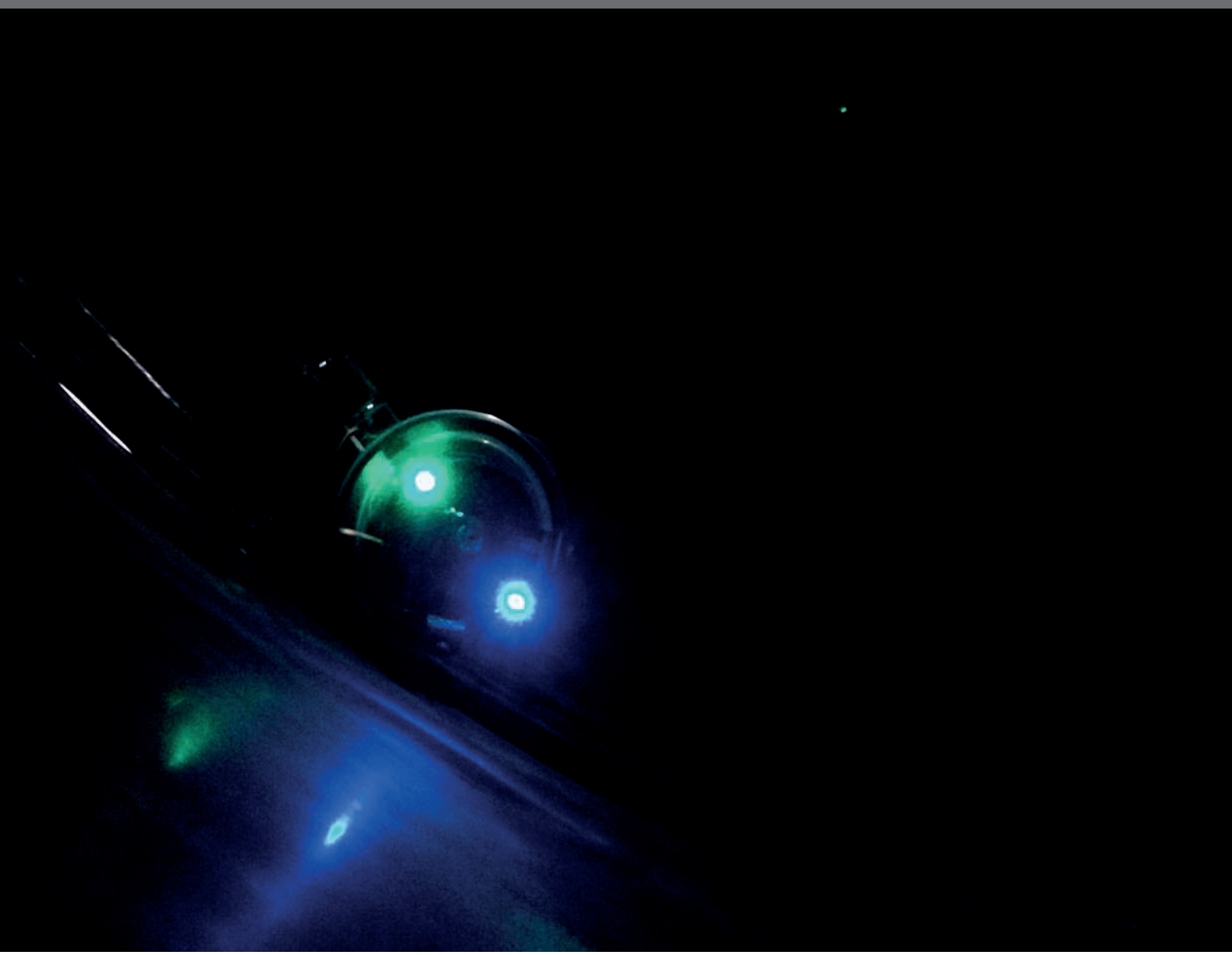
EVOLVABILITY, ENVIRONMENTS, EMBODIMENT, & EMERGENCE IN ROBOTICS

EDITED BY: John H. Long Jr., Eric Aaron and Stéphane Doncieux
PUBLISHED IN: Frontiers in Robotics and AI



frontiers Research Topics

frontiers

ISSN 1664-8714
ISBN 978-2-88945-622-2
DOI 10.3389/978-2-88945-622-2

About Frontiers

Frontiers is more than just an open-access publisher of scholarly articles: it is a pioneering approach to the world of academia, radically improving the way scholarly research is managed. The grand vision of Frontiers is a world where all people have an equal opportunity to seek, share and generate knowledge. Frontiers provides immediate and permanent online open access to all its publications, but this alone is not enough to realize our grand goals.

Frontiers Journal Series

The Frontiers Journal Series is a multi-tier and interdisciplinary set of open-access, online journals, promising a paradigm shift from the current review, selection and dissemination processes in academic publishing. All Frontiers journals are driven by researchers for researchers; therefore, they constitute a service to the scholarly community. At the same time, the Frontiers Journal Series operates on a revolutionary invention, the tiered publishing system, initially addressing specific communities of scholars, and gradually climbing up to broader public understanding, thus serving the interests of the lay society, too.

Dedication to Quality

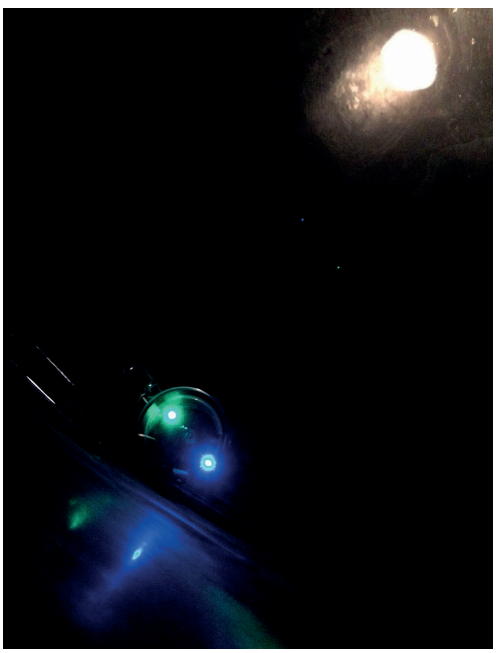
Each Frontiers article is a landmark of the highest quality, thanks to genuinely collaborative interactions between authors and review editors, who include some of the world's best academicians. Research must be certified by peers before entering a stream of knowledge that may eventually reach the public - and shape society; therefore, Frontiers only applies the most rigorous and unbiased reviews.
Frontiers revolutionizes research publishing by freely delivering the most outstanding research, evaluated with no bias from both the academic and social point of view. By applying the most advanced information technologies, Frontiers is catapulting scholarly publishing into a new generation.

What are Frontiers Research Topics?

Frontiers Research Topics are very popular trademarks of the Frontiers Journals Series: they are collections of at least ten articles, all centered on a particular subject. With their unique mix of varied contributions from Original Research to Review Articles, Frontiers Research Topics unify the most influential researchers, the latest key findings and historical advances in a hot research area! Find out more on how to host your own Frontiers Research Topic or contribute to one as an author by contacting the Frontiers Editorial Office: researchtopics@frontiersin.org

EVOLVABILITY, ENVIRONMENTS, EMBODIMENT, & EMERGENCE IN ROBOTICS

Topic Editors:
John H. Long Jr., Vassar College, United States
Eric Aaron, Colby College, United States
Stéphane Doncieux, Institut des Systèmes Intelligents et de Robotique (ISIR), Sorbonne University, CNRS, France



Tadro, a behaviorally autonomous swimming robot, navigates a circular pool to detect and harvest light from an overhead lamp. Lights on its bow (blue) and stern (green), used for motion capture, are reflected off the wall of the tank, while the overhead lamp is reflected on the water's surface. Tadros are but one example of embodied robots that are evolved to test scientific hypotheses.
Image: John Long.

Embodied and evolving systems — biological or robotic — are interacting networks of structure, function, information, and behavior. Understanding these complex systems is the goal of the research presented in this book. We address different questions and hypotheses about four essential topics in complex systems: evolvability, environments, embodiment, and emergence. Using a variety of approaches, we provide different perspectives on an overarching, unifying question: How can embodied and evolutionary robotics illuminate (1) principles underlying biological evolving systems and (2) general analytical frameworks for studying embodied evolving systems? The answer — model biological processes to operate, develop, and evolve situated, embodied robots.

Citation: Long, J. H., Aaron, E., Doncieux, S., eds. (2018). Evolvability, Environments, Embodiment, & Emergence in Robotics. Lausanne: Frontiers Media.
doi: 10.3389/978-2-88945-622-2

Table of Contents

frontiers in Robotics and AI

EDITORIAL
published: 31 August 2018
doi: 10.3389/frobt.2018.00103

Editorial: Evolvability, Environments, Embodiment & Emergence in Robotics

John H. Long Jr.[1,2]*, *Eric Aaron*[1,3,4] *and Stéphane Doncieux*[5]

[1] *Interdisciplinary Robotics Research Laboratory, Vassar College, Poughkeepsie, NY, United States,* [2] *Departments of Biology and Cognitive Science, Vassar College, Poughkeepsie, NY, United States,* [3] *Department of Computer Science, Vassar College, Poughkeepsie, NY, United States,* [4] *Department of Computer Science, Colby College, Waterville, ME, United States,* [5] *UMR 7222, ISIR, Sorbonne Universités, UPMC Univ Paris 06, Paris, France*

Keywords: evolvability, environments, embodiment, emergence, robustness, cognitive robotics, evolutionary robotics

Editorial on the Research Topic

Evolvability, Environments, Embodiment, & Emergence in Robotics

OPEN ACCESS

Edited by:
Geoff Nitschke,
University of Cape Town, South Africa

Reviewed by:
Heiko Hamann,
Universität zu Lübeck, Germany

***Correspondence:**
John H. Long Jr.
jolong@vassar.edu

Specialty section:
This article was submitted to Evolutionary Robotics, a section of the journal Frontiers in Robotics and AI

Received: *18 June 2018*
Accepted: *08 August 2018*
Published: *31 August 2018*

Citation:
Long JH Jr, Aaron E and Doncieux S (2018) Editorial: Evolvability, Environments, Embodiment & Emergence in Robotics. Front. Robot. AI 5:103. doi: 10.3389/frobt.2018.00103

We challenged researchers to grapple with four ideas and their interactions—evolvability, environments, embodiment, and emergence. These are complex drivers underlying the designs and actions of autonomous, mobile, and physical systems. How does a robot or robotic system come to have intelligent, goal-directed behavior? This question is, indeed, a grand challenge.

The articulation of a "grand challenge" frames a field's most important goals and the methods to achieve them. Grand challenges launched the *Frontiers in AI and Robotics* specialty sections of evolutionary robotics (Eiben, 2014), virtual environments (Slater, 2014), and computational intelligence (Prokopenko, 2014). Eiben (2014), for example, suggests three grand challenges: (1) the automatic generation of novel, original robots that are surprising in their design to humans, (2) self-reproduction in physical robots, and (3) open-ended evolution of physical robots in an open-ended environment. Stressing the complex, integrated nature of physical robots operating in the physical world, Doncieux et al. (2015) elaborate a methodological approach, emphasizing experiments designed to test specific hypotheses. This methodology is the hallmark of the work presented in this research topic, with the demands of rigorous experimentation forcing investigators to operationalize ideas.

Robust, expository experiments are founded upon robust, expository models. Integrating the domains of cognition, motion, and time into a hybrid system, Aaron creates the Dynamical Intention-Hybrid Dynamical Cognitive Agent (DI-HDCA) modeling framework. Across a range of task environments, the embodied DI-HDCA agents demonstrate behavior that is emergent, the on-going result of everything from micro-cognitive processes to embodied physical actuation interacting in a physical world. Explicitly modeling both continuous dynamics of and discrete transitions between behaviors allows the investigator to probe how DI-HDCAs continuously search for real-time reactive, goal-directed solutions.

Finding optimal solutions in a search space is also the goal of evolutionary robotics (ER), which enables computational evolution to optimize over a fitness function. In biological evolution,

such fitness-guided search generates a large diversity of species across ecological niches; in ER, generating diverse solutions is not always emphasized, but diversity can impact both the evolvability of populations and the robustness of individual agents, promoting exploration of multiple behavior spaces and choice among multiple behaviors. Pugh et al. investigate quality-diversity (QD) algorithms, which aim to discover all possibilities by rewarding the evolution of novelty. With case studies of robot maze navigation, they show how hybridized behavioral characterizations in QD algorithms may be key for advancing evolutionary exploration and, ultimately, evolvability.

While evolvability has many definitions, ranging from current adaptability to future capacity for innovation (Pigliucci, 2008), Lehman et al. argue that ER needs to focus on innovative creation. Creative potential is studied productively as a property of populations since they, not individuals, are the entities that evolve. As collections of related individuals, populations vary, and that variance provides the range of phenotypes upon which selection acts. But directional selection, which optimizes locally, reduces variance, slowing adaptation and evolvability. The authors show that divergent selection can generate variance within a population, increasing variance in the short term. Alternating between directional and divergent selection can mediate between local adaptation and global exploration.

Another potential way to increase evolvability is to find mechanisms that drive behavioral specialization within a population. Montanier et al. take on the challenge, investigating agents that can evolve specialized foraging behaviors for the acquisition of two different resources. Reproductive isolation, however it might be achieved, appears to be key to specialization. Thus, most importantly for ER, mating algorithms and scenarios should be treated carefully and justified, given their potential for creating and maintaining variance within a population. Mating and selection are separate evolutionary drivers.

Evolvability also depends on morphology. Cappelle et al. demonstrate that structural modularity also impacts the efficacy of evolution, when robust behavior is the goal of the search. Comparing modular and non-modular architectures, they found that robots with modular morphologies and controllers can more quickly adapt to new environments. Most promising is the connection of modularity to function: modules shaped by previous evolutionary history are predisposed to detect percepts in new combinations and new environments. Thus, evolved morphology that is modular, if present and preserved, endows a population with greater evolvability, as predicted by the Wankelmut benchmark (Schmickl et al., 2016).

Modularity, however, and evolvability by extension, need not be a direct target of selection. In a population of networks, for example, selection for a combination of performance and reduced connection costs evolves modularity (Clune et al., 2013). Such indirect evolution of modularity is tested for the first time in physical robots by Livingston et al.: They select for enhanced phototaxis of surface-swimmers, in which the genome encoded 60 weights of neural networks connecting photoresistors to motor outputs. With selection on behavior alone, over the course of 10 generations, the primary target of selection is network sparsity; modularity is a correlated evolutionary by-product. This work broadens our understanding of conditions under which modularity may evolve.

Evolution also depends on development. In addition to genetic processes, development introduces epigenetic operators that govern the mapping of genes into morphologies. Using physical robots, Brawer et al. create a developmental process for wiring sensors to motors. In one population, development is altered by wires' physical interactions; in another, those physical interactions are avoided. From identical starting points, these two different epigenetic operators guide different evolutionary responses, changes over generational time that are mediated by the epigenetic process of building working physical robots. This is the first demonstration employing physical robots to show that epigenetic operators can be created and used to complicate, in explicit ways, evolutionary search.

Let us return to our grand challenge: How does a robot or robotic system come to have intelligent, goal-directed behavior? This research topic identifies key processes: (1) the principled emergence of goal-directed behavior from a hierarchy of dynamical processes (Aaron); (2) opportunities for enhanced evolvability and robustness from selection for diversity in populations (Lehman et al.; Pugh et al.); (3) the evolution of specialized behavior from reproductive isolation (Montanier et al.); (4) the evolution of robust behavior from modular systems, which may be evolved indirectly (Capelle et al.; Livingston et al.); and (5) the creation of phenotypic variation and enriched evolutionary possibilities from epigenetic developmental processes (Brawer et al.).

The intersection and interaction of these mechanisms provides ample opportunity to explore the evolution of intelligent behavior. The search continues for principled approaches for their integration in physically embodied systems, biological, or computational.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct and intellectual contribution to the work, and approved it for publication.

FUNDING

JL was funded by National Science Foundation, INSPIRE, Special Projects (grant no. 1344227).

ACKNOWLEDGMENTS

We appreciate the work of the editorial and production staff of *Frontiers in Robotics and AI* and Frontiers Research Topics who encouraged us to undertake this project and helped with its implementation. We also acknowledge the work of the authors, editors, and reviewers who made this project possible.

REFERENCES

Clune, J., Mouret, J. B., and Lipson, H. (2013). The evolutionary origins of modularity. *Proc. R. Soc. B* 280:20122863. doi: 10.1098/rspb.2012.2863

Doncieux, S., Bredeche, N., Mouret, J. B., and Eiben, A. E. G. (2015). Evolutionary robotics: what, why, and where to. *Front. Robot. AI* 2:4. doi: 10.3389/frobt.2015.00004

Eiben, A. E. (2014). Grand challenges for evolutionary robotics. *Front. Robot. AI* 1:4. doi: 10.3389/frobt.2014.00004

Pigliucci, M. (2008). Is evolvability evolvable? *Nat. Revi. Genet.* 9, 75–82. doi: 10.1038/nrg2278

Prokopenko, M. (2014). Grand challenges for computational intelligence. *Front. Robot. AI* 1:2. doi: 10.3389/frobt.2014.00002

Schmickl, T., Zahadat, P., and Hamann, H. (2016). *Sooner Than Expected: Hitting the Wall of Complexity in Evolution. arXiv* [preprint]. Available online at: https://arxiv.org/abs/1609.07722 (Accessed July 12, 2018).

Slater, M. (2014). Grand challenges in virtual environments. *Front. Robot. AI* 1:3. doi: 10.3389/frobt.2014.00003

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 17 November 2016
doi: 10.3389/frobt.2016.00066

Dynamical Intention: Integrated Intelligence Modeling for Goal-Directed Embodied Agents

Eric Aaron[1,2]*

[1] Department of Computer Science, Vassar College, Poughkeepsie, NY, USA, [2] Interdisciplinary Robotics Research Laboratory, Vassar College, Poughkeepsie, NY, USA

Intelligent embodied robots are integrated systems: as they move continuously through their environments, executing behaviors and carrying out tasks, components for low-level and high-level intelligence are integrated in the robot's cognitive system, and cognitive and physical processes combine to create their behavior. For a modeling framework to enable the design and analysis of such integrated intelligence, the underlying representations in the design of the robot should be dynamically sensitive, capable of reflecting both continuous motion and micro-cognitive influences, while also directly representing the necessary beliefs and intentions for goal-directed behavior. In this paper, a *dynamical intention*-based modeling framework is presented that satisfies these criteria, along with a *hybrid dynamical cognitive agent* (*HDCA*) framework for employing dynamical intentions in embodied agents. This dynamical intention-HDCA (*DI-HDCA*) modeling framework is a fusion of concepts from spreading activation networks, hybrid dynamical system models, and the *BDI* (belief–desire–intention) theory of goal-directed reasoning, adapted and employed unconventionally to meet entailments of environment and embodiment. The paper presents two kinds of autonomous agent learning results that demonstrate dynamical intentions and the multi-faceted integration they enable in embodied robots: with a simulated service robot in a grid-world office environment, reactive-level learning minimizes reliance on deliberative-level intelligence, enabling task sequencing and action selection to be distributed over both deliberative and reactive levels; and with a simulated game of Tag, the cognitive–physical integration of an autonomous agent enables the straightforward learning of a user-specified strategy during gameplay, without interruption to the game. In addition, the paper argues that dynamical intentions are consistent with cognitive theory underlying goal-directed behavior, and that DI-HDCA modeling may facilitate the study of emergent behaviors in embodied agents.

Keywords: intelligence modeling, learning, embodiment, hybrid systems, hybrid dynamical systems, machine learning, action selection, cognitive robotics

OPEN ACCESS

Edited by:
John Rieffel,
Union College, USA

Reviewed by:
José Antonio Becerra Permuy,
University of A Coruña, Spain
Michael Spranger,
Sony Computer Science Laboratories, Japan

***Correspondence:**
Eric Aaron
eaaron@cs.vassar.edu

Specialty section:
This article was submitted to Evolutionary Robotics, a section of the journal Frontiers in Robotics and AI

Received: *20 April 2016*
Accepted: *19 October 2016*
Published: *17 November 2016*

Citation:
Aaron E (2016) Dynamical Intention: Integrated Intelligence Modeling for Goal-Directed Embodied Agents. Front. Robot. AI 3:66. doi: 10.3389/frobt.2016.00066

1. INTRODUCTION

Embodied robots can encompass everything from low-level motor control to navigation, goal-directed behavior and high-level cognition in one complex, cognitive–physical system. Accordingly, when considering modeling frameworks for the design, development, and deeper understanding of such robots and their behaviors, there are many desired criteria and required constraints for

their models. This paper presents one such framework, anchored by *dynamical intention* modeling (Aaron and Admoni, 2010; Aaron et al., 2011) to represent cognitive elements underlying goal-directed behavior in embodied robots. With dynamical intention modeling and the accompanying *hybrid dynamical cognitive agent* (*HDCA*) framework, essential components that are often treated separately – including reactive and deliberative intelligence, and cognitive and physical behaviors – are unified in a modeling framework that supports high-level behavioral design, low-level cognitive and physical representations, and machine learning methods for integrated, autonomous learning in response to robots' environments.

Dynamical intention modeling and the HDCA framework for integrated dynamical intelligence are influenced by several observations about models of intelligent embodied agents, biological and robotic, in dynamic environments:

- Embodied agents are integrated systems, complete autonomous agents embedded in an environment (Pfeifer and Bongard, 2006). Their high-level cognitive intelligence, low-level cognitive intelligence, and physical actions and behaviors are essential system components, and they should be modeled and analyzed together, reflecting their integration.
- Goal-directed behavior of embodied agents moving through their environments is necessarily the result of the agents' integration across cognitive and physical components. For models to better support both production and analysis of goal-directed behavior, the relevant cognitive and physical components should be integrated in the model.
- In dynamic, unpredictable environments with arbitrary asynchrony, agents should be capable of appropriately dynamic responses and learning. If the environment cannot be known *a priori*, then ideally, models would not impose *a priori* restrictions on the granularity of possible responses in the environment. Similarly, because embodied agents are sensibly modeled as moving continuously through space and time, models should ideally support continuous space and time representations, without pre-imposed discretizations.
- Typically, models allowing only low-level representations do not effectively extend to high-level representations: for example, models that describe only kinematics of leg movement do not extend to pathfinding on large maps, and cognitive models describing only subsymbolic processes do not extend to representations of intentions guiding goal-directed planning.
- Conventional AI models of goal-directed behavior are frequently founded on high-level propositional representations, such as the goals, beliefs, and intentions of agents carrying out planning for the behavior [e.g., Georgeff and Lansky (1987)]. These representations do not readily support integration with low-level, continuous-time processes; they do not readily support cognitive–physical integration without imposing restrictions that may be ill-suited in unpredictable environments. Ideally, intelligence models would represent cognitive elements such as beliefs and intentions in a framework consistent with agents as integrated systems.

For the design and analysis of navigating, goal-directed embodied agents, a model of integrated intelligence would ideally represent and unify the cognitive and physical components – and interactions among them – underlying robust behavior in unpredictably dynamic environments. This paper presents the dynamical intention-HDCA (*DI-HDCA*) framework for integrated dynamical intelligence models for embodied agents, discussing its background, specifications, and foundation for extensions. Two different kinds of dynamical intention-based integration are presented, reactive–deliberative integration and cognitive–physical integration, as are required for fully integrated embodied agents. Moreover, the paper conceptually contextualizes this modeling framework in specific motivations based on the roles of embodiment and environment in agent behavior.

The DI-HDCA framework fuses ideas from cognitive modeling and general system modeling in a new synthesis, often employing them unconventionally to support the requirements of embodied intelligence. For instance, the foundation of a DI-HDCA model is a finite-state machine that combines continuous and discrete dynamics in a *hybrid automaton* (Alur et al., 2000): states (*modes*) represent continuously evolving actions or behaviors described by systems of differential equations; each mode also has conditions governing when discrete transitions to other modes occur, and what discrete changes in system state occur as part of these transitions.

