int_{\mathbb{R}^n} p(\mathbf{x}_k|\mathbf{z}_k) \log \frac{1}{p(\mathbf{x}_k|\mathbf{z}_k)} d\mathbf{x}_k.$$

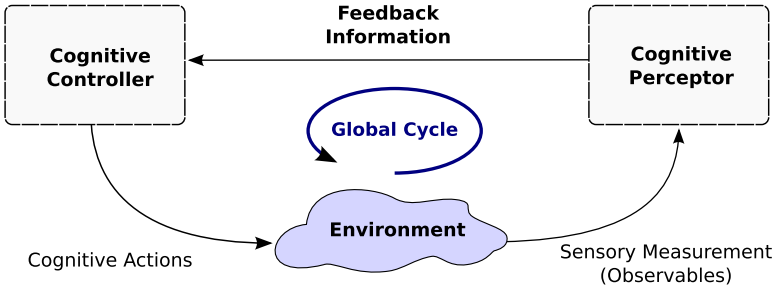

FIGURE 1. Block diagram of the global perception-action cycle in a cognitive dynamic system.

By definition [9], the function of cognitive control is defined as follows:

> To control the entropic state (i.e., state of the perceptor), such that the target's estimated state continues to be reliable across time.

Cognitive control therefore requires the entropic state, which is computed in the perceptor and then passed to the cognitive controller as *feedback information*.

B. CYCLIC DIRECTED INFORMATION FLOW

The *global perception-action cycle*, depicted in Fig. 1, plays a key role in a cognitive dynamic system; it is said to be global, in that it embodies the perceptor in the right-hand side of the figure, the cognitive controller in the left-hand side of the figure, and the surrounding environment, thereby constituting a closed-loop feedback system. In descriptive terms, the global perception-action cycle operates on the observables (measurements) of the environment, so as to separate *relevant information* about the environment from irrelevant information that is not needed. The lack of sufficient relevant information extracted from the observables is attributed to the unavoidable uncertainties in the environment as well as design imperfections in the perceptor. The entropic state introduced in sub-section *A* is indeed a measure of the lack of sufficient information. The entropic state supplies the *feedback information*, which is sent to the cognitive controller by the perceptor. With this feedback information at hand, the cognitive controller acts on the environment, producing a change in the observables. Correspondingly, this change affects the amount of relevant information about the environment, which is extracted from the new observables. A change is thereby produced in the feedback information and with it, a new action is taken on the environment by the cognitive controller in the next perception-action cycle. Continuing in this manner from one cycle of perception-action to the next, the cognitive dynamic system experiences a *cyclic directed information flow*, as illustrated in Fig. 1.

In addition to feedback information directed from the perceptor to the cognitive controller, there is also a feedforward information link from the cognitive controller to the perceptor. In other words, the perceptor and the cognitive controller are reciprocally coupled. This important link is illustrated in Fig. 3, and will be discussed later in Section VI.

III. FORMALISM OF THE LEARNING PROCESS IN COGNITIVE CONTROL

Previously in Section II, we introduced learning and planning as the two important processes in the execution of cognitive control. In actual fact, the aims of both learning and planning processes are to improve an entity called *cognitive policy*. By definition, cognitive policy is the probability distribution of cognitive actions at the perception-action cycle $k+1$, which includes the influence of action taken in cycle k. Let $\pi_k(c, c')$ denote the cognitive policy at cycle k, defined as follows:

$$\pi_k(c, c') = \mathbb{P}[c_{k+1} = c' | c_k = c]; \quad \text{with} \quad c, c' \in \mathcal{C},$$

where $\mathcal{C}$ is the cognitive action-space, c and c' are two cognitive actions, and $\mathbb{P}$ is a probability measure.

The cognitive policy should pertain to the long-term value of cognitive actions. In order to formalize a long-term value for each cognitive action, an immediate *reward* has to be defined. To this end, the *incremental deviation* in the entropic state from one cycle to the next, denoted by $\Delta_1 H_k$, is defined by

$$\Delta_1 H_k = H_{k-1} - H_k. \tag{3}$$

where H_{k-1} and H_k are the entropic states at the preceding and current cycles $k-1$ and k, respectively. Note that $\Delta_1 H_k$ could assume a positive or negative value, depending on conditional changes in the environment. The *entropic reward* for cognitive control at cycle k, denoted by r_k, is now defined as an arbitrary function of two entities: the entropic-state's absolute value, $|H_{k|k}|$, and the incremental deviation $\Delta_1 H_k$, as shown by:

$$r_k = g_k(|H_k|, \Delta_1 H_k) \tag{4}$$

where, g_k is an arbitrary scalar-valued operator. For example, the entropic reward in (4) may take the following form:

$$r_k = \frac{\Delta_1 H_k}{|H_k|} \tag{5}$$

Remark 1: Computation of the entropic reward r_k requires knowledge of the incremental deviation $\Delta_1 H$, defined in (4). To satisfy (4), it follows therefore that we need a short-term memory that accounts for the preceding entropic-state H_{k-1}.

As a result, after taking a cognitive action, a positive r_{k+1} indicates a decreasing deviation that can be considered as an immediate reward for the taken action. Conversely, a negative r_{k+1} demonstrates a cost against the selected action.

We may now define the following *value-to-go function* for the cognitive controller:

$$J(c) = \mathbb{E}^{\pi}[r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + \cdots \mid c_k = c] \quad (6)$$

where $\gamma \in [0, 1)$ denotes a *discount factor* that decreases the effect of future actions, and $\mathbb{E}$ denotes the expected value operator for which the expected value is calculated using the policy distribution π_k.

Lemma 1: $J(c)$ satisfies the following recursion:

$$J(c) = \mathcal{R}(c) + \gamma \Sigma_{c'} \pi_k(c, c') J(c') \quad (7)$$

where $\mathcal{R}(c) = \mathbb{E}^{\pi}[r_{k+1} | c_k = c]$ denotes the expected immediate reward at cycle $k + 1$ of the currently selected action c at cycle k.

Proof: Using the linear property of the expected value operator [12], we may expand (6) as follows:

$$\begin{aligned} J(c) &= \mathbb{E}^{\pi}[r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + \cdots \mid c_k = c] \\ &= \mathbb{E}^{\pi}[r_{k+1} | c_k = c] + \gamma \mathbb{E}^{\pi}[\Sigma_{j=0}^{\infty} \gamma^j r_{k+j+2} | c_k = c] \end{aligned}$$

In the second line of the equation, the first term is the expected immediate reward $\mathcal{R}(c)$. The second term lacks c_{k+1} in the condition to be the action-value of one-step future action. Therefore, using the *total probability theorem,*[5] we may write:

$$\begin{aligned} J(c) &= \mathcal{R}(c) + \gamma \mathbb{E}^{\pi}[\Sigma_{j=0}^{\infty} \gamma^j r_{k+j+2} | c_k = c] \\ &= \mathcal{R}(c) + \gamma \Sigma_{c'} \mathbb{P}[c_{k+1} = c' | c_k = c] \\ &\quad \times \mathbb{E}^{\pi}[\Sigma_{j=0}^{\infty} \gamma^j r_{k+j+2} | c_k = c, c_{k+1} = c'] \\ &= \mathcal{R}(c) + \gamma \Sigma_{c'} \pi_k(c, c') J(c') \end{aligned}$$

■

It is noteworthy that (7) has the flavor of Bellman's equation for dynamic programming, on which more will be said in the next section. In order to have a *recursive* algorithm, we may express the recursion in the following form:

$$J(c) \leftarrow \mathcal{R}(c) + \gamma \Sigma_{c'} \pi_k(c, c') J(c') \quad (8)$$

With recursion in mind and for the sake of flexibility, on every cycle of the recursion, (8) becomes more of practical value in an algorithmic sense by having $J(c)$ plus a weighted incremental update, as shown by

$$J(c) \leftarrow J(c) + \alpha[\mathcal{R}(c) + \gamma \Sigma_{c'} \pi_k(c, c') J(c') - J(c)] \quad (9)$$

where $\alpha > 0$ is a *learning parameter*. On the basis of the recursion described in (9), we may formulate Algorithm 1, which updates the value-to-go function from one cycle of perception-action to the next.

