Agent-Based Brain Modeling by Means of Hierarchical Cooperative Coevolution

Michail Maniadakis*,†
Foundation for Science and Technology—Hellas
University of Crete

Panos Trahanias*,**,†
Foundation for Science and Technology—Hellas
University of Crete

Abstract We address the development of brain-inspired models that will be embedded in robotic systems to support their cognitive abilities. We introduce a novel agent-based coevolutionary computational framework for modeling assemblies of brain areas. Specifically, self-organized agent structures are employed to represent brain areas. In order to support the design of agents, we introduce a hierarchical cooperative coevolutionary (HCCE) scheme that effectively specifies the structural details of autonomous, yet cooperating system components. The design process is facilitated by the capability of the HCCE-based design mechanism to investigate the performance of the model in lesion conditions. Interestingly, HCCE also provides a consistent mechanism to reconfigure (if necessary) the structure of agents, facilitating follow-up modeling efforts. Implemented models are embedded in a simulated robot to support its behavioral capabilities, also demonstrating the validity of the proposed computational framework.

Keywords Coevolution, brain modeling, robotics, working memory, lesion model, epigenesis

1 Introduction

The long-term vision of developing artificial organisms with advanced cognitive abilities has given new impetus to brain modeling studies. Since mammals constitute the category of biological organisms that exhibit the highest level of intelligence, they can be used as an excellent prototype for the development of machines with enhanced cognitive abilities. In this endeavor, environmental interaction is of the utmost importance, because it is difficult to investigate the mammalian brain without embedding the models into a body to interact with its environment. Robots can provide a useful means for assessing brain models. Therefore, biologically inspired robotic systems and brain science can support each other in developing efficient artificial brain models.

The cognitive capabilities of mammals are supported by their central nervous system (CNS). The CNS consists of several interconnected modules with different functionalities [9, 31]. Recently, many computational models have been implemented, trying to explain and reproduce the functionality of partial brain areas [2, 7, 12, 42, 45, 70]. Unfortunately, each of these models operates at a different level of description and explanation, based on different assumptions. In other words, they seem to
form a heterogeneous collection, such that computational differences among them render their integration very difficult [72]. As a result, it is currently very difficult to implement large-scale brainlike systems. A consistent computational framework is necessary to support both the design and the integration of partial brain models, facilitating the long-term process of modeling the mammalian CNS [27]. In the present work we address this particular task, introducing a novel computational framework for engineering brain models. By following the proposed approach, we aim at systematically developing brain-inspired systems that will furnish robots with advanced cognitive abilities.

Recently, we have introduced a coevolutionary method to implement partial brain models [42]. In summary, each brain area is modeled by an agent [15, 30], emphasizing the autonomy and the special features of the area. Agents are represented by neural networks that capture the basic anatomical principles of the mammalian CNS. Each agent is necessary to develop a functionality similar to the corresponding brain area, after a certain amount of robot-environment interaction [9, 68]. An evolutionary process specifies the detailed structure of the biologically inspired cognitive system [25, 58, 63]. Instead of adopting a unimodal evolutionary approach, we employ a cooperative coevolutionary method that effectively addresses the specialized structure of each agent [53].

In the present work, we propose a hierarchical extension of this approach, exploiting the inherent ability of coevolutionary methods to integrate system components successfully. We introduce a new hierarchical cooperative coevolutionary (HCCE) scheme, which allows the coevolution of a large number of species (populations), organized in successively larger groups. By assigning each agent (representing a brain area) to a species, we are capable of addressing both the specialized characteristics of the agent and the composite characteristics of the overall system. The combination of partial autonomy and cooperative performance in a single design method seems particularly appropriate for engineering brainlike systems. Both of them are provided by the proposed approach, as a direct consequence of combining the distributed modeling (specifically, agent-based modeling) with the distributed design methodology (specifically, the HCCE scheme).

Following recent trends in studying computational models in lesion conditions [1, 18, 44, 52], our method facilitates systematic modeling of brain lesion experiments. Specifically, lesions are simulated by deactivating one or more system components (neural agents). The coevolutionary design procedure investigates the pre- and post-lesion performance of the model, utilizing separate fitness functions for indicating the performance of the model when all components are present and also indicating its performance when some components are deactivated. Hence, following the proposed design method, biological lesion findings can be systematically replicated, enforcing the similarity of the implemented model to the brain prototype.

Unfortunately, the construction of large-scale brain models is difficult to accomplish by developing a very complicated system from scratch. An alternative approach could be based on implementing partial models of brain areas, which are gradually extended to more complex and more efficient ones. Along this line, the capability of redesigning the model is an important feature for a computational framework that succeeds in long-term design procedures. This is because initial design steps impose constraints on the computational structure, which may harm forthcoming modeling efforts. Therefore, it is necessary to have a consistent method that systematically refines partial structures, being able to guarantee the cooperation of the redesigned components (and potentially some completely new ones) with those that remain unchanged. Following this approach, existing cortical area models can be systematically reutilized in order to implement successively more complex brainlike systems.

Due to the distributed nature of both the model and the design mechanism, the computational framework proposed in the current work can effectively address individual system components. Therefore, it provides a consistent mechanism to combine partial models [39] and, whenever necessary, redesign them [41] in order to advance their functional characteristics. This feature makes the proposed engineering approach very effective in implementing large-scale brainlike systems.

It is noted that other approaches employing artificial neural network components to represent brain areas have also appeared in the literature [33, 35, 66]. However, they suffer with respect to scalability, because they are not supported by a (semi)automated design procedure that facilitates the
reusability of substructures (e.g., by means of evolution [23]). Thus, they cannot be easily employed as a general-purpose computational framework for engineering brain models.

Following research efforts that link cognitive capabilities of robots with brain science [61, 64], the implemented models are embedded in a simulated robot to furnish it with cognitive capabilities. The robotic platform supports interaction with the environment and the assessment of the models. Consequently, having evaluated system components at different stages of the design procedure, increased reliability is offered to the final model, and the long-term vision of developing artificial organisms with cognitive capabilities is facilitated.

The rest of the article is organized as follows. In Section 2 we formally present the agent structures representing brain areas and their connectivity. Then, we present the HCCE scheme, which is employed for designing brain models consisting of autonomous, yet cooperating agents. Experimental results of the proposed computational framework are presented in Section 4. In particular, we describe the incremental modeling of brain areas involved in working memory, and additionally we evaluate the effectiveness of HCCE, comparing it with enforced subpopulation coevolution [19] and with unimodal evolution. In the last section, we highlight the basic features of the proposed method and we suggest interesting research directions for future work.

2 Computational Model

Agents are deemed an appropriate theoretical tool for modeling complex, distributed systems. At the same time, the brain is often described as a group of cooperating specialists that achieve the overall cognitive function by splitting the task into smaller elements [56]. Thus, an agent-based approach seems suitable to support brain modeling efforts, mainly due to the distributed organization of the central nervous system. Agent technology facilitates the development of distributed systems consisting of cooperative and/or interactive parts, supporting their flexibility, autonomy, subjectivity, and situatedness in a specific environment [15, 48]. From a designer’s point of view, it supports problem decomposition, abstraction of partial models, and scalability toward global problem solution [30]. In the current work we take advantage of the above beneficial features, adopting an agent-based representation of brain areas.

We have implemented two different agent components for representing cortical areas and their connectivity. Briefly, a neural-network-based agent consisting of excitatory and inhibitory neurons is utilized to represent brain areas. This module is called the cortical agent, and serves as the main processing unit for the implemented models. In order to simulate the interaction of brain areas, cortical agents need to exchange information by means of brainlike pathways. A link agent is responsible for transferring neural activity from one cortical module to the other. Only the excitatory neurons of a cortical agent formulate long-distance axon projections implementing intercortical connectivity. In order to facilitate the uniformity of the proposed modeling methodology, sensory inputs are represented by special kinds of cortical agents without any processing power. They consist of virtual excitatory neurons, being capable only of formulating intercortical axon projections. Therefore, the same link structure can be employed for implementing both input-to-cortical-agent connectivity and cortical-agent-to-cortical-agent connectivity. Overall, the human designer can utilize an appropriate number of links and cortical agents to simulate any desired connectivity of brain areas.

We note that the computational structures presented below are not restrictive for the approach proposed in the current work, but rather serve as a guide to how the agent-based coevolutionary framework can support engineering of brain-inspired models. In future works, additional constraints can be integrated to increase the biological reliability of agents, or, alternatively, a completely new structure with emphasis on its biological characteristics can be used, to implement brain models with enhanced biological reliability.

1 Other approaches to brain representation also exist that are however outside the scope of this article.
2.1 Working Example: A Minimal Modeling Task

In order to better describe the proposed computational framework for implementing brainlike models, we introduce a simple working example that will serve our detailed presentation in Sections 2 and 3. Let us assume that we are interested in a very small part of the mammalian central nervous system consisting of only two cortical areas \( A \) and \( B \). Cortical area \( A \) receives sensory input from the environment and, after processing, projects its activation to cortical area \( B \), which serves as the output. Additionally, we assume that areas \( A \) and \( B \) have different roles in the composite cortical system, but they have to cooperate to accomplish a satisfactory joint functionality. This assumption is typical for the mammalian central nervous system (e.g., different brain areas serve visual or motor competencies that however effectively cooperate to achieve complex real-life behaviors).

Let us now assume that we want to implement a model of \( A \) and \( B \) interaction that will be employed in a robotic application. We represent the connectivity of cortical areas \( A \) and \( B \) by utilizing a combination of link and cortical agents. This is demonstrated in Figure 1. At a given time \( t \), link agent \( L1 \) transfers information from sensors to the cortical agent representing area \( A \). Then, a second link agent \( L2 \) projects the neural activation of \( A \) to another cortical agent representing the output area \( B \). Neural activation in \( B \) is directly applied to the actuators of the robot, guiding its movement. In the next time step, the robot interacts with the environment and some of its sensors are activated. Sensor activity is mapped to the sensory module, and the processing cycle is repeated.

The computational implementation of the components employed to design the abovementioned model are described below.

2.2 Input Module

As is mentioned above, a sensory input module is represented by a simplified cortical agent consisting only of virtual excitatory neurons. Each neuron of the module corresponds to one sensor of a particular sensory modality. These neurons lack processing power, and their output is directly set according to the activation of the corresponding sensor. Still, this kind of virtual excitatory neuron can have axon projections to cortical agents that perform information processing. Long-distance intercortical connectivity is implemented by link agents described in the next subsection.

2.3 Link Agents

Link agents aim at supporting connectivity among cortical modules. Using a link agent, any two cortical modules can be connected. The formulation of link agents is in line with the representation of cortical agents by rectangular planes with uniformly distributed excitatory and inhibitory neurons (see Section 2.4). Only excitatory neurons are used as outputs of the effluent cortical agent. Therefore, link agents are responsible for connecting the excitatory neurons of the projecting cortical agent to the receiving cortical agent. This is represented graphically in Figure 2, demonstrating how link agents \( L1 \) and \( L2 \) transfer information to cortical agents \( A \) and \( B \). We recall that sensory inputs are represented as a special case of cortical agents consisting of virtual excitatory neurons. Therefore, sensory inputs can be connected to normal cortical agents using link agents.

---

Figure 1. The hypothetical connectivity of agents for the working example serving the explanation of the proposed approach. Cortical agents are represented with blocks, while link agents are represented with double arrows.
The axons of projecting neurons are completely described by their \((x, y)\) coordinates on the receiving plane. Cortical planes have a predefined dimension, implying that axons projecting beyond the borders of the plane are not activated. As a result, it is not necessary that all excitatory neurons project their outputs on the receiving plane. This is illustrated graphically in Figure 2, where active projections are represented by \(\times\) on their termination. Projections outside the cortical plane are shown without a terminal point, and thus they are deemed deactivated. When the locations of axons on the receiving cortical plane are defined, synapses between axon terminals and the excitatory or inhibitory neurons can be specified. Synapse specification is based on the structure of the receiving cortical plane. This process is described in detail in Section 2.4.

The flexibility of link agents, projecting their axons on any desired position of the receiving cortical plane, is in contrast to our previous model that employs prespecified axon projection coordinates [38, 42]. Following the flexible projection approach, more power is provided to the proposed modeling approach for performing incremental design steps, supporting the reusability of the implemented models.