The dynamical intention framework underlying cognitive models is influenced by the *belief–desire–intention* (*BDI*) theory of practical reasoning and its many implementations [e.g., Georgeff and Lansky (1987) and successors], which established the effectiveness of BDI elements (beliefs, desires, and intentions) as a foundation for goal-directed intelligence. Unlike conventional BDI agents, however, dynamical intention models link BDI elements in a continuously evolving system inspired by *spreading activation networks* (Collins and Loftus, 1975; Maes, 1989). Each BDI element in this dynamical intention framework is represented by an activation value indicating its salience "in mind" (e.g., intensity of a commitment to an intention, intensity of a belief). The continuous evolution of these cognitive activation values is governed by differential equations, with cognitive elements affecting the rates of change in activations of other cognitive elements, as described in sections 2.3 and 2.4. These dynamical cognitive representations can be employed for both low-level reactive intelligence and high-level deliberative planning (Aaron and Admoni, 2010), enabling integration of the two levels.

The particular physical motion of DI-HDCAs (i.e., navigation in dynamic environments) is not central to the DI-HDCA framework, as discussed in section 3.2, except that it too is governed by dynamical systems. This enables further integration: physical and cognitive components in DI-HDCAs are represented in the common language of differential equations, which is critical to the learning demonstrations in section 5.

These are the components of the general framework of dynamical intention and DI-HDCA modeling. The remainder of the paper further elaborates on these components and presents example DI-HDCAs, which illuminate general concepts and are employed in various proofs of concept.[1] For example, the paper

[1]The specific agents described in this paper are far from an exhaustive demonstration of the DI-HDCA modeling framework. To distinguish the general DI-HDCA

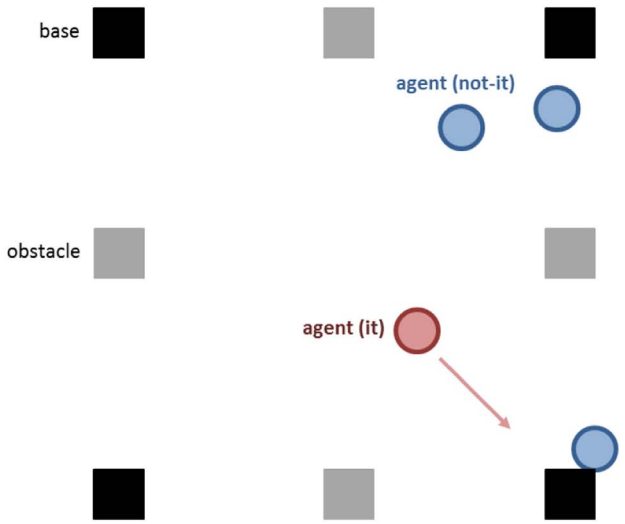


FIGURE 1 | Diagram of a Tag game environment, containing bases (darker squares), obstacles (lighter squares), and agents (circles) playing the game. Both kinds of Tag players are represented, one It player and three non-It players.

presents a simulated service robot in a grid-world office environment, for two kinds of demonstrations: how conventionally deliberative-level intelligence can be distributed over reactive-level processes in DI-HDCA models; and how new kinds of machine learning can be facilitated by dynamical intention representations. Indeed, with dynamical intention-based learning, the robot approximates deliberative rule-based performance with only reactive-level learning, minimizing reliance on deliberation and supporting dynamically responsive, adaptive behavior.

In addition, the paper presents experiments with DI-HDCAs as autonomous players in a real-time, human-interactive simulation of the child's game Tag. In Tag, a player designated as "It" attempts to touch ("tag") other players, who try to avoid being tagged. Safe locations called *bases* are in the Tag variant in this paper, as shown in **Figure 1**, so that players touching a base cannot be tagged. If a non-It player P_i does get tagged by It (call the It player P_j, distinct from P_i), then P_i becomes the new It, P_j is no longer It, and the game continues with players (including P_j) avoiding being tagged. This game is well suited for demonstrations of embodied intelligence: agents employ complex cognitive strategies while navigating in an unpredictably dynamic environment. Demonstrations from Tag games in this paper illustrate cognitive–physical integration in DI-HDCAs, with agents' jointly altering cognitive and physical performance to meet new specifications for their strategies without interrupting gameplay.

The contributions of this paper include:

- A broad description of dynamical intention and HDCA modeling, significantly expanding upon more narrowly focused presentations in Aaron and Admoni (2010) and Aaron et al. (2011). This includes the motivation and proper contextualization of DI-HDCA modeling as a response to entailments of environment and embodiment.
- A survey of previously described DI-HDCA learning methods and experimental results in both the Tag game and office environments mentioned above (Aaron and Admoni, 2010; Aaron et al., 2011), demonstrating the role of DI-HDCA modeling in adaptive integrated intelligence.
- Several new experimental results and substantially expanded analyses, including statistical analyses of data that were previously only qualitatively described.

This paper is the first comprehensive presentation of integrated intelligence for DI-HDCAs – encompassing physical-level components for motion and navigation and cognitive-level components for reactive and deliberative intelligence – and the first casting of DI-HDCA concepts that directly exposes the elements of embodied agency underlying those concepts. In addition, section 6 briefly discusses potential extensions of the present work in new contexts, including possibilities of verifying DI-HDCA models and applying the DI-HDCA modeling framework to study emergent properties of embodied intelligence.

2. THE DI-HDCA MODELING FRAMEWORK

The DI-HDCA modeling framework is specifically designed for, and constrained by, the demands of embodied autonomous intelligent agents navigating in dynamic environments. It is a synthesis of three concepts – BDI theory, spreading activation networks, and hybrid system models – that are employed unconventionally to enable formally specified yet broadly expressive agent models. This section presents the background and foundational ideas on which the DI-HDCA framework is based, analyzing the roles of embodiment and environment in modeling goal-directed agents, and then discussing cognitive modeling and hybrid system modeling in that context.

2.1. Environment

In principle, goal-directed agents need not be embodied [e.g., many BDI-based planning agents (Georgeff and Lansky, 1987)], but with or without embodiment, environment constrains what factors and features may be elements of effective agent models. Some problem solving agents operate in fully known, unchanging environments, which constrains the kinds of reasoning they need; for example, pathfinding problems can be solved prior to navigation for perfect performance. Other agents might operate in stationary environments that are not fully known in advance, so problems might not be solvable ahead of time, but information once discovered would not be changed, which could simplify machine learning or other adaptation needed in this environment. Such stationary environments are not realistic for the present context, however, so this paper restricts consideration to only dynamic and unpredictable environments.

For goal-directed behavior, agents must do some kind of planning or task sequencing, potentially employing propositional reasoning-based deliberative intelligence. As an environmental constraint, however, this paper additionally considers only environments in which deliberation is not sufficient, and some kind of

framework from specific agents, a phrase such as "in this paper" will formulaically be used to indicate specific focus.

reactive intelligence is also necessary. This reactivity requirement is not identical to the above criterion of "dynamic and unpredictable" – one could imagine environments in which deliberation sufficed for all unpredictable changes – but it is related.

In such environments, both reactive- and deliberative-level intelligence – and their combinations – are essential for goal-directed embodied robots. DI-HDCA modeling integrates deliberative and reactive intelligence through shared representations of cognitive elements: the same elements that support reflexive, reactive responses can also be employed for task sequencing and other conventionally deliberative-level intelligence. These shared, dynamically sensitive representations allow goal-directed reasoning to be distributed over both reactive and deliberative levels; the particular agent models in section 4 exemplify this distributed approach. Thus, DI-HDCA modeling does not deny deliberation, but it can minimize *reliance* on deliberation for more robustly responsive and adaptive agents.

2.2. Embodiment

Section 2.1 noted that an agent's environment could be incompletely known or unknowable, but for real-world robotics, one might potentially instead view the *embodiment* of the robot as the primary factor introducing such unpredictability: from dirt on a floor that affects a wheel's traction to moving obstacles (e.g., people) in hallways navigated by service robots, embodiment seems critical to why embodied robots need to respond and adapt at unpredicted times, to unpredicted situations.

Indeed, in a real-world environment for a robot, unpredictability is general, but that may not be strictly due to embodiment. If embodiment is considered separate from real-world constraints, it is imaginable *in theory* that a goal-directed embodied agent and its world might be fully deterministic and known in advance. This may seem laughably implausible to anyone who has worked with real robots, but in principle, it seems that unpredictability need not follow from embodiment alone.

Similarly, it might initially seem that reasons for continuum-based modeling of time and space – to represent continuous agent motion through space, and through time – are due to attributes of and constraints from the environment. Indeed, one could assert that continuous time and space are environmental properties: once unpredictability and the need for reactive responses are part of the environment, continuous time and space representations are then needed to fully represent the environment. It is not clear, however, that the environment would actually need to be *fully* represented for successful goal-directed behavior by a *non-embodied* agent. Perhaps the needed reactivity for a non-embodied agent could be achieved with a discretized time and space model, with limited granularity of representation; the asynchrony in the environment could be arbitrary, but perhaps that complexity need not be imposed in full upon the agent model.

DI-HDCA models do represent continuous space and time, however, with embodiment rather than environment as the practical motivation. Conventionally, real-world embodied systems are modeled as moving continuously through space, often by differential equations. Because these continuous representations are well established as useful for modeling, they have been adopted for DI-HDCA models.

The effects of this design decision pervade the DI-HDCA modeling framework: because DI-HDCA models should be integrated, and continuous time and space representations are useful, added entailments arise. A navigation model sensitive to continuous time variations is needed. Reactivity should be modeled on a continuous-time scale, for integration with continuous-modeled motion. The cognitive model should thus also be modeled with real-time dynamics, for sensitivity to real-time changes in the environment. Then, as cognitive model elements are real-time dynamic parts of the environment of other cognitive elements (e.g., beliefs are parts of the cognitive environment that affects intentions), and cognitive elements are sensitive to real-time environmental variations, the cognitive model should represent micro-cognitive variations and effects throughout all cognitive components. This can be viewed as part of reactive–deliberative integration, in the context of a continuous time and space model.

For a fully integrated agent model, however, the effects cannot stop within the cognitive system. Full integration between cognitive and physical components entails that models should not restrict micro-level cognitive changes from affecting physical elements. Indeed, if a modeling framework represents arbitrary levels of detail, enabling representations of arbitrarily unpredictable environments, then integrated agent models should permit micro-cognitive effects to cause micro-physical effects (and vice versa); indeed, any cognitive element should be able to somehow affect any physical element (and vice versa). In the DI-HDCA framework, one could design models with pre-imposed constraints on the extent of cognitive–physical integration – e.g., that the agent's heading angle for navigation has no effect on the activation of a particular desire to complete a task – but to support fully integrated models, the framework allows for models without such constraints.

The constraints from environment and embodiment therefore entail continuum-valued representations for both cognitive and physical elements of the model, and simultaneous integration across reactive and deliberative intelligence and cognitive and physical components. This is achieved in DI-HDCA models by expressing all continuously varying elements in the unifying language of differential equations, in a hybrid dynamical system model (see section 2.4). This does not entail that *all* model elements must be continuously varying, but critical cognitive and physical elements should vary continuously, and the agents described in sections 4 and 5 exemplify these ideas.

2.3. Cognitive Modeling and Goal-Directed Reasoning

DI-HDCAs can be viewed as having *physical* and *cognitive* system components, represented by the differential equations and variables describing behaviors conventionally considered physical or cognitive, respectively. Because DI-HDCA modeling is designed for embodied agents moving through environments, models can contain continuously time-varying representations of physical elements conventionally useful for modeling motion, such as *xy*-location, velocity, or heading angle; because DI-HDCA modeling is also designed for integrated, goal-directed intelligence of these navigating embodied agents, models also contain continuously time-varying representations of *cognitive elements* conventionally

useful for modeling goal-directed behavior. These cognitive elements are derived from the *BDI* (*belief–desire–intention*) theory of practical reasoning (Bratman, 1987) and the many agent-based implementations of it [Georgeff and Lansky (1987) and many successors].

BDI theory recognizes the critical role of intentions as cognitive elements of practical reasoning, distinguishing intentions from desires. Although all three kinds of cognitive elements influence behavior selection and planning of task sequences, beliefs, desires, and intentions are distinct in their roles: in particular, desires (i.e., desired goals or conditions in the world) may conflict, whereas intentions are *conduct-controlling* pro-attitudes, reflecting commitment to behaviors and resisting reconsideration or conflict. A BDI-based approach provides a broader cognitive framework for goal-directed agents than conventional hybrid reactive–deliberative architectures [e.g., Arkin (1990), Gat (1998)], subsumption architectures [e.g., Brooks (1986)], or other behavioral robotics approaches that do not employ distinct desires and intentions as cognitive elements for action selection and task sequencing.

Conventional applications of BDI theory to computational agents, however, do not explicitly support all the entailments of embodiment or the multi-tiered integration described in this paper. For example, BDI implementations are not conventionally based on continuous models of time and space, and action selection and task sequencing are typically the result of deliberative processes, employing propositional representations of beliefs, desires, and intentions. Designing BDI-based agents without continuum-valued representations seems apt for some contexts – without the requirements of embodiment, continuous representations might needlessly complicate agent design and analysis – but for embodied, goal-directed mobile robots, continuous-modeled cognitive and physical representations can be beneficial, particularly to support the integration inherent in such robots. Moreover, continuum-valued cognitive representations support dynamicist perspectives of cognition (Port and van Gelder, 1995; van Gelder, 1998; Beer, 2000; Spivey, 2007), and they enable sensitivity to real-time micro-cognitive variations that can cascade into macro-level cognitive effects.

In DI-HDCA models, cognitive elements are represented by continuously varying activation values, where an activation value represents the salience "in mind" of the related cognitive element. As examples, beliefs with high activations are "strongly held," desires with near-zero activations are not "strongly felt," and high-active intentions indicate high priorities on the related actions. Because all cognitive elements are represented this way, and activations can vary in real time, interactions among them can be represented by an unconventional *spreading activation network*. Spreading activation networks are well-established models with applications in both cognitive psychology (Collins and Loftus, 1975) and agent modeling (Maes, 1989), based on neuroscience-influenced ideas that activations of cognitive elements can affect activations of other cognitive elements. Spreading activation networks are related to other connectionism-inspired approaches, including Haazebroek et al. (2011), which employs ideas from the *theory of event coding* to model action and cognition; similar to DI-HDCA modeling, the work in Haazebroek et al. (2011) emphasizes shared representations for integrating across levels of action and cognition, but the DI-HDCA framework is explicitly focused on dynamically sensitive representations of intentions, desires, and beliefs for goal-directed navigating agents.

Because cognitive activation values are governed by differential equations in DI-HDCA models, the spreading activation framework employed is unconventional: instead of the activation of an element directly having excitatory or inhibitory effects on activations of other elements, the activation of an element affects the *rates of change* in activations of other elements. That is, an activation of one element serves as part of a term in the differential equation describing the variation in another element. As a small, constrained example, consider this part of a differential equation, where B_P stands for the activation on the belief of P, $k > 0$ is a constant, I_A stands for the activation on the intention for action A, and the dotted variable $\dot{I}_A$ stands for the rate of change in I_A:

$$\dot{I}_A = \ldots + k \cdot B_P + \ldots. \tag{1}$$

This encodes excitatory and inhibitory effects on I_A: if $B_P > 0$, $\dot{I}_A$ will increase, for an excitatory effect on I_A over time; if $B_P < 0$, $\dot{I}_A$ will decrease, for an inhibitory effect. The magnitude of coefficient k in that equation serves to intensify or diminish the effect of B_P on I_A, an observation that is exploited in mechanisms for DI-HDCA learning (see section 2.5). A system of such differential equations, in which activations of cognitive elements are parts of differential equations for other cognitive elements, thus models a spreading activation network. Although this network may be viewed as unconventional due to the layer of indirection induced by the differential equations, it might also be viewed as appropriate for continuous-time environments with arbitrary asynchrony. Activation is not passed through the model in synchronized lock step nor in pre-determined quantities, and the quantity of spread activation is time-varying and responsive to changes in the system, fitting a DI-HDCA's environment.

The BDI-based framework of dynamical intentions presented in this paper is not the only agent model with dynamical systems-based elements that can be viewed as representing intentions. The dual dynamics framework (Hertzberg et al., 1998; Jaeger and Christaller, 1998) represents *activation dynamics* as different from *target dynamics*, analogous to intentions and navigation dynamics in DI-HDCAs. *Dynamic neural field* approaches (Schöner et al., 1995; Erlhagen and Bicho, 2006; Richter et al., 2012; Sandamirskaya et al., 2013), based on neuroscientific principles, also associate activations of cognitive entities with actuations of behaviors. The DI-HDCA framework shares the emphasis on dynamics with these approaches but is less tightly coupled with low-level sensorimotor systems, emphasizing cognitive dynamics of typically higher-level constructs of desires and intentions, which can directly support the high-level behavioral design and analysis desirable for many embodied robotics applications.

2.4. Hybrid Dynamical System Modeling

Continuous dynamics are essential for cognitive and physical elements in DI-HDCA models, but discrete dynamics are also important for behavioral modeling. Robots are productively designed

and understood in terms of discretely delineated behaviors, with transitions between those behaviors. The idea that discrete changes between behaviors can occur when some threshold condition is met has been employed in contexts ranging from logical models ("if *condition* then *begin action A*") to neural models (e.g., threshold for neurons firing) and beyond, including system models that combine continuous and discrete dynamics.

To support both continuous dynamics and discrete behavioral design, a DI-HDCA model can be expressed as a *hybrid automaton* model of a *hybrid dynamical system* (*HDS*), which explicitly represents and distinguishes continuous and discrete system dynamics (Alur et al., 2000). A hybrid automaton is a finite-state machine in which each state (*mode*) is a continuous behavior, specified by differential equations describing system dynamics in that mode. HDS models have been employed for many complex applications, including navigating robots or virtual agents [e.g., Egerstedt (2000) and Aaron et al. (2002)], and for the present application, the structures of a DI-HDCA model correspond naturally to elements of an HDS. For DI-HDCAs, each behavior might be specified as a mode, describing the physical and cognitive dynamics governing the robot while executing that behavior. **Figure 2** illustrates a mode in a DI-HDCA model, showing cognitive elements interconnected in a dynamical system model. The physical elements (e.g., position, velocity) are also governed by differential equations in each mode, and because all physical and cognitive elements are represented as variables in a dynamical system, any one of them can be part of any differential equation in the system – i.e., for integration, any element can affect the dynamical change in any other element.

In DI-HDCA behavior, transitions between modes occur when threshold conditions (*guards*) are met, and transitions are represented as instantaneous changes in behavior, which may be accompanied by discrete changes in values of elements in the model. For example, when some action A_i is completed, the robot might transition to the mode for action A_j, and the activation on the intention for A_i might instantaneously drop, as the robot no longer intends to carry out A_i. **Figure 3** illustrates a mode-transition system for a DI-HDCA, situating the mode from **Figure 2** in a full model. The connections between modes indicate available transitions: at any given moment, an agent is in exactly one mode (call it M_i), describing its behavior at that moment; when guard conditions in mode M_i are met, the agent transitions to some other mode M_j connected to M_i in the model.

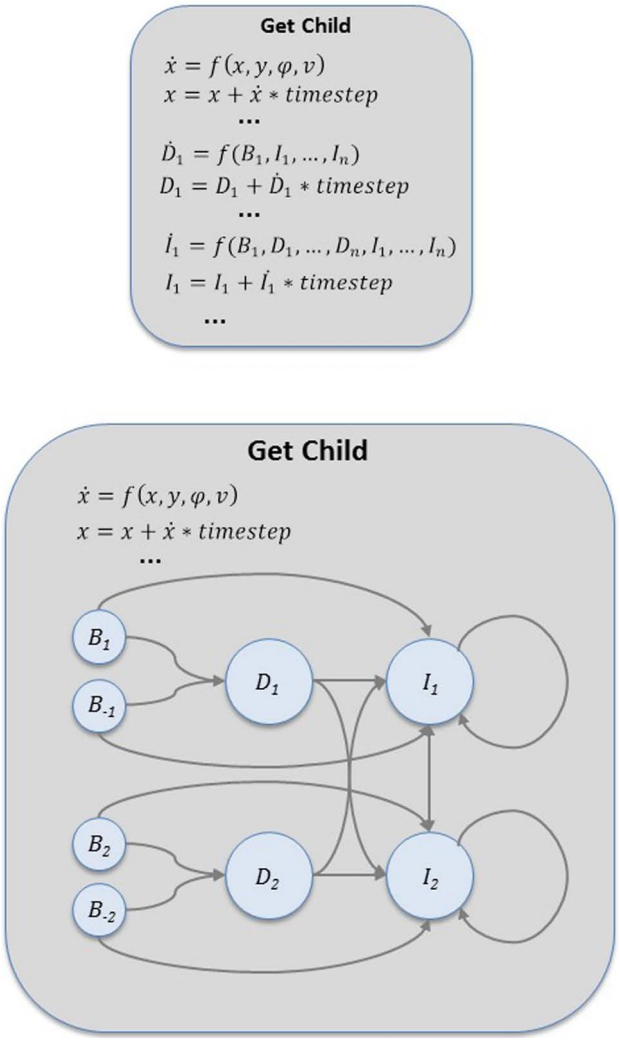


FIGURE 2 | Visualization of cognitive elements in a behavior (a *mode*) in the hybrid dynamical system model of a DI-HDCA service robot (see section 4). BDI-based cognitive elements – beliefs, desires, and intentions – are interconnected with excitatory and inhibitory links, expressed by differential equations in a dynamical system, which can be viewed as an atypical spreading activation network. Because both cognitive-level and physical-level components are expressed in the shared language of dynamical systems, physical behavior components such as speed or heading angle (not visualized here) can in principle also be part of the dynamical system.

2.5. DI-HDCA Learning

Because cognitive elements are represented as parts of terms in differential equations (**Figure 2**), they can affect each other's activations and any behavior based on those activations. For example, with action selection or task sequencing based on which intentions have the greatest activation values, any cognitive element can influence every intention's activation in the network, thus affecting action selection. Moreover, because physical elements (e.g., position, velocity) are also represented in that dynamical system, they can in principle also affect activation values and task sequencing. This interconnectedness is central to integration in DI-HDCA modeling.

This interconnectedness is also central to straightforward methods by which DI-HDCAs can learn from experience. As background, note that the magnitude of the effect of element E_i on element E_j in the dynamical system is expressed by the coefficient $c_{i,j}$ in the related term, as in this example:

$$\dot{E}_j = \ldots + c_{i,j}E_i + \ldots. \quad (2)$$

Here, if coefficient $c_{i,j}$ became a greater positive number, the activation on E_i would have a stronger direct effect on $E_j - c_{i,j}$ represents the link from E_i to E_j. Thus, if an agent's experience suggested that E_i should have a different effect on E_j, learning that new effect would only require altering that coefficient.