[5]For random variables X, Y and Z defined in Ω_X, Ω_Y, and Ω_Z, respectively, the total probability theorem [12] says:

$$\mathbb{E}[X|Y = y] = \sum_{z \in \Omega_Z} \mathbb{P}[Z = z|Y = y] \mathbb{E}[X|Y = y, Z = z].$$

Algorithm 1 A Value-to-go Updating Algorithm Under Lemma 1

1 **Varables:**
2 J := value-to-go function
3 γ := discount factor, $\gamma \in [0, 1)$
4 α := learning parameter, $\alpha > 0$
5 **Inputs:**
6 $\mathcal{R}(c)$:= expected reward of action c
7 π := learning policy
8 **Updating:**
9 **for** *all cognitive actions* $c \in \mathcal{C}$ **do**
10 $\quad J(c) \leftarrow J(c) + \alpha[\mathcal{R}(c) + \gamma \Sigma_{c' \in \mathcal{C}} \pi_k(c, c') J(c') - J(c)]$
11 **end**

From an implementation perspective, the term $\Sigma_{c' \in \mathcal{C}} \pi_k(c, c') J(c')$ in line 10 of Algorithm 1 may be substituted by $mean\{J(c)\}$, simply by considering $\pi_k(c, c')$ to be a uniform distribution here.[6] This method is called *off-policy* and is known also to be convergent to the optimal policy [15].

Hereafter, the recursive algorithm based on (9) is referred to as the *cognitive control learning algorithm*. This algorithm has been derived by exploiting the cyclic information flow that is a characteristic of the global perception-action cycle. With $mean\{J(c)\}$ substituted in line 10, examination of Algorithm 1 immediately reveals that this algorithm follows a linear law of computational complexity with respect to the number of actions taken by the cognitive controller, which is the cardinality of the cognitive action-space $\mathcal{C}$.

From a practical perspective, linearity of the algorithm by itself is not adequate. To be more precise, convergence as well as optimality of the algorithm would have to be justified theoretically. With this objective in mind, we propose to shift gear from the cognitive perspective and appeal to Bellman's dynamic programming, which is known to be both convergent and optimal [16], [17].

IV. COGNITIVE CONTROL LEARNING ALGORITHM VIEWED AS A SPECIAL CASE OF BELLMAN'S DYNAMIC PROGRAMMING

Bellman's celebrated dynamic programming algorithm [16], [17] was first described in the literature about fifty five years ago; yet it remains to occupy an important place in the study of optimal control. The optimality manifests itself in terms of maximizing a long-term value-to-go function; it is formally defined over time by means of immediate rewards. In its basic form, Bellman's dynamic programming deals with finite-horizon problems. However, from an analytic perspective, the preferred mathematical approach is to deal with

[6]With $mean\{J\}$ substituted in (9), the learning rule becomes similar to the traditional Q-learning algorithm [13], [14], yet it differs from it in two basic fronts: First, Q-learning uses $max\{J\}$ as an approximation, and second, (9) is calculated for all the cognitive actions, whereas in Q-learning, the update is only for the current state and action. In Section IX, we have chosen traditional Q-learning as a frame of reference for comparison in our computational experiment.

Algorithm 2 A Value-to-go Updating Algorithm for a Generic Dynamic Programming

Varables:
$\tilde{J}$:= value-to-go function
γ := discount factor, $\gamma \in [0, 1)$
α := learning parameter, $\alpha > 0$
Inputs:
$T^a_{ss'}$:= transition probability
$R^a_{ss'}$:= expected reward
$\tilde{\pi}$:= learning policy
Updating:
for *all states* $s \in \mathcal{S}$ **do**
  for *all actions* $a \in \mathcal{A}$ **do**
    $\tilde{J}(s,a) \leftarrow \tilde{J}(s,a) + \alpha[\sum_{s' \in \mathcal{S}} T^a_{ss'}[R^a_{ss'} + \gamma \sum_{a' \in \mathcal{A}} \tilde{\pi}_k(s,a')\,\tilde{J} - \tilde{J}(s',a')]$
  end
end

infinite-horizon problems, where the rewards are considered over an infinite number of cycles.

In dynamic programming, a system is defined by its set of states $\mathcal{S}$ and set of actions $\mathcal{A}$. On a cycle-by-cycle basis, the system has a transition from state $s \in \mathcal{S}$ at cycle k to $s' \in \mathcal{S}$ at cycle $k+1$ as a result of action $a \in \mathcal{A}$. This transition results in an immediate reward $r_{k+1} \in \mathbb{R}$. The state-action-based *value-to-go* function is then defined by the formula:

$$\tilde{J}(s,a) = \mathbb{E}^{\tilde{\pi}}[r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + \cdots | s_k = s, a_k = a],$$

for which, $\tilde{\pi}_k(s,a) = \mathbb{P}[a_{k+1} = a | s_k = s]$ is the state-based *policy* when the system is in state s; the tilde in $\tilde{\pi}_k(s,a)$ is intended to differentiate it from the policy $\pi(c,c')$ used in the previous section. As mentioned previously, $\mathbb{P}$ denotes a probability measure and $\mathbb{E}^{\tilde{\pi}}$ denotes the expected value operator with respect to the policy $\tilde{\pi}$. In Appendix A, it is shown that $\tilde{J}(s,a)$ obeys *Bellman's equation* for dynamic programming as follows:

$$\tilde{J}(s,a) = \sum_{s' \in \mathcal{S}} T^a_{ss'}[R^a_{ss'} + \gamma \sum_{a' \in \mathcal{A}} \tilde{\pi}_k(s,a')\,\tilde{J}(s',a')] \quad (10)$$

where the transition probability $T^a_{ss'}$ and the immediate expected reward $R^a_{ss'}$ are respectively defined by the following pair of equations:

$$\begin{cases} T^a_{ss'} = \mathbb{P}[s_{k+1} = s' | s_k = s,\ a_k = a], \\ R^a_{ss'} = \mathbb{E}^{\tilde{\pi}}[r_{k+1} | s_{k+1} = s', s_k = s, a_k = a] \end{cases} \quad (11)$$

The optimal value-to-go function, denoted by $\tilde{J}^*$, is obtained by maximizing the sum of all the terms in (10) with respect to action a. Unfortunately, the end result of this maximization is an exponential growth in computational complexity, known as the *curse of dimensionality* [17]. Nevertheless, the algorithm is known to be convergent as well as optimal [15]. Algorithm 2 describes a dynamic programming algorithm corresponding to (10). Inclusion of the two *nested for-loops* in Algorithm 2 (lines 10 and 11) is indeed the root of the curse of dimensionality problem.

The cognitive control learning algorithm, described in Section III, is indeed *state-free*. On the other hand, in light of the fact that Bellman's dynamic programming is *state-dependant*, the question to be addressed is:

> How do we make Bellman's dynamic programming to be on par with the cognitive control learning algorithm, such that both of them are state-free?