2.4 Cortical Agent

Each cortical agent consists of a predefined population of excitatory and inhibitory neurons located on a boundary-limited cortical plane (see Figure 2). The numbers of excitatory and inhibitory neurons are specified at design time by humans. Both sets of neurons are uniformly distributed, forming an excitatory and an inhibitory neural grid on the cortical plane. The axon terminals coming from projecting links are also located on the same plane (Figure 2). One-way synapses are formed among axons, excitatory neurons, and inhibitory neurons to support information processing. Synapse specification is based on the postsynaptic neuron as proposed in [58]. Overall, six synapse types can be specified, namely \(ae\) (axon to excitatory), \(ee\) (excitatory to excitatory), \(ie\) (inhibitory to excitatory), \(ai\) (axon to inhibitory), \(ei\) (excitatory to inhibitory), and \(ii\) (inhibitory to inhibitory).

Synapses are formulated according to the general rule of locality [55], which is simulated here by utilizing circular neighborhoods. All excitatory neurons share common neighborhood measures, that is, radii \(n_ao\) for specifying their connectivity with axons, \(n_eo\) for specifying their connectivity with the other excitatory neurons, and \(n_io\) for specifying their connectivity with inhibitory neurons. This process is demonstrated in Figure 3, explaining further the example of Figure 2. In particular, only the synapse definition for cortical agent \(B\) is shown. The first row of Figure 3a depicts the definition of axon-to-excitatory-neuron synapses. A circular neighborhood is centered on an excitatory neuron, and the axon projections located within the circle form a synapse with the neuron. Then the
circular neighborhood is centered on the next excitatory neuron, specifying its synapses with axon projections. This process is repeated for all excitatory neurons of the cortical agent. A similar process is followed for specifying excitatory-to-excitatory neural connectivity (row 2 of Figure 3a) and inhibitory-to-excitatory connectivity (row 3 of Figure 3a). The total number of synapses transferring information to excitatory neurons is depicted in Figure 3b. In a similar way, the connectivity of inhibitory neurons is based on neighborhood measures $n_{ax}$, $n_{ex}$, and $n_{in}$ specifying their connectivity with axons, excitatory

![Diagram](image)

Figure 3. The definition of synapses for cortical agent $B$ of Figure 2. Different neighborhood measures are used for each type of synapses. Part (a) demonstrates the definition of synapses towards excitatory neurons. Part (b) presents the total number of synapses toward excitatory neurons. Part (c) demonstrates the definition of synapses toward inhibitory neurons. Part (d) presents the total number of synapses toward inhibitory neurons. Finally, part (e) presents the overall internal connectivity in cortical agent $B$. 
neurons, and inhibitory neurons. The process of synapse specification for inhibitory neurons is depicted in Figure 3e and 3d. We note that in the current example there are no synapses connecting two inhibitory neurons, because they are not located in the neighborhoods of one another (see row 3 of Figure 3e). The total number of synapses in cortical agent B is shown in Figure 3e. Overall, six neighborhood values are necessary to specify the internal connectivity of a cortical agent.

Both excitatory and inhibitory neurons follow the Wilson-Cowan model with sigmoid activation, similar to [69]. Let $x_p$ represent the firing rate of a neuron of type $p \in \{e, i\}$ (either excitatory or inhibitory). Then, following the incoming neural activity, $x_p$ is updated according to the equation

$$\mu \Delta x_p = -x_p + S(W_e x + W_i y) - W_i y$$

where $\mu$ presents the membrane time constant, $W_{ep} \in [0, 1]$ are the weights of synapses with axons, $W_{ip} \in [0, 1]$ are the weights of synapses with excitatory neurons, and $W_{ep} \in [0, 1]$ are the weights of synapses with inhibitory neurons. Additionally, $S(y) = 1/(1 + e^{\alpha (y - \beta)})$ is the nonlinear sigmoid function, where $\beta$ and $\alpha$ stand for the threshold and the slope, respectively. All excitatory neurons of a cortical plane share common parameters $\mu$, $\alpha$, and $\beta$. The same is also true for inhibitory neurons, using parameters $\mu_i$, $\alpha_i$, and $\beta_i$.

The weights of synapses are not static, but are adjusted at run time, according to the experiences of the robot [48]. This is similar to epigenetic learning, which makes an important contribution to the performance of the mammalian brain [68]. Specifically, all six types of synapses (both with an inhibitory and an excitatory effect [57]) are assigned a Hebbian-like biologically plausible learning rule, similar to [14], enforcing experience-based subjective learning of robots.

We have implemented a pool of 10 Hebbian-like rules that can be appropriately combined to produce a wide range of functionalities. Thus, adequate flexibility is offered to each component of the model for developing the desired behavior. The rules have been selected based on their simplicity and their previous application in a variety of tasks [6, 13, 22, 32, 49, 50, 62]. Still, the architecture of cortical agents is open and amenable to other learning rules with desirable characteristics with regard to either model performance or biological plausibility.

Learning rules are encoded by unique identification numbers (ids) in the range $\{1, \ldots, 10\}$, also facilitating their assignment to synapse types. Assuming that there is a synapse with strength $w_{ab}$ from neuron $a$ with activation $x_a$ to neuron $b$ with activation $x_b$, then the employed learning rules are described below:

1. Differential decorrelation [6]: $\Delta w_{ab} = -\Delta x_a x_b$, where $\Delta$ is approximated by its discrete-time counterpart $\Delta x(t) = x(t) - x(t - 1)$.

2. Differential correlation [6]: $\Delta w_{ab} = \Delta x_a x_b$, where $\Delta$ is as above.

3. Postsynaptic [13]: $\Delta w_{ab} = w_{ab}(x_a - 1.0)x_b + (1.0 - w_{ab})x_a x_b$.

4. Presynaptic [13]: $\Delta w_{ab} = w_{ab}(x_b - 1.0)x_a + (1.0 - w_{ab})x_a x_b$.

5. Covariance [13]:

$$\Delta w_{ab} = \begin{cases} (1 - w_{ab})t & \text{if } t > 0 \\ w_{ab}t & \text{otherwise} \end{cases}$$

where $t = \tanh(2 - 4|x_a - x_b|)$.

2. Epigenesis, here, includes all learning processes during a lifetime.
6. Connectedness [22]: \( \Delta w_{ab} = 1 - w_{ab} \).

7. Kohonen [32]: \( \Delta w_{ab} = x_a - w_{ab} \).

8. Principal-component analyzer [49]: \( \Delta w_{ab} = \frac{1}{\lambda_a} (x_a - \lambda_a w_{ab}) \).

9. Anti-Hebbian I [50]: \( \Delta w_{ab} = k - x_a x_b, k > 0 \) a small forgetting factor to avoid vanishing.

10. Anti-Hebbian II [62]: \( \Delta w_{ab} = k - 2x_a x_b / (x_a^2 + 1) \), where \( k \) is as above.

Each synapse is assigned a learning rule that adjusts its synaptic weight at run time, highlighting subjective understanding of the organism about the world. Six rules are necessary to specify the internal learning dynamics of a cortical agent. In particular, rule \( r_{ax} \) adjusts axon to excitatory neuron synapses, rule \( r_{ea} \) adjusts excitatory to excitatory neuron synapses, and rule \( r_{ae} \) adjusts inhibitory to excitatory neuron synapses. In a similar way, synapses toward inhibitory postsynaptic neurons are adjusted according to rules \( r_{ai} \), \( r_{ei} \), and \( r_{ij} \).

2.5 Agent Design Specification

In the previous subsections we have presented the general structure of input modules and link and cortical agents. In order to simulate a pathway of brain areas, the human designer should assemble an appropriate number of these components. Additionally, the configuration of cortical and link agents has to be parametrically adjusted. In this subsection we summarize the parameters that have to be specified in order to completely define cortical and link agents. We note that the structure of input modules involves only the number of virtual neurons, and thus no parametric adjustment is necessary.

We start with cortical agents, described by a plane with prespecified dimension (in the current study it is \([0, 100] \times [0, 100]\)) and a prespecified number of excitatory and inhibitory neurons. All other structural details are parameterized. Specifically, for each cortical agent in the model, the neighborhood radii \( n_{x_a}, n_{y_a}, n_{x_b}, n_{y_b} \in [1, 40] \) and \( n_{x_a}, n_{y_a}, n_{x_b}, n_{y_b} \in [1, 40] \) used for the definition of synapse sets are specified by six real values. The neural parameters \( \mu_x, \alpha_x, \beta_x \), and \( \mu_y, \alpha_y, \beta_y \) are defined by six more real values \( \mu_x, \mu_y \in [0, 1], \alpha_x, \alpha_y \in [0.1, 6] \), and \( \beta_x, \beta_y \in [-10, 10] \). Additionally, six integers specify the identifiers of the learning rules \( r_{ax}, r_{ea}, r_{ae}, r_{ai}, \) and \( r_{ei}, r_{ij} \in \{1, \ldots, 10\} \), which adjust synapse weights at run time. In summary, 18 parameters are necessary to specify the complete configuration of a cortical agent.

In order to configure a link agent, it is necessary to know the number of excitatory neurons of the efferent projecting cortical agent. This is because only excitatory neurons have axon projections, specifying intercortical connectivity. For example, for the link agent transferring information from cortical agent \( A \), having \( N_{x} \) excitatory neurons and \( N_{y} \) inhibitory neurons, to cortical agent \( B \), the projection coordinates of \( N_{x} \) axons need to be specified. This is done by utilizing \( 2N_{x} \) real values, specifying the \((x, y)\) coordinates of all axons. All excitatory variables take values in the range \([-5, 105]\). It is recalled that the \( x \) and \( y \) dimensions of cortical agents are in \([0, 100]\). Therefore, axon projections having an \( x \) or \( y \) coordinate in \([-5, 0]\) or \([100, 105]\) are outside the cortical plane, and they are deemed deactivated.

In order to design a computational model consisting of many components (cortical and link agents), we need to specify the full details of their structure by giving appropriate values to the above-mentioned parameters. This task is addressed by an artificial evolutionary mechanism facilitating systematic exploration of agents’ configurations and optimal parameter selection. Furthermore, in order to support the autonomy of agents, a coevolutionary method is employed, evolving a separate population for each agent of the model. The details of this mechanism are described in Section 3.

2.6 Reinforcement Learning

Reinforcement learning models have become very popular in robotic applications in recent years [75]. Despite the effectiveness of reinforcement learning approaches, the biological reliability of this learning scheme has been criticized. However, some researchers have suggested that Hebbian learning
mechanisms can facilitate training based on reinforcement signals (e.g., [51]). This is because the self-organized dynamics of cortical agents can adjust synapses supporting reinforcement learning skills of the artificial organism. In the present work, a variety of self-organization dynamics can be implemented by properly mixing the Hebbian-like learning rules described in Section 2.4. Therefore, computational models with advanced reinforcement learning skills can be implemented. Similar to our approach, [4, 71] have also evolved Hebbian rules to accomplish reinforcement training.

The idea behind Hebbian reinforcement learning is to treat the reward as an ordinary signal that can be properly given as input to a pre- and a postsynaptic neuron, in order to coordinate their activations [37]. The learning rule adjusting the weight of the synapse connecting the pre- and postsynaptic neurons is then responsible for either strengthening or weakening their connection. In other words, the external reinforcement signal takes advantage of the internal plasticity dynamics of the agent, modulating its performance.

3 Hierarchical Cooperative Coevolution

An effective optimization mechanism is necessary to support the configuration of complex brainlike systems, facilitating optimal selection of parameter values. Artificial evolution could effectively address this issue, because it is capable of handling complex structures, and additionally it provides a means to systematically map biological-like features on computational systems. However, the majority of applications that involve evolutionary processes employ a single genotype to encode global solution representations. Using this unimodal approach, it is not possible to sufficiently explore partial solutions corresponding to the components of the composite system [53]. Due to the distributed nature of brain-inspired computational systems, a design approach that also follows a distributed architecture would be particularly appropriate. This is because a distributed design mechanism can sufficiently investigate the specialties of system components, and at the same time address their coupled performance in the composite system.

Coevolutionary algorithms have been recently proposed, facilitating exploration in problems consisting of many decomposable substructures [5]. They involve two or more populations with interactive performance, each one evolving one component of the whole problem. Following the coevolutionary approach, different populations are allowed to evolve separately using their own evolutionary parameters, providing increased exploration competencies. Distinct populations are usually referred to as species in the coevolutionary literature, and thus both terms will be employed henceforth interchangeably.