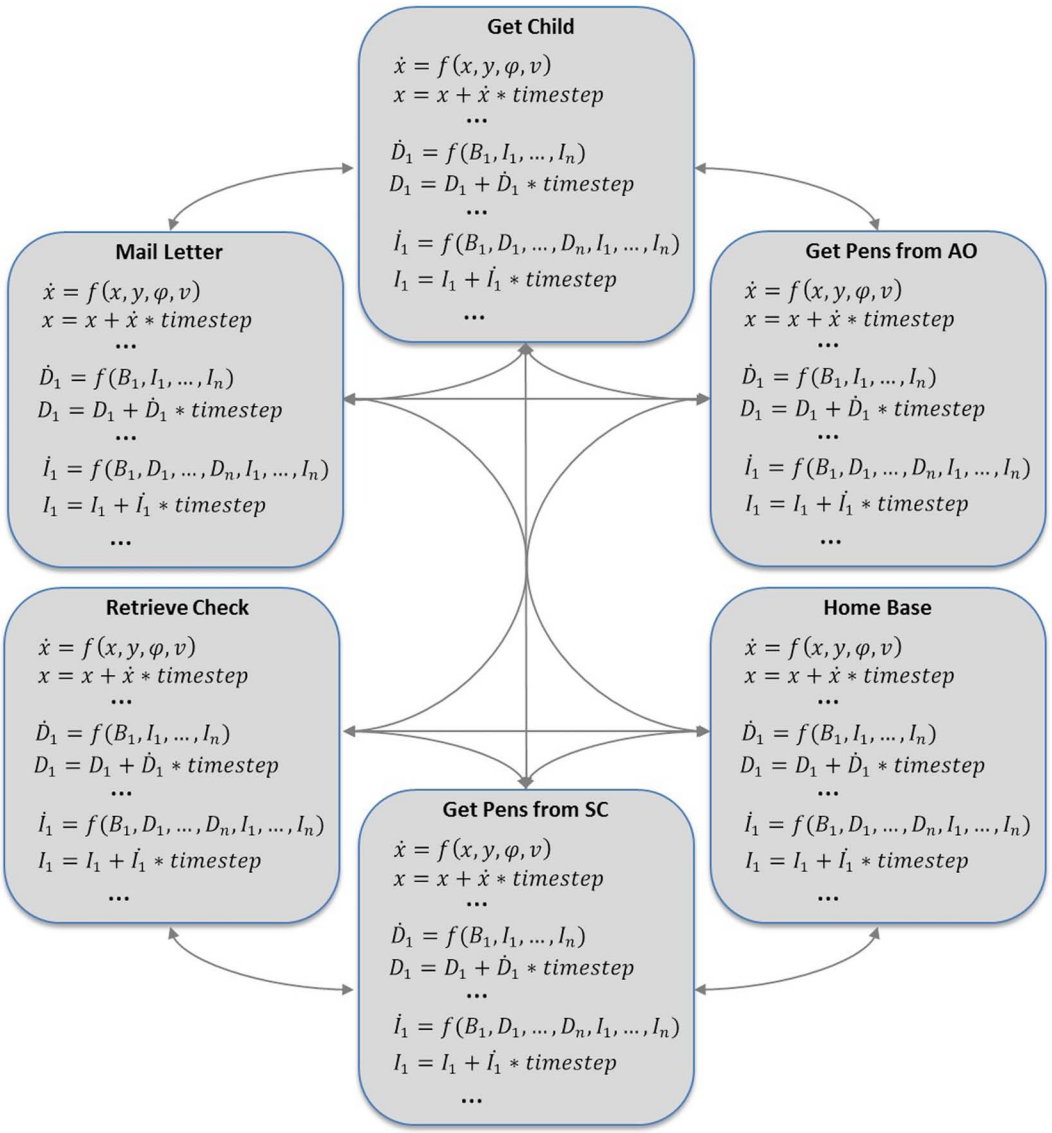


FIGURE 3 | Visualization of a full mode-level system model of a DI-HDCA service robot (see section 4). Individual modes (see **Figure 2**) are in a state-transition model: at any moment, the agent is in one behavior mode; the change to a different behavior is modeled by a transition from one mode to another. It is not necessary in DI-HDCA models that every mode be connected to every other mode, but there is no restriction against a fully connected system such as the one shown here. Every cognitive and physical element in the DI-HDCA can be present in each mode, for use in that mode's dynamical system.

With this, DI-HDCA learning of new strategies for action selection or dynamic task re-sequencing – aspects of agent intelligence that are often expressed as deliberative in agent models – could require only that the appropriate coefficients change values. For example, if feedback suggests that some belief B should affect action selection, the agent can learn that connection by adjusting coefficients relating belief B to the appropriate intentions. Because intentions are the cognitive elements representing actions, this can suffice to bring about the learned adaptation; no new rules or complex mechanisms are required.

Although the relationship between beliefs and intentions is an especially important one, DI-HDCA learning is not restricted to those elements. If feedback suggested, for instance, that increased salience of a desire D is not productive during some action A, connections could be learned to lower the activation on intention I_A corresponding to action A whenever the activation of D is a large value. Moreover, if faster speed of an agent is not productive when the agent is in action A (action A might require acute perception or good traction for motion), the agent could learn to calibrate the activation of I_A based on speed. Because of the full interconnectedness of the cognitive–physical representations enabled by DI-HDCA models, any such relationship could be straightforwardly learned by altering the weights of links between elements.

From the perspective of an agent designer, this mechanism can effectively refine agent behavior to meet specifications, even in interactive environments (see section 5). From the perspective of a scientist modeling and analyzing behavior, this mechanism

enables the study of connectionism-inspired learning – learning occurs by changing weights of links between elements – with phenomena as low-level as speed and as high-level as intention. [In context, it can also be viewed as a form of reinforcement learning; see Aaron and Admoni (2010).] The integration encoded in DI-HDCA models enables such straightforward learning approaches to be exceptionally effective in DI-HDCAs.

3. AGENT IMPLEMENTATION

Dynamical intentions can be implemented in multiple ways to be consistent with distinguishing properties of intention in BDI theory. Similarly, the reactive navigation intelligence in DI-HDCAs can vary with different agent implementations. Nothing intrinsic to the DI-HDCA framework *fully* defines such options, although some constraints are imposed (e.g., navigation models are expressed as differential equations, for integration with dynamical intention). Below, this section presents general background regarding DI-HDCA implementation and simulation for the experiments in sections 4 and 5, including the navigation system and a brief summary of the factors for adherence to BDI properties.

3.1. Distinguishing Properties of BDI Intentions

As described in section 2.3, the BDI-based cognitive elements of DI-HDCAs are represented by dynamically varying activation values. For agents implemented in the demonstrations described in this paper, cognitive activations are bounded to be within $[-10, 10]$. Low-magnitude activation values (i.e., near 0) indicate low salience of the associated concepts, whereas greater magnitudes of activations represent more importance or intensity of the associated concepts; for example, a desire with near-zero activation would indicate relative apathy regarding the associated concept, while a belief with high activation would be strongly held and a high-active intention would indicate greater importance of and commitment to the related task or behavior. Activations with negative values indicate salience of the opposing concept – e.g., an intention with activation -2 indicates a mild commitment not to do the associated task, and a belief with activation -9 indicates that the opposite or negation of the associated concept is strongly held.

For the agents implemented in this paper, beliefs and desires can conflict with each other. For instance, if an agent model included both beliefs B_{amIt} representing that the agent is It in a Tag game and B_{notIt} representing that the agent is not It, the model need not preclude them from having simultaneously high activations. DI-HDCAs could be designed to disallow conflicting beliefs, and doing so could benefit some applications, but for the explorations of computational intelligence in this paper, such conflicts were not explicitly disallowed. Similarly, it is possible for conflicting desires to have simultaneously high activations, representing an agent intensely desiring to do two things when only one at a time is possible.

The philosophical foundations of BDI agents assert that desires can conflict with each other but intentions resist conflict with each other. This is one of the *distinguishing properties* of intentions noted in Bratman (1987), part of explicitly establishing desires and intentions as distinct cognitive elements. For this paper, DI-HDCAs are implemented with mechanisms consistent with distinguishing properties that apply to this dynamical account of intention[2]:

- Intentions are *conduct-controlling* cognitive elements.
- When salient, intentions *resist reconsideration*.
- When salient, intentions *resist conflict* with other intentions.

It is straightforward to implement that intentions control conduct: in the state-transition system representing a DI-HDCA's behaviors (see **Figure 3**), conditions for entering and exiting a mode specify that the highest-active intention determines agent state. Initially, the agent must begin in the mode corresponding to its highest-active intention, e.g., in mode *Init*, when intention I_{Init} has the highest activation of any intention. Then, a transition to another mode *Other* occurs only when intention I_{Other} becomes highest-active, which can happen in two ways: behavior *Init* becomes completed, so the activation of I_{Init} is set to a low value (e.g., -10) and intention I_{Other} becomes highest-active; or the cognitive activation values change over time, as governed by the dynamical system, and the activation value of I_{Other} evolves to become greater than I_{Init}.

For reconsideration resistance, the implemented mechanism [described in Aaron and Admoni (2009, 2010)] encodes that a high-active intention I_a tends to minimize other intentions' impacts on I_a, and this effect becomes more pronounced as the activation of I_a grows. For intentions I_a and I_b ($b \neq a$), the differential equation for $\dot{I}_a$ includes the following structure:

$$\dot{I}_a = \ldots \; - k_i \cdot \mathrm{PF}(I_a) \cdot \mathrm{NCF}(I_b) \cdot I_b + \; \ldots . \tag{3}$$

Persistence factor PF is defined as

$$\mathrm{PF}(I_a) = 1 - \frac{|I_a|}{\sum_i |I_i| + \epsilon}, \tag{4}$$

where i ranges over all intentions and the $\epsilon > 0$ term prevents division by 0. Then, $\mathrm{PF}(I_a)$ multiplies every intention I_b in the equation for $\dot{I}_a$ (for $b \neq a$), so as I_a grows in magnitude relative to other intentions, contributions of every I_b are diminished, and when $\mathrm{PF}(I_a) = 1$ (i.e., $I_a = 0$), such contributions are unaffected. The denominator is designed to model I_a as less reconsideration resistant when other intentions are highly active.

The implemented mechanism for conflict resistance among intentions is also in coefficients in cognitive dynamical systems. In this paper, every intention in agents' cognitive systems is negatively interconnected with every other intention, with a *non-conflict factor* NCF as part of the differential equation for every intention. [Recall from equation (3) that $\dot{I}_a = \ldots \; - k_i \cdot \mathrm{PF}(I_a) \cdot \mathrm{NCF}(I_b) \cdot I_b \ldots .$] The non-conflict factor function is:

$$\mathrm{NCF}(I_b) = \left(1 + 1.6\left(\frac{I_b}{10}\right)^8 + 0.8\left(\frac{I_b}{10}\right)^9\right). \tag{5}$$

[2]These are not the only properties of intention described or emphasized in Bratman (1987); these properties, however, can apply to reactive-level intention, not requiring, e.g., future-directedness incompatible with reactive cognition.

This NCF component is applied similar to PF: in the differential equation for I_a, each term for an intention I_b is multiplied by NCF(I_b) (although unlike PF, it is possible that $a = b$). Thus, NCF decreases activation levels for conflicting intentions (and increases them for non-conflicting intentions, e.g., when $a = b$). The constants in equation (5) were chosen for agents in this paper by the agent designer after thought experiments and evaluation of preliminary tests; with different choices of constants, other DI-HDCAs could perform differently in the same general framework.

To test NCF effectiveness, simulations were run that isolated effects of NCF: agents did not navigate, and persistence factor PF was removed from the cognitive system; experiments compared a control group without NCF to an experimental group with NCF for results. Each group was identical in all other ways, containing ten agents ($A_1, \ldots, A_{10}$) with cognitive elements designed for the office scenario in section 4. Each of the ten agents had identical cognitive activation values except for initial activations on intentions; for intentions, each agent A_i's initial activations were i/3 times these baseline values:

MailLetter	GetChild	RetrieveCheck	HomeBase	GetPensSC	GetPensAO
3.1	2	1	1	1	1

The *rate of change* in activation on intention I_{ML} corresponding to the *MailLetter* behavior was then measured. On average, over the first 30 s of test runs, agents with non-conflict factor NCF in operation and the highest level of initial activation had a lower rate of decrease in activation of intention I_{ML} compared with agents in the baseline condition. The effect was reversed at medium levels of initial activation, as indicated by marginally significant ($p = 0.052$) interaction. For the baseline agent, mean rates of change were -0.228 when medium-active and -0.232 when high-active; for the NCF agent, -0.279 when medium-active and -0.191 when high-active, as presented in **Figure 4**. (All statistical analyses in the paper were conducted with SPSS, version 23.)

Examination of distinguishing BDI properties for DI-HDCAs is not complete, but the implemented mechanisms suggest that dynamical intentions can be consistent with BDI properties, and they demonstrate the environmental sensitivity and design control capable in the DI-HDCA framework.

3.2. Navigation

Although some agent navigation for this paper is simple, straight-line motion (see section 5.1), most agent navigation in both the Tag game and the office grid-world (section 4) is instead similar to the potential-based reactive navigation of Schöner et al. (1995), Large et al. (1999), Goldenstein et al. (2001), and Aaron and Mendoza (2011). This system models environments as consisting of *actors* (the navigating agents), *obstacles* that repel actors, and *targets* that serve as goal locations, attracting actors. Actors, obstacles, and targets can be either moving or stationary, and actors can be treated as obstacles or targets by other actors. In the Tag game scenario for experiments in section 5, for instance, non-It players might consider It actors as obstacles, and an It player may have an actor as its target. To illustrate the system and suggest the mathematics underlying it, the dynamics of this navigation system are briefly summarized here.

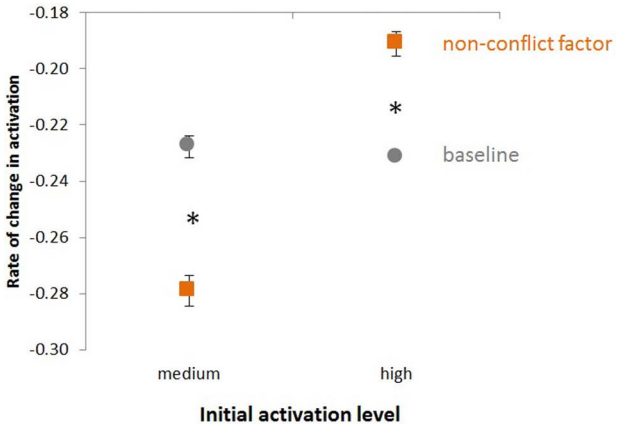


FIGURE 4 | Intentions with high activation avoid conflict with other high-active intentions. *A priori* contrasts indicate significant differences ($p < 0.05$) between means, as shown by asterisks. The main effects of activation level and type of agent are significantly different ($p < 0.05$) in a 2 × 3 ANOVA, with a low activation level condition included but not shown for clarity and brevity.

Non-linear *angular attractor* and *repeller* functions represent targets and obstacles, and their weighted contributions are dynamically combined to calculate an actor's angular velocity in real-time response to the environment. Heading angle ϕ is computed by a non-linear system of the form:

$$\dot{\phi} = f(\phi, \mathbf{env}) = |w_{tar}| f_{tar} + |w_{obs}| f_{obs} + n, \qquad (6)$$

where f_{tar} and f_{obs} are the attractor and repeller functions for the system, and w_{tar} and w_{obs} are their weights in the calculation. (Noise term n helps prevent the system from becoming trapped at critical points.) The weights themselves are determined by computing fixed points of another non-linear system [see Large et al. (1999) for details]. Other parameters and details are also concealed in the terms presented above. For instance, a repeller function f_{obs} depends on parameters that determine how much influence obstacles have on an actor. This is only a partial overview of the navigation system, but it suggests the complexity involved in modeling it and exposes the significant non-linearity in the agent models' physical components and navigation intelligence.

Although this navigation system integrates cleanly into dynamical intention-based intelligence, it is not the only option. For example, instead of abstracting navigation to position, heading, velocity, etc., as the above system does, one might employ a more physically grounded model for motion of a wheeled robot: the robot would have volume and mass; acceleration would be critical to the model, as would friction on the wheels and drag through the air. Such a physically detailed model would also integrate cleanly with DI-HDCA intelligence, as long as the system of motion was expressed in the language of differential equations, so any element of the system could straightforwardly affect any differential equation in the system – cognitive or physical – to effect the desired integration.

```
while time < endtime

  for k = 1:numOfAgents
    ag = agentArray(k);

    switch ag.mode
      case HIDE_RA % mode name
        ag = hide_ra(ag);
      case HIDE_U
        ag = hide_u(ag);
        ...
    end % end switch

    agentArray(k) = ag;

  end % end for loop

end % end while loop
```

FIGURE 5 | The basic code structure of the simulator in MATLAB.

```
function agent = hide_ra(agent)
   ...
   if guard1
       ...
     agent = setMode(agent,...);
     return
   elseif guard2
       ...
     agent = setMode(agent,...);
     return
   end % end if-else block

   ...
   xd = agent.vel * cos(...);
   yd = agent.vel * sin(...);
   newx = oldx + (xd*timestep);
   newy = oldy + (yd*timestep);
   agent.posn = [newx newy];
   ...
```

FIGURE 6 | The basic code structure of a mode in MATLAB for a hybrid dynamical agent.

3.3. Simulation

The simulations for this paper are implemented in MATLAB, although other choices could also be good for implementing DI-HDCAs. At each time step, the simulation updates the state of each agent according to the behavior mode governing the evolution of that agent. The modes themselves are implemented as functions, containing both the propositional guards for transitions to other modes and the dynamical systems describing the behavior; executing a mode function on an agent either induces a transition to another mode or updates the state of the agent. As shown in **Figure 5**, the simulator loops through every agent, identifying the proper mode function to execute for that agent.

Figure 6 contains a sample code skeleton for a mode. In each mode, a list of mode-transition guards is checked, and if a guard is true, the mode-transition corresponding to the first true guard is taken. This transition is effected by discrete changes in the state of the agent, including setting a new mode value for the agent; the main loop will then simulate the agent in the appropriate new mode during the next time step. If no guard is true, the agent's state is updated according to the dynamical system in the mode. To simplify this implementation, all discrete or deliberative dynamics in the agents in sections 4 and 5 occur during these instantaneous transitions; representing deliberation during mode execution is an interesting extension of the current implementation, but it requires giving temporal dynamics to deliberation that is not typically modeled as temporally dynamic, and that complication was not engaged in the present work.

4. REACTIVE-LEVEL LEARNING AND DELIBERATIVE-LEVEL INTELLIGENCE

Part of the integrated intelligence of DI-HDCAs is the distribution of goal-directed intelligence over both reactive and deliberative processes: task sequencing and action selection are often considered to be deliberative-level intelligence, but with dynamical intention modeling, some can be handled by reactive-level intelligence and learning. This enhancement of reactive-level intelligence reflects a fundamental motivation of dynamical intention modeling and DI-HDCA design: reactive-level intelligence can be enhanced without denying deliberative intelligence; DI-HDCAs minimize *reliance* on deliberative intelligence, for greater robustness in unpredictable environments.

This section discusses dynamical intention-based learning methods for DI-HDCAs and describes demonstrations of agents learning to approximate deliberative, rule-based behavior. In particular, this section emphasizes how deliberative-level intelligence is distributed over reactive-level processing and learning. Although the idea of hybrid reactive–deliberative systems is not novel to DI-HDCAs, and deliberative-level intelligence that employs the same representations as reactive systems is not extraordinary (e.g., a planner that uses the location of a robot, where location is altered by reactive navigation) in hybrid agents, DI-HDCA modeling emphasizes "the other direction" of distribution of intelligence: instead of low-level reactive representations being employed by high-level logical planners, DI-HDCAs' dynamical intentions enable conventionally high-level intelligence such as task sequencing and action selection to be distributed down to reactive, lower-level systems.

To demonstrate this reactive–deliberative integration, experiments consider a simulated service robot carrying out tasks in a grid-world office environment, illustrated in **Figure 7**, requiring navigation to various locations (see section 4.1 for task descriptions). To demonstrate the effects of reactive-level learning, three agents were compared: one had straightforward deliberative rules explicitly encoded to improve efficiency, the second was a reactive agent without dynamical intention-based learning, and the third agent employed dynamical intentions and reactive-level learning to approximate the rule-based performance of the first agent without requiring explicit deliberative rules. Two kinds of DI-HDCA learning were implemented for these experiments: a *Hebbian* learning method that strengthens connections among cognitive elements that are concurrently salient (i.e., with concurrently high activation values); and *belief–intention* (*BI*) learning for task-specific associations of beliefs and intentions. The Hebbian and BI learning methods were originally presented and qualitatively described in Aaron and Admoni (2010); this section summarizes

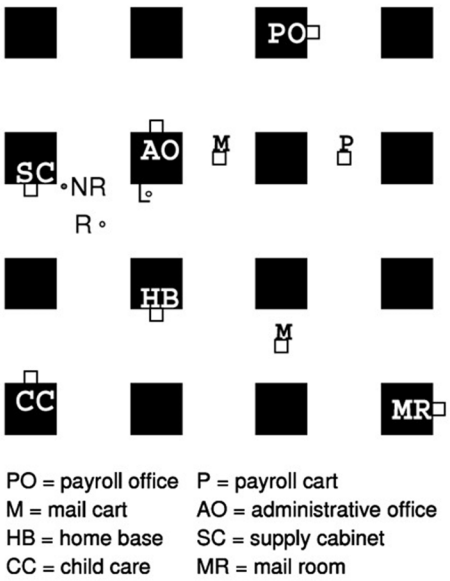


FIGURE 7 | Simulation in progress, screenshot. A map of the office environment, top, shows offices and obstacles (black squares), targets (white squares abutting offices), and three agents (*L*, *R*, and *NR*). Below the grid-world map is a list of target locations corresponding to agent tasks.

these learning methods and presents new analyses demonstrating their effectiveness.

4.1. The Office Grid-World: An Overview

An office environment for a simulated service robot provides a context in which navigation, action selection, and task sequencing are all essential. The particular office environment for these demonstrations (see **Figure 7**) is a simplified grid-world – e.g., mail carts in hallways are stationary, not moving obstacles – although future experiments in the same environment could more fully exploit DI-HDCA reactivity. In experiments, service robots can carry out six tasks, each with an associated target location: *MailLetter*, which requires navigating to the mail room (labeled *MR* in **Figure 7**); *GetChild*, with navigation to the child care center *CC*; *RetrieveCheck*, at payroll office *PO*; *HomeBase*, at home base *HB*; *GetPensFromSC*, at supply closet *SC*; and *GetPensFromAO*, at administrative office *AO*. Agents are therefore implemented with six behavior modes, one for each task, and cognitive elements including one intention for each behavior (e.g., I_{ML} for *MailLetter*, I_{GC} for *GetChild*), related beliefs (e.g., B_{ML} for having a letter to mail), and related desires (e.g., D_{GP} for the desire to get pens). These foundations enable experiments to focus on reactive and deliberative task sequencing intelligence, and this brief presentation emphasizes only the central elements for the results presented in this paper. In particular, perception and navigation intelligence are limited and not emphasized in these experiments; for additional details, see Aaron and Admoni (2010).

As introduced above, three kinds of robot agents were compared in DI-HDCA learning experiments. One agent A_R (for *Rules*) employed two straightforwardly encoded deliberative rules: a sorting-based *distance bias* to prefer task sequencing that co-prioritizes tasks with proximate target locations; and the *minimal-effort rule* to avoid redundancy such as needlessly going to both the supply closet and the administrative office to get pens. The second agent A_{NRL} (*Non-Rules/Non-Learning*) was identical to A_R except it lacked the relevant deliberative rules; it employed DI-HDCA task sequencing – intention activations determined its current task – but had no DI-HDCA learning implemented. The third agent A_{L} (*Learning*) employed dynamical intentions and reactive-level Hebbian and BI learning to approximate the rule-based performance of the first agent without requiring explicit deliberative rules. In the next sections below, both general expositions and specific applications to these agents are presented, for both Hebbian and BI learning, although the experimental results presented here focus primarily on BI learning.

4.2. Hebbian Learning

Inspired by observations about neuronal interconnections in Hebb (1949), Hebbian learning in these DI-HDCAs strengthens connections between co-active cognitive elements (i.e., elements that concurrently have high activation values). This broadly general dynamical intention-based Hebbian learning method could in principle apply to any elements, but for these demonstrations, it is only employed to enhance connections among intentions associated with target locations that are near each other: the closer the locations, the stronger the connection between the associated intentions.

For DI-HDCAs in this paper, the mechanism for Hebbian learning is based on a limited model of perception and additional structure in the cognitive dynamical system that allows perception to affect intention activations. Training for Hebbian learning consists of each agent simply navigating in its environment. For these demonstrations, training consists of an agent taking a prespecified route through the office environment that passes close to all target locations for tasks (e.g., mail room, supply cabinet); training stops at the completion of that route. (Different training routines or stopping criteria could result in different learning; this choice suffices for the present demonstrations.) Each agent has a radius of perception r_p roughly equal to one-quarter of the length of the grid-world, so it accurately perceives target locations within distance r_p of it as it moves. During training runs for Hebbian learning, coefficients encoding interconnections between intentions have their values increased (until stopping criteria are reached) based on the proximity of target locations. In particular, for intentions I_a and I_b (corresponding to tasks a and b, where $a \neq b$) and associated target locations L_a and L_b, if both L_a and L_b have been recently perceived by the agent, the following coefficients become greater in the cognitive dynamical system:

- The coefficient $k_{a,b}$ on intention I_b in the equation $\dot{I}_a = \ldots\ k_{a,b} \cdot I_b \ldots$ gets larger by an amount proportional to how recently L_b has been perceived.
- The coefficient $k_{b,a}$ on intention I_a in the equation $\dot{I}_b = \ldots\ k_{b,a} \cdot I_a \ldots$ gets larger by an amount proportional to how recently L_a has been perceived.