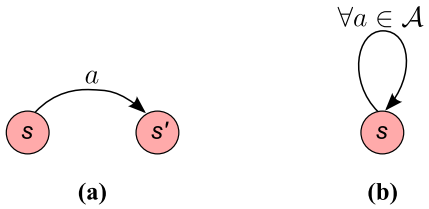

FIGURE 2. Graphical illustration of state transition in dynamic programming: (a) generic model, and (b) special case of Model 1.

To this end, consider the two models depicted graphically in Fig. 2. In a generic sense, part (*a*) of the figure illustrates the transition from state $s_k = s$ at time k to a new state $s_{k+1} = s'$ at time $k+1$ under the influence of action $a_k = a \in \mathcal{A}$, as it would be in Bellman's dynamic programming. On the other hand, part (*b*) of the figure depicts a "special" transition that involves a single state s and therefore a graphical representation of the following model:

Model 1:

- State-space contains only one state, that is $\mathcal{S} = \{s\}$,
- There exists a self-loop for s, including all the actions in the action space, i.e., $\mathbb{P}[s_{k+1} = s | s_k = s, a_k = a] = 1,\ \forall a \in \mathcal{A}$.

Model 1 is a valid model that lends itself to the application of Bellman's dynamic programming; moreover, the application of dynamic programming to Model 1 will not affect the properties of optimality and convergence, which are basic to dynamic programming [7]. The idea behind using Model 1 is to remove dependence of the dynamic programming algorithm on the states, as it would be in the cognitive control learning algorithm.

We next show that the following lemma holds:

Lemma 2: Dynamic programming of Model 1 is equivalent to the cognitive control learning algorithm.

Proof: It suffices to show that Bellman's equation for Model 1 is identical to the recursion equation in Lemma 1. Assume that the action-space of Model 1 is the same as the cognitive action-space in the previous section, that is, $\mathcal{A} = \mathcal{C}$. Because Model 1 has only one state, the outer summation in Bellman's equation (10) has only one term with the transition probability $T^a_{ss'}$ being one (due to the second property of Model 1). Additionally, the current and next states s_k and s_{k+1} in the condition of $R^a_{ss'}$ are always equal to s; hence, they add no additional information to $R^a_{ss'}$, and they are there-

fore redundant in $R^a_{ss'}$. We may thus formally write:

$$\begin{aligned} R^a_{ss'} &= \mathbb{E}[r_{k+1}|s_{k+1} = s, s_k = s, a_k = a] \\ &= \mathbb{E}[r_{k+1}|a_k = a] = \mathcal{R}(a) \end{aligned}$$

Similarly, since in Bellman's dynamic programming, current actions are independent of previous actions, we may express the corresponding policy:

$$\begin{aligned} \tilde{\pi}_k(s, a') &= \mathbb{P}[a_{k+1} = a'|s_k = s] \\ &= \mathbb{P}[a_{k+1} = a'|s_k = s, a_k = a] \\ &= \mathbb{P}[a_{k+1} = a'|a_k = a] \\ &= \pi_k(a, a') \end{aligned}$$

Substituting $R^a_{ss'}$ and $\tilde{\pi}_k$ in (10) will then prove the lemma. □

On the basis of Lemma 2, we may now state that the cognitive control learning algorithm is indeed a special case of of dynamic programming. Accordingly, the cognitive control learning algorithm inherits the basic properties of dynamic programming, namely, convergence and optimality. We may now conclude the section with the following statement:

> The cognitive control learning algorithm is not only linear, but also convergent to the optimal policy.

V. OPTIMALITY VS. CONVERGENCE-RATE IN ONLINE IMPLEMENTATION

Thus far, we have addressed optimality and convergence of the cognitive control learning algorithm. However, there are two other practical issues relating to the convergence rate of the learning process, which are described as follows:

1) To implement the *for-loop* in Algorithm 1, the expected immediate rewards should be known for *all* the actions in the action space $\mathcal{C}$. In reality, the immediate reward is available only for the currently selected action, which can replace its expected value. Hence, there would be $M = |\mathcal{C}|$ perception-action cycles required to collect information about all the actions. To overcome this first issue we propose to use *planning*, which is to be described in Section VI.
2) If we were to explore all the M cognitive actions in the action space $\mathcal{C}$, we would end up with a cognitive controller of poor performance in the exploration period. To overcome this second issue, we propose to use the ϵ-greedy strategy, which is to be discussed in Section VII.

Thus, through the use of planning and ϵ-greedy strategy, an efficient convergence rate with optimal performance for *on-line* applications is assured.

VI. FORMALISM OF THE PLANNING PROCESS IN COGNITIVE CONTROL

Planning is defined as the process of using *predicted* future rewards in order to improve our knowledge of the value-to-go function $J(c)$. Hence, the planning process plays a key role in speeding up the convergence rate of the cognitive controller. To this end, predicted values of entropic rewards are therefore required.

Referring to (1), pertaining to the state-space model of the environment, we may infer the following points:

1) If the probability density function of the noise terms in (1) is known, then the entropic state can be predicted one cycle into the future by using the Bayesian filtering framework of the perceptor.
2) The predicted entropic reward in the cognitive controller is then computed for the next hypothesized cycle.

In what follows next, this two-step procedure is illustrated in an example involving a Gaussian environment. This example will then be used in our computational experiment.

A. PREDICTING THE ENTROPIC REWARD IN A GAUSSIAN ENVIRONMENT

Consider a target with arbitrary dynamics in a Gaussian environment, with the state and measurement vectors denoted by $\mathbf{x}$ and $\mathbf{z}$, respectively. Since the noise terms in (1) are both Gaussian, the posterior $p(\mathbf{x}_k|\mathbf{z}_k)$ at each cycle is simply reduced to its mean value and covariance matrix. Let the entropic state be expressed by Shannon's entropy of the Gaussian posterior [11], namely:

$$H_{k|k} = \frac{1}{2}\log(\det\{(2\pi e)\mathbf{P}_{k|k}\}) \tag{12}$$

where det{.} denotes the determinant operator, and the matrix $\mathbf{P}_{k|k}$ is the covariance matrix of the posterior at cycle k, given the measurement also at cycle k. Since the logarithm is a monotonic function, (12) may be simplified to express the entropic state as follows:

$$H_{k|k} = \det\{\mathbf{P}_{k|k}\} \tag{13}$$

Based on this definition, a one-step *predicted* entropic state $H_{k+1|k} = \det(\mathbf{P}_{k+1|k})$ is found if we know the predicted covariance $\mathbf{P}_{k+1|k}$. To that end, the Kalman filter,[7] operating as the perceptor, provides $\mathbf{P}_{k+1|k}$ simply by knowing the system noise covariance matrix $\mathbf{Q}_k$ and measurement noise covariance matrix $\mathbf{R}_{k+1}$ [18]. Assuming that these two covariance matrices are given, we may compute the predicted entropic state of the perceptor. This process may be repeated to achieve further stages of prediction into the future, namely $H_{k+j|k}$, $j = 1, \ldots, l$, for l-step look-ahead horizon in time. Having all the $H_{k+j|k}$, predicted future rewards can then be calculated using equation (4), and we may therefore benefit from a planning process as well. □

The issue that emphasizes the need for planning is the time required for having actual rewards. In a cognitive dynamic system, we need to wait for one cycle to the next in order to access new rewards, and thereby proceed with the cognitive

[7] In this context, if the process and/or measurement dynamics are nonlinear, then the Kalman filter may be replaced by a nonlinear version such as the extended Kalman filter (EKF), unscented Kalman filter (UKF), or cubature Kalman filter (CKF); the CKF will be employed in our computational experiment in Section IX.

control learning algorithm, cycle by cycle. Unfortunately, Fig. 1 lacks a *feedforward* link from the controller to the perceptor. In such a scenario with an action library involving M possible actions (i.e., $|\mathcal{C}| = M$), there would have to be M global perception-action cycles for exploring the complete action library. If the time T seconds are taken for each global perception-action cycle, then there would have to be MT seconds needed to cover the entire action library. In order to mitigate such a long-windowed exploration phase, we propose to introduce a *feedforward* link, which connects the controller to the perceptor, as depicted in Fig. 3. The feedforward information is a hypothesized future action, which is to be selected for a planning stage. In so doing, a new so-called *internally composite cycle* [19] is therefore created, which completely bypasses the environment. Accordingly, the duration τ taken by such a cycle will be small compared to that of the global perception-action cycle, T. The practical benefit of introducing the internally composite cycle in Fig. 3 is the fact that the perceptor and the cognitive controller are now reciprocally coupled with each other, resulting in an exploration phase that is considerably shorter than in Fig. 1 by the factor T/τ.