The implementation of brainlike cognitive systems fits very well with coevolutionary approaches, because separate species can be utilized to perform design decisions for each component of the computational model, effectively addressing the role and the particular characteristics of the agents representing brain areas. At the same time, the distributed nature of the coevolutionary scheme facilitates the integration of system components and the formation of successively more complex structures. Finally, due to the advantageous capability of coevolution to address the characteristics of each component, coevolution supports partial redesign of existing models and their gradual improvement.

Most of the coevolutionary approaches presented in the literature can be classified as competitive [59] or cooperative [53]. Competitive approaches are based on an antagonistic scenario, where the success of one species implies the failure of the other. In contrast, cooperative approaches follow a synergistic scenario, where individuals are rewarded when they successfully cooperate with individuals from the other species. Since brain modeling aims at the cooperative performance of partial structures representing brain areas, in the following we only consider cooperative coevolution.

During the last few years, a large number of cooperative coevolutionary schemes has been proposed in the literature. However, in most of the schemes the significance of choosing the appropriate collaborator is overlooked [73, 74]. The majority of existing applications consider only the case of cooperating with the best individual from a species [34, 53] or a randomly selected set of individuals [5, 20]. Evidently, the coevolutionary process could be supported by the maintenance of successful assemblies of cooperators, as proposed in [46].
Recently, we introduced a two-level evolutionary scheme [38, 42] that aims at the successful selection of cooperators among species, as a means of improving the performance of coevolutionary algorithms. Specifically, besides separate evolution of each component, our method employs an additional evolutionary process to select the most appropriate individuals from partial populations. These optimally selected individuals are put together to construct successful solutions for the overall problem.

The present work extends this method to a hierarchical multilevel architecture developing a powerful HCCE scheme that serves as a design mechanism for implementing brain-inspired computational systems. The work described in [11] presents a first attempt toward formulating a hierarchy of coevolved species. In contrast with [11], however, our approach employs groups of coevolved species providing the opportunity to form significantly larger assemblies of cooperating components and, at the same time, emphasizes the independence of substructures by utilizing multiple semiautonomous criteria to guide partial evolutionary processes.

Below we describe the proposed HCCE scheme, focusing on the design of brain-inspired computational systems. For the sake of clarity of the presentation, we will continue working on the hypothetical modeling problem introduced in Section 2.1.

### 3.1 Hierarchical Organization

In the present study we utilize HCCE to optimally design brain-inspired cognitive systems consisting of cortical and link agents serving as the primitive components of our models. The proposed HCCE scheme employs many partial evolutionary processes, each one designing one component of the model. The evolved populations (species) consist of individuals encoding candidate configurations of primitive components (either a cortical or a link agent\(^3\)). Therefore we call these populations primitive structure (PS) species. Additionally, we use higher-level evolutionary processes that aim at combining configurations of primitive components. In this case, the evolved populations encode candidate assemblies of primitive components (cortical and link agents). These higher-level processes are responsible for coordinating the evolution of groups of PS processes. Therefore, we call them coevolved groups (CGs), as they are responsible for enforcing the cooperation between components of the model. It is noted that a CG can also be a member of another CG. Thus, several CGs can be organized hierarchically in a treelike architecture (for example, see Figure 4).

In order to give a specific example of an HCCE scheme, we turn back to the problem described in Section 2.1, assuming that we want to design an HCCE process that will specify the structure of the model presented in Figure 1. Four PS species are employed to explore the structure of primitive components A, L1, B, and L2, searching for optimal cortical and link configurations. We assume that the functionality of the overall system aims at the accomplishment of task T by the robot. Additionally, in order to highlight the specialized roles of A and B, we assume that cortical agent A should support the accomplishment of subtask T1, while cortical agent B should support the accomplishment

---

\(^3\) Following the discussion in Section 2.5, input modules are virtual components without processing power. Their structure is static and predefined. Thus, they are not subject to optimization.
of subtask T2 (for example, the composite task T could correspond to a goal following behavior, with subtask T1 corresponding to goal identification, and subtask T2 corresponding to motion direction shifting). The specialized roles of A and B are addressed by grouping the components of the model in two CGs having separate design objectives (Figure 4). In particular, CG1 encodes assemblies of candidate structures for A and L1, searching for those configurations that successfully accomplish tasks T and T1. In a similar way, CG2 is searching for B and L2 configurations that are capable of accomplishing tasks T and T2. Finally, a top-level CG3 supports integration of CG1 and CG2 components to a successful composite model aiming at the accomplishment of global task T. Overall, a three-level HCCE process is necessary for implementing the underlying model.

3.2 Encoding

In the following we describe the structure of HCCE genotypes that encode primitive structures (PSs) and coevolutionary groups (CGs).

3.2.1 Chromosome Structure

We have implemented a general purpose chromosome\(^4\) that can be properly adjusted to encode both PSs and CGs. Specifically, the individuals used in all (partial) evolutionary processes are described by (i) an identification number (id) and (ii) two different types of variables encoding the evolved parameters. The general form of the chromosome is demonstrated in Figure 5a. The unique identification number of an individual is preserved during the coevolutionary process, making possible the definition of assemblies of individuals (i.e., assemblies of components). We turn now to the encoding of the information that can be represented by the two types of variables. The first type takes values from a set of unordered numbers (e.g., \{1, 5, 7, 2\}, with the ordering of the elements being of no significance). These variables are called set variables, and they are employed to store ids, encoding the relationship between id-assigned elements of the model. The second type of variables is allowed to take values within a range (e.g., \[0, 1\]); therefore, they are called range variables, and they are employed to search the continuous parameter domains. The values of set variables and range variables are encoded in the genome by integers and real numbers, respectively. They are graphically represented with dotted and solid boxes (see Figure 5a).

---

\(^4\) The genotype is designed in an abstract form, capable of encoding a variety of computational structures. Thus, neural agents of any level of biological plausibility can be encoded and evolved.
3.2.2 Encoding Components and Assemblies
In order to encode the detailed configurations of cortical and link agents, appropriately modified instances of the general chromosome are utilized. Specifically, according to the description of cortical agents, their structure is completely specified by 18 variables (see Section 2.5). These variables are mapped on the genotype as follows. Six set variables encode the ids of the learning rules responsible for performing real-time adjustment of synaptic weights, six range variables encode neighborhood radii necessary for synapse definition, and six range variables encode neural parameters of excitatory and inhibitory neurons. Overall, the chromosome utilized to encode the structure of cortical agents is formulated as shown in Figure 5b.

Following the description of link agents, they are completely defined by the coordinates of axon projections (see Section 2.5). In particular, a link structure transferring neural activation from a cortical agent with N, excitatory neurons to another cortical agent will have N, axons, and therefore 2N, range variables are necessary to encode the (x, y) coordinates of all axons. No set variables are necessary for encoding link agents. The chromosome used for encoding the structure of link agents is illustrated in Figure 5c.

The individuals of coevolved agent groups (CGs) encode assemblies consisting of PSs and other CGs trying to coordinate lower-level partial evolutionary processes. In order for a CG to guide the evolutionary process of S species, it has to encode assemblies of length S. This is achieved by utilizing S set variables, each one linked with one lower-level species. A set variable can be assigned any identification number of an individual from the corresponding lower-level species. No range variables are used for CG chromosomes. A graphical illustration of the chromosome employed by a CG's species is given in Figure 5d.

3.3 HCCE Internal Structure
The HCCE scheme that optimizes a brain-inspired computational system employs several partial evolutionary processes being organized in a treelike hierarchical manner. In particular, CG species are used as branches of the tree, while PS species are used as leaves (e.g., Figure 4). In the following, we present the internal structure of HCCE, describing how CG individuals are used to define assemblies of components.

We turn back to our working example and the HCCE process of Figure 4. A snapshot of this process is shown in Figure 6, demonstrating the formulation of assemblies of cortical and link agents. In order to simplify the figure and make it more easily readable, we do not show the detailed encoding of cortical and link agents. Each variable on the chromosome of a CG individual encodes the id of a candidate partial solution at the lower level. The arrows connecting individuals among species illustrate how the HCCE builds candidate composite solutions. For example, the individual with id = 7 of species CG3 encodes a solution consisting of partial assemblies with id = 19 at CG1 and id = 3 at CG2. Further analyzing the assembly at CG1, it consists of the individual with id = 14 at species A, and the individual with id = 21 at species L1. In the same way, analyzing the assembly of CG2 with id = 3, it consists of the individual with id = 4 at species B, and the individual with id = 5 at species L2. The abovementioned individuals from species A, L1, B, L2 will be decoded to detailed agent structures formulating the complete candidate solution described by individual with id = 7 of species CG3.

Obviously, individuals (candidate configurations) of A and L1 can be members of more than one partial assembly in CG1, while B and L2 individuals can be members of more than one assembly in CG2. This is indicated in Figure 6 by the arrows pointing at PS individuals. In a similar way, partial assemblies described by CG1 and CG2 individuals can participate in many CG3 global assemblies. This is true, for example, for the individual with id = 9 of CG1, and for the individual with id = 16 of CG2.

The distributed architecture of the coevolutionary scheme facilitates the segmentation of the problem space to smaller parts that can be more easily explored. This is because the evolution of each PS species explores the parameter space of only one primitive system component (either a cortical or a link agent). Besides the distributed architecture, the proposed scheme is also hierarchically organized.
The evolution of CG species searches within PS populations, finding those individuals that can successfully cooperate. Fortunately, by following this approach the population of CG individuals memorizes the best assemblies of components across consecutive evolutionary generations. In that way, the best CG individuals are used as a basis to drive the composite coevolutionary procedure.

3.4 Fitness Assignment
The individuals of the HCCE scheme are evaluated by formulating and testing all encoded problem solutions. This is done by sequentially accessing populations, starting from the highest level. The individuals of CGs at intermediate levels are used as guides to select cooperators among PS species. Then, PS individuals are decoded to detailed cortical and link agents, which are put together to construct a candidate solution for the complete problem. This solution is subsequently evaluated, testing the coupled performance of agent structures.

Although the majority of existing cooperative coevolutionary methods assume that all species share a common fitness function (e.g., [5, 19, 34, 73]), the proposed HCCE scheme is capable of using separate fitness functions for each coevolved species. This is a clear advantage for the coevolution of agents, because separate design objectives can be defined for each agent, effectively addressing its specialized characteristics.

Specifically, all individuals of a species $s$ are evaluated by a global fitness function $f_s$. Additionally, separate partial fitness functions $f_{s,t}$ can be defined, each one evaluating the ability of a candidate solution to serve task $t$. Then, partial fitness values are combined in a multiplicative manner to estimate the global fitness:

$$f_s = \prod_t f_{s,t}$$

(2)
The multiplication operator favors individuals that can accomplish (at least partly) all tasks, distinguishing them from those that fail in any one of them.

All PS species grouped on the same CG share common objectives, which implies that they share the same fitness measures. For the example at hand, the fitness function of species $A$ and $L1$ on a task $i$ is equal to the fitness function of $CG1$ (i.e., $f_{A,i} = f_{L1,i} = f_{CG1,i}$). The same is also true for species $B$, $L2$, and $CG2$ (i.e., $f_{A,i} = f_{L2,i} = f_{CG2,i}$). The fitness functions of $CG1$, $CG2$, and $CG3$ on a task $i$ can be different in general. This is because each CG species should evaluate the accomplishment of task $i$ according to the objectives of the underlying group of agents.

The cooperativity selection process at CG populations will potentially select a lower-level individual to participate in many assemblies. This is, for example, the case for the individual with id $= 14$ of species $A$, in Figure 6. Let us assume that an individual participates in $K$ assemblies, which means that it will get $K$ fitness values $f_{i,j}$ regarding the accomplishment of the $i$th task. Thus, it is given $K$ chances to demonstrate its suitability for the task, which is estimated by

$$f_{i,j} = \max_k \left\{ f_{i,k}^j \right\}, \quad k \in \{1, \ldots, K\}$$

where $f_{i,k}^j$ is the fitness value of the $k$th solution formulated with the membership of the individual under discussion. Partial fitness values obtained by Equation 3 are subsequently used in Equation 2 for estimating the global fitness of individuals.