Because this occurs only when both L_a and L_b have been recently perceived, only proximate target locations contribute to the strengthening of connections between associated intentions, and there is greater co-activation between intentions when

the target locations are perceived closer to each other during training.

Additional details are in Aaron and Admoni (2010) about how coefficients are altered during training (including a Hebbian scaling constant c_1 that affects the changes in $k_{a,b}$ and $k_{b,a}$), the mechanism by which recency of perception is implemented to result in the learning described here, and the effects of Hebbian learning without BI learning. The above description only summarizes the details necessary for the presentation of integrated Hebbian and BI learning in section 4.4 below.

4.3. Belief–Intention Learning

Intentions and beliefs have an especially important conceptual relationship regarding task completion: completion of a task T likely results in a strong belief that T has been completed; unless T needs to be repeated, the belief that T is completed would influence intention I_T to have a negative value, so the agent would intend not to do task T again. Belief–intention (*BI*) learning, which alters cognitive connections between beliefs and intentions, is therefore especially significant for DI-HDCAs. For experiments in this paper, BI learning trains agents to relate intentions to beliefs in ways that might typically be encoded in propositional rules such as the minimal-effort rule (see section 4.1), but without any proposition-based learning. Details about BI learning, originally presented in Aaron and Admoni (2010), are summarized below.

In these experiments, the BI learning mechanism requires that coefficients relating beliefs to intentions have the form

$$\begin{aligned} \mathrm{IC}(I_a, B_b) &= k_{a,b} \cdot [r_{a,b} \cdot C_{a,b} + (1 - r_{a,b})] \\ \mathrm{IC}(I_a, B_{\bar{b}}) &= k_{a,\bar{b}} \cdot [r_{a,\bar{b}} \cdot C_{a,\bar{b}} + (1 - r_{a,\bar{b}})]. \end{aligned} \tag{7}$$

Variables a and b ($a \neq b$) refer to tasks, ranging over the six behaviors for agents; as convention, the $k_{a,b}$ values are designer-chosen scalars, I_a is the intention associated with task a, and B_b ($B_{\bar{b}}$, respectively) is the belief associated with task b having been completed (not completed). Coefficient $\mathrm{IC}(I_a, B_b)$ ($\mathrm{IC}(I_a, B_{\bar{b}})$) is then placed as the coefficient on term B_b ($B_{\bar{b}}$) in the differential equation for intention I_a:

$$\begin{aligned} \dot{I}_a = \ldots \;\; & k_{a,b} \cdot [r_{a,b} \cdot C_{a,b} + (1 - r_{a,b})] \cdot B_b \\ & + k_{a,\bar{b}} \cdot [r_{a,\bar{b}} \cdot C_{a,\bar{b}} + (1 - r_{a,\bar{b}})] \cdot B_{\bar{b}} \;\; \ldots . \end{aligned} \tag{8}$$

The $r_{a,b}$ and $C_{a,b}$ values can be designer selected for specific applications. For this motivating example application – learning behavior consistent with the deliberative minimal-effort rule, avoiding redundant tasks when relevant but otherwise leaving cognition unaffected [see Aaron and Admoni (2010) for additional details] – $r_{a,b} = 1$ exactly when belief B_b should affect intention I_a, otherwise $r_{a,b} = 0$ (similarly for $r_{a,\bar{b}}$ and $B_{\bar{b}}$), i.e., $r_{a,b} = 1$ exactly when a, b correspond to redundant tasks, which here are the pen-related tasks *GetPensFromSC* and *GetPensFromAO*. The $C_{a,b}$ values specify how B_b affects $\dot{I}_a$ when $r_{a,b} = 1$; for this example, $C_{a,b} = C_{a,\bar{b}} = c\frac{B_{\bar{b}} - 10}{-20}$, so $B_b, B_{\bar{b}}$ both do not effect I_a when beliefs reflect that task b has not yet been completed ($B_{\bar{b}} = 10$), but after b has been completed ($B_{\bar{b}} = -10$), the coefficient on I_a drops rapidly, preventing a redundant errand.

These $r_{a,b}$ and $C_{a,b}$ terms are not modified due to BI learning, however. As with this Hebbian learning, this BI learning modifies coefficients $k_{a,b}$ during training. Training consists of an agent running errands in its office; the stopping criteria are met if that errand run ended with the agent having completed exactly one of the two pen-related tasks. If the errand run stopped but it was not the case that exactly one pen-related task had been completed, the scalar parts $k_{a,b}$ (for $a \neq b$) in coefficients described in equation (7) are modified as follows:

$$k_{a,b} = k_{a,b} \cdot [1 + r_{a,b}(\gamma_{a,b} - 1)]. \tag{9}$$

(Scalars $k_{a,\bar{b}}$ are similarly modified.) The pre-specified scalar $\gamma_{a,b} > 1$ encodes the extent of the modification. In this implementation, therefore, when tasks are not redundant, $r_{a,b} = 0$ and $k_{a,b}$ is unchanged; when learning could lead to minimal-effort rule-like behavior, $r_{a,b} = 1$ and the inhibitory link between belief B_b and I_a is strengthened. Thus, once one pen-related task is completed, activation on the intention to do the other rapidly drops.

To demonstrate the effect of BI learning (Hebbian learning is not part of these demonstrations), two agents were compared: agent A_{BI}, which had been trained with BI learning to approximate the minimal-effort rule; and agent A_{NBI}, identical to A_{BI} but without training by BI learning. For these agents, the $\gamma_{a,b}$ parameter values were all 1.2, and the initial activations on desires and intentions are as presented here:

	MailLetter	GetChild	RetrieveCheck	HomeBase	GetPensSC	GetPensAO
Initial desires	3	9	8	2	1	*n/a*
Initial intentions	6	9.3	10	1	3	3

Recall that agents have only one cognitive element for desires to get pens – noted as $D_{GetPensSC}$ in the above listing – which has the expected excitatory effect on both *GetPensSC* and *GetPensAO* behaviors; there is no separate $D_{GetPensAO}$ element, which is noted by the value *n/a* for the activation for $D_{GetPensAO}$ above. Agent A_{BI} had seven training runs following the procedure described above, each starting from the same position near the supply cabinet on the left side of the office, and cognitive coefficients were adjusted during training. After training, agents A_{BI} and A_{NBI} were tested, with each test consisting of the agent autonomously running errands in its office; test runs began from 16 intersections in the office grid-world. Two facets of agent behavior were measured: redundancy, whether redundant tasks were completed by the agent, and speed, how long it took the agent to complete its run.

The redundancy measure was qualitatively described in Aaron and Admoni (2010): after training, agent A_{BI} completed exactly one pen-related task on all 16 errand runs, completely avoiding redundancy and adhering to the minimal-effort rule; agent A_{NBI}, in contrast, redundantly completed both pen-related tasks on 8 of its 16 errand runs. The speed measure, not previously statistically analyzed, is presented in **Figure 8**. The completion times of runs varied as expected depending on starting position: the agents' first errand was to go to Payroll Office *PO* on the map (**Figure 7**), so

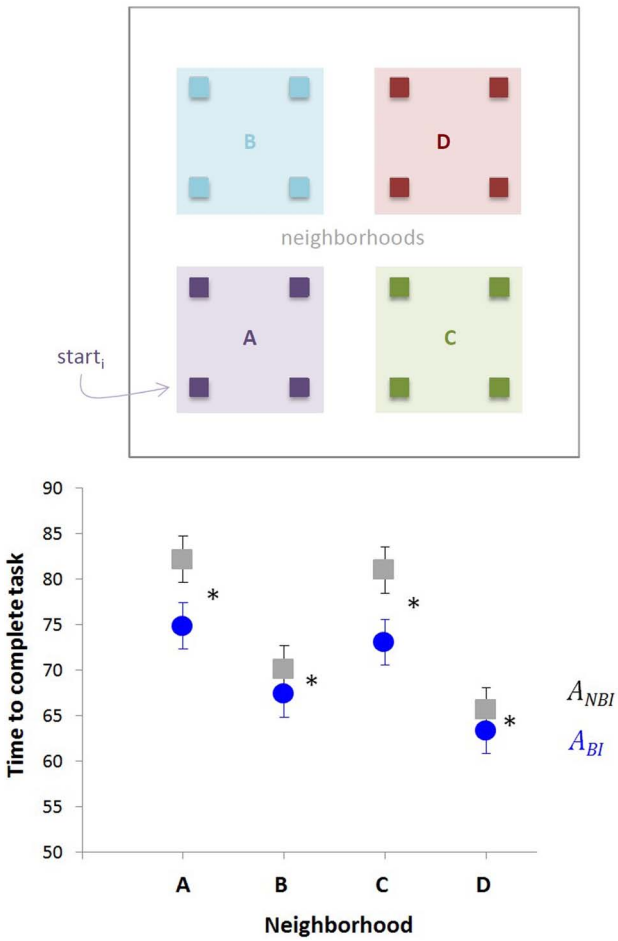


FIGURE 8 | Belief–intention (BI) learning improves navigation. Agents started from one of 16 different positions, as indicated by the small squares on the map (top). For purposes of analysis, four neighborhoods were defined. The time for each agent to reach the target was measured (bottom). Agents that undergo BI learning navigate significantly faster to the target than agents without learning (ANOVA, significant main effects of agent type and neighborhood $p < 0.05$, with asterisk indicating the difference between agents). Neighborhoods are the same as in **Figure 9**.

runs starting farther from *PO* tended to take longer. Completion time data were therefore considered in four neighborhoods, each corresponding to a quadrant (lower/upper, left/right) of the map, and each containing four of the 16 starting points; a depiction of the neighborhoods is presented with the results in **Figure 8**. In every neighborhood, from every starting location, agent A_{BI} completed its run faster than agent A_{NBI}: in neighborhoods *A*, *B*, *C*, and *D*, respectively, the mean times to complete the runs are 74.885, 67.417, 73.073, and 63.375 s for A_{BI}, and 82.198, 70.167, 80.958, and 65.573 s for A_{NBI}.

These experiments suggest the effectiveness of BI learning for improving efficiency, enabling deliberation-level intelligence without proposition-based deliberative reasoning. Other implementations of BI learning are certainly possible for DI-HDCAs, but this simple example illustrates fundamental ideas about how learning can alter connections between beliefs and intentions to train agent behavior.

4.4. Integrating Hebbian and BI Learning

The Hebbian and BI learning methods described above can be straightforwardly integrated: because they alter disjoint sets of cognitive connections, nothing additional is needed to employ both methods together. For demonstrations of *integrated Hebbian–BI* (*HBI*) learning described in this paper, agents employ the mechanisms in sections 4.2 and 4.3 without alteration. These procedures and some results were originally in Aaron and Admoni (2010); this section summarizes the experiments run to demonstrate HBI learning and presents new and expanded statistical analyses of data from these experiments.

Training for HBI learning is consistent with procedures described above. A training run consists of an agent autonomously running errands in its office environment. Training concludes after a training run meets two conditions: the agent performs exactly one of the two pen-related tasks, suggesting learning of the minimal-effort rule; and the time taken by the errand run is not less than the time taken by the previous run, suggesting adequate learning of the distance bias. (Because DI-HDCAs in these experiments move at constant speed, time and distance are equivalent measures.) Training of agent A_L (*Learning*) consisted of 18 training runs beginning at the same location and with the same cognitive activation values and parameters as for the BI learning in section 4.3, along with Hebbian scaling constant $c_1 = 4 \times 10^4$.

As described in section 1, these experiments compared agent A_L to two other agents: A_{NRL}, which is identical to the pre-learning state of agent A_L; and A_R, which is identical to A_{NRL} except with propositional, deliberative encodings of the distance-bias and minimal-effort rule. For experiments, tests were run from 16 starting locations, consisting of each agent running errands as in the experiments of section 4.3. The redundancy of agents' runs – i.e., did they execute both pen-related tasks – and the average time of completion of agents' runs were measured and compared across the three agent types.

Considering task redundancy, the behavior of A_R in these tests was dictated by its deliberative rules, as expected: it always retrieved pens from the administrative office, so it never went to the supply closet. By comparison, HBI learning agent A_L also went to exactly one of those two locations on every run – indeed, on 15 of the 16 test runs, the dynamical intention-guided A_L performed exactly the same task sequence as A_R – but untrained agent A_{NRL} went to both locations on every run. Because A_{NRL} was identical to A_L without the integrated HBI training, the reactive-level learning clearly reduced redundancy, bringing about the same performance as A_R without additional deliberation.

Considering errand run completion times, A_L finished every run faster than A_{NRL}, but slower than A_R. As with the results in **Figure 8**, completion time data for these agents were considered in the same four neighborhoods. Results are in **Figure 9**. In every neighborhood, completion times of agents A_R and A_L are statistically indistinguishable, indicating that HBI learning enabled the DI-HDCA agent to approximate rule-based behavior without explicit deliberative rules. Moreover, in every neighborhood, both A_R and A_L were statistically different from A_{NRL}, demonstrating that learning distinguished A_L from A_{NRL}. As shown in **Figure 9**, the mean completion times (in seconds) to complete the errand runs in neighborhoods *A*, *B*, *C*, and *D* (respectively) are: 73.76,

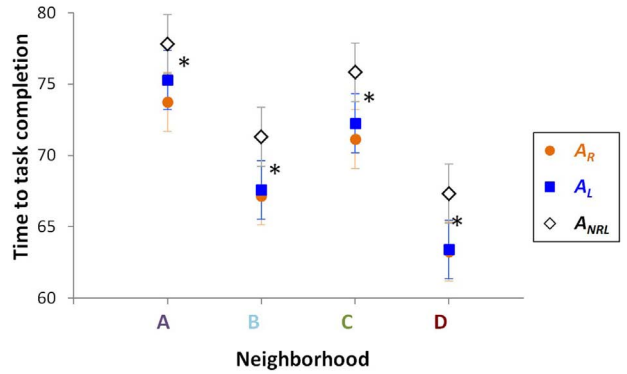


FIGURE 9 | Hebbian–BI (HBI) learning. The three types of agent started from one of 16 different positions, grouped into the same neighborhoods as in **Figure 8**. The time for each agent to complete its errand run task was measured. Using a 3 × 4 fully factorial ANOVA, a significant ($p < 0.05$) effect of type of agent and neighborhood was detected. *Post hoc* t-tests determined that in all neighborhoods A_R and A_L were statistically indistinguishable and that both were statistically different from A_{NRL}, as indicated by the asterisks. When the marker for A_R is not visible, it is hidden behind the marker for A_L.

67.187, 71.146, and 63.271 for A_R; 75.302, 67.583, 72.25, and 63.406 for A_L; and 77.822, 71.302, 75.823, and 67.333 for A_{NR}.

4.5. Discussion

The above results demonstrate that dynamical intention-based, reactive-level learning can train agents to closely approximate deliberative-level intelligence and rule-based behavior in these experimental conditions, without reliance on deliberative structures. DI-HDCAs do not learn *explicit propositional rules*; agents learn reactive-level tendencies generally (though not entirely) in accord with the guiding rules. This enables deliberative-level intelligence to be distributed to reactive-level processes, for hybrid intelligence that retains the benefits of both deliberative goal-based performance and reactive responsiveness.

The generality of the tasks and this domain suggest that learned behavior can generalize beyond an agent's training set, and that similar learning processes could generalize to other task domains. Moreover, Hebbian and BI learning alter only cognitive connections between some beliefs and intentions, but different DI-HDCA learning methods could incorporate other cognitive elements (including desires) or other connections among elements. Indeed, the underlying modeling framework of excitatory and inhibitory links among dynamically responsive cognitive elements is general enough to enable (if not encourage!) different approaches to DI-HDCA learning.

5. COGNITIVE–PHYSICAL INTEGRATION AND ONLINE LEARNING

Along with DI-HDCAs' integration of reactive- and deliberative-level intelligence, which arises from shared cognitive representations across both levels, cognitive–physical integration arises from both cognitive and physical system components being expressed in the shared language of dynamical systems. As a demonstration domain for integrated cognitive–physical learning for DI-HDCAs, interactive simulated Tag games – i.e., requiring agent interactions with people and not just other agents – provide some especially important elements: an unpredictable environment; a requirement for navigation intelligence, including target seeking and obstacle avoidance; and the possibility of both simple and complex behaviors and strategies.

Tag has continuous, real-time play rather than turn-taking, so online learning during gameplay might be preferable to learning that interrupts play or occurs only after games. Moreover, in a user-interactive environment, agents might be asked to learn things specified by a user during gameplay – for instance, an agent might be playing too well, making the game too difficult, and the user could instruct the agent to modify some but not all of its strategy during play, for a more enjoyable game. In such a multi-faceted modification, as described in section 5.2, the agent might need to modify both its physical speed and its cognitive strategy for the desired behavior, learning during gameplay and without direct user feedback.

DI-HDCAs' cognitive–physical integration can make it straightforward to learn this altered behavior. Because agent speed is represented by a variable in the agent's shared cognitive–physical dynamical system, cognitive variation can directly respond to physical variation: speed can be directly employed as a parameter in learning that alters the agent's cognitive network, so real-time micro-variations in speed can result in real-time micro-variations in cognitive-level strategy. As results in section 5.2 show, this straightforward approach can be effective for online learning in Tag-game demonstrations.

The remainder of section 5 further describes the Tag environment and related experiments. Although this is not the deepest instance of cognitive–physical integration possible in DI-HDCA models [see Aaron et al. (2011) for a brief mention of physical actions considered "involuntary" affecting cognitive elements considered "subconscious"], it illustrates the effect that cognitive–physical integration can have on adaptive agent behavior, and it illuminates the central role of dynamical intention modeling in integrated intelligence.

5.1. The Tag-World: An Overview

In the Tag game environment – called "Tag-world" here, analogous to "grid-world " in section 4 – interactions between the user and agents are standard: each It player pursues some non-It player; each non-It agent avoids It players. To make the game more adversarial, for these demonstrations, two players at a time are It. The field of play (**Figure 10**) is a square with bases near the corners, obstacles between bases, and multiple players; players are penalized for touching an obstacle, requiring that they stay frozen for a specified duration, during which they are vulnerable to getting tagged by an It player. In addition, players touching base cannot become It, but they cannot stay on base too long, to prevent play from degenerating into all players staying on base and none getting tagged.

These Tag games are populated by two kinds of autonomous agents: cognitive Tag agents (*CTAs*), the focal agents in these experiments, with dynamical intention-based cognitive systems and relatively complex strategies; and simple Tag agents (*STAs*) with relatively basic strategies, serving as additional players in the game. For navigation, sometimes agents use straight-line motion

that is not obstacle avoidant to move to a target location; when obstacle avoidance is needed, the navigation is the same kind as for the agents in section 4, as described in section 3.2. Tag-world and these agents were originally described in Aaron et al. (2011), which also contains details not included in the brief summary here; below, this section describes only details needed for the experiments involving CTAs, and it presents the results of online learning for CTAs, including qualitative description and new statistical analysis.

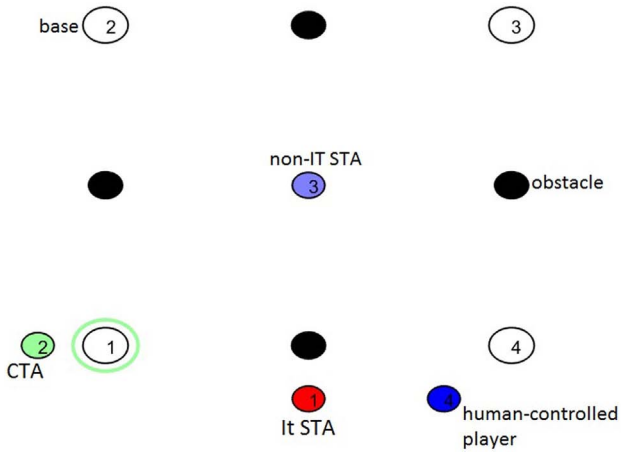


FIGURE 10 | Simulated interactive Tag game. The screenshot (with added labels) illustrates field layout and Tag players. Elements include bases, obstacles, human-controlled players, It agents, cognitive Tag agents, and simple Tag agents. [Aaron et al. (2011) contain additional details about these Tag simulations.]

An STA, when not It, simply runs clockwise from base to base, ideally avoiding being tagged. When an STA becomes It, it chooses from two possible *It-actions*: it either chases the user (the person playing the game) or it chases another agent. An STA's cognitive structure is a very simple dynamical intention-based system, straightforwardly supporting only this behavior; specifics of STA action selection are not central to results in this paper. In contrast, a CTA contains more complex intelligence and cognitive–physical integration; **Figure 11** shows the mode-level architecture of CTAs in these experiments. When a CTA *C* is not It, it will try to execute all of the following behaviors in a game: *runBases*, the simple base-running strategy that STAs have; *getMitten*, retrieving its mitten (which, as children sometimes do, this agent drops in every game); *protect*, protecting a friend from being tagged; and *readyToTag*, actively seeking to become It, to tag an adversary. The *getMitten* action is implemented by selecting a time when, wherever *C* is, its mitten drops; soon after, *C* finds the mitten's location, and cognitive activations evolve until, in general, mitten-retrieval becomes *C*'s highest priority. To enable *protect* and *readyToTag*, *C* has beliefs of affinities for each player in the game; *C* will protect a non-It player with maximal affinity during *protect* and pursue a non-It player with minimal affinity during *readyToTag*. When a CTA is It, it either follows through on a *readyToTag* action and pursues its selected adversary, or it selects between chasing the user or another agent, as STAs do. [Additional details of STAs and CTAs, not central to results in this paper, are in Aaron et al. (2011)].

The cognitive dynamical systems in these agents connect BDI cognitive elements in intuitive ways. For example, the equations governing activations of the desire to run bases, the intention to

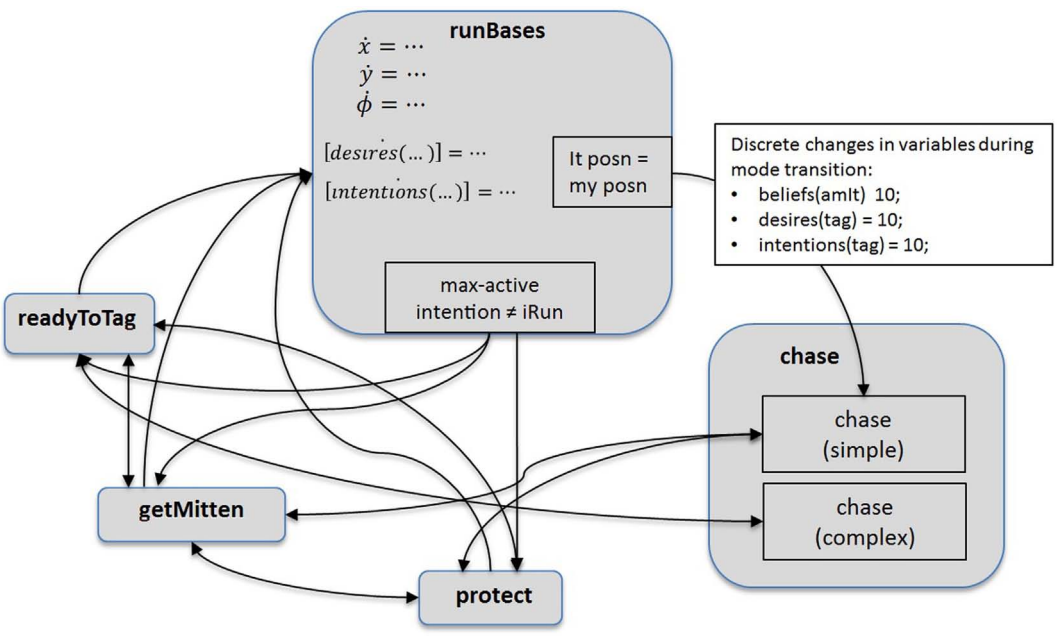


FIGURE 11 | Behavior mode-level architecture of a cognitive Tag agent (CTA). Each mode also has a self-transition, omitted here to avoid visual clutter. [Aaron et al. (2011) contain additional details.]

tag another player, and the intention to run bases contain the following structure:

$$\begin{aligned} &\dots \\ \dot{dRun} &= -c_1 \cdot bAmIt - c_2 \cdot iTag + c_3 \cdot iRun \\ \dot{iTag} &= d_1 \cdot bAmIt + d_2 \cdot dTag - d_3 \cdot dRun \\ &\quad + d_4 \cdot iTag - d_5 \cdot iRun \\ \dot{iRun} &= -e_1 \cdot bAmIt - e_2 \cdot dTag + e_3 \cdot dRun - e_4 \cdot iTag \\ &\quad + e_5 \cdot iRun \\ &\dots \end{aligned} \tag{10}$$

Additional structure is also present in equations for these cognitive elements, and additional equations are present for other cognitive elements. [The specific components for distinguishing properties of BDI intentions and the experimental results in section 4, however, are not present in agents for these experiments. For additional details about these cognitive systems, see Aaron et al. (2011).] The online learning of DI-HDCAs in these examples is based on the interconnections encoded in these equations, similar to the mechanism in section 4, as further described below.