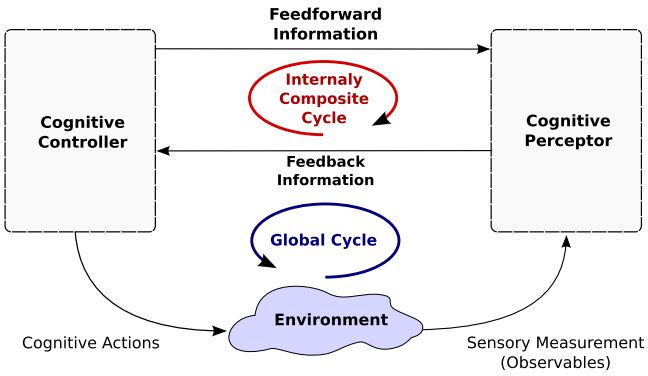

FIGURE 3. Block-diagram illustrating the combined presence of feedback information as well as feedforward information links.

Building on the scenario illustrated in Fig. 3, the two distinct but similar phases of learning and planning may now be implemented together, as follows:

1) **Learning**, which is based on actual values of the pair of entropic rewards at cycles k and $k-1$ as in (3) and (4), reproduced here for convenience of presentation:

$$g(|H_{k|k}|, \Delta_1 H), \quad \Delta_1 H = H_{k-1|k-1} - H_{k|k}$$

2) **Planning**, which is based on predicted values of the entropic reward; for example, at cycle $k+1$ and the actual reward at the current cycle k, we have the predicted reward defined by:

$$g(|H_{k+1|k}|, \Delta_2 H), \quad \Delta_2 H = H_{k|k} - H_{k+1|k}$$

Recall that learning is based on Lemma 1; equally, this lemma also applies to planning because conceptually speaking, both learning and planning perform the same required task. Note, however, learning is processed *only once* in each global perception-action cycle, which involves a single selected cognitive action; that is because learning is based on actual reward. On the other hand, in Fig. 3, planning is performed for any number of internally composite cycles and any number of hypothesized future actions in each of such cycles. Hence, specially in problems with very large number of possible actions (compared to the number of global perception-action cycles), a cognitive controller with learning only and therefore no planning may not perform on average much better than random action-selection. It follows therefore that planning is an essential requirement for policy convergence.

VII. EXPLORE/EXPLOIT TRADEOFF FOR COGNITIVE CONTROL

As discussed previously in Section V, in order to collect information about all the cognitive actions, the cognitive controller has to invest several global cycles, especially at the beginning of the experiment. During this phase, which is complete exploration of the cognitive action-space, the selected cognitive action in each cycle may result in completely poor performance. In particular, for problems with large set of cognitive actions, the resulting efficiency of the learning algorithm may remain unacceptable for a long period of time. Planning helps to mitigate this issue considerably, yet there is another auxiliary approach to smoothen the exploration process as much as possible, as discussed next.

In the cognitive control learning algorithm, and generally in dynamic programming, two different steps exist:

1) Updating the value-to-go function, J,
2) Updating the policy, π.

Note that updating J requires the knowledge of π, and vice versa. Hence, different approaches may be taken to update J and π, one after the other. When designing an algorithm to shape the cognitive policy on a cyclic basis, the following two extreme approaches may then be taken:

- In the first approach, the cognitive controller will explore the entire action-space uniformly *without regard to* the value-to-go function as guidance. This strategy is called *pure explore*.
- In direct contrast, at each cycle, the cognitive controller may select an action that maximizes the value-to-go function J. This strategy is called *pure exploit*.

These two pure strategies are both extreme and clearly in conflict with each other. In reality, a *mixed strategy* is therefore desirable; namely, it is most of the time optimal in terms of value-to-go maximization, while at the same time, the strategy also involves exploration of other actions.

A commonly used mixed strategy as a compromise between the two mentioned pure strategies is called ϵ-greedy strategy [20], as follows:

- With the probability of ϵ (e.g., 5%), the cognitive controller selects action randomly (pure explore),
- With the probability of $1-\epsilon$ (e.g., 95%), the cognitive controller selects action based on the maximum value

criterion (pure exploit). In this case, the action selection is completely aligned with the value-to-go function, hence the term *greedy*.

Furthermore, in cognitive control, the explore/exploit tradeoff may be performed separately in two stages:

1) For the cognitive policy, we use an ϵ-greedy strategy, in which all the cognitive actions have the chance of being selected at least with a small but nonzero probability ϵ; this means most of the time, the policy is greedy but not always.
2) In the planning phase, instead of selecting m (out of $M = |\mathcal{C}|$) "random" cognitive actions, which is complete exploration, we may select the m cognitive actions based on some prior knowledge. In such a case, the selection of m cognitive actions is driven by some selection prior probability distribution based on the policy.

Deployment of the explore/exploit tradeoff in cognitive control may be viewed as a facilitator of *attention* as one of the basic principles of cognition. Therefore, a cognitive controller empowered with the explore/exploit tradeoff tries to allocate computational resources in such a way that it remains focused on the knowledge gained about the environment, but the controller does not fall into local optimal actions and thereby miss the big picture.

VIII. STRUCTURAL COMPOSITION OF THE COGNITIVE CONTROLLER

Having the three constituents of perception, feedback information, and control, we may incorporate all three of them to propose a framework for cognitive control in a state-space modelled environment, as described next.

1) STRUCTURE

To incorporate planning and learning, a basic and simple structure is suggested by Sutton and Barto, called Dyna [15]. However, Dyna lacks state-space modelling and the inclusion of Bayesian perception with cyclic directed information flow, required in cognitive control. Thus, inspired by Dyna and having cognitive control in mind, we propose a new structure depicted in Fig. 4. This structure consists of two parts: (a) and (b) for ease of understanding. A global perception-action cycle is initiated in the perceptor at the right-hand side of Fig. 4-a, where Bayesian perception is performed. The feedback information to be controlled will then be the