The fitness assignment process is explained in detail by means of our working example. We remind the reader that according to the employed scenario, the composite model should accomplish task $T$, while partial models should develop their own partial functionalities. Specifically, the components $A$ and $L1$ should develop the behavior described by subtask $T1$, while the components $B$ and $L2$ should develop the behavior described by subtask $T2$. Summarizing, the individuals of population $CG3$ are evaluated on task $T$, the individuals of populations $CG1$, $A$, and $L1$, are evaluated on $T$ and $T1$, and the individuals of populations $CG2$, $B$, and $L2$, are evaluated on $T$ and $T2$. Following the formulation introduced in Equation 2, the above scenario is described mathematically by the following equations:

$$f_{CG3} = f_{CG3,T}, \quad f_{CG1} = f_{CG1,T} \cdot f_{CG1,T1}, \quad f_{CG2} = f_{CG2,T} \cdot f_{CG2,T2}$$

For simplicity we assume that $f_{CG3,T} = f_{CG1,T} = f_{CG2,T}$; in general they can be different. Additionally, we recall that PS species share the same fitness measures with their higher-level CG. This implies that $f_{A,T} = f_{L1,T} = f_{CG1,T}$ for $T$, and $f_{A,T1} = f_{L1,T1} = f_{CG1,T1}$ for $T1$. Furthermore, according to Equation 2, the global fitness functions are

$$f_A = f_{A,T} \cdot f_{A,T1} \quad \text{and} \quad f_{L1} = f_{L1,T} \cdot f_{L1,T1}$$

In a similar way, $f_{B,T} = f_{L2,T} = f_{CG2,T}$ and $f_{B,T2} = f_{L2,T2} = f_{CG2,T2}$, while according to Equation 2, the global fitness functions are

$$f_B = f_{B,T} \cdot f_{B,T2} \quad \text{and} \quad f_{L2} = f_{L2,T} \cdot f_{L2,T2}$$

Let us now turn to the snapshot of our working example. For the sake of brevity, we discuss fitness assignment only for $CG3$, $CG2$, $B$, and $L2$. The assigned fitness values are illustrated in Figure 7, where we have zoomed out the species $A$ and $L1$. Let us start from the top-level species
CG3, assuming that its individuals have been evaluated on task T. Each individual is assigned only one score; therefore its fitness equals that score (see also Equation 4). We turn now to CG2. Let us examine the individual with id = 16, which participates in two assemblies of CG3. Its ability to serve task T (i.e., \( f_{CG2,T} \)) will be evaluated with the maximum of the respective fitness values. Additionally, CG2 individuals are assigned separate fitness values for accomplishing task T2. Thus, the individual with id = 16 is assigned one more partial fitness value, \( f_{CG2,T2} \). Then, according to Equation 4 (see also Equation 2), its global fitness \( f_{CG2} \) is estimated by the product of partial fitnesses. The same process is repeated for the rest of the individuals of CG2.

We turn now to the individuals of PS species B and L2. Let us focus first on the individual of B with id = 1, which has multiple participation, evaluated many times on the accomplishment of tasks T and T2. Therefore, its partial fitnesses regarding the two tasks are estimated by the maxima of the respective values, and finally it is assigned a high global score. However, the individual with id = 4 of species B participates in only one assembly, and therefore it will be assigned the scores of this particular assembly. We note that although it receives a high score for its participation in task T2, it receives zero for its participation in T, and consequently its global score according to Equation 6 will be zero. Additionally, there are individuals that receive high global scores, even if none of the assemblies they participate in performed successfully in all tasks. For example, consider the individual with id = 5 of species L2. It participates in two assemblies, one of them receiving a high score in T and a low score in T2, while the other receives a high score in T2 but a low score in T. This is probably because its collaborators in the one case are capable of accomplishing T but not T2, while in the other case, the other set of collaborators are capable of accomplishing T2 but not T. However, the individual with id = 5 will be assigned two high partial scores, because it is capable of successfully serving both tasks. As a result its global fitness value will be high.
Intuitively, the fitness assignment mechanism discussed above—described mathematically by Equations 2 and 3—aims at distinguishing the successfully designed components of the model from those that are unsuccessfully designed. The most successful ones are favored during reproduction for the new generations, while the less successful ones are subject to changes, facilitating exploration of the problem space.

3.5 Lesion Simulation

Following recent trends in studying computational models in lesion conditions [1, 52, 65], the proposed modeling approach can easily simulate brain lesion experiments [40]. This is because the distributed, agent-based representation of brain areas facilitates lesion simulation by simply eliminating the appropriate agent components. Additionally, the HCCE process is capable of designing the model with consideration of its performance in the underlying operating conditions (i.e., pre- and post-lesion) by employing an appropriate number of partial fitness measures.

The design of a computational system that mimics the results of a neuroscientific experiment should be based on the behavior of animals in pre- and post-lesion conditions. In order to simulate real-life performance, we design the computational counterparts of animal behaviors described by tasks $T_{\text{pre}}$ and $T_{\text{post}}$. The artificial organism should be capable of accomplishing the task $T_{\text{pre}}$ in pre-lesion operating conditions (all agents of the model are active), while it should accomplish $T_{\text{post}}$ in post-lesion conditions (some agents are deactivated). We note that this is not an easily accomplished objective, because typically, artificial neural network systems completely collapse after eliminating a small number of neurons. However, the HCCE-based optimization mechanism is capable of designing the model, enforcing the accomplishment of tasks $T_{\text{pre}}$ and $T_{\text{post}}$ for different operating conditions of the computational system. This is because separate partial fitness functions $f_{T_{\text{pre}}}$ and $f_{T_{\text{post}}}$ can be used for $T_{\text{pre}}$ and $T_{\text{post}}$ tasks. The components participating only in the pre-lesion operation of the model will be designed according to $f_{T_{\text{pre}}}$, while the components participating in both the pre- and post-lesion operation will be designed according to both $f_{T_{\text{pre}}}$ and $f_{T_{\text{post}}}$.

For example, let us slightly modify the scenario of our working example. We assume that the composite model consisting of $A$, $L1$, $B$, and $L2$ should accomplish the pre-lesion task $T_{\text{pre}}$ (e.g., a goal-following behavior). Additionally, we assume that system behavior is impaired after lesion of $B$ and $L2$. However, the remaining components are still capable of performing cognitive processes in $A$, accomplishing the task $T_{\text{post}}$ (e.g., a goal identification task).

We turn now to the fitness functions that will guide the evolution of HCCE species. According to the above-described scenario, the components $A$ and $L1$ should support both $T_{\text{pre}}$ and $T_{\text{post}}$ tasks. Therefore, according to Equation 2 the species CG1, $A$, and $L1$ are evolved by the following fitness functions:

$$F_{\text{CG1}} = F_{\text{CG1}, T_{\text{pre}}} \cdot F_{\text{CG1}, T_{\text{post}}}, \quad F_A = F_{A, T_{\text{pre}}} \cdot F_{A, T_{\text{post}}}, \quad F_{L1} = F_{L1, T_{\text{pre}}} \cdot F_{L1, T_{\text{post}}} \quad (7)$$

In contrast, the components $B$ and $L2$ should support only the accomplishment of the task $T_{\text{pre}}$. Therefore, the fitness functions used by CG2, $B$, and $L2$ are

$$F_{\text{CG2}} = F_{\text{CG2}, T_{\text{pre}}}, \quad F_B = F_{B, T_{\text{pre}}}, \quad F_{L2} = F_{L2, T_{\text{pre}}} \quad (8)$$

Finally, the top level CG3 should integrate partial models to a composite system, taking into account all relevant tasks. Therefore, the fitness function for CG3 is

$$F_{\text{CG3}} = F_{\text{CG3}, T_{\text{pre}}} \cdot F_{\text{CG3}, T_{\text{post}}} \quad (9)$$

5 Theoretically, CG3 could aim at only accomplishing task $T_{\text{pre}}$. However, it has been experimentally proven that HCCE processes addressing lesion experiments are more successful when all tasks are targeted by the highest level.
3.6 Evolutionary Procedure

After presenting HCCE architecture for optimizing brain-inspired computational systems, we turn to the evolutionary operators applied on partial populations.

3.6.1 Crossover and Mutation Operators

Based on the general genome structure described in Section 3.2 (see also Figure 5), we have implemented crossover and mutation operators that perform separately on set variables and range variables. During the mating process, the usual single-point crossover is applied to both set variables and range variables. This is demonstrated graphically in Figure 8a. Mutation is implemented in a different way for the two kinds of variables. In particular, in the case of range variables, mutation corresponds to additive noise. In the case of set variables, mutation corresponds to a random assignment of a new id value. Both mutation cases are demonstrated in Figure 8b.

Turning back to the working example, mutations on the individuals of species A and B correspond to changes in the learning dynamics of the cortical agent, and/or the synaptic connectivity, and/or the parameters of excitatory and inhibitory neurons. In the case of species L1 and L2, mutation corresponds to changes in the axon projection coordinates (which subsequently affect the synaptic connectivity of the receiving cortical agents). In the case of CG species, mutation corresponds to the selection of a new individual from the lower-level species.

It is worth emphasizing that the mutation of set variables is different for PS and CG individuals. This is because in the case of PS, set variables encode learning-rule identifiers. Thus, mutation corresponds to a random assignment of a new learning rule. In the case of CG, set variables encode identifiers of individuals at the lower species. Thus, mutation corresponds to the probabilistic selection of a new individual, based on fitness scores. Following this approach, the best-fitted individuals are most probably selected to participate in the new assemblies.

3.6.2 Replication Operator

Due to the probabilistic nature of the assembly configuration process performed in CG species, there are individuals from the lower levels that are multiply selected to participate in many assemblies. At

---

Figure 8. A hypothetical example demonstrating the application of (a) crossover operator and (b) mutation operator.
the same time, some individuals in the same species might exist that are not offered any cooperation. They are henceforth termed noncooperative.

The individuals with multiple participation have many chances to demonstrate their suitability on given tasks. This fact supports a fitness assignment process that aims at distinguishing successfully from unsuccessfully designed components. However, having a large number of multiple cooperations is generally a drawback for the coevolutionary process. This is because different cooperators would demand evolution of the same individual in different directions.

Noncooperative individuals can be utilized to decrease the multiplicity of cooperations for those that are heavily reused. This is achieved by employing a new genetic operator termed replication\(^6\) [42]. Briefly, for each noncooperative individual \(x\) of a species, replication identifies the fittest individual \(y\) with more than \(\text{max} \) cooperations. The genome of \(y\) is then copied to \(x\), and \(x\) is assigned \(\text{max} - 1\) cooperations of \(y\), by updating the appropriate individuals of the population at the higher level. After replication, individuals \(x\) and \(y\) are allowed to evolve separately, following different evolutionary directions. Thus, replication enforces the coevolutionary process to exploit the whole population of individuals in each partial species. The application of the replication operator for the species at the top level is not applicable, since there is no other, higher evolutionary process.

Intuitively, \(\text{max}\) balances the exploration-versus-exploitation dynamics of the coevolutionary procedure. High values of the replication threshold indicate that the assemblies of individuals of partial species remain largely unaffected, in order to be employed as cooperators for the individuals of the rest of the species. Thus, the dynamics of the coevolutionary procedure put more emphasis on exploiting the current results. In contrast, low values of the replication threshold \(\text{max}\) prevent individuals from participating in many assemblies, enforcing their independent evolution in many different directions. Therefore, in that case, the dynamics of the coevolutionary procedure put more emphasis on the exploration of the search space.

### 3.6.3 Evolutionary Step

Just after individual testing and the application of fitness assignment, the individuals of each species are sorted according to their global fitness values. The HCCE scheme is evolved in synchronous evolutionary steps for all partial populations. Specifically, starting from the highest level of the hierarchy and moving downward, each species is sequentially applied the genetic operators described above. At first, replication reduces the very large number of cooperations for individuals. Then, a predefined percentage of individuals are probabilistically crossed over. Finally, mutation is applied in a small percentage of the resulting population to preserve diversity.

At the end of the evolutionary step, a new set of candidate problem solutions have been implemented, and they are ready for testing on given tasks. The cycle of testing-evaluation-evolution is repeated for a predefined number of generations.

### 4 Experimental Methodology

The suitability of the proposed computational framework on engineering brain models is illustrated by incrementally designing a brainlike computational system that supports the cognitive abilities of a simulated robot. Specifically, we start by modeling the cortical areas involved in working memory (WM), investigating how WM is utilized in accomplishing delayed response (DR) tasks. Additionally, in order to evaluate the effectiveness of the proposed design procedure, we compare HCCE with the enforced subpopulation scheme of cooperative coevolution and with ordinary, unimodal evolution. Then, we investigate the possibility of making incremental design steps, incorporating reinforcement

---

\(^6\) The proposed operator does not aim to be a computational representative of the biological DNA replication, although they share some characteristics.
learning (RL) skills into the previously implemented model. Specifically, it is shown that the agent-based coevolutionary framework facilitates both the integration of new agent structures to the model and the redesign of preconfigured components according to an enhanced set of objectives, advancing the capabilities of the overall system.