5.2. Agent Learning and Cognitive–Physical Integration

The motivation for the Tag-world learning demonstrations below was to approximate what a human game-player might want during play: a user might specify agent behavior to change, within desired bounds, to improve the gameplay experience. For example, a user might have been tagged so quickly by It agents that the game was not a fair challenge, but when agents were non-It players, their behavior was good for gameplay. Based on this idea, a CTA was tasked to learn from a simulated user request to change one aspect of gameplay without affecting another, exemplifying an arbitrary user choice, unrelated to agent design and substantively changing behavior.

As preparation, *control condition* behavior for CTAs was determined by letting a game play extensively (more than 8000 simulated seconds), with an automated user for replicability. In this game setup, when a CTA C_{ctrl} became It, C_{ctrl} would almost always tag some other player in less than 25 simulated seconds (average: 12.85 s). In addition, the value a_{ctrl} of the average number of bases reached per execution of the *runBases* behavior, over the full game, was $a_{ctrl} = 4.01$.

For the learning demonstrations, the CTA would learn a goal with two components: *speed change*, requiring *speed-only* learning; and *base-running maintenance*, requiring *speed-and-bases (SB)* learning.

- *Speed change*: after becoming It, C should optimally tag some other player between 25 and 45 s later. Speed-only training (and thus partial SB training, see below) occurs when C transitions out of *chase* mode. If the time C was It is outside of the desired range (25–45 s), C is trained to become slower or faster, as appropriate, by a factor depending on how far outside of the desired range C was It.
- *Base-running maintenance*: despite the effects of speed-only learning, C should only minimally change the value a_C of the average number of bases reached during each *runBases* behavior. SB training occurs when C transitions out of *runBases* mode: a_C is updated, and coefficients in cognitive differential equations are altered to train C to approach the control value of 4.01 in the future. As a partial example, if $a_C < 4.01$, coefficients in the differential equation governing *iRun* are altered so that C tends to remain longer in *runBases*, encouraging greater a_C in the future. The amounts altered depend on values such as the velocity of C when training occurs, exemplifying cognitive–physical integration: values of physical variables affect cognitive adjustments.

To focus these demonstrations, the connections modified during training were pre-selected, though the adjustments were autonomous.

Feedback for SB learning is given by the expected two measurements: how long until C tagged another player (agent or user) when C was It; and how many times C reached a base when in *runBases* behavior. Learning occurs when C transitions out of two behaviors:

- When C transitions out of *chase*, if the time t that agent C was It is less than minimum desired time $t_{\min}$ (here, $t_{\min}$ is 25 s), then C becomes slower, multiplying its speed by $1 - m * \frac{(t_{\min} - t)}{t_{\min}}$, where $m = 0.05$ controls the effect of the change. This is designed to approach $1 - m$ when C tags its target almost immediately, for maximal change, and approach 1 when C tags its target near the time of $t_{\min}$, for minimal change. Similarly, if C takes more than some maximum desired time $t_{\max}$ (here, 45 s) to tag a player, then it becomes faster, multiplying its velocity by the similarly designed factor $1 + m * \frac{(t_{\max} - t)}{t_{\max}}$.
- When C transitions out of *runBases*, learning occurs under two circumstances: either when the agent transitions out because another behavior's intention becomes highest-active and the agent has touched more or fewer bases than the desired number, or when the agent is tagged by an It player after having touched more bases than the desired number. (If the agent is tagged after having touched fewer bases than the desired number, it is not possible to know whether it would have touched fewer or more bases than the desired number, and learning does not happen in that situation.) In either of these cases, the value of the average number of bases touched by C (call it a_C) each time it was in *runBases* is re-computed, and if that average is either more or less than the desired average, coefficients in C's cognitive system are altered, based on agent-specific *learning factor lF*.

Cognitive interconnections can be altered in two ways by this learning procedure: multiplying by *lF*, or multiplying by $\frac{1}{lF}$. Coefficients in cognitive differential equations for which higher values would intuitively make C evolve out of *runBases* faster are multiplied by *lF*; coefficients for which high values would intuitively make C evolve out of the behavior more slowly are multiplied by $\frac{1}{lF}$. Therefore, if C were touching too many bases per runBases behavior and needed to transition out of it more quickly, *lF* would be increased during play, and coefficient-altering learning would be applied. Similarly, *lF* would be decreased if C needed to stay in *runBases* longer to achieve its goals.

Learning factor *lF* is defined as $lF = s \cdot f$, the current speed *s* of *C* multiplied by a factor *f*, the value of which is described below. This way, an increase (decrease, respectively) in *s* straightforwardly results in a tendency to touch more (fewer) bases before transitioning out of *runBases*. Factor *f* is also varied during gameplay. Initially, $f = \frac{1}{s_{\text{init}}}$, where s_{init} is the speed of *C* at the beginning of the game (for these demonstrations, that speed was $\frac{1}{50}$ of the size of the Tag field per simulated second). Then, the value of *f* can be changed when *C* transitions out of *runBases*, as part of the learning procedure. Any time the agent exits *runBases*, the average number of bases a_C touched during the *runBases* behavior is computed; then, if at the time of the current transition out of *runBases*, a_C is greater than the desired value of a_{ctrl} (here, 4.01), *f* is multiplied by 1.2. Similarly, if a_C is below 4.01 at the time of a voluntary transition, *lF* is multiplied by 0.8. This definition of *f* and its alteration during play completes the definition of *lF* to have the desired properties.

With this definition of *lF*, learning is implemented by altering cognitive coefficients as appropriate. For this paper, the coefficients intuitively presumed to increase the rate at which the system evolves out of *runBases*, and which were therefore multiplied by *lF*, are: in the equation for $\dot{I}_{protect}$, the coefficient for $D_{protect}$, and a positive constant term; in the equation for $\dot{I}_{runBases}$, the coefficients for I_{tag}, $I_{protect}$, I_{mitten}, and a negative constant term. Similarly, the coefficients multiplied by $\frac{1}{lF}$ are: in the equation for $\dot{I}_{runBases}$, the coefficients for $D_{runBases}$ and $I_{runBases}$. This is not meant to be a comprehensive list of all coefficients that intuitively affect the speed with which *C* evolves out of *runBases*, but rather a sample sufficient to affect the behavior of *C* and illustrate ideas of DI-HDCA learning.

Demonstrations showed agent *C* successfully learning integrated cognitive–physical behavior during play: *C* slowed to spend more time as It before tagging another player (average: 32.62 s) while also maintaining a bases average of $a_C = 4.21$, very close to 4.01. Additionally, speed-only learning without full SB learning resulted in $a_C = 2.19$ in otherwise identical conditions, suggesting the importance of integrated learning for the desired goal. [see Aaron et al. (2011) for additional details.] To quantitatively analyze performance and test the hypothesis that the type of learning alters the performance of the agents, univariate ANOVA was used, with results presented in **Figure 12**. For the dependent measure of number of bases touched, the mean value for speed-and-bases learning was significantly higher [$F(1, 18) = 15.358$, $p = 0.001$, $\eta^2 = 0.460$] than the mean for the speed-only learning (**Figure 12A**). For the variance in the number of bases touched, the mean value for speed-and-bases learning was significantly higher [$F(1, 18) = 131.624$, $p < 0.000$, $\eta^2 = 0.880$] than the mean for the speed-only learning (**Figure 12B**).

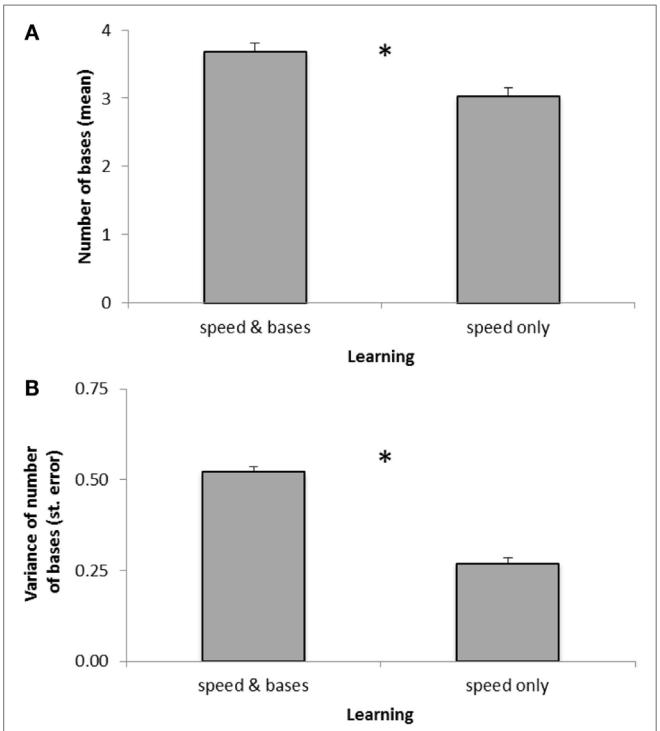


FIGURE 12 | Learning in the Tag game. **(A)** Number of bases that agents touched. With learning based on speed and number of bases, agents touched significantly more bases. **(B)** Variance in number of bases touched. With learning based on speed and number of bases, agents had significantly greater variance.

5.3. Discussion

The above results show DI-HDCAs' cognitive–physical integration as a substrate for online learning of multi-faceted, real-time interactive gameplay. The cognitive–physical integration makes the learning straightforward for DI-HDCAs: the extent to which speed or the agent's cognitive network needs to be modified is not known *a priori*, but cognitive–physical integration enables small adjustments in one to bring about small adjustments in the other, so the integrated agent system can find the desired balance.

Other experiments presented in Aaron et al. (2011), although illustrative of cognitive–physical integration in DI-HDCAs, were not related to agent learning and hence not presented above. Specific values were varied in controlled environments, to investigate the particular effects that might result. For example, many game segments were simulated with identical CTA *C*; initially, *C*'s intentions implied task order [*readyToTag*, *runBases*, *protect*, *getMitten*]. Across simulations, two factors varied – when *C* dropped its mitten; and when *C* was tagged by the user (automated, for replicability) – to illuminate micro-level cognitive and physical effects in gameplay. As *mitten-drop* grew later with *get-tagged* held constant, for example, the time at which *C* moved from *readyToTag* into *runBases* was not affected, but the time at which *C* then entered *protect* tended to get *earlier*. In addition, for particular values of *mitten-drop* and *get-tagged*, *C* entered *protect* mode – in which movement is not obstacle-avoidant – at an inopportune moment and ran straight into an It player. This sequence of events and ensuing cascade of effects illustrates how engaging, unscripted behavior that could be considered emergent can arise in the DI-HDCA framework. Emergent behavior and the DI-HDCA framework are also briefly discussed in section 6 below.

6. CONCLUSION AND DISCUSSION

The DI-HDCA modeling framework is a fusion of ideas from BDI theory, spreading activation networks, and hybrid dynamical system models, each adapted and employed in new ways that are influenced by entailments of environment and embodiment. DI-HDCA modeling embraces BDI theory and spreading activation networks for cognitive modeling, adapting them to real-time varying environments, continuum-valued representations, and multi-tiered integration across a model. Representing DI-HDCA models in a formal HDS enables behavioral design, and it supports cognitive–physical integration in each behavior mode. Moreover, because all physical and cognitive elements have the real-time evolution of their activation values expressed by differential equations in the same dynamical system, any elements can affect any other in the integrated agent model.

The DI-HDCA framework's expansive integration also supports the agent learning demonstrated in sections 4 and 5 of this paper, employing both reactive–deliberative and cognitive–physical integration for adaptive behavior of navigating, goal-directed agents. Experiments demonstrate that DI-HDCA modeling can enable the distribution of typically deliberative task sequencing intelligence onto reactive-level processes, and that cognitive–physical integration can enable straightforward online learning in interactive simulations. These experiments are not an exhaustive demonstration of the capacity of DI-HDCA models nor a *full* exploration of the integration and adaptation possible for DI-HDCAs – for example, they considered the reactive–deliberative and cognitive–physical dimensions independently, not jointly – but they illuminate the role of this integrated intelligence modeling and suggest the value of further exploration.

There are many possible directions in which the presently described DI-HDCA framework could be extended. In the general context of reactive and deliberative systems, extensions of dynamical intention-based reactive systems illustrated here could potentially serve as reactive adjuncts to deliberation in hybrid reactive–deliberative systems, augmenting deliberative methods with enhanced reactive intelligence; this could reduce reliance on deliberation and extend reactive benefits of responsiveness and adaptability in incompletely known environments. There is also the perhaps more ambitious potential that DI-HDCA models could extend to fully replace some deliberative systems, representing the necessary rule-based behavior in the reactive DI-HDCA framework. Neither of these approaches is currently fully explored, and it is not the intent of this paper to prescribe one of these two approaches or endorse one over the other; both seem interesting to explore.

The specific details of DI-HDCA modeling presented in this paper can also be altered in further explorations. For example, in this paper, one activation value represents both salience and cognitive intensity or commitment "in mind," but those qualities need not be conflated: within this general modeling framework, agents could be very aware (high salience) of a mild desire (low intensity), with individual elements in the cognitive networks representing each of those qualities; the models presented in this paper could straightforwardly adopt such new elements in their cognitive dynamical systems. In addition, deliberation could be modeled differently in the DI-HDCA framework, with specific deliberation-behavior modes that represent the time during deliberation; these could be incorporated without altering reactive cognitive representations. Such extensions were not necessary, however, for the demonstrations of reactive-level learning and cognitive–physical integration in this paper.

Even within the models already developed, the capacity of dynamical intention modeling to enhance reactive-level intelligence and minimize reliance on deliberation is not confined to agent learning methods such as those presented above. Reactive task re-sequencing for DI-HDCAs, as discussed in Aaron and Admoni (2009), can enable agents with internally inconsistent cognitive elements to smoothly correct inconsistencies without deliberation: an agent with a high-active intention I_{ML} to mail a letter but also a high-active belief $B_{\overline{ML}}$ that it does not have a letter to mail can reactively re-order its task sequence, without propositional planning. The cognitive network enables the high activation on $B_{\overline{ML}}$ to have an inhibitory affect on I_{ML} until *mailLetter* is no longer a high-priority task for the agent; indeed, in the demonstration reported in Aaron and Admoni (2009), the activation on I_{ML} becomes negative and the *mailLetter* task is not completed, consistent with the belief. The agent can invoke deliberative planning when needed, but for this cognitive inconsistency, reactive activation changes governed by cognitive differential equations suffice for task re-sequencing.

Expanding the scope of planning in DI-HDCAs could involve a deeper exploration of reactive planning. At present, a plan for DI-HDCAs is represented as a sequence of activation values on intentions: at any moment, the plan is the ordering of those intentions from high priority (to be completed first) to low priority. Additional study of mechanisms for planning, and for reasoning about time in this modeling context, could yield both interesting cognitive insights and more robust, reliable robots. Relatedly, applications of DI-HDCA modeling to agents with predictive intelligence is also a potentially productive extension. Because DI-HDCA models are based on differential equations, there is inherently a predictive model in the system: at any moment, the current values of time derivatives could straightforwardly be employed to linearly extrapolate any system value to any time in the future. This capacity is not tested in the present work in this paper, but it might be employed to further enhance DI-HDCA behavior, including incorporating such predictions into agent learning methods.

The modeling of DI-HDCAs as hybrid dynamical systems is also influenced by concerns of agent reliability. There are formal logics and computational methods to analytically verify some properties of hybrid dynamical systems, and in principle, a DI-HDCA model could perhaps be analytically proved to be designed correctly according to specifications. (Indeed, some STAs in the Tag scenario had designs amenable to verification, although that analysis was not performed.) In practice, however, it is extremely difficult to analyze properties of arbitrarily complex hybrid dynamical systems; indeed, reasoning about approximations to a system may be needed in cases where exact reasoning about the desired system is computationally impossible (Alur et al., 1995, 2000). For that reason, verifiability is not presently

a primary concern underlying DI-HDCA modeling, but as the verifiable correctness of complex computational agents becomes more important, it may become more beneficial to have models of intelligent robots grounded in a framework that enables verification. Moreover, there are promising HDS-related approaches to creating verifiably correct behaviors, such as synthesizing robot controllers from formal specifications [e.g., Wong et al. (2014)]; the gap between such approaches and DI-HDCA modeling is sizable, but less than the gap between such approaches and models without formal foundations.

The DI-HDCA framework may also be an apt candidate for studying emergence and mechanisms of emergence. The DI-HDCA framework enables and encourages low-level behavior design while also expressing higher-level behavioral abstractions. On a fundamental level, these are the elements needed to begin an analysis of emergent behaviors: a lower level, with respect to which behaviors can be emergent; a higher level, in which emergent behaviors can be described; and a formalized foundation in which patterns can be recognized and considered emergent. Consider, for instance, how artificial neural networks can be parts of studies involving emergence: behaviors arise that are not readily or properly described as behaviors of the network itself. Similarly, any higher-level agent behavior would not be considered emergent with respect to a system if it is already encoded in that system. With DI-HDCAs, the high-level behaviors explicitly represented as HDS modes could be a baseline against which newly recognized behaviors could be compared for determining emergence; such potentially emergent behaviors could arise from low-level cognitive and physical dynamics and interconnections, analogous to behavior arising from a neural network, without explicit high-level encoding. Moreover, because of the flexibly expressive HDS modeling, a wide variety of candidate mechanisms for generating or recognizing potentially emergent behaviors could be implemented, for a formalized approach to studying emergence.

Embodied robots are complex integrated systems, and DI-HDCA modeling represents that complexity in a structured framework that enables effective analysis and design, with new approaches to integrated intelligence and learning that can improve robot performance. Although extensions of the present work could explore narrowly construed task domains (e.g., an automated robot arm for manufacturing, designed to make only one specific weld), that is not suggested here. By design, the DI-HDCA framework is not primarily for narrowly delineated, domain-specific problems; instead, it illustrates what a modeling framework for integrated embodied intelligence might contain, which can be broadly applied to complex scenarios. For the general study and robust implementation of embodied intelligence, models expressing both broad scope and integration seem well suited, and the DI-HDCA modeling framework is designed for behaviors both low-level and high-level, both cognitive and physical, and their interactions in embodied agents.

AUTHOR CONTRIBUTIONS

EA conceived of the theory, worked with collaborators (see Acknowledgments) to conduct experiments and analyses, and wrote the manuscript.

ACKNOWLEDGMENTS

The author gratefully acknowledges: Henny Admoni for the observation that led to the Hebbian learning approach in this paper and in Aaron and Admoni (2010); Henny Admoni and Juan Pablo Mendoza for contributions to simulations, data, and figures in papers [e.g., Aaron and Admoni (2010) and Aaron et al. (2011)] foundational to the current presentation; and John Long for statistical analyses and figures in this paper. Thanks also to all of the above, to Jim Marshall for especially inspirational conversations that helped advance this work, and to reviewers of this paper for thoughtful and helpful comments.

FUNDING

A Research Committee award from The Lucy Maynard Salmon Research Fund of Vassar College funded publication of this article.

REFERENCES

Aaron, E., and Admoni, H. (2009). "A framework for dynamical intention in hybrid navigating agents," in *Hybrid Artificial Intelligence Systems* (Berlin, Heidelberg: Springer-Verlag), 18–25.

Aaron, E., and Admoni, H. (2010). Action selection and task sequence learning for hybrid dynamical cognitive agents. *Rob. Auton. Syst.* 58, 1049–1056. doi:10.1016/j.robot.2010.05.006

Aaron, E., Ivančić, F., and Metaxas, D. (2002). "Hybrid system models of navigation strategies for games and animations," in *HSCC 2002, Lecture Notes in Computer Science* (Berlin, Heidelberg: Springer-Verlag), 7–20.

Aaron, E., and Mendoza, J. P. (2011). "Dynamic obstacle representations for robot and virtual agent navigation," in *Proceedings of the Canadian Conference on Artificial Intelligence* (Heidelberg, New York: Springer-Verlag), 1–12.

Aaron, E., Mendoza, J. P., and Admoni, H. (2011). "Integrated dynamical intelligence for interactive embodied agents," in *ICAART 2011 – Proceedings of the 3rd International Conference on Agents and Artificial Intelligence* (Setubal: SCITEPRESS), 296–301. doi:10.5220/0003188102960301

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., Ho, P.-H., Nicollin, X., et al. (1995). The algorithmic analysis of hybrid systems. *Theor. Comp. Sci.* 138, 3–34. doi:10.1016/0304-3975(94)00202-T

Alur, R., Henzinger, T., Lafferriere, G., and Pappas, G. (2000). Discrete abstractions of hybrid systems. *Proc. IEEE* 88, 971–984. doi:10.1109/5.871304

Arkin, R. C. (1990). Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Rob. Auton. Syst.* 6, 105–122. doi:10.1016/S0921-8890(05)80031-4

Beer, R. D. (2000). Dynamical approaches to cognitive science. *Trends Cogn. Sci.* 4, 91–99. doi:10.1016/S1364-6613(99)01440-0

Bratman, M. (1987). *Intentions, Plans, and Practical Reason*. Cambridge, MA: Harvard University Press.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE J. Robot. Autom.* RA-2, 14–23. doi:10.1109/JRA.1986.1087032

Collins, A. M., and Loftus, E. F. (1975). A spreading activation theory of semantic priming. *Psychol. Rev.* 82, 407–428. doi:10.1037/0033-295X.82.6.407

Egerstedt, M. (2000). "Behavior based robotics using hybrid automata," in *HSCC 2000 Lecture Notes in Computer Science* (Berlin, Heidelberg: Springer-Verlag), 103–116.

Erlhagen, W., and Bicho, E. (2006). The dynamic neural field approach to cognitive robotics. *J. Neural Eng.* 3, R36–R54. doi:10.1088/1741-2560/3/3/R02

Gat, E. (1998). "On three-layer architectures," in *Artificial Intelligence and Mobile Robots*, eds D. Kortenkamp, R. P. Bonnasso, and R. Murphy (Menlo Park, CA: AAAI Press), 195–210.

Georgeff, M., and Lansky, A. (1987). "Reactive reasoning and planning," in *AAAI-87* (Menlo Park, CA: AAAI Press), 677–682.

Goldenstein, S., Karavelas, M., Metaxas, D., Guibas, L., Aaron, E., and Goswami, A. (2001). Scalable nonlinear dynamical systems for agent steering and crowd simulation. *Comput. Graphics* 25, 983–998. doi:10.1016/S0097-8493(01)00153-4

Haazebroek, P., van Dantzig, S., and Hommel, B. (2011). A computational model of perception and action for cognitive robotics. *Cogn. Process.* 12, 355–365. doi:10.1007/s10339-011-0408-x

Hebb, D. O. (1949). *The Organization of Behavior*. New York, NY: John Wiley & Sons, Inc.

Hertzberg, J., Jaeger, H., Morignot, P., and Zimmer, U. (1998). "A framework for plan execution in behavior-based robots," in *Proceedings of ISIC/ISAS* (Piscataway, NJ: IEEE).

Jaeger, H., and Christaller, T. (1998). Dual dynamics: designing behavior systems for autonomous robots. *Artif. Life Rob.* 2, 108–112. doi:10.1007/BF02471165

Large, E., Christensen, H., and Bajcsy, R. (1999). Scaling the dynamic approach to path planning and control: competition among behavioral constraints. *Int. J. Robot. Res.* 18, 37–58. doi:10.1177/027836499901800103

Maes, P. (1989). "The dynamics of action selection," in *IJCAI-89* (San Mateo, CA: Morgan Kaufmann), 991–997.

Pfeifer, R., and Bongard, J. (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. Cambridge, MA: MIT Press.