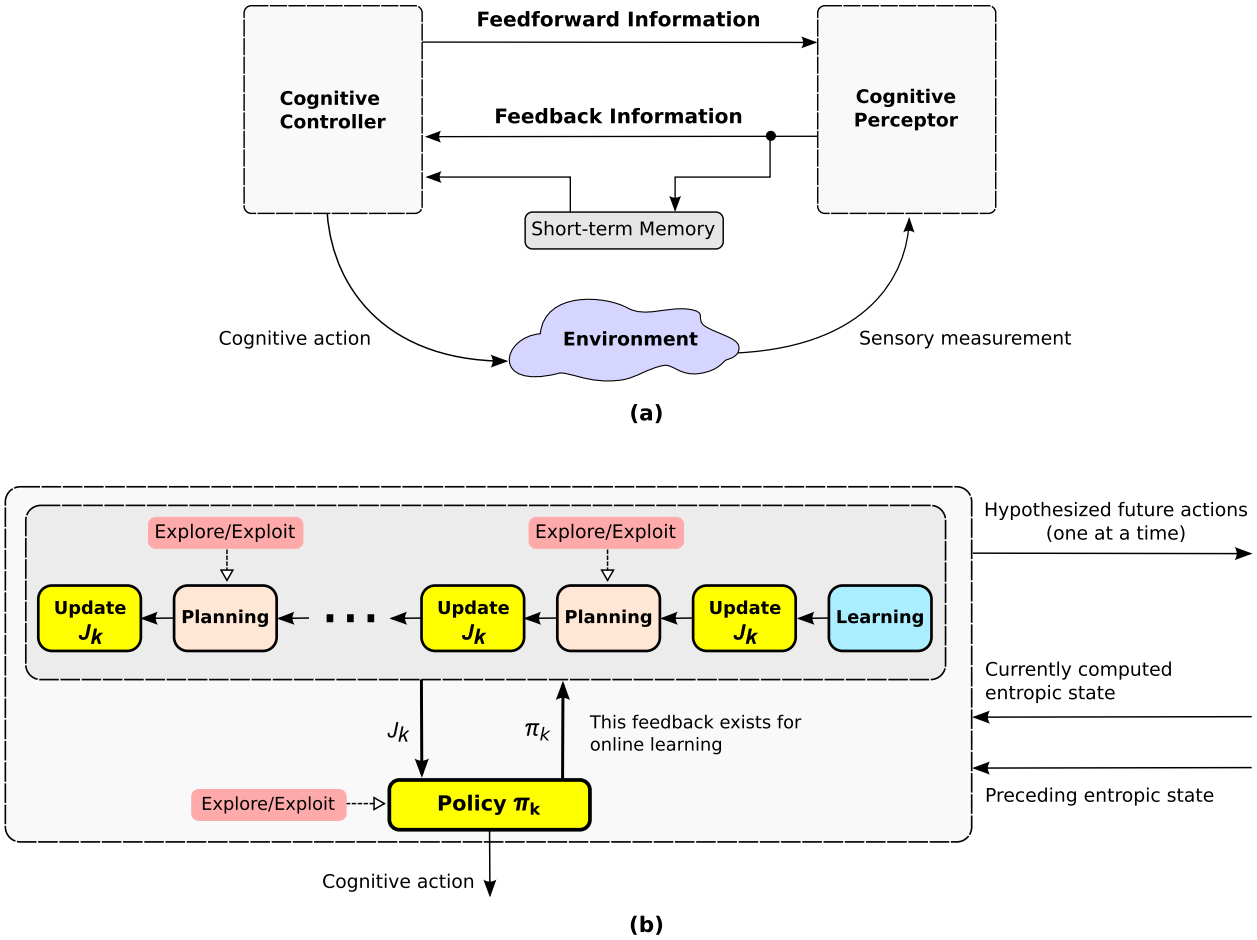

FIGURE 4. Block diagrammatic description of cognitive control: (a) cyclic directed information flow, and (b) illustration of algorithmic process in the cognitive controller.

entropic-state $H_{k|k}$, which is passed to the cognitive controller at the left-hand side of Fig. 4-a. At the same time, as explained in Remark 1, $H_{k|k}$ is also preserved in a short-term memory for the next cycle; it is short-term because in each cycle, the previous value will be overwritten. Then, in the cognitive controller, learning and planning are performed in the manner depicted in Fig. 4-b. It is noteworthy that in Fig. 4-b, the processes of learning and planning are performed in a serial manner.[8] To be specific, learning is performed, the result of which is an updated value-to-go function $J(c)$ for the preceding action. Then, we have a number of planning stages, each of which gives rise to a particular value-to-go update. In practice, the number of planning stages is dependant on the application of interest.

The explore/exploit tradeoff, explained in Section VII, is carried out in two different places: one place pertains to planning, and the other one pertains to policy-making. At the end, a cognitive action is selected from the derived policy and applied to the environment; and with it, the next global perception-action cycle is initiated. This framework is indeed the underlying structure for implementing cognitive control.

2) COMPLETE ALGORITHM

Algorithm 3 defines implementation of the cognitive controller, as described above, under Structure. "Updates" in lines 22 and 32 of the algorithm refer to the implementation of equation (7) for the currently selected cognitive action in learning and the hypothesized predictive action in planning, respectively. Also, line 30 in Algorithm 3 is implemented using the state-space model, as explained previously in Section II-A. Finally, the explore/exploit tradeoff is applied both in line 26 of the algorithm, where attention is deployed over some specific cognitive actions, namely the set $\mathcal{C}_1$, and the point where the cognitive policy π is shaped as ϵ-greedy in line 36 of the algorithm.

Algorithm 3 A Complete Algorithm to Implement Lemma 2, Which Embodies Both Learning and Planning

1 **Varables:**
2 $\mathcal{C}$:= set of all cognitive actions
3 $\mathcal{C}_1$:= set of selected cognitive actions for planning
4 J := value-to-go function
5 π := control policy
6 *memLearning* := short-term memory for learning
7 *memPlanning* := short-term memory for planning
8 c := selected cognitive action
9 r := computed reward
10 k := time step
11 **Initialization:**
12 $k \leftarrow 0$;
13 *memLearning* $\leftarrow H_0$;
14 $c \leftarrow$ a random cognitive action;
15 Apply c to the environment;
16 **repeat**
17 $k \leftarrow k + 1$;
18 $H_{k|k} \leftarrow$ *Input*(entropic_state) from Perceptor;
19 ---
20 **Learning:**
21 $r \leftarrow g_k(H_{k|k},\ (memLearning - H_{k|k}))$;
22 Update J;
23 *memLearning* $\leftarrow H_{k|k}$;
24 ---
25 **Planning:**
26 Select $\mathcal{C}_1 \subseteq \mathcal{C}$;
27 **for** *all cognitive actions* $c \in \mathcal{C}_1$ **do**
28 **for** *i=1 to num_prediction_steps* **do**
29 *memPlanning* $\leftarrow H_{k+i-1|k}$;
30 compute $H_{k+i|k}$ using c;
31 $r \leftarrow g_k(H_{k+i|k},\ (memPlanning - H_{k+i|k}))$;
32 Update J;
33 **end**
34 **end**
35 ---
36 Update π by J;
37 Select c based on π;
38 Apply c to the environment;
39 **until** *perception-action cycles are finished*;

IX. COMPUTATIONAL EXPERIMENT: COGNITIVE TRACKING RADAR

In what follows, we will demonstrate the information-processing power of the cognitive controller applied to a cognitive radar system, where the emphasis is on tracking performance. To be specific, we consider the tracking of a falling object is space, using a radar with 10 measurements per second, based on the benchmark example presented in [6] and [22]. Here, the cognitive actions "change" the radar transmitter's waveform parameters on a cycle-by-cycle basis in order to correspondingly control noise in the receiver via the environment.