4.1 Simulation Environment
The implemented models are embedded in a simulated mobile robot that facilitates environmental interaction. We employ a two-wheeled simulated robotic platform equipped with eight object proximity sensors, eight light sensors, and eight positive reward sensors, all of them uniformly distributed (see Figure 9). All 24 sensors take values in the range [0, 1], with 1 representing maximum activation.

The environment of the robot consists of a wall, light sources, and positive reward areas. The robot uses object proximity sensors to sense walls when they are at a distance less than 100 points. The activity of the proximity sensor increases linearly from zero to one as the robot approaches the wall. The light source is represented by a circular area with a predefined radius of 150 points. Light sensors have maximum activation when the robot is located at the center of the circle. The sensation of light decreases linearly to zero as the robot moves to the circumference of the circle. The positive reward is also simulated by a circular area. The radius of the reward area is 90 points. The robot is sensing the reward when it is located in the circle. The amount of reward changes linearly from zero to one as the robot is moved from the circumference of the circle to the center.

The simulated robot is moved by two wheels that are moved independently of one another. For each wheel, we assume the existence of a pair of speeds, operating in an agonist-antagonist mode. One of them directs the wheel forward, and the other backward. Both speeds are in the range [0, 0.5]. The difference between forward and backward speeds determines the motion of the wheel. Overall, four speed values are necessary to determine the speed of the whole robot in every simulation step.

4.2 Working-Memory Model
The first set of experiments aims at modeling the posterior parietal cortex (PPC), prefrontal cortex (PFC), primary motor cortex (M1), and spinal cord (SC), emphasizing their role in the development of WM and the accomplishment of DR tasks. Following well-established knowledge from the field of neuroscience, M1 encodes primitive motor commands that are expressed to actions by means of SC. PPC-PFC reciprocal interaction operates in a higher-level encoding WM [7], which is used to develop plans of future actions. PFC activation is then passed to M1, which modulates its performance according to higher-level orders. Additionally, several experiments have highlighted the performance of

![Figure 9. A schematic representation of the simulated robot.](image-url)
these structures in lesion conditions. In particular, it is well known that a PFC lesion affects the planning ability of the organism, resulting in reduced ability to move purposefully [54].

In the past, several studies have tried to model the abovementioned cortical areas. For example, computational models of M1 have been developed in [2, 70], which, however, do not emphasize the self-organized understanding of environmental characteristics by the organism. Existing PFC computational models emphasize WM activity by means of recurrent circuits [7, 28]. Still, these models are not operative, in the sense that they are not linked to other structures to affect their performance. Additionally, computational models aiming at the accomplishment of memory-guided tasks have been proposed in [45, 76], but they employ compact artificial neural network structures, without specific assumptions on the functionality of partial brain areas.

### 4.2.1 Model and Tasks

The model is implemented following the agent-based coevolutionary computational framework, demonstrating the ability of the latter to design complex systems consisting of autonomous yet cooperative components. Separate agents are utilized to represent each substructure of the mammalian central nervous system. Specifically, the investigated brain areas are simulated by using four cortical agents that are properly connected via link agents (Figure 10).

In order to design a computational model that mimics the functionality of brain areas involved in WM, the experimental process reproduces a brain lesion scenario. Three partial tasks are designed, highlighting the role of each agent in the model. In particular, the composite computational model should be capable of accomplishing a DR task, simulating the pre-lesion performance of animals [16]. In brief, a light cue is presented to the simulated robot, which has to memorize the side of light cue appearance in order to make a future choice related to 90° turning, left or right. Similar tasks have also been discussed in other studies (e.g., [76]). The accomplishment of the DR task is further supported by two partial behaviors. The first accounts for the development of WM-like activation in PPC and PFC, which are the brain structures most closely linked to WM [7]. The second accounts for purposeless motion by M1 when a lesion occurs on the higher-level structures, simulating the post-lesion performance of animals [54]. The three tasks are presented below, starting with the simpler ones.

---

7 It is known that the spinal cord is less plastic than the cortex and it should be modeled with a specialized structure. However, in order to simplify the presentation of results, in the present study all modules are represented with the same computational component.
4.2.1.1. Wall avoidance task The first task accounts for primitive motion abilities without purposeful planning. For mobile robots, a task with the above characteristics is wall avoidance navigation. Thus, for the needs of the present study, the isolated performance of M1-SC structures aims at navigating the robot to avoid wall bumps. The simulated robot starts from a predefined location in the top of the maze but with a random initial direction, and it is tested for \( M = 1,500 \) simulation steps. The successful accomplishment of the task is evaluated by the function

\[
E_{\text{wall}} = \left( \sum_{M} (sl + sr - 1)(1.0 - p^2) \right) \left( 1 - \frac{2}{M} \left| \sum_{M} \frac{sl - sr}{sl \cdot sr} \right| \right)^3 \left( 1 - 2 \sqrt{\frac{B}{M}} \right)^3
\]  

(10)

In the above equation, \( sl \) and \( sr \) are the instant speeds of the left and the right wheel, \( p \) is the maximum instant activation of distance sensors, and \( B \) is the total number of robot bumps. The first factor supports forward movement far from the walls, the second supports straight movement without unreasonable spinning, and the last minimizes the number of robot bumps on the walls. The larger the value of \( E_{\text{wall}} \), the better the performance of the robot in wall avoidance navigation.

4.2.1.2. Working memory task The second task aims at the development of WM, that is, the ability to store goal-related information in order to guide forthcoming actions. The robot starts from a predefined initial position in the top of the maze, but with a random direction in the range \([-85^\circ, -95^\circ]\). The task lasts 300 simulation steps, and the robot is driven by a simple human-hardwired controller that avoids wall bumps. In the current experimental scenario, a light cue is presented on the left or right side of the simulated robot for the initial 40 simulation steps, and then disappears. WM aims at encoding the side of light cue presentation, developing different patterns of persistent PFC activity, for a short future period (simulation steps 41 to 250).

Two different states \( l \) and \( r \) are defined, associated respectively to the left and the right side of the light source appearance. For each state, separate activation averages \( a^l \) and \( a^r \) are computed, with \( j \) identifying one of the \( N_e \) excitatory neurons at PFC. The average is over a period of \( M = 41-250 \) simulation steps. The activation of inhibitory neurons at PFC is not considered, since only excitatory neurons encode effenter information. The formation of WM patterns related to the side of the light cue appearance is evaluated by considering the persistence of activation in the PFC agent:

\[
E_{\text{wm}} = \min \left\{ \sum_{j, a^l > a^l} (a^l - a^l), \sum_{j, a^r > a^r} (a^r - a^r) \right\} \cdot \left( \frac{v_l}{m_l} + \frac{v_r}{m_r} \right)
\]  

(11)

The first factor of Equation 11 supports separate representation of the states \( l \) and \( r \) at PFC, by different sets of active neurons. The second factor enforces the consistency of PFC activation, with \( m_l, v_l, m_r, \) and \( v_r \) being the mean and variance of average activation at the corresponding states:

\[
m_l = \frac{1}{N_e} \sum_j a^l_j, \quad v_l = \frac{1}{N_e} \sum_j (m_l - a^l_j)^2
\]

\[
m_r = \frac{1}{N_e} \sum_j a^r_j, \quad v_r = \frac{1}{N_e} \sum_j (m_r - a^r_j)^2
\]

If persistently few, but the same, neurons are activated during the observed period, the second factor of Equation 11 will get a high value. If activation is not consistent, different neurons are activated in every simulation step, and this factor will get a low value. Overall, high values of \( E_{\text{wm}} \) indicate successful development of working memory patterns.
4.2.1.3. Same-side delayed response task Finally, the third task aims to combine the above behaviors, formulating a more complex one. The successful integration of partial behaviors is demonstrated by means of the same-side (SS) DR task. The robot is initialized to a predefined starting position in the top of the maze with a random direction in the range $[-85^\circ, -95^\circ]$, similarly to the WM task described above. The duration of the task is 300 simulation steps, and it is separated into a sampling and a response phase. In the sampling phase, a light cue is presented on the left or right side of the simulated robot for 40 simulation steps. During the response phase, which lasts 260 simulation steps, the light source disappears, and the robot drives freely to the end of the corridor, memorizing the side of light cue appearance. Then, it has to make a choice: a $90^\circ$ turn left or right. The robot response is considered correct if it turns to the side where the light source had appeared during the sampling phase.

In order to evaluate the behavior of the simulated robot, a target location is defined on each side of the maze, depending on the position of the light cue sample. The robot has to approximate the target location without crashing into the walls. The successful approximation to a target location $x$ is estimated by

$$G^x = \left(1 + 3 \left(1 - \frac{d}{D} \right)\right)^3 \left(1 - 2\sqrt{\frac{B}{M}}\right)$$

where $d$ is the minimum Euclidean distance between the target and the robot, $D$ is the Euclidean distance between the target and the starting location of the robot, and $B$ is the total number of robot bumps.

The accomplishment of the SS response task is evaluated by means of two subtasks, testing separately for the right and the left turning of the simulated robot. Each time, different target locations are employed to evaluate the performance of the robot. Hence, the total accomplishment of the memory-guided SS DR task is evaluated according to

$$E_{ss} = G^r \cdot G^l$$

which implies high scores for both subtasks. The larger the value of $E_{ss}$, the better the accomplishment of the SS task by the robot.

4.2.2 Coevolutionary Experimental Protocol
We turn now to the design of the model by means of the HCCE scheme. According to the lesion scenario followed in the present study, each agent needs to serve more than one task. This guides the classification of PS species that evolve the components of the model into CG species. The tasks served by each group of agents are illustrated in Figure 11, at the right side of each CG. The structures under CG1 are related to M1-SC interactions, and they need to serve both the wall avoidance and the SS response task. The structures under CG2 are related to PFC and its afferent and efferent projections. They need to serve WM persistent activation and the SS response task. The structures under CG3 are related to PPC and its afferent projections, which have to support WM activation only. Finally, the top level CG4 enforces cooperation within partial configurations, facilitating the accomplishment of all three tasks, in both the pre- and the post-lesion operating modes.

The individuals of the coevolutionary scheme encoding candidate problem solutions (agent configurations) are tested as follows. The individuals of the top-level species are accessed one by one. Each individual of CG4 guides cooperator selection among its lower-level CG and PS species. Individuals of PS species are decoded to detailed agent structures, and they are put together to formulate a solution for the composite problem. Then, the model is tested on the accomplishment
of the SS response task. Next, PPC-PFC interaction is isolated by deactivating the agents under CG1. The remaining structures are tested on the WM task. Finally, CG1 agents are reactivated, and now CG2 structures are deactivated to simulate PFC lesion. The remaining agents are tested on the accomplishment of wall avoidance navigation. After all these tests and the assignment of fitness values, we go back to CG4, selecting a new individual. The testing loop continues until all CG4 individuals have been examined.

The individuals in all species are assigned a combination of evaluation indices for the accomplishment of the tasks described above. Specifically, the agents grouped under CG1 serve the success of two tasks, namely wall avoidance and SS response. Thus, the fitness function employed for the evolution of CG1 and its lower-level species is based on the measures evaluating the success of these tasks. Following the formulation introduced in Equations 2 and 3, we have

\[
f_{CG1} = f_{CG1,1} \cdot f_{CG1,2} \quad \text{with} \quad f_{CG1,1}^k = E_{wa}, \quad f_{CG1,2}^k = \sqrt{E_{ss}}
\]  

where \( k \) represents each membership of an individual in a proposed solution.

Similarly, CG2 components support the accomplishment of WM and SS response tasks. Thus, the fitness function that guides the evolution of CG2 and its lower-level species is defined by means of the corresponding evaluation measures:

\[
f_{CG2} = f_{CG2,1} \cdot f_{CG2,2} \quad \text{with} \quad f_{CG2,1}^k = E_{wm}^2, \quad f_{CG2,2}^k = \sqrt{E_{ss}}
\]  

where \( k \) is as above.