Port, R., and van Gelder, T. J. (1995). *Mind as Motion: Explorations in the Dynamics of Cognition*. Cambridge, MA: MIT Press.

Richter, M., Sandamirskaya, Y., and Schöner, G. (2012). "A robotic architecture for action selection and behavioral organization inspired by human cognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (Piscataway, NJ: IEEE), 2457–2464.

Sandamirskaya, Y., Zibner, S. K. U., Schneegans, S., and Schöner, G. (2013). Using dynamic field theory to extend the embodiment stance toward higher cognition. *New Ideas Psychol.* 31, 322–339. doi:10.1016/j.newideapsych.2013.01.002

Schöner, G., Dose, M., and Engels, C. (1995). Dynamics of behavior: theory and applications for autonomous robot architectures. *Robot. Auton. Syst.* 16, 213–245. doi:10.1016/0921-8890(95)00049-6

Spivey, M. (2007). *The Continuity of Mind*. New York, NY: Oxford University Press.

van Gelder, T. (1998). The dynamical hypothesis in cognitive science. *Behav. Brain Sci.* 21, 615–665. doi:10.1017/S0140525X98001733

Wong, K. W., Ehlers, R., and Kress-Gazit, H. (2014). "Correct high-level robot behavior in environments with unexpected events," in *Robotics: Science and Systems Conference (RSS14)*. doi:10.15607/RSS.2014.X.012

Conflict of Interest Statement: The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 12 July 2016
doi: 10.3389/frobt.2016.00040

Quality Diversity: A New Frontier for Evolutionary Computation

*Justin K. Pugh, Lisa B. Soros and Kenneth O. Stanley**

Evolutionary Complexity Research Group, Department of Computer Science, University of Central Florida, Orlando, FL, USA

While evolutionary computation and evolutionary robotics take inspiration from nature, they have long focused mainly on problems of performance optimization. Yet, evolution in nature can be interpreted as more nuanced than a process of simple optimization. In particular, natural evolution is a divergent search that optimizes locally within each niche as it simultaneously diversifies. This tendency to discover both quality and diversity at the same time differs from many of the conventional algorithms of machine learning, and also thereby suggests a different foundation for inferring the approach of greatest potential for evolutionary algorithms. In fact, several recent evolutionary algorithms called *quality diversity (QD) algorithms* (e.g., novelty search with local competition and MAP-Elites) have drawn inspiration from this more nuanced view, aiming to fill a space of possibilities with the best possible example of each type of achievable behavior. The result is a new class of algorithms that return an archive of diverse, high-quality behaviors in a single run. The aim in this paper is to study the application of QD algorithms in challenging environments (in particular complex mazes) to establish their best practices for ambitious domains in the future. In addition to providing insight into cases when QD succeeds and fails, a new approach is investigated that hybridizes multiple views of behaviors (called *behavior characterizations*) in the same run, which succeeds in overcoming some of the challenges associated with searching for QD with respect to a behavior characterization that is not necessarily sufficient for generating both quality and diversity at the same time.

Keywords: novelty search, non-objective search, quality diversity, behavioral diversity, evolutionary computation, neuroevolution

OPEN ACCESS

Edited by:
John Howard Long, Vassar College, USA

Reviewed by:
Andrés Faíña Rodríguez-Vila, IT University of Copenhagen, Denmark
Leonardo Trujillo, Tijuana Institute of Technology, Mexico

***Correspondence:**
Kenneth O. Stanley
kstanley@cs.ucf.edu

Specialty section:
This article was submitted to Evolutionary Robotics, a section of the journal Frontiers in Robotics and AI

Received: *31 March 2016*
Accepted: *23 June 2016*
Published: *12 July 2016*

Citation:
Pugh JK, Soros LB and Stanley KO (2016) Quality Diversity: A New Frontier for Evolutionary Computation. Front. Robot. AI 3:40. doi: 10.3389/frobt.2016.00040

1. INTRODUCTION

The products of nature have long served as inspiration for the investigation and practice of evolutionary algorithms and evolutionary robotics (Cliff et al., 1993; Nolfi and Floreano, 2000; Stanley, 2011). Yet the ability of such algorithms to match the complexity and sophistication of nature has frustratingly lagged, as researchers in the fields often observe (Stanley and Miikkulainen, 2003; Doncieux et al., 2015). This observation then often becomes the motivation for developing more sophisticated algorithms and encodings. Yet, even more fundamental than the question of how to abstract the most brilliant achievements of nature into an algorithm of commensurate power is a question less often explicitly asked: for what practical purpose is evolution actually suited anyway?

Until recently, throughout a large swath of the field, the implicit yet resounding answer to this question has been *objective optimization* (Mitchell, 1997; De Jong, 2002; Bishop, 2006). Various early pioneers in evolutionary computation independently inferred from their observations of nature that evolution can serve when abstracted artificially as a powerful optimization algorithm (Fogel et al., 1966; Holland, 1975; Goldberg and Richardson, 1987; Goldberg, 1989; Schwefel, 1993). The

discoveries of evolution in nature, such as the flight of birds or the intelligence of the human brain, suggested to these pioneers that if fitness pressure is calibrated to push selection toward an ambitious objective then evolution can become a tool of directed design and creation.

The casting of evolution as an algorithm for optimization naturally pitted evolutionary computation against the many subfields of machine learning invested in optimization, leading to conflicts and critiques, sometimes portraying evolutionary computation as *ad hoc*, unprincipled, less effective than other more theoretically based optimization methods, or even outdated (and therefore often given less room in modern texts on machine learning) (Bäck et al., 1997; Bishop, 2006). While sometimes overly harsh or uninformed, even if such critiques are accepted, the perplexing question still hangs in the background of how it is possible that evolution in nature has managed to produce artifacts far beyond the capacity of *any* subfield of machine learning or optimization. If evolution is apparently so unmatched in power in nature, then why is there even a debate about its ability to compete with other approaches to optimization?

While one possible answer is that we have yet to uncover the deepest principles that unlock its true potential as an objective optimizer, a more intriguing possibility is that the real virtue of evolution is not in the end optimization at all. This suggestion goes beyond Herb Simon's assertion that evolution is a *satisficer* rather than an optimizer (Simon, 1957), which casts evolution almost as a poor man's optimizer. Rather, the hypothesis is that evolution is indeed phenomenally virtuosic at *something*, but that something is simply not optimization. This perspective can help to explain how it could be possible for evolution to produce sensational results in nature yet frustratingly modest ones in computation: *we may be using it wrong*. Perhaps the analogy with optimization was a mistake.

Indeed, it is difficult to imagine that evolution in nature is structured in the same way as a conventional optimization algorithm: there is no obvious unifying objective and organisms are often rewarded for being different in addition to being better. For example, organisms that are sufficiently different from their predecessors may establish a new *niche* in which they enjoy greatly reduced competition and thus are more likely to survive (Kirschner and Gerhart, 1998; Lehman and Stanley, 2013). Contrary to the tendency of optimization algorithms to converge over time to a single "best" solution, natural evolution instead exhibits a remarkable tendency toward *divergence* – continually accumulating myriad different ways of being. This observation is the crux of an alternative perspective in evolutionary computation (EC) that has been gaining momentum in recent years: *evolution as a machine for diversification* rather than optimization.

Inspired by this alternate view of natural evolution's apparent strength as a diversifier, a new evolutionary algorithm called novelty search (NS) (Lehman and Stanley, 2008, 2011a) was introduced, which searches only for behavioral diversity without any underlying objective pressure. Surprisingly, in some domains (particularly those that are deceptive), NS quickly finds the global optimum even when objective-based approaches consistently fail. The counterintuitive result that NS can sometimes find the best solutions without explicitly searching for them has since sparked considerable research interest in applying NS and methods like it to solving problems that were previously considered to be too difficult (Lehman and Stanley, 2008, 2010, 2011a,b; Kistemaker and Whiteson, 2011; Mouret, 2011; Risi et al., 2011; Mouret and Doncieux, 2012; Cully and Mouret, 2013; Gomes and Christensen, 2013; Gomes et al., 2013; Liapis et al., 2013b; Martinez et al., 2013; Naredo and Trujillo, 2013). Novelty search has effectively demonstrated that evolution's talent for diversification can itself be harnessed as a powerful tool for seeking a near-optimum, instead of the conventional notion of "survival of the fittest." However, this view ignores the intrinsic value of diversity itself, treating it merely as a "means to an end" of finding the global optimum as usual.

In a true departure from conventional optimization, which seeks only the single best-performing solution, a new search paradigm has begun to emerge within EC where the effort focuses instead on finding various viable solutions, similar to how evolution in nature has discovered over billions of years a vast assortment of unique species, each of which are capable of orchestrating the complex system of biological processes necessary to sustain life. More precisely, the goal of this new type of search, called *quality diversity* (QD), is to find a maximally diverse collection of individuals (with respect to a space of possible behaviors) in which each member is as high performing as possible. In service of this goal, QD algorithms, such as novelty search, with local competition (NSLC) (Lehman and Stanley, 2011b) and MAP-Elites (Mouret and Clune, 2015) carefully balance a drive toward increasing diversity with localized searches for quality in an analogy with nature where species face the strongest competition from within their own niche. In this way, search can move toward different behaviors, while simultaneously improving behaviors that have already been discovered.

An important aspect of QD that differentiates it from other approaches designed to return multiple results is that in QD, diversity between individuals is measured with respect to their *behavior* (the actions and features of an individual over the course of its lifetime). The experimenter selects some subset of behavioral features of interest to form a *behavior characterization* (BC), thus defining a space of possible behaviors. The assumption in QD is that all parts of the behavior space are considered equally important. This assumption contrasts with non-QD approaches that assign priority to higher-performing regions, which draw inspiration from the idea of returning multiple local optima (Mahfoud, 1995; Trujillo et al., 2008, 2011). Instead, the goal of QD is to sample all regions of the behavior space (at some granularity), returning the best possible performance within each region. In other words, diversity takes priority over quality[1] and therefore QD algorithms must be careful to avoid driving search away from lower-performing regions. More formally, the behavior space must be divided into t niches $\{N_1, \ldots, N_t\}$ that together cover the entire space. That is, every point in the behavior space belongs to some niche N_i. Then, the task of QD is to maximize a quality measure Q within every niche.[2]

[1] Approaches that desire to return a handful of the best local optima (i.e., where quality takes priority over diversity) may be better served by the term *diverse quality*.

[2] For any niche N_i where no point has been discovered, Q_i is defined as 0.

Although QD algorithms can be applied to traditional optimization-oriented tasks where they may even perform well due to their ability to overcome the problem of deception, the deeper promise of QD is to push beyond what is possible through simple optimization. In particular, QD has potential applications in the areas of computational creativity (Boden, 2006) and open-ended evolution (Standish, 2003; Bedau, 2008), where the hope is to automatically generate an endless procession of uniquely interesting artifacts. In more restricted search spaces, QD algorithms have also been called "illumination algorithms" because they effectively reveal the best possible performance achievable in each region of the phenotype space (Mouret and Clune, 2015). The types of problems inspired by QD (many of which are discussed in the next section) favor approaches that explore many promising directions at the same time and thus represent an opportunity for evolutionary algorithms to establish a more unique profile within the broader machine learning community where focused single-solution approaches such as backpropagation (Rumelhart et al., 1986) and support vector machines (Cortes and Vapnik, 1995) have historically dominated.

To help accelerate research in this emerging area, the aim of this paper is to establish a standard framework for understanding and comparing different approaches to searching for QD. [This paper is in effect a major expansion on the theme of our earlier conference paper that first introduced the term *quality diversity* (Pugh et al., 2015).] The hope is to unify early works in this emerging field and to promote the design of better QD algorithms in the future. To that end, this paper introduces a benchmark domain in the form of a series of maze-navigation tasks of varying difficulty that are paired with a quantifiable measure of the performance of QD algorithms called the *QD-score*. Experimental results in these mazes comparing current state-of-the-art approaches as well as several novel variants thereof reveal important insights into the application of QD algorithms that extend beyond individual methods.

One such insight concerns the importance of considering how diversity is *characterized* when applying QD algorithms. Specifically, some choices of characterization can make finding QD more difficult, which on sufficiently deceptive problems can translate into an inability to find the best solutions altogether. This apparent weakness presents a problem for researchers interested in finding QD with respect to a non-optimal characterization (i.e., one which inhibits finding the best-performing individuals) because standard practice suggests driving search with the same notion of diversity that you are ultimately interested in discovering (Trujillo et al., 2008, 2011; Lehman and Stanley, 2011b; Cully and Mouret, 2013; Szerlip and Stanley, 2013; Mouret and Clune, 2015). A solution to this problem is presented and then empirically validated in the form of new QD algorithms that drive search with multiple characterizations simultaneously. These new multi-characterization approaches, together with an increased understanding of the types of characterizations that are ideal for driving search effectively, enable the application of QD algorithms to various more difficult domains in the future.

Nature has discovered organisms both diverse and highly optimized within their own niches. This kind of divergent creative phenomenon differs from the typical convergent objective-driven process seen in search algorithms across machine learning, evolutionary computation (EC), and evolutionary robotics. By bringing QD now to the forefront of EC and providing a framework for understanding and comparing its available algorithms, the hope is that the field can progress more confidently from this initial foundation. In some cases (as later results will show), QD may even produce results beyond what an objective-driven process can accomplish, but it more broadly offers the chance to uncover a large swath of uniquely intriguing possibilities within a vast space and during just a single run.

2. BACKGROUND

This section begins with a review of historical precedent for the study of QD, followed by highlights of recent works in the area and the questions they raise.

2.1. Before QD

Early work in *multi-modal function optimization* [MMFO; Mahfoud (1995)] foreshadowed the later arrival of QD. The aim of MMFO is to discover multiple local optima within a search space, which naturally yields a diversity of solutions. However, its main difference from QD is that MMFO traditionally focuses on *genetic* diversity and tends to apply only to simple phenotypes, such as mathematical functions, where the genotype and phenotype are in effect the same (Mahfoud, 1995); QD reflects a later shift in interest toward *behavioral* diversity and is often applied in domains such as evolutionary robotics where the relationship between genome and behavior is complex. The main limitation of genetic diversity is that it is susceptible to genetic aliasing, which means that genomes that are different may nevertheless behave similarly. Such aliasing, which is amplified especially in the presence of indirect genotype to phenotype mappings (Hornby and Pollack, 2001, 2002; Bongard, 2002; Stanley and Miikkulainen, 2003; Stanley, 2007), is thus counterproductive to find various behaviors, as shown empirically by Trujillo et al. (2011).

Another subject of research related to QD is multi-objective optimization (MOO) (Deb et al., 2002). In MOO, the search algorithm aims to uncover the key trade-offs (called the Pareto front) among two or more objectives set by the user. Similar to QD, MOO returns a set of top candidates rather than a single winner, but MOO is still ultimately driven toward specific objectives (though more than one) and therefore intrinsically convergent. By contrast, QD is a genuine *divergent* form of search driven explicitly to move *away* in the search space from where it has visited before. The unique effect is thus to reveal a sampling of the spectrum of possible behaviors latent in a search space.

2.2. Early Divergent Search Algorithms

Interest in divergence in evolutionary algorithms was sparked initially by the surprising observation that some problems are solved more reliably by searching for novel behaviors than by searching for their objective (Lehman and Stanley, 2008, 2011a). This approach, called *novelty search*, revealed just how pervasive and costly deception can be in otherwise unremarkable domains. It also showed that searching divergently instead of convergently can sometimes sidestep deception to uncover desirable parts of

the search space. The benefits of divergent search were further confirmed through similar experiments by Mouret and Doncieux (2009) and Mouret and Doncieux (2012), who use the synonymous term *behavioral diversity*.

These initial studies were followed by a wave of interest in divergent search algorithms as researchers explored their potential for discovery and open-endedness (Risi et al., 2009, 2011; Soltoggio and Jones, 2009; Doucette and Heywood, 2010; Goldsby and Cheng, 2010; Graening et al., 2010; Krcah, 2010; Kistemaker and Whiteson, 2011; Woolley and Stanley, 2011; Gomes and Christensen, 2013; Gomes et al., 2013; Liapis et al., 2013a,b; Martinez et al., 2013; Morse et al., 2013; Naredo and Trujillo, 2013; Risi and Stanley, 2013). However, a clear missing ingredient from pure novelty or behavioral diversity techniques is a complementary notion of objective quality. The radical shift away from objectives toward divergence discards with it the ability to specify up front what is good and what is bad, and to have that notion influence the search. The loss of this convenient notion set the stage for its reemergence in QD algorithms.

2.3. Quality Diversity (QD) Algorithms

Early works on behavioral diversity (Mouret and Doncieux, 2009, 2012) and some more recent studies of combining novelty with objectives (Gomes et al., 2015) investigate the idea of hybridizing novelty with fitness. The means of such combination is usually through a multi-objective framework (Mouret and Doncieux, 2009, 2012), though a weighted combination is also possible (Gomes et al., 2015). However, in either case the notion of fitness (i.e., quality) is *global*, which means in effect that the component of the search pushing toward quality focuses its effort exclusively on the part of the search space where performance is dominant over all other areas of the search space. Thus, such approaches are not generally focused on QD. In an approach closer to QD, Trujillo et al. (2008, 2011) achieve multiple functional behaviors through behavioral speciation and fitness sharing. However, while this approach finds several locally optimal behaviors, it is still governed by global fitness because it preferentially explores higher-performing niches. A preferential push toward global quality in any approach reintroduces a strong convergent force into the search. While that may help in some cases if the aim is to discover a single near-global optimum or a handful of high-performing local optima, QD algorithms aim instead to explore the entire behavior space.

In particular, the hope in QD is to uncover as many diverse behavioral niches as possible, but where each niche is represented by a candidate of the highest possible quality for that niche. That way, the result is a kind of *illumination* of the best of all the diverse possibilities that exist (Mouret and Clune, 2015). The original QD algorithm of this type, called *novelty search with local competition* (NSLC), hybridizes novelty search with local fitness competitions only among individuals with similar behaviors, yielding a broad population of numerous simultaneous local competitions in diverse behavioral niches (Lehman and Stanley, 2011b). In its first demonstration, NSLC uncovered a collection of effective virtual creature morphologies and walking strategies in a single run, thereby demonstrating the unique benefits of QD (Lehman and Stanley, 2011b).

The potential to uncover a diverse collection of high-quality alternatives in a single run inspired further investigations and algorithms. For example, Szerlip and Stanley (2013) evolve diverse ambulating two-dimensional creatures made of sticks, also in a single run. Cully and Mouret (2013) evolve a collection of walking behaviors for hexapod robots that functions as a repertoire of skills for a robot. In a hint at the wide applicability of QD, Szerlip et al. (2015) evolve diverse low-level feature detectors for neural networks that are later aggregated into a combined classifier network. Revealing that QD encompasses more than a single algorithm, Mouret and Clune (2015) and Cully et al. (2015) introduce an alternate QD algorithm called *multi-dimensional archive of phenotypic elites* (MAP-Elites) that collects elite versions of diverse behavior within individual bins in a behavioral map. In yet another QD application, MAP-Elites collects diverse walking strategies for a robot that can be adapted in response to different kinds of damage (Cully et al., 2015). Other applications of MAP-Elites include generating sets of images for fooling deep networks (Nguyen et al., 2015a), and for exposing the space of concepts encoded inside a deep network as two-dimensional images (Nguyen et al., 2015b) and as three-dimensional models (Lehman et al., 2016).

The quickly expanding set of applications of QD motivates the need for a systematic study of its best practices, which is the aim in this paper. For that purpose, a key concept in any QD algorithm is the *behavior characterization* (BC). The BC is usually a vector that describes the chronology of actions taken by an individual during its evaluation but can also describe other salient aspects of an individual's behavior or phenotype. This vector is then used to compute its novelty compared with other individuals (or its location in the behavior map in MAP-Elites), thereby driving the diversity component of QD. An important property of the BC is its *alignment* with the notion of quality, which refers to the *degree to which finding novelty tends also to lead to higher fitness*. For example, in a maze, if the BC is based on the final position reached, then it is highly aligned because eventually an agent that continues to find new final positions will find the endpoint of the maze. While BC alignment can be difficult to measure *a priori* (just as the shape of fitness landscapes are not known *a priori* for any challenging problems of interest), a BC's degree of alignment can be anticipated by considering two key properties of highly aligned BCs: (1) each behavior is associated with only a narrow range of fitness values (e.g., a robot's final position in a maze is associated with exactly one fitness value) and (2) the maximum possible fitness in adjacent regions of behavior space correlates (e.g., nearby positions in a maze generally have similar fitness).

By contrast, interestingly, most published QD applications involve finding diversity with respect to an *unaligned* BC because usually the notion of diversity that we find interesting is not intrinsically aligned with quality (Lehman and Stanley, 2011b; Cully and Mouret, 2013; Szerlip and Stanley, 2013; Mouret and Clune, 2015). For example, seeking creatures of different morphologies or with different numbers of legs does not naturally lead to higher-quality walking, yet we are nevertheless interested in finding such creatures. So far, QD has succeeded even despite a lack of such alignment, leaning heavily on the quality component of the algorithm to push otherwise unaligned notions of diversity toward

good performance. For example, a previous study of QD that is precedent for the present paper finds that unaligned BCs often lead to a slower discovery of passable solutions (Pugh et al., 2015).

The key question raised by this previous study is whether QD algorithms with unaligned BCs ultimately stop working entirely when the domain is sufficiently hard. The maze used to test unaligned BCs in Pugh et al. (2015) is relatively simple and easy to solve; what would happen with much more complex mazes? This paper takes this next step with mazes of a scale beyond mazes in previous studies of QD and examines the effect on finding QD with respect to unaligned BCs in these new mazes. Not only does this study reveal how QD holds up in more complex domains but it also surveys new strategies for mitigating the effects of BC misalignment (which is usually the most desirable and intuitive way to setup QD). Such insight is not only critical to the future of QD as a nascent field but also important for the general progress of machine learning outside of conventional convergent closed problems, where the potential might be open-ended and the aim to collect broad possibilities rather than to converge to a final answer.

3. DOMAINS

In the experiments described in this paper, simulated wheeled robots navigate mazes of varying complexity. To aid in their navigation, they are equipped with six rangefinder sensors (five of which are evenly distributed across the front half of the robot, and one pointing backwards) and four pie-slice sensors that indicate both the relative direction and distance to the goal. These sensors serve as inputs to an evolved neural controller with only a single continuous output that controls the degree to which the robot turns left or right (the robot always moves forward at a constant speed). The task is to evolve neural controllers that successfully guide the robot to a goal at the end of the maze.

Successfully navigating a maze can be challenging because it requires learning a complex mapping between sensors and effectors. In part, because sources of deception (and thus difficulty) are often visually apparent (e.g., dead ends), maze navigation has become a canonical domain for evolutionary robotics experiments [e.g., Lehman and Stanley (2011a) and Velez and Clune (2014)]. Another benefit of maze domains is that they tend to be computationally inexpensive, allowing many more evaluations than more intensive physics-based simulations. For this reason, maze domains are ideal for studying evolutionary dynamics over long timescales – an endeavor that would be impossible in a more computationally expensive domain.

In this study, three mazes of varying difficulty assess the efficacy of QD algorithms that will be described in the next section. In an important departure from mazes designed as an optimization problem [e.g., the HardMaze domain in Lehman and Stanley (2011a)], where there is often only a single "correct" path, the mazes here are intentionally designed with *multiple* viable paths to reach the goal. This unusual maze design makes it possible to investigate evolution's potential for finding QD. Thus, the task in this study is not simply to find an agent that reaches the goal, but to find all of the different ways of driving through the maze (including those who do not necessarily reach the goal at all).

An important feature of all three mazes is the *deceptive trap* – areas of the maze that appear to be close to the goal (in terms of Euclidean distance) but, because of the presence of obstructions, do not actually offer a short drivable path to the goal. These traps, often in the form of an easily accessible corridor that terminates at a dead end before reaching the goal, represent local optima in the search space, and serve to deceive algorithms that simply follow a gradient of increasing fitness (Lehman and Stanley, 2011a). Thus, mazes containing such traps serve as a metaphor for complex domains in general, where successfully finding solutions requires a clever search algorithm aimed at innovation rather than straightforward optimization.