The target state is $\mathbf{x} = [x_1,\ x_2,\ x_3]^T$, where x_1, x_2 and x_3 denote the altitude, velocity and ballistic coefficient, respectively; the ballistic coefficient depends on the target's mass, shape, cross-sectional area, and air density. The measurement vector $\mathbf{z} = [r,\ \dot{r}]^T$, consists of radar's range and range-rate. The extended state-space model is then defined by the following set of equations, involving both the state-space model as well as the entropic-state model:

$$\begin{cases} \mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{v}_k \\ \mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{w}_k(\theta_{k-1}) \\ H_k = \det\{\mathbf{P}_{k|k}\} \end{cases}$$

where the vector θ_{k-1} refers to the waveform transmitted at the previous cycle, $k-1$. For details of the functions $\mathbf{f}(.)$ and $\mathbf{h}(.)$, the reader is referred to [6]. Both noise terms, $\mathbf{v}_k$ and $\mathbf{w}_k$, are assumed to be white and zero-mean Gaussian. The system

[8] In the human brain, we have a similar scenario to that described in Fig. 4-b. Learning and planning use the same resources in the prefrontal cortex; where both learning and planning require organization in the time domain, with learning being current and planning being predictive [21].

noise has the following covariance matrix [6]:

$$\mathbf{Q} = \begin{bmatrix} q_1\frac{\delta^3}{3} & q_1\frac{\delta^2}{2} & 0 \\ q_1\frac{\delta^2}{2} & q_1\delta & 0 \\ 0 & 0 & q_2\delta \end{bmatrix}$$

where $q_1 = 0.01$, $q_2 = 0.01$, and $\delta = 1$. To model the measurement noise covariance matrix $\mathbf{R}$ as a function of waveform parameters, we use the model developed by Kershaw and Evans [23]. There, it is shown that for the transmit waveform, combining linear frequency modulation with Gaussian amplitude modulation, the measurement noise covariance matrix is defined by

$$\mathbf{R}(\theta_{k-1}) = \begin{bmatrix} \frac{c^2\lambda^2}{2\eta} & -\frac{c^2 b\lambda^2}{2\pi f_c \eta} \\ -\frac{c^2 b\lambda^2}{2\pi f_c \eta} & \frac{c^2}{(2\pi f_c)^2\eta}\left(\frac{1}{2\lambda^2} + 2b^2\lambda^2\right) \end{bmatrix}$$

where, the constants f_c and η are the carrier frequency and the received signal-to-noise ratio (SNR), respectively, and $c = 2.9979 \times 10^8\ m/s$ is the speed of light. Finally, $\theta = [\lambda,\ b]^T$ is the waveform-parameter vector, which is adjustable by the cognitive controller for matching the transmitted waveform to the environment as closely as possible.

For the Bayesian filter in the perceptor, a cubature Kalman filter (CKF) [24] has been used, which provides the estimated state covariance matrix $\mathbf{P}_{k|k}$ at cycle k. The entropic-state is then determined by $H_{k|k} = \det\{\mathbf{P}_{k|k}\}$, as in (13). For the entropic reward function, $r_k = |\log(|\Delta H|)|.\mathrm{sgn}(\Delta H)$, with $\Delta H = H_{k-1|k-1} - H_{k|k}$ has been used, where $\mathrm{sgn}(\cdot)$ denotes the standard signum function. This entropic reward also includes the right algebraic sign, which is required to guide the controller correctly. In the cognitive controller (i.e., radar transmitter), θ is changed at each perception-action cycle, which gives rise to 382 possible cognitive actions (382 is the number of different combinations for the transmit-waveform library). On each cycle, the cognitive action taken by the cognitive controller will affect the measurement noise covariance matrix. The time allowed for the experiment is five seconds for scenario 1 and 25 seconds for scenario 2; we therefore have to consider 50 and 250 perception-action cycles, respectively. All the simulations are performed over 1000 Monte Carlo runs to minimize the effect of randomness. It is also noteworthy that Algorithm 3, just like any other learning algorithm, is sensitive to the design parameters; as such, it is important to fine-tune the parameters for a given problem of interest.

In what follows, we describe two different experimental scenarios, one dealing with planning and the other comparing three different controllers.

Scenario 1: The Impact of Planning on Cognitive Control

In this experiment, we conduct three distinct case-studies:

1) *Absence of cognitive control*, that is, there is no feedback information form the receiver to the transmitter. In effect, in so far as the receiver is concerned, the CKF acts entirely on its own. As illustrated in Fig. 5, the green diamond-line at the top of the figure refers to the fixed-waveform radar, where there is no cognitive action at all. Nevertheless, because the CKF is an integral part of the perceptor, the learning curve decreases almost two orders of magnitude in the course of 50 cycles.
2) *Cognitive learning with no planning*, in which the recursive algorithm of (9) operates on its own in the cognitive controller. As explained in Section VI, since the total number of cycles is far less than the entire number of possible cognitive actions (50 vs. 382), the red bar-line in Fig. 5 is not that much better than the case study involving the fixed transmit waveform.
3) *Cognitive learning with planning*, for which we retain learning, but this time we also add planning. Implementing *explore*-only in the planning phase (see Section VII), this third case-study is repeated for three different choices of $|\mathcal{C}_1|$ (see line 26 of Algorithm 1): (i) only one random cognitive action (blue triangle-line), (ii) two random cognitive actions (black circle-line), and (iii) three random cognitive actions (cyan square-line). In the case of $|\mathcal{C}_1| = 1$, although one planning is still much less than the entire number of cognitive actions, it is enough to demonstrate a considerable improvement compared to the case with learning only. As for the other two cases, they both show more than four orders of magnitude improvement in the entropic-state reduction compared to the radar with fixed waveform.

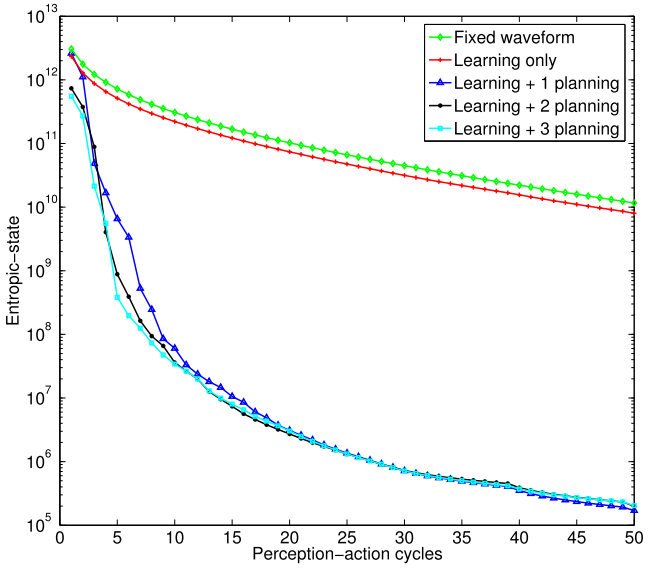

FIGURE 5. The impact of planning on cognitive control in Scenario 1.

Scenario 2: Comparison of Learning Curves of Three Different Cognitive Controllers

We refer back to the two different cognitive controller described in the Introduction, and compare them experimentally with the new cognitive controller described

in this paper. Thus, the study involves the following three different configurations with the same cubature Kalman filter for the cognitive radar receiver (perceptor):

1) *Cognitive controller using dynamic optimization:* This optimization algorithm is a simplified version of Bellman's dynamic programming [6], in that it does not account for the future impact of the currently selected action. The reason is that at each perception-action cycle, we must compute the change in the entropic state for *all* the actions in the action-space. Therefore, from a practical perspective, the computational throughput of dynamic optimization is extremely high. To account for this practical difficulty, the depth of horizon is reduced to unity; in other words, there is no provision in looking into the future. Even so, the computational throughput is too heavy and therefore of limited practical applications.[9] The learning curve of this first cognitive controller is depicted by the blue line in Fig. 6.