The third group, CG3, evolves PPC and all link agents projecting on it. These structures need to serve only the development of WM activation in PFC. The fitness function employed for the evolution of CG3 is defined by

\[
f_{CG3} = f_{CG3,1} \quad \text{with} \quad f_{CG3,1}^k = E_{wm}
\]  

where \( k \) is as above.

Additionally, the top-level evolutionary process CG4 enforces the integration of partial configurations in a composite model, aiming at the successful accomplishment of all three of the tasks. The fitness function guiding the evolution of CG4 supports simultaneous success on wall avoidance, working memory, and same-side response tasks. It is defined according to the formulation introduced in Equations 2 and 3 by

\[
f_{CG4} = f_{CG4,1} \cdot f_{CG4,2} \cdot f_{CG4,3} \quad \text{with} \quad f_{CG4,1}^k = \sqrt{E_{wa}}, \quad f_{CG4,2}^k = E_{wm}^2, \quad f_{CG4,3}^k = E_{ss}
\]  

where \( k \) is as above.
Following the fitness functions discussed above (Equations 14–17), different species enforce the accomplishment of each task with a different weight. For example, compared to CG1, the fitness function that guides the evolution of CG4 enforces more the accomplishment of same-side response task than wall avoidance (see definitions of $f^{\text{CG1A}}$, $f^{\text{CG1B}}$, $f^{\text{CG4A}}$, and $f^{\text{CG4B}}$).

The coevolutionary process described above employed populations of 200 individuals for all PS species, 300 individuals for CG1, CG2, and CG3, and 400 individuals for CG4. Each parameter encoded in an individual of a PS species has a 2% probability of being mutated. The parameters of the individuals in CG species are mutated with probability 0.4%. For both kinds of species, individuals are crossed over with probability 60%. Additionally, an elitist evolutionary strategy was followed in each evolutionary step, with the seven best individuals of each species copied unchanged in the new generation, supporting the robustness of the coevolutionary process.

### 4.2.3 Results

After 170 evolutionary epochs, the process converged successfully. Sample results of robot performance on each task are illustrated in Figures 12, 13, and 14. First, the composite model exploits the interaction of partial structures successfully accomplishing the SS DR task (Figure 12). This behavior is based on the development of separate activation patterns at PFC, which encode the side of light cue appearance and memorize it for the future time period (Figure 13). Moreover, when a lesion occurs at PFC, the overall system does not collapse; it is still able to drive the simulated robot in a purposeless manner, following a wall avoidance policy (Figure 14). We note that we have tested the behavior of the simulated robot in the SS task after PFC lesion, and we have obtained random DRs (both to the left and to the right), produced each time by input sensory variations.

Overall, the obtained results have shown that brain imaging findings are successfully replicated by the model. This is achieved by means of the powerful HCCE process, which is able to consider and further specify the performance of the artificial system in both the pre- and post-lesion conditions. To the best of our knowledge, no other modeling framework provides this capability.

### 4.3 Comparing HCCE, ESP, and Unimodal Evolution

In the current set of experiments, we investigate the suitability of HCCE for designing distributed brainlike models. Specifically, we utilize as a test bed the problem discussed in the previous paragraphs, in order to compare HCCE with two other evolutionary schemes, namely, enforced sub-populations (ESP) [19] and ordinary unimodal evolution.

![Figure 12. A sample result of robot performance in the same-side delayed-response task, for (a) left-side and (b) right-side light cue presence. Goal positions are marked with double circles.](image-url)
4.3.1 Hierarchical Cooperative Coevolution

In order to evaluate the speed and robustness of the HCCE scheme, we perform six independent runs of the coevolutionary procedure discussed in Section 4.2.2. The obtained results are illustrated in Figure 15, where each column corresponds to a different run. In the first run, the progress of the HCCE scheme is initially slow, but after approximately 100 evolutionary epochs, the probabilistic search identifies a promising evolutionary direction, which is efficiently exploited to identify a set of successful solutions. In the following two runs, we see that the coevolutionary process is rather unstable. Specifically, the evolution of species $CG4$ is not able to formulate successful assemblies of cooperators that will be preserved in the consecutive epochs. This fact additionally affects the progress of evolution in species $CG2$ and $CG3$, which are trapped in suboptimal solutions. In the fourth run, the progress of the composite coevolutionary scheme develops slowly, and simultaneously for all species. The coevolutionary procedure is terminated without reaching the success rate of the first run. Still, the evolutionary progress has not stabilized, which means that more epochs are necessary for estimating a sufficiently good result. The fifth run is similar to the first. The progress of the HCCE procedure is initially slow, until a promising assembly of cooperators is identified. After a small unstable period in the coevolutionary procedure, an effective assembly is preserved, also driving the other individuals in an area of successful solutions. Finally, the last run is similar to the fourth. The evolution of each $CG$ proceeds without rapid changes. However, the convergence is a bit faster than in the fourth run, and thus the composite procedure is able to find solutions with nearly optimum fitness values.

In an attempt to formulate general comments on the progress of the HCCE scheme, we can state that the WM development task is critical for the success of the composite scheme. Note that the evolution of $CG3$ aims only at the accomplishment of WM; see Equation 16. Thus, by observing the
third row of Figure 15, we realize that whenever the solution of the WM task is stalled, the composite coevolutionary procedure does not converge successfully. This is explained by the fact that the WM development task is actually a subtask of the SS DR task. As a result, if WM is not sufficiently developed, the simulated robot cannot remember the sample cue to express the appropriate delayed response.

4.3.2 Enforced Subpopulation
Additionally, we investigate whether a different coevolutionary scheme is capable of solving the same problem, successfully specifying the structure of cortical and link agents. In particular, we approach the problem discussed above by utilizing the ESP coevolutionary scheme. In the current work, we have implemented the ESP algorithm described in [19], without however activating the stagnation check that practically reinitializes populations when the process gets stalled.

Specifically, ESP can be employed in two different ways to approach the problem at hand. In the first case, all populations of the ESP scheme are evolved according to a common set of objectives, utilizing the same fitness function. Hence, the results of accomplishing the three tasks (namely, wall avoidance, WM development, and SS delayed response) by either the composite or the eliminated configurations of the model are combined into a single measure. Like the function $f_{CCA}$ that evolves the top-level CG of the HCCE scheme (see Equation 17), the fitness of ESP individuals in all populations is measured by

$$f = \sqrt{E_{wa} \cdot E_{wm}^2 \cdot E_{ss}}$$

(18)

This objective implies that the progress of ESP evolution is directly comparable with the progress of the HCCE scheme. Twelve different species are employed to specify the structure of the 12
components of the model. All species are evolved according to the criteria described by Equation 18. We call this approach of ESP homogeneous, since all species share a common fitness function. According to [19], this is the standard approach of ESP.

Alternatively, we could highlight the specialized role of each component of the model, by employing several different fitness functions to evolve simultaneously each species of the ESP scheme. Similar to the HCCE configuration described in Section 4.2.2, we group the species of the coevolutionary process into three categories, each one evolved according to different design objectives.

Specifically, three different fitness functions are utilized. The first drives evolution of the species exploring the structures of M1, SC, L6, L7, and L8. Similar to Equation 14, it is described by

$$f = E_{wm} \cdot \sqrt{E_{ss}}$$

(19)

The second evolves the species specifying the structure of PFC, L3, and L5. This is similar to Equation 15, and it is described by

$$f = E_{wm}^2 \cdot \sqrt{E_{ss}}$$

(20)

The third fitness function drives the evolution of the species corresponding to PPC, L1, L2, and L4, and similar to Equation 16, it is described by

$$f = E_{wm}$$

(21)

We call this approach of ESP heterogeneous, because different species are evolved according to different fitness functions.

Similar to HCCE, each population evolving configurations of a component of the model consists of 200 individuals. In both the homogeneous and the heterogeneous ESP approaches, 2,000 individuals encoding assemblies of components are randomly created in each epoch. These complex assemblies aim at identifying successful solutions to the composite problem. Overall, each individual representing a candidate configuration of a cortical or a link agent participates in approximately 10 complete solution assemblies. The average fitness of individuals drives the evolution of each species. We note that in contrast to HCCE, the population of 2,000 complete solution assemblies of ESP is not evolved, but is reinitialized in each evolutionary epoch [19]. The success of these assemblies can be measured by

$$f = \sqrt{E_{wa} \cdot E_{wm}^2 \cdot E_{ss}}$$

(22)

which is similar to $f_{CCA}$ (see Equation 17). Thus, the progress of ESP evolution is directly comparable with the progress of the HCCE scheme.

We performed six independent runs of the ESP homogeneous and heterogeneous schemes, which were evolved for 170 epochs. The probabilities of applying crossover and mutation operators over the individuals encoding a cortical or a link agent were the same as the probabilities of the HCCE scheme. The results of these processes are illustrated in Figures 16 and 17. These results are directly comparable with the last row of Figure 15. Evidently, none of the ESP processes was successful. Additionally, no significant differences can be identified in the effectiveness of the two approaches. This is mainly because neither ESP process is explicitly directed toward constructing successful complex assemblies. In contrast, it is expected that due to the large number of complex

---

8 The original ESP formulation follows only the homogeneous scheme. In the present study, the heterogeneous ESP scheme has been introduced, investigating an alternative ESP approach to the problem at hand.
assemblies being evaluated, satisfactory assemblies will be randomly formulated. Unfortunately, as is indicated by the present results, this is not the case when many components need to be coevolved.

### 4.3.3 Unimodal Evolution

Finally, we approach the same problem by utilizing a unimodal evolutionary scheme. In particular, a single, large chromosome is employed to encode the structure of all cortical and link agents of the model. Hence the parts of the genotype corresponding to candidate configurations of system components participate in only one composite solution. Following the unimodal approach, it is not possible to evolve system components separately. Thus, the role of substructures cannot be indicated by partial fitness functions. In other words, the evolution of agents cannot follow their own evolutionary directions. Still, the separate role of each agent in the model can be revealed by testing the performance of composite and partial solutions in accomplishing the underlying three tasks.

The objective function that guides the evolutionary process is defined according to the fitness function of the top-level CG of the HCCE scheme. Hence, similar to $f_{CG}$ (see Equation 17), the fitness function is given by

$$f = \sqrt{E_{wa}} \cdot E_{win}^2 \cdot E_{sr}$$

which implies that the progress of unimodal evolution is directly comparable with the progress of the HCCE scheme. In the current set of experiments, a population of 400 individuals is evolved for 170 steps. The probabilities of applying crossover and mutation operators over the configuration of a cortical or a link agent are the same as for the HCCE scheme.

We performed six independent runs of the unimodal evolutionary process. The results of each process are shown in Figure 18. These results are directly comparable with the last row of Figure 15. Evidently, none of the ordinary evolutionary processes was successful. This is because ordinary evolution employs a single population with individuals encoding the overall composite solution, and additionally employs a single fitness function that is not able to address the role of each component to the system. These results highlight the unsuitability of unimodal evolution to design distributed structures consisting of autonomous components and, additionally, highlight the need for a specialized scheme able to consider explicitly the individual characteristics of substructures.
### 4.3.4 Comments

In the present set of experiments we have utilized three different evolutionary methods — namely, HCCE, ESP, and ordinary unimodal evolution — to address the design of the brainlike computational model. The results obtained are illustrated in Figures 15, 16, 17, and 18. The expected, best-fitness values of the methods are 1,178 for HCCE, 317 and 214 for the homogeneous and heterogeneous ESP, and 223 for unimodal evolution. Obviously, under this criterion, the HCCE procedure outperforms both ESP and unimodal processes. Even if a systematic study (investigating many population sizes as well as many mutation and crossover rates) is necessary to make conclusions about the effectiveness of these methods, current results indicate that HCCE is more suitable to address problems that require the special characteristics of substructures to be explored. In particular, even the best of the ESP and the unimodal results are not as good as the worst case of the HCCE. This is because the proposed coevolutionary scheme is able to evolve large distributed systems, enforcing successful cooperation among their autonomous components. Furthermore, our previous study [36] showed that the replication operator significantly facilitates the successful convergence of the composite coevolutionary process, because it conveys information from the higher to the lower levels of the hierarchy, in order to modulate and coordinate partial evolutionary processes.

Due to the embodiment of the cognitive system in the simulated robotic platform and the observation of robot performance on several tasks (each one testing a large number of simulation steps), all evolutionary processes demanded several hours to run for 170 evolutionary epochs. The experiments have been performed on a PC with an Intel Pentium 4 processor at 3.00 GHz and 512 Mb of RAM. Each HCCE run evolved for approximately 10 h, the ESP homogeneous and heterogeneous runs evolved for approximately 45 h, and unimodal evolution also evolved for approximately 10 h. This is illustrated graphically in Figure 19.