The first maze, introduced by Pugh et al. (2015) as the "QD-maze" but called the **small maze** (**Figure 1**) in this article, provides an initial example of this multi-path maze design. In this maze, individuals must escape from a single deceptive trap surrounding the start point after which the maze opens up, allowing various possible strategies for reaching the goal. Importantly, agents are given a considerable amount of time to allow for the expression of complicated and roundabout strategies (such as crossing back and forth across the map several times before driving to the goal). The ideal QD algorithm would eventually find *all* strategies for driving around the maze, including those that do not end up reaching the goal.

Of course, the most interesting real-world problems are often complex and challenging, and if maze navigation domains are to serve as proxies for such problems then they too should display non-trivial complexity. For this reason, two additional *gauntlet* mazes are introduced. While each maintains the important design principle of multiple viable paths to the goal, these mazes are intentionally made more difficult by (1) adding several larger and more pronounced deceptive traps, (2) setting strict maximum

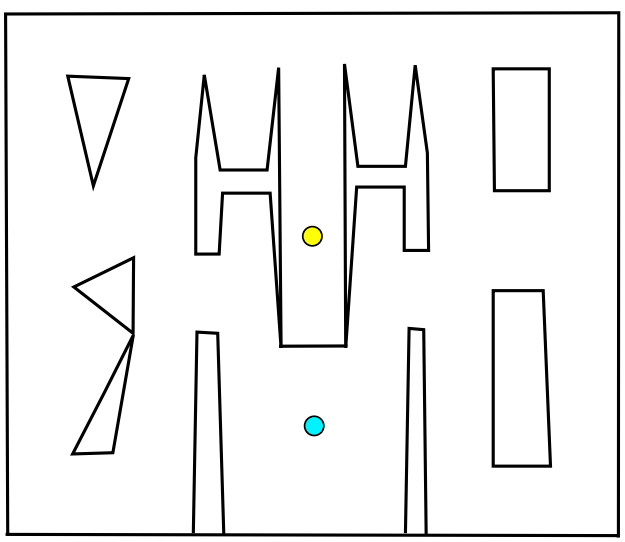


FIGURE 1 | Small maze. Individuals start at the yellow circle (top) and must navigate to the goal point, marked with a blue circle (bottom). The relative openness and lenient time constraint allow a range of different techniques for reaching the goal.

evaluation times such that individuals cannot wander down a deceptive trap and then subsequently reach the goal point, and (3) placing the goal point at the end of a series of chained sub-mazes that are each approximately as hard as the small maze and the original HardMaze from Lehman and Stanley (2011a). Navigating to the end of the **asymmetric gauntlet** (**Figure 2**) and **symmetric gauntlet** (**Figure 3**) thus requires evolving complicated behavioral strategies and overcoming significant levels of deception. In this way, these mazes test both the ability to overcome multiple successive deceptive traps *and* cover the space of possible solutions in the same run. Parameter settings for each maze domain are presented in **Table 1**, which describes differences in maze size, time constraints, and agent sensor radius.

4. ALGORITHMS

This section describes each of the various algorithms considered in this study, beginning with those featured by Pugh et al. (2015). Because the pursuit of quality diversity has only recently become a subject of persistent research interest, there are only two main QD algorithms currently represented in the literature: NSLC (Lehman and Stanley, 2011b) and MAP-Elites (Cully et al., 2015; Mouret and Clune, 2015). To provide a broader perspective on the untapped algorithmic potential in this developing field, this study additionally features several novel variants of these core algorithms including some proposed improvements to MAP-Elites, which is mechanically very simple and thus particularly amenable to modification. Importantly, several other new variants specifically serve to address the problem of finding QD when the desired notion of diversity is unaligned with quality (and thus potentially incapable of finding the best solutions).

All of the following algorithm descriptions are partial in that they describe only selection and population maintenance mechanics for an underlying evolutionary algorithm. Algorithms new to this paper or otherwise not well represented in the literature are described with pseudocode in Appendix A. With the exception of Fitness (Section 4.1.1), which is implemented generationally (the entire population is replaced on every tick), all other algorithms are implemented as steady state (only a small portion of the population is replaced at a time). In particular, batches of 32 genomes are evaluated at a time to facilitate a modest amount of parallelism without substantially disrupting the composition of the population between batches.

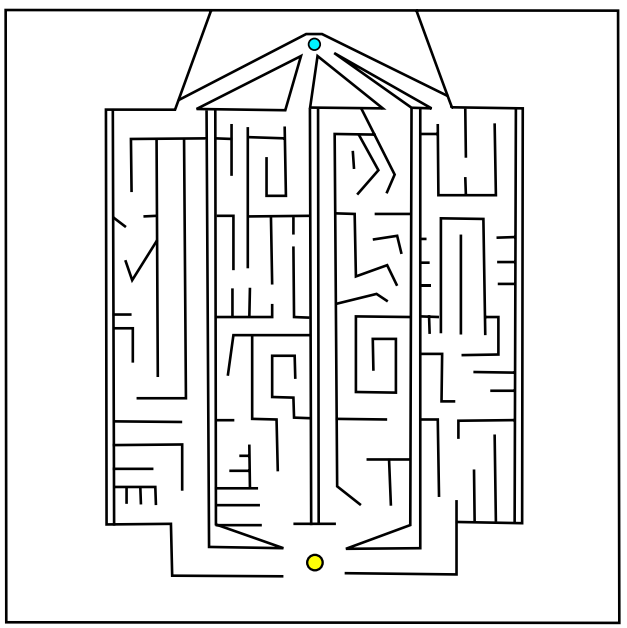


FIGURE 2 | Asymmetric gauntlet. Individuals start at the bottom point and must navigate to the goal at the top of the maze. Because of the variation between maze legs, reaching the goal is easier by some routes than others. The presence of multiple paths through the maze increases the various potential driving strategies that can reach the goal.

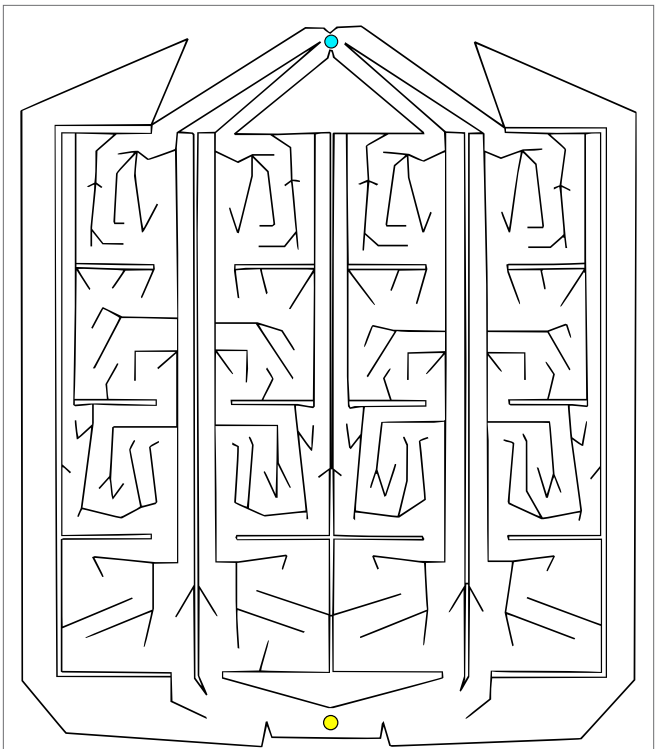


FIGURE 3 | Symmetric gauntlet. Individuals start at the point at the bottom and must navigate to the goal at the top of the maze. Each leg of the maze is an approximate mirror image of the neighboring legs (thus, all paths through the maze are similarly difficult to achieve). The presence of multiple legs allows various different driving strategies to be successful.

TABLE 1 | Maze-specific parameters.

	Small maze	Asymmetrical gauntlet	Symmetrical gauntlet
Evaluation time (ticks)	800	1300	2700
Maze dimensions (units)	900 × 770	6000 × 6000	4600 × 5280
Rangefinder length (units)	100	200	200
Velocity (units/tick)	6	6	6
Max turn rate (rads/tick)	0.21	0.21	0.21

Reflecting differences in design and difficulty, some parameters vary between mazes.

4.1. Controls

While not themselves QD algorithms, both of the following two controls are included in the study to establish the importance of specialized approaches that simultaneously balance drives toward behavioral diversity and locally increasing quality. The controls thus, respectively, exemplify searching with quality pressure alone and with diversity pressure alone.

4.1.1. [Fitness] Neuroevolution of Augmenting Topologies

The first control, which includes only a quality component, represents a traditional objective-oriented optimization approach and is implemented as standard generational NEAT (NeuroEvolution of Augmenting Topologies) (Stanley and Miikkulainen, 2002) with a population size of 500. Agents are rewarded according to the Euclidean distance between their final position and the goal point (this heuristic also underlies the quality component of all of the QD algorithms throughout this paper). NEAT includes a sophisticated speciation and fitness-sharing mechanism for maintaining genetic diversity across the population. However, as the canonical HardMaze experiments (Lehman and Stanley, 2008, 2011a) reveal, genetic diversity is often not enough to overcome the problem of deception on difficult maze-navigation tasks, which instead favor rewarding *behavioral* diversity (a critical component of QD algorithms).

4.1.2. [NS] Novelty Search

The *novelty search* (NS) (Lehman and Stanley, 2008, 2011a) algorithm represents the other extreme: searching only for behavioral diversity with no fixed objective at all (i.e., NS has no "quality component"). Novelty search works by rewarding *novelty* instead of fitness, where novelty measures how different an individual's behavior is from those who have been seen before. More formally, novelty is calculated by summing the distances to the k-nearest behaviors (in this paper, $k = 20$) from a set composed of the current population and an *archive* of past behaviors. The distance between two behaviors is simply the Euclidean distance between those behaviors when represented as a vector of numbers (which is the origin of the term *behavior characterization* or BC). While there exist several different strategies for managing the archive (Gomes et al., 2015), preliminary experiments indicated that an effective strategy is to add all individuals to an archive with a maximum size that is enforced by deleting those with the lowest novelty (the novelty of all archive members is recomputed before each deletion). In this study, NS has a population size of 500 and a maximum archive size of 2,500.

4.2. Quality Diversity Algorithms

Each of the following algorithms features both of the essential components of QD: pressure to discover more behavioral niches and a tendency toward increasing performance in niches that have already been discovered.

4.2.1. [NSLC] Novelty Search with Local Competition

Perhaps, the first true QD algorithm, *novelty search with local competition* (NSLC) (Lehman and Stanley, 2011b) combines the diversifying power of NS with a localized fitness pressure called *local competition* (LC), calculated as the proportion of an individual's k-nearest ($k = 20$) behavior neighbors that have a lower-fitness score. LC allows a quality-based reward to be assigned within behavioral neighborhoods without asserting that some neighborhoods are better than others. In NSLC, novelty and LC are combined by Pareto ranking following the practice of the NSGA-II multi-objective optimization algorithm (Deb et al., 2002).

4.2.2. [ME] Multi-Dimensional Archive of Phenotypic Elites

An alternative approach to QD is an algorithm called *Multi-dimensional Archive of Phenotypic Elites* (MAP-Elites or ME) (Cully et al., 2015; Mouret and Clune, 2015). The key difference in MAP-Elites is that niches are explicitly defined rather than passively emergent from a system of local competition; the behavior space is divided into a number of discrete behavior bins (often called a "grid" and created by discretizing each dimension of the BC) where each bin remembers the single fittest individual ever mapped to that bin. The set of filled bins constitutes the active population and evolution proceeds by selecting a bin at random (with equal probability) to produce an offspring, which is then mapped to a bin corresponding to its behavior where it may be either saved or discarded depending on whether its fitness is higher or lower than the current elite occupant. Because selection is uniform, the hope is to acquire diversity passively by virtue of the observation that more bins will tend to fill over time and, once filled, they will not be forgotten.

4.2.3. [MENOV] MAP-Elites + Novelty

Unlike NS and NSLC, the original formulation of MAP-Elites does not preferentially explore under-represented behaviors, potentially causing it to lag in its discovery of new types of behaviors. Conveniently, it is easy to augment MAP-Elites with a stronger focus on diversity by simply making selection proportional to novelty. In this variant, called *MAP-Elites + Novelty* (MENOV), whenever offspring are generated, they are also added to an archive of past behaviors (with a maximum size of 2,500, managed in the same way as in NS) that enables calculating a novelty score for all members in the MAP-Elites grid.

4.2.4. [MEPGD] MAP-Elites + Passive Genetic Diversity

The strict elitism at each bin in the original MAP-Elites formulation may eventually cause evolution to stagnate if all stepping stones to higher fitness require first making strides through lower-fitness space. Furthermore, genetic diversity is intrinsically limited when only a single individual is saved in each bin. Addressing both of these potential pitfalls without introducing any additional overhead, a new variant called *MAP-Elites + Passive Genetic Diversity* (MEPGD) saves two individuals in each bin instead of one.[3] Individuals with a lower fitness than the current elite still have a

[3]The number of extra slots per bin can conceivably be expanded to any number, where all slots except the first are governed by random replacement.

30% chance of being saved in the second slot, regardless of their fitness, thus allowing MEPGD to explore the potential of some lineages that would have otherwise been discarded. Importantly, these extra slots coincide with the set of MAP-Elites bins, which guarantees that they are behaviorally diverse.

4.3. Multi-BC Quality Diversity Algorithms

As discussed in Section 2.3, BCs that are strongly aligned with quality encourage the discovery of better behaviors simply by finding different behaviors, effectively bypassing the problem of deception that causes optimization-oriented search processes to become trapped in local optima. However, no such advantage exists for *unaligned* BCs, which often represent the most interesting and desirable types of diversity in practice. Worryingly, Pugh et al. (2015) find that such unaligned BCs actually negatively impact the performance of QD algorithms, which on sufficiently hard problems may translate into an outright failure to find the best-performing solutions. This observation raises the important question of how QD practitioners can find unaligned diversity without losing the ability to circumvent deception offered by aligned BCs.

This section offers a promising answer in the form of *driving search with multiple BCs simultaneously*. To explore this idea, the following algorithms represent various options for adapting QD to support more than one BC.

4.3.1. [NS–NS] Multi-BC Novelty Search

Novelty search can be extended to support multiple BCs simultaneously by calculating a separate novelty score for each BC and then combining these scores in a multi-objective formulation [via NSGA-II of Deb et al. (2002)]. Each BC maintains its own independent archive against which individuals are evaluated to determine their novelty score for that BC. There is only a single population where the breeding potential for each member is decided by a Pareto ranking according to the various novelty scores. Although NS is not itself a QD algorithm because it lacks a mechanism for discovered behaviors to increase in quality, when extended to include multiple BCs, NS–NS can conceivably achieve some success if one BC serves to find diversity while another promotes increasing quality (e.g., an unaligned BC paired with an aligned BC). Note that even such a pairing does not quite embody the spirit of QD because any tendency toward increasing quality that emerges from an aligned BC is not explicitly *local*.

4.3.2. [NS–NSLC] Multi-BC Novelty Search with Local Competition

The idea of NS–NS can then be expanded to include a drive toward locally increasing quality by adding a LC objective (in the same way as NSLC) where behavioral neighbors are decided by the unaligned BC that expresses the notion of diversity that the user is ultimately interested in collecting. The resulting NS–NSLC algorithm therefore includes three distinct objectives (combined via NSGA-II): (1) a quality-aligned novelty score to facilitate overcoming deception, (2) an unaligned novelty score for discovering new behaviors of interest, and (3) an unaligned LC score to promote competition within niches of similar behaviors.

4.3.3. [ME–ME] Multi-BC MAP-Elites

In an effort to maintain its characteristic simplicity, MAP-Elites is extended to support multiple BCs by maintaining a separate grid for each BC (with similar maximum sizes). On each iteration of ME–ME, an equal amount of parents are selected from each grid, and their resulting offspring are mapped to *both* grids (where the decisions to save or discard them are performed independently, e.g., between two grids, a single offspring may be saved in one, both, or neither).

4.3.4. [MENOV–MENOV] Multi-BC MAP-Elites + Novelty

Finally, MENOV is extended to MENOV–MENOV similarly to ME–ME except where each grid also maintains its own novelty archive and selection within each grid is proportional to novelty.

5. EXPERIMENTS

The behavior characterization (BC) determines the form of pressure that ultimately drives the diversity component of the search and thus must be selected carefully to complement the evolutionary algorithm. The experiments in this article explore two BCs that are each highly aligned or highly misaligned with the notion of quality (i.e., how close an agent is to arriving at the goal): **EndpointBC**, which is simply a two-dimensional vector containing the x and y coordinates of an individual's location at the end of its trial, and **DirectionBC**, a five-dimensional vector with entries indicating whether the individual was most frequently facing north (0.125), east (0.375), south (0.625), or west (0.875) for each fifth of its evaluation time. On the one hand, EndpointBC is thus highly aligned with the goal of navigating to the goal point because the continual discovery of new endpoints will eventually lead to finding the goal point. On the other hand, DirectionBC is largely orthogonal to quality because the direction a robot faces at a particular time step does not fully determine whether it solves the maze (more importantly, it is possible to visit all of the behaviors in the DirectionBC space without ever reaching the goal).

Consistent with the observation that the goal of QD applications in practice is often to find diversity with respect to an unaligned BC, the assumption in this study is that the goal is to find QD with respect to DirectionBC. Therefore, exploring how different approaches to QD interact with each of EndpointBC and DirectionBC makes it possible to address several important questions:

1. How well does the approach suggested by current literature (i.e., driving search with the very notion of diversity you are interested in collecting) work? Given that preliminary experiments from Pugh et al. (2015) show that DirectionBC sometimes results in suboptimal QD-scores even in the relatively simple small maze, the hypothesis is that this conventional approach will not be optimal on complex domains such as the gauntlet mazes.
2. Can diversity with respect to DirectionBC be found without searching for it explicitly? That is, EndpointBC has been shown to be effective for driving novelty search to find

solutions to deceptive mazes (Lehman and Stanley, 2008). However, it is unknown whether it is similarly effective at finding QD when diversity is measured on a separate unaligned metric. The hypothesis is that algorithms driven by EndpointBC should do well in terms of progress toward the goal, but may not necessarily result in high diversity with respect to DirectionBC.

3. Finally, can multiple BCs successfully be *combined* to compensate for the shortcomings of highly aligned BCs (lower diversity) and unaligned BCs (lower performance)?

Each of the algorithms from Section 4 is implemented with some combination of DirectionBC and EndpointBC for a total of fifteen treatments, enumerated in **Table 2**. Each treatment is run 20 times on the small maze, symmetric gauntlet, and asymmetric gauntlet for a total of 900 runs (300 per maze). Small maze runs ended after 250,000 evaluations and gauntlet runs ended after 1,000,000 evaluations (in each case, more than enough time for all algorithms to reach a performance plateau). Networks are evolved with a modified version of SharpNEAT 1.0 (Green, 2003) with mutation parameters validated by Pugh et al. (2015): 60% mutate connection, 10% add connection, and 0.5% add neuron. Networks are feedforward and restricted to asexual reproduction; other settings follow SharpNEAT 1.0 defaults. Maze-specific parameter settings are presented in **Table 1**.

In traditional maze navigation domains, an appropriate metric would be whether or not a robot was eventually able to navigate to the goal. However, this metric does not speak to an algorithm's propensity for discovering diversity in a search space. For this reason, Pugh et al. (2015) introduces a new QD metric [which is also similar to the "global reliability" metric in Mouret and Clune (2015)] that reflects both the quality and diversity of individuals found by evolution (including solutions and non-solutions). Diversity in this metric, called the *QD-score*, is measured with respect to a BC. In the experiments reported in this article, DirectionBC always characterizes diversity for the purpose of computing the QD-score *regardless of the behavior characterization driving search*. This approach reflects the usual idea that the desire is to see a wide diversity of solutions at the end of a run with respect to a BC that is not necessarily directly aligned with solving the problem.

To quantify how much of the space is explored by an algorithm for the QD-score, the entire behavior space is first discretized into a collection of t bins[4] $\{N_1, \ldots, N_t\}$ as in the MAP-Elites algorithm described in the previous section. Each bin corresponds to a unique combination of features from the individual's BC (in this case, DirectionBC) and represents a niche in the behavior space. Diversity is then quantified as the number of bins filled over the course of an evolutionary run. By *summing the highest fitness values found in each grid bin*, where Q_i represents the highest fitness achieved in bin N_i, it becomes possible to simultaneously quantify both quality and diversity as

$$\text{QD-score} = \sum_{i=1}^{t} Q_i.$$

This DirectionBC-based QD-score[5] is the primary metric in all mazes. Note that for the purpose of calculating QD-score, fitness is defined in a way that reflects the shortest drivable path between an agent ending location and the goal, respecting that agents cannot drive through walls. This special QD-score fitness is calculated by a breadth-first flood fill from the goal point, assigning fitness to locations in the maze that decreases linearly at each layer of the flood fill. Importantly, this fitness value, which draws a perfect non-deceptive gradient over the maze, is not available to any algorithms to drive search but merely appears during *post hoc* analysis to give an accurate accounting for how close each collected behavior is to solving the maze.

Due to the overwhelming historical focus on optimization and the recent realization that behavioral diversity can itself be a powerful tool for optimization (Lehman and Stanley, 2008; Mouret and Doncieux, 2009, 2012), this study includes an additional performance metric for the two gauntlets[6] that captures the spirit of this more conventional search paradigm: the *total maze progress* metric measures how close a run is to solving all four legs of the maze. More precisely, total maze progress is the sum of four progress measures (one per leg) where progress for each leg increases linearly as endpoints are discovered further along their solution path (calculated by means of the flood fill distance to the goal point). If E_i is the set of all flood fill fitness scores associated with endpoints discovered inside leg i then total maze progress can be quantified as

$$\text{total maze progress} = \sum_{i=1}^{4} \max E_i.$$

A maximum score is achieved by solving all four legs. Total maze progress therefore addresses the important question of how well-suited QD algorithms are to the task of optimizing toward a series of predefined targets.[7]

TABLE 2 | Fifteen treatments compared with three mazes.

No BC	DirectionBC	EndpointBC	Multi-BC
Fitness	NS_d	NS_e	NS_eNS_d
	$NSLC_d$	$NSLC_e$	NS_eNSLC_d
	ME_d	ME_e	ME_eME_d
	$MENOV_d$	$MENOV_e$	$MENOV_eMENOV_d$
	$MEPGD_d$	$MEPGD_e$	

Algorithms in different columns differ by the BC that drives search. Conventional QD algorithms are tested with each of DirectionBC and EndpointBC driving search (denoted by the subscripts d and e, respectively), while a new class of multi-BC QD algorithms drives search with both DirectionBC and EndpointBC simultaneously.

[4] In this study, $t = 1024$ when discretizing the DirectionBC behavior space.

[5] The original small maze results from Pugh et al. (2015) instead always measure diversity with respect to the BC that drives search and thus do not explore the idea of driving search with BCs other than those which characterize the dimensions of interest.

[6] No such metric is defined for the small maze because all treatments consistently and quickly find solutions and thus it does not represent a challenging optimization problem.

[7] Total maze progress is related to the interests of multimodal optimization, where the goal is often to find all of the global optima in a fitness landscape without regard to the behaviors or phenotypes that get there.

6. RESULTS

In all of the figures presented in this section, treatments are color coded according to which BC drives search: DirectionBC (subscript *d*) is drawn in blue, EndpointBC (subscript *e*) in yellow, multi-BC in green, and Fitness (which is not driven by any BC) in gray. For each treatment, results represent an average over 20 runs; error bars represent the SEM and can be interpreted to infer which differences are statistically significant when *p* values are not explicitly provided. In all reported cases, statistical significance is determined by an unpaired two-tailed Student's *t*-test.[8]

6.1. Total Maze Progress

The total maze progress achieved by each treatment after all evaluations is depicted in **Figure 4** (asymmetric gauntlet) and **Figure 5** (symmetric gauntlet). A maximum possible score of 400 corresponds to solving all four legs in the same run, although such scores are not observed in practice. Unlike in the small maze, where all treatments consistently find solutions in every run, many gauntlet runs (particularly those of Fitness or DirectionBC-driven treatments) do not find solutions at all, reflecting the increased difficulty in the gauntlet mazes. Of the two gauntlet mazes surveyed by this metric, higher scores are obtained by all treatments on the asymmetric gauntlet, indicating that the asymmetric gauntlet is comparatively easier to solve, thus establishing a continuum of difficulty between the three maze domains featured in this study: small maze (easiest), asymmetric gauntlet (harder), and symmetric gauntlet (hardest).