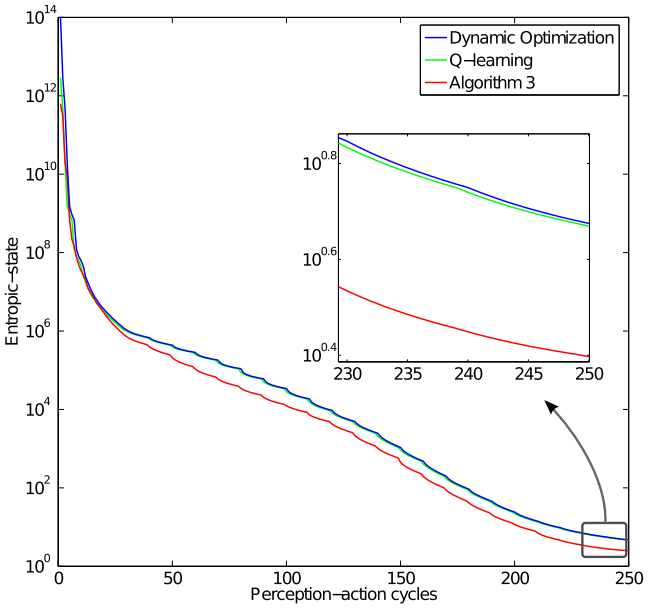

FIGURE 6. Comparative performance evaluation of three different cognitive control algorithms in Scenario 2.

2) *Cognitive controller, using Q-learning as well as planning:* To be specific, the learning process in the cognitive controller is performed using the traditional Q-learning algorithm [13], [14], which is made possible by exploiting the two-state model described in Section II-A. Moreover, the controller embodies planning with cardinality $|\mathcal{C}_1| = 3$. The learning curve of this second cognitive controller is depicted in Fig. 6 by the green line.
3) *The new cognitive controller,* which follows Algorithm 3. Specifically, it combines the use of the cognitive control learning algorithm as well as planning. The third and final learning curve (red line) in Fig. 6 accounts for the new cognitive controller. The planning part of this cognitive controller is also set to $|\mathcal{C}_1| = 3$. What is truly remarkable is the fact that the learning curve for the cognitive controller based on Algorithm 3 outperforms those of both Q-learning and dynamic optimization.

In this second scenario, the number of perception-action cycles has been set to 250 for the simple reason to allow for convergence to optimality.

It is important to note here that the numbers of floating-point operations (FLOPS) required for Algorithm 3 and Q-learning (both equipped with planning of $|\mathcal{C}_1| = 3$) are almost two orders of magnitude less than that of the method of dynamic optimization. Moreover, in the method of dynamic optimization, the computational load is unchangeable. In direct contrast, through the use of planning in Algorithm 3 (involving the selection of the planning set $\mathcal{C}_1$), we have complete design flexibility. Specifically, we may move anywhere from learning-only (least optimal, most computationally efficient), to any desirable number of planning stages that remains computationally efficient. This significant practical property of the new cognitive controller provides an information processing power to match the engineering design of the cognitive controller to any problem of interest, where levels of optimality and available computational resources are both specified.

X. CONCLUSION

A. COGNITIVE PROCESSING OF INFORMATION

The new cognitive controller in a cognitive dynamic system is inspired by the brain on two fundamental accounts: learning and planning:

A.1 The learning process in cognitive control is based on two basic ideas:

- The *entropic state* of the perceptor, which makes it possible to bypass the imperfect-state information problem that arises in the brain and other cognitive dynamic systems, such as the cognitive radar [19], [22].
- The *cyclic directed information flow*, which is attributed to the global perception-action cycle that defines the first principle of cognition [2], [25].

A.2 The planning process in cognitive control: This second process is inspired by the prefrontal cortex in the brain [19], [26]. Specifically, the cognitive controller in one side of the system is *reciprocally coupled* to the cognitive preceptor in the other side of the system. This reciprocal coupling, attributed to the combined use of *feedback information* from the perceptor to the controller as well as *feedforward information* from the controller to the perceptor, is the essence of the *shunt form* of perception-action cycle that completely bypasses the environment. In this paper we refer to this cycle as the internally composite cycle [19]; most importantly, it is this particular form of the perception-action

[9] Appendix B shows that the method of dynamic optimization may indeed be derived as a special case of the proposed algorithm in this paper.

cycle that accommodates the use of planning in the cognitive controller.

B. LINEARITY, CONVERGENCE, AND OPTIMALITY

These three intrinsic properties of the cognitive control learning algorithm are accounted for as follows:

- The linear law of computational complexity, measured in terms of actions taken on the environment, follows directly from the learning algorithm.
- Convergence and optimality of the learning algorithm follow from the proof that this algorithm is indeed a special case of the classic Bellman's dynamic programming.

C. ENGINEERING APPLICATION

Practical validity of the new cognitive controller has been demonstrated experimentally in a cognitive tracking radar benchmark example. Specifically, the new cognitive controller has been compared against two other different sub-optimal cognitive controllers: One controller involves dynamic optimization that is computationally expensive; the other controller involves the use of traditional Q-learning that is computationally tractable, but inefficient in performance.

ACKNOWLEDGMENT

The authors would like to thank the encouraging statements made by the two anonymous reviewers of the *IEEE Access* journal on novelty of the cognitive controller described herein for the first time.

APPENDIX A

In this appendix, we derive Bellman's equation (10). The proof is along the same line as the proof of Lemma 1.

Using the linear property of the expected value operator as well as the total probability theorem [12], we may expand $\tilde{J}$ as follows:

$$\begin{aligned}
\tilde{J}(s,a) &= \mathbb{E}^{\pi}[r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + \cdots \mid s_k = s, a_k = a] \\
&= \mathbb{E}^{\pi}[\Sigma_{j=0}^{\infty} \gamma^j r_{k+j+1} \mid s_k = s, a_k = a] \\
&= \sum_{s' \in \mathcal{S}} \mathbb{P}[s_{k+1} = s' | s_k = s, a_k = a] \\
&\quad \times \mathbb{E}^{\pi}[\Sigma_{j=0}^{\infty} \gamma^j r_{k+j+1} | s_k = s, a_k = a, s_{k+1} = s'] \\
&= \sum_{s' \in \mathcal{S}} T_{ss'}^{a} \\
&\quad \times \mathbb{E}^{\pi}[\Sigma_{j=0}^{\infty} \gamma^j r_{k+j+1} | s_k = s, a_k = a, s_{k+1} = s'] \\
&= \sum_{s' \in \mathcal{S}} T_{ss'}^{a} \\
&\quad \times \{\mathbb{E}^{\pi}[r_{k+1} | s_k = s, a_k = a, s_{k+1} = s'] + \gamma \mathbb{E}^{\pi} \\
&\quad [\Sigma_{j=0}^{\infty} \gamma^j r_{k+j+2} | s_k = s, a_k = a, s_{k+1} = s']\} \\
&= \sum_{s' \in \mathcal{S}} T_{ss'}^{a} \{R_{ss'}^{a} + \gamma \mathbb{E}^{\pi}[\Sigma_{j=0}^{\infty} \gamma^j r_{k+j+2} | s_k \\
&\qquad = s, a_k = a, s_{k+1} = s']\} \\
&= \sum_{s' \in \mathcal{S}} T_{ss'}^{a} \{R_{ss'}^{a} + \gamma \sum_{a' \in \mathcal{A}} \mathbb{P} \\
&\qquad [a_{k+1} = a' | s_k = s, a_k = a, s_{k+1} = s'] \\
&\qquad \times \mathbb{E}^{\pi}[\Sigma_{j=0}^{\infty} \gamma^j r_{k+j+2} | s_k = s, a_k = a, \\
&\qquad s_{k+1} = s', a_{k+1} = a']\} \\
&= \sum_{s' \in \mathcal{S}} T_{ss'}^{a} \{R_{ss'}^{a} + \gamma \sum_{a' \in \mathcal{A}} \tilde{\pi}_k(s,a') \tilde{J}(s',a')\}
\end{aligned}$$

□

APPENDIX B

In this appendix, we show that dynamic optimization used in the cognitive radar [6], it may be considered as a special case of the cognitive control learning algorithm, introduced in this paper.