The distribution of processing time is explained by the number of composite solution assemblies evaluated by HCCE, ESP, and the unimodal scheme in each evolutionary epoch. Specifically, the HCCE scheme evaluates 400 assemblies, ESP evaluates 2,000 assemblies, and unimodal evolution evaluates 400 assemblies. Thus, it is reasonable that ESP needs considerably more processing time, because it inherently performs more evaluations (the individuals encoding the component structures have to participate in many composite assemblies, in order to obtain an average estimate of their quality). Alas, despite the increased amount of computational resources spent, the quality of the obtained results is rather poor.

Overall, from the aforementioned set of experiments, we conclude that HCCE is more effective than both ESP and ordinary unimodal evolution for designing distributed systems consisting of many complex and autonomous components. Moreover, it has been illustrated that HCCE efficiently utilizes the available computational resources, being at least as fast as the unimodal evolution and much faster than ESP.

### 4.4 Advancing the Model

The previous subsections have demonstrated how the agent-based coevolutionary framework facilitates the development of a computational model that mimics brain operation. Specifically, we have implemented a model that develops WM-like activation, being able to solve the same-side (SS) DR

![Figure 18. Graphical illustration of the progress of six different unimodal evolutionary procedures. Each plot shows the maximum fitness value of individuals in a generation, against evolutionary epochs (compare to the last row of Figure 15).](image-url)
task (a light source appears to the simulated robot, which has to turn toward the side of the source). Obviously, the complementary task can also be defined, namely, opposite-side (OS), implying that the simulated robot should turn toward the opposite side from the light source.

In Section 4.2 we demonstrated that the HCCE-based design mechanism can be employed to implement models exhibiting the SS response strategy. Additional experiments\(^9\) have shown that, following a similar approach, we can design models solving the OS delayed response task. In both cases, however, the models are developed with the inborn ability to respond in the desired way. This is a common characteristic for the vast majority of existing brain models (e.g., [7, 26, 70]). Unfortunately, this is different from what happens in nature, because animals are able to adopt different strategies during their life. Thus, the question now arises whether we can design a single computational system that is able to adopt both the SS and the OS response strategy during a lifetime. In each case, the adopted response strategy will be specified by properly located environmental reward signals, which is also the case with animals. Fortunately, as discussed in Section 2.6, the neural agent structure employed in the current study is able to support reinforcement learning procedures.

### 4.4.1 Model and Tasks

In the following we investigate the possibility of extending the SS model (described in Section 4.2), thus developing an improved system with learning abilities. The new composite model is illustrated in Figure 20. In order to ease the design procedure, we avoid designing the composite model from scratch. Particularly, the current experimental process keeps in their original formulation the components that are less involved in the reinforcement learning procedure [namely, the posterior parietal cortex (PPC), primary motor cortex (M1), and spinal cord (SC)]. The structures mostly involved in the learning process are the prefrontal and premotor cortices (PFC, PM) [47]. The cortical agent representing PFC was also present in our previous model, and it needs to be redesigned in order to accommodate run-time adaptation abilities. PM is a new module that needs to be designed from scratch. Both PFC and PM modules receive information related to the reward stimuli, adapting accordingly the motion orders passed to the lower levels of the motor hierarchy. An additional module is utilized to strengthen reward information, effectively modulating PFC and PM operation. This module could represent the ventral tegmental area (VTA), which guides learning in the neocortex [31].

#### 4.4.1.1 Learning the opposite-side strategy

The training process of the simulated robot is separated into \(T\) trials. Each trial includes one sample-response pair, testing the memorization of

---

\(^9\) They are not presented here, in view of their extensive similarity to the ones described in Section 4.2.
sample cues by the simulated robot (left or right side of light source appearance) and the expression of the correct DR.

During each trial, the robot is initialized to a predefined starting position in the top of the maze with a random direction in the range $[-85^\circ, -95^\circ]$. Each trial lasts for $M = 300$ simulation steps, and is separated into a sample phase and a response phase. In the sample phase, a light cue is presented on the left or right side of the simulated robot for 40 simulation steps. During the response phase, which lasts 200 simulation steps, the light source disappears, and the robot drives freely to the end of the corridor. At the cross point, the robot has to decide which way to turn. According to the OS training process, the response is considered correct if the robot turns to the opposite side from the light cue appearance. In the case of a correct response, the robot drives toward the goal position, where a reward signal is located. If the robot makes the wrong turn, it will drive to an area where no reward exists, indicating that the currently adopted strategy is not correct.

The learning of the OS response strategy is tested over $T = 12$ consecutive trials, and the goal of the robot is to collect the maximum amount of reinforcement. Six trials evaluate robot turning to the left, and six trials evaluate robot turning to the right. The success of the training process is evaluated by

$$E_T = \left( \sum_{T, \text{left}} M \right) \left( \sum_{T, \text{right}} M \right) \left( 1 - \sqrt{\frac{B}{2TM}} \right)^3$$  \hspace{1cm} (24)

The first factor seeks maximum reward stimuli when the robot's correct response is the left side; the second seeks maximum reward when the correct response is the right side. The higher the reward the robot has received, the more successful was the reinforcement training process. The last factor minimizes the number of robot bumps on the walls.

Additionally, HCCE employs partial design criteria highlighting the special roles of agent components in the model. In particular, we explore the development of distinct WM-like activation patterns on PFC. Two different states $a$ and $b$ are defined, associated with the two possible sides of light source appearance. For each state, separate activation averages $p_{a}^t$ and $p_{b}^t$ are computed, with $t$ identifying PFC excitatory neurons. Similar to the procedure described in Section 4.2.1, for
each trial we consider neural activation only during simulation steps 41–250 (the same applies also for Equations 26 and 27 below). The formation of WM patterns is evaluated by

\[ E_{wm} = \min \left\{ \sum_{l,j,l_l > l_j} \left( p_{a,l} - p_{a,l_j} \right), \sum_{l,j,l_l > l_j} \left( p_{a,l} - p_{a,l_j} \right) \right\} \cdot \left( \frac{v_a}{m_a} + \frac{v_b}{m_b} \right) \]  

(25)

where \( m_a, v_a, m_b, \) and \( v_b \) are the means and variances of average activation in states \( a \) and \( b \). The first factor enforces consistent activation, while the second supports the development of separate activation patterns for each state \( a \) and \( b \).

Another criterion addresses the development of different planning orders in the PM component that should be passed to MI. Two different states \( r \) and \( f \) are defined, associated with the commands for right and for left turning. For each state, separate activation averages \( p_{k,r} \) and \( p_{k,f} \) are computed, with \( k \) identifying PM excitatory neurons. The successful development of distinct activation patterns for right and left turning is measured by

\[ E_r = \min \left\{ \sum_{k,j,k_j > k_j} \left( p_{k,r} - p_{k_j,r} \right), \sum_{k,j,k_j > k_j} \left( p_{k,f} - p_{k_j,f} \right) \right\} \cdot \left( \frac{v_r}{m_r} + \frac{v_f}{m_f} \right) \]  

(26)

The explanation of the measure is similar to that of Equation 25.

Finally, an additional criterion develops the difference between the activation patterns of the VTA structure, related to the two possible locations of the reward signal. Two different states \( x \) and \( y \) are defined, associated with the right and the left reward location. For each state, separate activation averages \( p_{x,r} \) and \( p_{y,f} \) are computed, with \( i \) identifying VTA neurons. This is described by

\[ E_r = \min \left\{ \sum_{l,j,l_l > l_j} \left( p_{x,r} - p_{x,l_j} \right), \sum_{l,j,l_l > l_j} \left( p_{y,f} - p_{y,l_j} \right) \right\} \cdot \left( \frac{v_x}{m_x} + \frac{v_y}{m_y} \right) \]  

(27)

The explanation of the measure is similar to that of Equation 25.

**4.4.1.2. Learning the same-side strategy** Just after testing the performance of the simulated robot on learning the OS strategy, all agent components are reinitialized, and we test now if the robot is able to adopt the SS response strategy. In that case, reward stimuli are re-localized, reinforcing delayed responses that are in accordance with the SS strategy. The process is again separated into \( T \) trials, and it is very similar to the one described above for the case of OS training. Specifically, each trial includes two sample-response pairs, but this time, due to the SS strategy, the reward stimulus is located on the same side that the light cue appeared on. The measure evaluating the adoption of the SS strategy by the robot is the as the one described in Equation 24. Furthermore, additional evaluation measures similar to those described in Equations 25, 26, and 27 highlight the roles of PFC, PM, and VTA structures in the composite model.

Overall, we employ two different sets of measures, namely \( \{ E_{wm,con}, E_{con,con}, E_{con,con}, E_{con,con} \} \) and \( \{ E_{wm,con}, E_{con,con}, E_{con,con}, E_{con,con} \} \), evaluating the ability of the simulated robot to adopt either the OS or the SS strategy after following the reward-based training processes, and additionally evaluating the distinct role of substructures in the composite model.

**4.4.2 Coevolutionary Experimental Protocol**

We turn now to the design of the model by means of the HCCE scheme. The hierarchical coevolutionary process that redesigns and extends the preexisting model is illustrated in Figure 21.
The species below CG1 and CG3 are depicted with dashed lines, highlighting that the original structures of these components are kept in the current procedure (they have been designed in the experiment described in Section 4.2). Thus, the species depicted with dashed lines are not evolved.

According to the current experimental scenario, two learning procedures are tested to validate the adoption of the OS and SS response strategies. Partial fitness functions should additionally highlight the specialized role of each component in the model. Specifically, the fitness function employed for the evolution of CG2 and its lower-level species evaluates the success of OS and SS learning procedures and the development of WM activity in PFC. Following the formulation introduced in Equations 2 and 3, this is described mathematically by

$$f_{CG2} = f_{CG2,1} \cdot f_{CG2,2} \quad \text{with} \quad f_{CG2,1}^k = E_{w,os} \cdot E_{r,os}, \quad f_{CG2,2}^k = E_{w,ss} \cdot E_{r,ss}$$ (28)

where $k$ represents each membership of an individual in a proposed solution.

The agent structures grouped under CG4 serve the success in OS and SS learning and the development of the appropriate higher level motion commands in PM. Thus, the fitness function employed for the evolution of CG4 is

$$f_{CG4} = f_{CG4,1} \cdot f_{CG4,2} \quad \text{with} \quad f_{CG4,1}^k = E_{s,os} \cdot E_{r,os}, \quad f_{CG4,2}^k = E_{s,ss} \cdot E_{r,ss}$$ (29)

where $k$ is as above.

The agent structures grouped under CG5 support OS and SS learning and the development of different reward patterns in VTA. Thus, the fitness function employed for the evolution of CG5 is

$$f_{CG5} = f_{CG5,1} \cdot f_{CG5,2} \quad \text{with} \quad f_{CG5,1}^k = E_{s,os} \cdot E_{r,os}, \quad f_{CG5,2}^k = E_{s,ss} \cdot E_{r,ss}$$ (30)

where $k$ is as above.

Finally, the top-level species CG6 integrates partial configurations in a composite model, enforcing the cooperation of substructures. Particularly, it facilitates the accomplishment of both learning processes, and additionally highlights the role of each cortical agent in the model. The fitness function employed for the evolution of CG6 is defined accordingly by

$$f_{CG6} = f_{CG6,1} \cdot f_{CG6,2} \quad \text{with} \quad f_{CG6,1}^k = E_{r,os} \cdot \sqrt{E_{w,os} \cdot E_{r,os}},$$

$$f_{CG6,2}^k = E_{r,ss} \cdot \sqrt{E_{w,ss} \cdot E_{r,ss}},$$ (31)

where $k$ is as above.
The hierarchical coevolutionary process employed populations of 200 individuals for all PS species, 300 individuals for $CG_2$, $CG_4$, and $CG_5$, and 400 individuals for $CG_6$. The mutation and crossover rates were the same as those presented in Section 4.2.2. The elitist strategy also applies here, in order to support the success of the coevolutionary procedure.

4.4.3 Results

After 70 evolutionary epochs, the HCCE process converged successfully. Sample results of the simulated robot adopting the OS and SS strategies are illustrated in Figures 22 and 23. In both cases, the responses of the robot in the first two trials (columns 2 and 3) are incorrect. However, in the third trial (column 4), the robot tries another strategy, which is successful, and then continues it for all the remaining trials. Obviously, HCCE has successfully redesigned the previous computational structure, formulating an improved model with run-time strategy adaptation abilities.

In order to get a better idea about the effect of reinforcement signals on the performance of the simulated robot, we have tested the responses of the robot in the SS and OS tasks when (i) no reward is provided, (ii) only the right-side reward is provided, and (iii) only the left-side reward is provided. Each test includes 10 trials, with the light appearing interchangeably on the left and right sides. The observed robot behavior is as follows: (i) In the case of no reward, for both the SS and the OS task, the robot gives an always-right response for the first six trials, while in the seventh trial it gives a response to the left, continuing with an OS response for trials 8–10. In experiment (ii) we test the case of right-side-only reward. In the OS task, the robot starts with two exploratory trials; then it continues with two always-left trials and three correct OS trials. However, since the left reward signal is missing, the robot cannot stabilize to the correct OS strategy, and switches again to the always-left response for one trial, and again to the correct OS response for two trials. Additionally, during the SS task, the robot starts with an always-right response that is switched in the fourth trial to an always-left response. This changes to an OS response on the seventh trial, and an always-left response on the tenth trial. In experiment (iii) we test the case of left-side-only reward. The response pattern of the robot is the same for the OS and the SS tasks. It starts with an always-right response, and switches to an OS response on the seventh trial.

According to the results described above, the robot has a tendency to respond following either the OS, or the always-right, or the always-left strategy. In other words, when one or two reward signals are missing, there was no case giving an SS response pattern. Intuitively, the robot gives repeated responses to the same side, trying to identify which side of the light cue will provide a reward. In the case that no reward is provided after some trials, the robot switches to OS, which seems to be used as a default strategy (similar behavioral preferences have also been studied in [29]).

![Figure 22](image)

Figure 22. A sample result of simulated robot performance in the same-side response task. The first column illustrates sample cues. The rest of the columns (2–7) demonstrate the response of the robot in consecutive trials. The robot always starts at the top of the maze. R depicts the side of the reward. The first row illustrates robot responses when the light sample appears to the right. In a similar way, the second row illustrates robot responses when the light sample appears to the left.
Figure 23. A sample result of robot performance in the opposite-side response task. The first column illustrates sample cues. The rest of the columns (2–7) demonstrate the response of the robot in consecutive trials, with the simulated robot always starting at the top of the maze. R depicts the side of the reward. The first row illustrates robot responses when the light sample appears to the right. In a similar way, the second row illustrates robot responses when the light sample appears to the left.

The abovementioned experiments highlight the importance of the reward signal, which helps the robot to correctly adopt both the OS and the SS strategy.

In summary, the present experimental procedure demonstrates the power of the agent-based coevolutionary framework to redesign the model of Section 4.2, in order to enhance its behavioral capabilities. The distributed HCCE-based design mechanism is particularly appropriate to enforce cooperation among new and preexisting components. It is noted that the ability of partial redesign is an important characteristic for an effective computational framework that aims to support long-term design procedures, such as brain modeling.

4.5 Internal Dynamics—Emergent Characteristics

The current work introduces a new engineering perspective in designing brain-inspired cognitive systems. In particular, we propose a computational framework that follows an agent-based representation of brain areas, and an HCCE-based optimization mechanism for specifying the details of the model. The coevolutionary scheme employs separate fitness criteria to evolve each component of the model, thus being able to address its specialized characteristics. This HCCE feature significantly supports the design of large cognitive systems, because practically, it is very difficult to handle them in a compact form (i.e., ignoring information regarding their components) [53]. In other words, it is very difficult to obtain partial behaviors in a pure emergent way. Especially when we are dealing with complex structures, this is unlikely to happen, due to the very large number of parameters that have to be explored. In the current study, we exploit neuroscientific findings addressing the role of brain areas, in order to specify fitness measures that enforce the development of brain-area-like functionalities by the components of the model. Therefore, our work concentrates on the engineering of the models and how we can systematically map on them brainlike characteristics. However, the implemented systems have developed additional brainlike features that are not prespecified by the designer. These are summarized below, concentrating our discussion on the features appearing consistently in all successful solutions obtained from independent coevolutionary runs.

We start from the component representing SC, noting that oscillatory neural activity has emerged in its internal dynamics. This is clearly shown by letting the simulated robot move in a simplified freespace environment without obstacles. The activity of the two motor neuron pairs (each one responsible for driving one wheel; see Section 4.1) is demonstrated in Figure 24. This oscillatory activity is properly modulated when the robot drives in an environment with obstacles, avoiding collisions. The activity of motor neuron pairs for the case of wall avoidance navigation is shown in Figure 25. We note that SC takes input from the M1 agent. Therefore, the oscillatory dynamics emerging in our model seems to be very effective in accepting and executing higher-level motion orders. This
is also the case with natural systems, in that the vast majority of animals have adopted oscillatory motion mechanisms.

A time-structured neural activation is also observed in M1 neurons. Specifically, Figure 26 illustrates the activation of M1 excitatory neurons during wall avoidance navigation. We can easily observe the existence of temporally repeated activation patterns, and the formulation of neuron groups having synergistic activation. Similar collective organization (i.e., temporal structure and grouping) has been also observed in the rat brain [8, 43, 60]. Additionally, we have investigated the role of neural activation patterns and how they affect the behavior of the simulated robot during navigation. Our findings are depicted in Figure 27. Obviously, different activation patterns correspond to left-directed and right-directed robot maneuvering. In other words, neurons with direction-selective activations have been implemented in the M1 agent. This is similar to the functionality of biological neurons in the motor cortex having motion-direction correlated activity [17].

The observed neuron groups in M1 are not only responsible for driving the robot in a wall avoidance mode, but additionally, they operate as input gates accepting higher-level orders for either left- or right-directed motion. This is demonstrated in Figures 28a and 29a, showing the activation of an M1 neuron with left-direction selective activation, during multiple SS and OS responses. Obviously, this neuron is mostly active when a left response is decided by the higher-level modules. However, it is also occasionally active when a right response is decided. This is because M1 has to execute higher-level orders, being at the same time responsible for avoiding bumps on the walls. Thus, when a right response is planned by PFC and the robot senses a proximal wall at its right side, the underlying M1 neuron fires in order to avoid a crash, instantly directing the robot to the left. A similar activation pattern has been observed for M1 neurons with right-direction selective activation.

Additionally, two different WM patterns are formulated in PFC, encoding higher-level orders for a left or right response. The activation of a PFC neuron encoding right-directed responses for both the SS and OS training is illustrated in Figures 28(b), and 29(b). The PFC module also contains neurons following a complementary activation pattern, encoding response orders to the left. Additionally, the agent representing PM shows neural activation patterns similar to PFC, but with a reduced level
of maximum activation. Both the activation patterns in PFC and in PM are developed due to the objectives of the coevolutionary design procedure. However, the agent representing PPC is free to develop any desired behavior that supports WM-task accomplishment (see Equation 16 in Section 4.2).

After investigating PPC's role, we found that its activity is significantly correlated with PFC WM activation. This emergent functionality is in accordance with PPC's role as reported in brain imaging studies (e.g., [10]). In particular, PPC supports WM by making all the appropriate comparisons and associative recalls to determine the pattern to be memorized. In the current model, this brain-like functionality emerges due to (i) the brain-inspired position of the PPC agent in the whole system (i.e., PPC mediates the projection of environmental information on PFC), and (ii) the role assigned to PPC. Therefore, despite the fact that PPC's role is not constrained by the human designer, the HCCE-based design methodology forces the underlying agent to develop rather natural performance.

Finally, we would like to comment on the Hebbian rules assigned to the synapses of cortical agents, specifying the run-time dynamics of the model. We have examined successful cortical agent configurations obtained from different coevolutionary runs, without however identifying any consistently

Figure 25. The oscillatory activation of the (a) left, and (b) right spinal cord motor neurons during wall avoidance navigation. Neural dynamics are properly modified according to the sensory input, driving the simulated robot without wall bumps.

Figure 26. The activation of M1 agent neurons during wall avoidance navigation. Each line corresponds to one of the 16 neurons. The activity is shown in grayscale, with black corresponding to full activity. Two patterns of repeated neural activity are easily identified, highlighted with solid and dashed rectangles.
Figure 27. Six pairs of (i) robot navigation paths and (ii) the corresponding neural activities in M1 agent. The first row demonstrates leftward robot turnings, while the second demonstrates rightward turnings. Obviously, M1 neurons have developed direction-selective activity.

appearing rule pattern (i.e., the same synapse type is assigned a different rule in each particular solution). This means that it is the combination of Hebbian rules that actually matters, rather than a rule assigned in a specific set of synapses. Unfortunately, as is discussed in [42], it is very difficult to study the combination of many different Hebbian rules. Therefore, it is currently not feasible to formulate concrete conclusions about their interaction and the dynamics shaped internally in the model.

5 Conclusions and Future Work

The work presented here constitutes a first attempt toward a rigorous computational framework that facilitates the implementation of brainlike cognitive systems for robotic applications. The results obtained attest to its validity and effectiveness in modeling partial brain areas and replicating biological behaviors.

The proposed computational framework makes a twofold contribution. First, neural agents are utilized to represent brain areas and their connectivity. The agent-based representation is in accordance with the distributed nature of the brain prototype. Due to the inherent autonomy of agents, the proposed representation supports problem decomposition into small, tractable, and progressively solved tasks whose results are easily integrated into larger structures. Second, a distributed optimization method is employed to design the composite brain-inspired model. We have introduced a novel hierarchical cooperative coevolutionary (HCCE) scheme that is capable of designing the autonomous components of the model, addressing both their specialized characteristics and their coupling to a single complex system. In summary, the proposed agent-based coevolutionary framework facilitates:

- the design of distributed brain-inspired systems, explicitly addressing the role of each component in the model;
- the computational replication of neuroscientific findings from lesion studies, as a means to support the reliability of the model;
the gradual advancement of the model, being able to integrate new components and additionally redesign some of the previously existing ones.

For comparative purposes, we have also employed enforced subpopulation coevolution and ordinary unimodal evolution to approach the current modeling problems, without however either of them being successful. It has been experimentally demonstrated that HCCE is the only effective method (of the three tested) to evolve systems consisting of many components, investigating the functionality of the composite structure in different operating conditions. Particularly, the capability of HCCE to coevolve a large number of system components makes it one of the best-suited methods to successfully tackle the implementation of complex brain models. We should note here that the hierarchical formulation of the coevolutionary scheme does not imply that the model should perform in a hierarchical mode. The performance of partial structures can be either hierarchical or completely parallel. Hence, the coevolutionary design mechanism does not impose any constraints on simulating the connectivity of brain areas.

We would like to note that the specification of each agent role by using separate fitness criteria is analogous to contemporary brain-modeling approaches employing neural networks with well-known internal dynamics (classifiers, associators, etc.) to represent brain areas [3, 21, 24, 67]. Thus, both the proposed and the preexisting approaches proceed by constraining the brain model design process. Still, we believe that our approach is more general, because the designer specifies only the desired

Figure 28. The activations over time of (a) an M1 neuron with left-direction selective firing, and (b) a PFC neuron encoding WM and directing the robot to the right, during the SS training process depicted in Figure 22. In both cases, six trials are illustrated, separated by dashed lines.
outcome, rather than the specific computational details of the model. As a result, neural agents are free to develop any kind of internal dynamics necessary for the model to be functional.

The current study shows that the proposed computational framework assists implemented models to acquire additional brainlike characteristics that are not specified by the human designer. More specifically, the proposed approach facilitates modeling of (i) the training of animals and (ii) the effect that cortical lesions have on animal behavior. Therefore, following the agent-based coevolutionary framework, existing data from brain imaging experiments can be systematically exploited to support brain modeling efforts. The more neuroscientific data the model is able to replicate, the more reliable the roles of agent components in the composite model become. In the future, we aim at implementing brainlike systems that accomplish many different tasks following many different training procedures and additionally replicate various lesion effects on the modeled cortical areas.

Finally, the proposed coevolutionary approach can also be utilized in contexts different from brain modeling, investigating systems consisting of any kind of components (to design complex modular mechanical structures, teams of cooperating robots, etc.). Thus, HCCE can potentially be used as a general purpose too for modeling distributed systems.

Acknowledgments
The work presented in this article has been partly supported by the European Commission-funded project MATHESIS, under contract IST-027574.
References