At each perception-action cycle, the cost-function in the dynamic optimization algorithm is equivalent to the entropic state in this paper, and it is predicted for all the actions in the action library, using the Kershaw and Evans model [23]. The action that has the minimum cost is then selected as the optimal action.

Turning back to cognitive control, recall the learning update in Algorithm 1 (line 11):

$$J(c) \leftarrow J(c) + \alpha[\mathcal{R}(c) + \gamma \Sigma_{c' \in \mathcal{C}} \pi_k(c,c') J(c') - J(c)]$$

Substituting $\alpha = 1$ and $\gamma = 0$ yields the following:

$$J(c) \leftarrow \mathcal{R}(c) \tag{14}$$

which implies that under the assumptions of $\alpha = 1$ and $\gamma = 0$, the value-to-go function in cognitive control turns into the immediate reward.

Next, consider the substitution of $\mathcal{C}_1 \leftarrow \mathcal{C}$, in algorithm 3 (line 26). This case is equivalent to having *complete* planning at each perception-action cycle, which is clearly a possible choice.

Combining the learning and planning processes, discussed above, we have then exactly the same algorithm as dynamic optimization. To be more specific, in the case of having complete planning with unitary learning factor and no future inclusion (zero-discount), the new cognitive controller is reduced to dynamic optimization, and therefore the new cognitive controller embodies features that do not exist in dynamic optimization. Hence, it is not surprising that in Scenario 2 of Section IX, the learning curve for dynamic optimization deviates from that of the new cognitive controller.

REFERENCES

[1] E. K. Miller and J. D. Cohen, "An integrative theory of prefrontal cortex function," *Annu. Rev. Neurosci.*, vol. 24, no. 1, pp. 167–202, Mar. 2001.
[2] J. M. Fuster, *Cortex and Mind: Unifying Cognition*. London, U.K.: Oxford Univ. Press, 2003.

[3] S. Haykin, "Cognitive dynamic systems," *Proc. IEEE*, vol. 94, no. 11, pp. 1910–1911, Nov. 2006.
[4] S. Haykin, "Cognitive radio: Brain-empowered wireless communications," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 2, pp. 201–220, Feb. 2005.
[5] S. Haykin, "Cognitive radar: A way of the future," *IEEE Signal Process. Mag.*, vol. 23, no. 1, pp. 30–40, Jan. 2006.
[6] S. Haykin, A. Zia, Y. Xue, and I. Arasaratnam, "Control theoretic approach to tracking radar: First step towards cognition," *Digit. Signal Process.*, vol. 21, no. 5, pp. 576–585, Sep. 2011.
[7] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA, USA: Athena Scientific, 2005.
[8] S. Haykin, "Cognitive dynamic systems: Radar, control, and radio [Point of View]," *Proc. IEEE*, vol. 100, no. 7, pp. 2095–2103, Jul. 2012.
[9] S. Haykin, M. Fatemi, P. Setoodeh, and Y. Xue, "Cognitive control," *Proc. IEEE*, vol. 100, no. 12, pp. 3156–3169, Dec. 2012.
[10] W. Rudin, *Principles of Mathematical Analysis* (International Series in Pure and Applied Mathematics), 3rd ed. New York, NY, USA: McGraw-Hill, 1976.
[11] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. New York, NY, USA: Wiley, 2006.
[12] D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to Probability*, 2nd ed. Belmont, MA, USA: Athena Scientific, 2008.
[13] C. J. C. H. Watkins, "Learning form delayed rewards," Ph.D. dissertation, Dept Comput. Sci., Cambridge Univ., Cambridge, U.K., 1989.
[14] C. J. C. H. Watkins and P. Dayan, "*Q*-learning," *Machine Learn.*, vol. 8, nos. 3–4, pp. 279–292, May 1992.
[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1998.
[16] R. E. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton Univ. Press, 1957.
[17] R. E. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton, NJ, USA: Princeton Univ. Press, 1961.
[18] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation With Applications to Tracking and Navigation*. New York, NY, USA: Wiley, 2001.
[19] S. Haykin and J. M. Fuster, "On cognitive dynamic systems: Cognitive neuroscience and engineering learning from each other," *Proc. IEEE*, vol. 102, no. 4, pp. 608–628, Apr. 2014.
[20] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd ed. New York, NY, USA: Wiley, 2011.
[21] J. M. Fuster and S. Haykin, private communication, Apr. 2014.
[22] S. Haykin, Y. Xue, and P. Setoodeh, "Cognitive radar: Step toward bridging the gap between neuroscience and engineering," *Proc. IEEE*, vol. 100, no. 11, pp. 3102–3130, Nov. 2012.
[23] D. J. Kershaw and R. J. Evans, "Optimal waveform selection for tracking systems," *IEEE Trans. Inf. Theory*, vol. 40, no. 5, pp. 1536–1550, Sep. 1994.
[24] I. Arasaratnam and S. Haykin, "Cubature Kalman filters," *IEEE Trans. Autom. Control*, vol. 54, no. 6, pp. 1254–1269, Jun. 2009.
[25] S. Haykin, *Cognitive Dynamic Systems*. Cambridge, U.K.: Cambridge Univ. Press, Mar. 2012.
[26] J. M. Fuster, "The prefrontal cortex makes the brain a preadaptive system," *Proc. IEEE*, vol. 102, no. 4, pp. 417–426, Apr. 2014.

MEHDI FATEMI is currently pursuing the Ph.D. degree with the McMaster School of Computational Science and Engineering, Hamilton, ON, Canada. He received the master's (Hons.) degree in computational science interdisciplinary program from Memorial University, NL, Canada, and the B.Sc. degree in physics from the College of Sciences, Shiraz University, Shiraz, Iran. He has been a fellow with the School of Graduate Studies, Memorial University, which is the highest honorary SGS title and the only one reflecting on the transcript. He was a recipient of the Ontario Graduate Scholarship from the Government of Ontario, the Stelco Graduate Bursaries and McMaster Graduate Scholarship.

He has been with the National Research Council, Canada, for three years as a Graduate Student Researcher and then as a Research Engineer. For more than 12 years, he has extensive experience in computational science and engineering, which includes project consultation, technical training, high-end educational training, and tutorial conducting. He has authored and co-authored scholarly publications, including the seminal paper entitled *Cognitive Control*, which has been featured as a cover story for the centennial issue of Proceedings of the IEEE.

Mr. Fatemi is the Co-Founder and the Chair of the CSE/SHARCNET Seminar Series initiative, which has been a notably successful program. His research interests include high-performance and GPU computing, stochastic complex systems, probability theory, control theory, estimation theory, data science, and network science.

SIMON HAYKIN received the B.Sc. (first class honors), Ph.D., and D.Sc. degrees in electrical engineering from the University of Birmingham, Birmingham, U.K. He is a Distinguished University Professor in the Faculty of Engineering, McMaster University, Hamilton, ON, Canada. For much for the past 15 years, he has focused his entire research program on learning from the human brain and applying it to a new generation of cognitive dynamic systems, exemplified by cognitive radar, cognitive control, and cognitive radio. He is the author/coauthor of over 50 books on communication systems (analog and digital), adaptive filter theory, neural networks and learning machines, and cognitive dynamic systems.

Prof. Haykin is a fellow of the Royal Society of Canada. He is the recipient of the Honorary Doctor of Technical Sciences (ETH, Zurich, Switzerland), the Booker Gold Medal from URSI for his outstanding contributions to radar and wireless communications, as well as many other medals and awards.