While in both gauntlets the performance of Fitness is subpar as expected [it is known to struggle with deception in maze domains; Lehman and Stanley (2008)], a perhaps more surprising result is that all DirectionBC-driven treatments perform significantly worse than Fitness in terms of ability to find solutions ($p < 0.05$) to both the asymmetric gauntlet (**Figure 4**) and the symmetric gauntlet (**Figure 5**). Indeed, of these treatments on the symmetric gauntlet, only $MEPGD_d$ finds any solutions at all (two solutions found across all 20 runs).

On the other hand, all approaches that include an aligned BC (EndpointBC) always perform significantly better than Fitness ($p < 0.001$). Of those treatments that include EndpointBC, single-BC approaches tend to perform better (with respect to solving the maze) than multi-BC approaches that also include DirectionBC (**Figures 4** and **5**). This phenomenon is especially apparent on the harder symmetric gauntlet (**Figure 5**), where $NSLC_e$ and NS_e perform significantly better than the multi-BC variants ($p < 0.05$). Of particular interest is that specialized QD algorithms such as $NSLC_e$ are competitive with the currently accepted method for overcoming deception on maze tasks: NS_e (Lehman and Stanley, 2008, 2011a). On the symmetric gauntlet (**Figure 5**), there is some evidence that $NSLC_e$ may actually be *better* than NS_e, although the evidence is not strong enough to establish statistical significance ($p = 0.093$).

Of the successful MAP-Elites variants (those driven by EndpointBC), $MEPGD_e$ performs significantly better than the core ME_e on both gauntlets (**Figures 4** and **5**), while $MENOV_e$ in neither case is significantly different than ME_e.

6.2. QD-*Score*

The final QD-score achieved by each treatment after all evaluations is depicted in **Figure 6** (small maze), **Figure 7** (asymmetric gauntlet), and **Figure 8** (symmetric gauntlet).

6.2.1. Small Maze

Reflecting its lack of challenging complexity, the best treatments on the small maze (**Figure 6**) consistently achieve near the

[8]The simple Student's *t*-test is chosen intentionally to avoid Type II errors, which are more likely when adjusting for multiple comparisons. As such, the results here are intended to highlight potential differences between treatments, not to establish a definitive ranking.

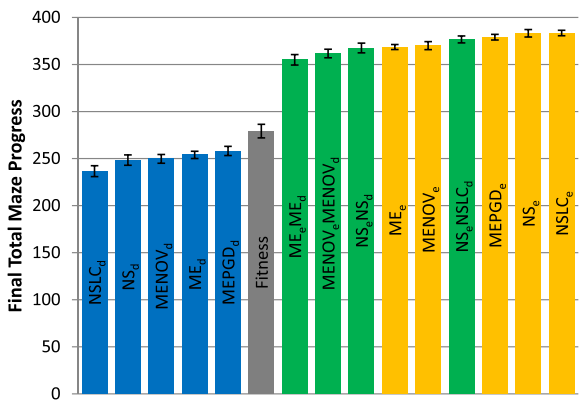


FIGURE 4 | Total maze progress (asymmetric gauntlet). The final total maze progress achieved by each of fifteen treatments (**Table 2**) after 1,000,000 evaluations on the asymmetric gauntlet is shown (averaged over 20 runs). Bars are color coded according to which BC drives search (see section 6) and error bars represent SE. A maximum possible score of 400 corresponds to solving all four legs of the maze.

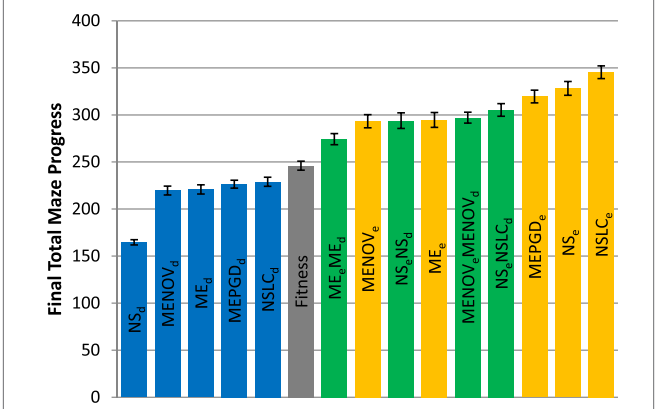


FIGURE 5 | Total maze progress (symmetric gauntlet). The final total maze progress achieved by each of fifteen treatments (**Table 2**) after 1,000,000 evaluations on the symmetric gauntlet is shown (averaged over 20 runs). Bars are color coded according to which BC drives search (see section 6) and error bars represent SE. A maximum possible score of 400 corresponds to solving all four legs of the maze.

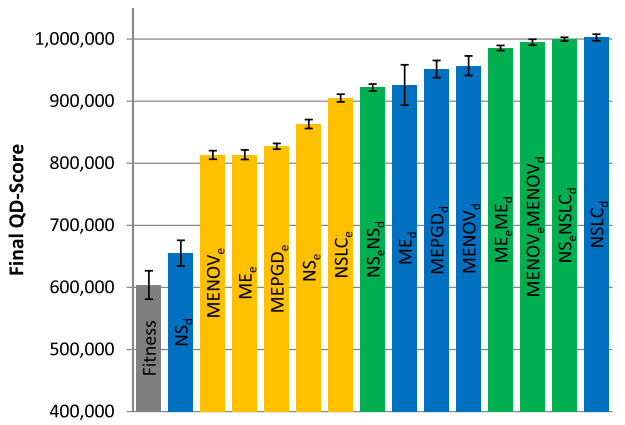


FIGURE 6 | Final QD-score (small maze). The QD-score achieved by each of fifteen treatments (**Table 2**) after 250,000 evaluations on the small maze is shown (averaged over 20 runs). Bars are color coded according to which BC drives search (see section 6) and error bars represent SE. In all cases, QD-score is measured with respect to DirectionBC. The maximum possible QD-score on the small maze is 1,024,000 corresponding to a perfect solution in all 1,024 bins.

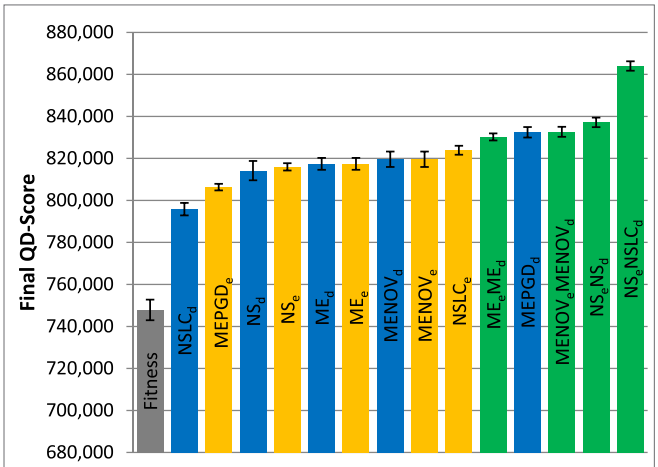


FIGURE 7 | Final QD-score (asymmetric gauntlet). The QD-score achieved by each of fifteen treatments (**Table 2**) after 1,000,000 evaluations on the asymmetric gauntlet is shown (averaged over 20 runs). Bars are color coded according to which BC drives search (see section 6) and error bars represent SE. In all cases, QD-score is measured with respect to DirectionBC.

maximum possible QD-score of 1,024,000, which corresponds to finding a maze solution (score = 1000) in each of the 1,024 bins. On this relatively simple task, QD-score is dominated by the multi-BC QD approaches (ME$_e$ME$_d$, MENOV$_e$MENOV$_d$, NS$_e$NSLC$_d$) and by NSLC$_d$. EndpointBC-driven approaches perform poorly in comparison, though not as poorly as Fitness and NS$_d$.

Of interest is the comparatively large variance on the MAP-Elites variants (when driven by DirectionBC) versus their NS-based counterparts (**Figure 6**). In particular, the most extreme such variance, on ME$_d$, is caused by three outliers. While most runs of ME$_d$ score between 950 K and 1000 K, the outlier runs obtain scores of 791 K, 680 K, and 421 K. In each of these runs, the grid is completely filled (representing maximum diversity), but many bins contain low-quality behaviors (mostly representing agents that exclusively drive around inside the main deceptive trap). Thus, the higher variance observed by ME$_d$ here is indicative of MAP-Elites sometimes becoming stuck in local optima.

6.2.2. Gauntlet Mazes

Because of the strict time constraints on the gauntlet mazes, many types of behaviors (i.e., bins in the QD grid) can never represent full solutions even in theory; thus, the maximum possible QD-score on the gauntlet mazes is lower than on the small maze and also difficult to achieve in practice. This limitation is reflected by the comparatively lower scores observed in both the asymmetric gauntlet (**Figure 7**) and the symmetric gauntlet (**Figure 8**).

While DirectionBC-driven QD achieves the highest scores on the small maze (**Figure 6**), this trend is reversed as domain difficulty increases. On the asymmetric gauntlet, treatments driven by EndpointBC perform similarly to those driven by DirectionBC (**Figure 7**). On the comparatively more difficult symmetric gauntlet, the trend is completely reversed, with EndpointBC-driven

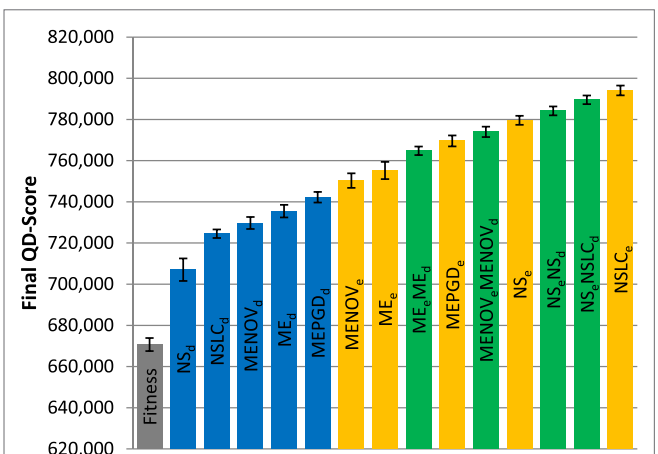


FIGURE 8 | Final QD-score (symmetric gauntlet). The QD-score achieved by each of fifteen treatments (**Table 2**) after 1,000,000 evaluations on the symmetric gauntlet is shown (averaged over 20 runs). Bars are color coded according to which BC drives search (see section 6) and error bars represent SE. In all cases, QD-score is measured with respect to DirectionBC.

approaches performing significantly better than those who are driven only by DirectionBC (**Figure 8**).

Consistently with the small maze, in each of the gauntlets, QD-score is dominated by the best multi-BC treatments. Specifically, in all three mazes, NS$_e$NSLC$_d$ is consistently among the best-performing treatments (**Figures 6–8**). On the asymmetric gauntlet (**Figure 7**), its lead is unmatched, while on the symmetric gauntlet it is tied with NSLC$_e$ (**Figure 8**; the difference between NSLC$_e$ and NS$_e$NSLC$_d$ is not statistically significant, $p = 0.163$).

However, it turns out that even though their final QD-scores are similar on the symmetric gauntlet, $NSLC_e$ and NS_eNSLC_d do exhibit different learning curves. To highlight this difference on the symmetric gauntlet, it is instructive to graph the development of QD-score over the course of evolution. **Figure 9** depicts the QD-score over time for the best-performing method from each of the three classes in **Table 2**: EndpointBC, DirectionBC, and multi-BC. The general trend displayed by each treatment is representative of the other treatments in its respective class, e.g., DirectionBC-driven treatments tend to increase quickly and then plateau at a low score. On the other hand, $NSLC_e$ increases relatively slowly before ultimately reaching a much higher score. Combining the best of both of these options, NS_eNSLC_d quickly reaches high scores (**Figure 9**). Thus, overall the hybrid NS_eNSLC_d proves a competitive choice for maximizing QD-score on all the variant mazes.

7. DISCUSSION

Within the wide-ranging field of EC, two distinct types of research goals are relevant in the context of this investigation. The first and historically most dominant focus of EC is the task of optimization: harnessing the powerful natural mechanism of "survival of the fittest" to reach some predefined target (or series of targets). It is this goal that lies at the heart of the vast majority of EC literature, thereby seemingly aligning the ambitions of EC with the broader practice of machine learning, where optimization is treated as the essence of learning itself. However, having only recently begun to garner attention, the second type of research goal is less familiar, though it offers promising new opportunities for discovery and advancement uniquely accessible to EC. This goal, called *quality diversity*, represents a fundamental departure from optimization because instead the idea is to explore all of what is possible rather than to find only the best option. While the primary intention of this paper is to promote the theory and practice of QD, the results of this study have implications relevant to both paradigms.

The recent realization that pursuing behavioral diversity can help to overcome deception (Lehman and Stanley, 2008, 2011a; Mouret and Doncieux, 2009, 2012) has offered a source of hope to solving the problem of deception that permeates almost every optimization space of interest. However, deception does not exclusively affect optimization and not all notions of "behavioral diversity" are equally capable of thwarting it. As this study reveals, simply focusing on QD does not make evolution immune to the problem of deception because the quality-seeking *component* of QD algorithms can itself fall victim to it. In this study, which features maze tasks of varying difficulty, QD algorithms following the compass of an unaligned BC are incrementally less able to find QD as the level of deception increases from one maze to another (**Figures 6–8**). Further highlighting the problem of deception even in QD, surprisingly, as maze difficulty increases, it actually becomes more effective to drive search with an aligned BC even when you are collecting unaligned QD (**Figure 8**). The primary lesson is that searching for diversity with respect to an unaligned BC does not circumvent the problem of deception (doing so is mostly orthogonal to overcoming deception, similar to the shortcomings of pursuing *genetic* diversity). However, for QD practitioners, the magic bullet cannot be to simply drive search with some other, better-suited BC because that only leaves the diversity of interest to be collected *coincidentally*. Instead, this study offers the promising new idea of driving search with multiple BCs simultaneously. The so-called multi-BC algorithms (such as NS–NSLC) allow search to be driven both by the desired notion of diversity and a separate BC that is well equipped to circumvent deception, thus unlocking the best parts of the search space for discovery.

An alternative approach not tested in this study is to simply concatenate both an aligned and unaligned BC into a single BC with more dimensions. In considering this option, it is important to note that the behavior space grows *exponentially* with the size of the BC. Thus, such an approach generally cannot be applied with MAP-Elites because the number of bins would also grow exponentially until the grid no longer resembles a reasonably sized population. While concatenating multiple BCs into a single monolithic BC is still tractable with NS-based algorithms, the resulting vast behavior space may present an additional challenge over the multi-BC approaches tested here.

Following the lesson originally presented by Lehman and Stanley (2008), this study reconfirms that a powerful strategy for finding solutions to a difficult maze is to abandon the objective entirely and simply search for behavioral diversity (e.g., NS_e in **Figures 4** and **5**). This conclusion itself has implications outside of maze solving that apply more generally to all of optimization. However, an important observation is that not just any type of behavioral diversity is successful. Indeed, with regard to being able to solve difficult mazes, searching for diversity with respect to an unaligned BC (such as DirectionBC) does even worse than purely objective search (**Figure 5**). The more general lesson is that *BCs that are aligned with the notion of quality* (such as EndpointBC in this study) are the key ingredient to overcoming deception on difficult problems. Furthermore, this study offers the additional insight that QD algorithms themselves offer a means for "the objective-less search for behavioral diversity" (i.e., novelty search) to be rectified with their missing objective in a way that allows search to respect the ultimate goal (e.g.,

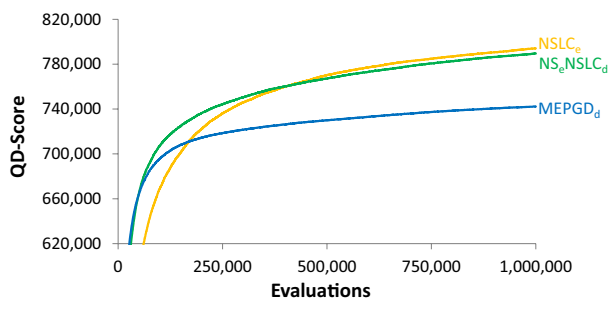


FIGURE 9 | QD-score over time (symmetric gauntlet). The progression of the QD-score on the symmetric gauntlet for the best-performing treatment from each class (according to the BC that drives search; DirectionBC: $MEPGD_d$, EndpointBC: $NSLC_e$, multi-BC: NS_eNSLC_d) is graphed over time (averaged over 20 runs). The main result is that multi-BC treatments exhibit the best characteristics of each component BC: both increasing quickly and reaching high scores.

solving the maze) without re-introducing the problem of deception. Specifically, $NSLC_e$ in this paper demonstrates that it can optimize in the presence of strong deception at least as well as regular novelty search (**Figure 4**) and, in fact, might even be better (**Figure 5**). This result suggests that while the ultimate goals of QD are distinct from optimization, advancements in QD can themselves directly benefit optimization.

As a relatively new approach to QD, MAP-Elites remains a largely unexplored paradigm at the time of this writing. The core MAP-Elites algorithm offers the significant appeal of a very simple algorithm (requiring in effect only a few lines of code) that powerfully distills the essence of QD. Because it is relatively new, its performance in this study serves to confirm that it is largely impacted by BCs similarly to NS-based methods (suggesting some general principles for QD across different algorithms), but of course much room remains for MAP-Elites in particular to be improved. MENOV and MEPGD in this study both suggest that the core ME algorithm can be fruitfully augmented in part to overcome any restrictive effects of its strict elitism. Some ideas not tested here include MEPGDNOV (combining genetic diversity with novelty) and MEPGD–MEPGD (ME with genetic diversity and multiple BCs). The genetic diversity component might also benefit from expansion to more than one diversity candidate per bin.

An important question raised by the results is how to decide whether a particular domain is hard (therefore likely requiring more than one BC) and in such cases whether the chosen aligned BC is sufficiently aligned to overcome the threat of deception. One way to decide whether it may be necessary to include multiple BCs is simply to run a pure fitness-based (objective-driven) search. If such a search consistently finds solutions in multiple runs then the domain can be considered sufficiently easy that multiple BCs may be unnecessary. On the other hand, if the domain proves too difficult for fitness, then it is important to make sure that the aligned BC is, in fact, sufficiently aligned to complement the unaligned BC. One way to test for alignment is to try running simple NS only with the candidate aligned BC. If the alignment is effective, such a search should at least do better than fitness-based search, which would then validate that the BC in question can complement an unaligned BC in a multi-BC hybrid.

Interestingly, while the results support that in sufficiently easy domains a more naive single-BC approach with an unaligned BC may work to collect QD, it does not appear harmful in any case to take the safer multi-BC approach even then. Thus, even though it has been customary so far in the QD-related literature to rely on a single unaligned BC (Lehman and Stanley, 2011b; Cully and Mouret, 2013; Szerlip and Stanley, 2013; Mouret and Clune, 2015), it may be possible to revisit some of the domains of the past with multiple BCs and achieve even better performance.

More broadly, the multi-BC approach and our new understanding of the implications of alignment can help us in the future to achieve significantly more impressive results with QD than seen in the past. Even domains that might have seemed inexplicably out of reach might now become accessible through the application of multiple BCs. Thus, because QD represents a promising direction exclusive to evolutionary techniques, it is in the interest not just of those working in QD, but EC and evolutionary robotics as a whole to seek out and propose such future domains. The potential for such ideas is foreshadowed by QD results already published in the diverse domains of morphology evolution (Lehman and Stanley, 2011b; Szerlip and Stanley, 2013; Mouret and Clune, 2015), robot control and adaptation (Cully and Mouret, 2013; Cully et al., 2015; Mouret and Clune, 2015), image generation (Nguyen et al., 2015a,b), and three-dimensional object evolution (Lehman et al., 2016). With increasing interest in the field, these domains may be just the beginning for QD.

8. CONCLUSION

In an attempt to unify and investigate the emerging field of quality diversity (QD), this paper compared various QD algorithm variants and controls in three different maze domains, two of a higher level of complexity than previously seen in such maze-based studies. By pushing the mazes to a higher level of complexity than seen in QD before, the study was able to expose conditions under which QD effectively breaks down. It turns out that driving the diversity component of QD algorithms exclusively by a BC unaligned with quality (which is heretofore common practice) performs relatively poorly under such difficult conditions. However, on a positive note, methods that hybridize more than one BC (one aligned and one unaligned) tend to perform well at the same time as finding the kind of diversity desired, suggesting a promising path forward for QD in the future as it is applied in increasingly ambitious domains.

AUTHOR CONTRIBUTIONS

JP helped conceive the work and led experimental design, implementation, and writing. LS helped conceive the work and contributed to experimental design, implementation, and writing. KS provided vision and oversight, helped conceive the work, and contributed to experimental design and writing.

FUNDING

This work was supported by the National Science Foundation under grant no. IIS-1421925. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

Bäck, T., Hammel, U., and Schwefel, H.-P. (1997). Evolutionary computation: comments on the history and current state. *IEEE Trans. Evol. Comput.* 1, 3–17. doi:10.1109/4235.585888

Bedau, M. (2008). "The arrow of complexity hypothesis (abstract)," in *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI)*, eds S. Bullock, J. Noble, R. Watson, and M. Bedau (Cambridge, MA: MIT Press), 750.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*, Vol. 1. New York, NY: Springer.

Boden, M. (2006). *Mind as Machine: A History of Cognitive Science*. USA: Oxford University Press.

Bongard, J. C. (2002). "Evolving modular genetic regulatory networks," in *Proceedings of the 2002 Congress on Evolutionary Computation, Honolulu* (Washington, DC: IEEE).

Cliff, D., Harvey, I., and Husbands, P. (1993). Explorations in evolutionary robotics. *Adapt. Behav.* 2, 73–110. doi:10.1177/105971239300200104

Cortes, C., and Vapnik, V. (1995). Support-vector networks. *Mach. Learn.* 20, 273–297. doi:10.1007/BF00994018

Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. (2015). Robots that can adapt like animals. *Nature* 521, 503–507. doi:10.1038/nature14422

Cully, A., and Mouret, J.-B. (2013). "Behavioral repertoire learning in robotics," in *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation (GECCO '13)* (New York, NY: ACM), 175–182.

De Jong, K. A. (2002). *Evolutionary Computation: A Unified Perspective*. Cambridge, MA: MIT Press.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* 6, 182–197. doi:10.1109/4235.996017

Doncieux, S., Bredeche, N., Mouret, J.-B., and Eiben, A. G. (2015). Evolutionary robotics: what, why, and where to. *Front. Robot. AI* 2:4. doi:10.3389/frobt.2015.00004

Doucette, J., and Heywood, M. I. (2010). "Novelty-based fitness: an evaluation under the santa fe trail," in *Proceedings of the European Conference on Genetic Programming (EuroGP-2010), Istanbul* (Berlin, Heidelberg: Springer-Verlag), 50–61.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence Through Simulated Evolution*. New York, NY: John Wiley & Sons.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley.

Goldberg, D. E., and Richardson, J. (1987). "Genetic algorithms with sharing for multimodal function optimization," in *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, ed. J. J. Grefenstette (Hillsdale, NJ: Lawrence Erlbaum), 41–49.

Goldsby, H. J., and Cheng, B. H. (2010). "Automatically discovering properties that specify the latent behavior of UML models," in *Model Driven Engineering Languages and Systems* (Berlin, Heidelberg: Springer), 316–330.

Gomes, J., and Christensen, A. L. (2013). "Generic behaviour similarity measures for evolutionary swarm robotics," in *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation (GECCO '13)* (New York, NY: ACM), 199–206.

Gomes, J., Mariano, P., and Christensen, A. L. (2015). "Devising effective novelty search algorithms: a comprehensive empirical study," in *Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation, Madrid* (New York, NY: ACM), 943–950.

Gomes, J., Urbano, P., and Christensen, A. L. (2013). Evolution of swarm robotics systems with novelty search. *Swarm Intell.* 7, 115–144. doi:10.1007/s11721-013-0081-z

Graening, L., Aulig, N., and Olhofer, M. (2010). "Towards directed open-ended search by a novelty guided evolution strategy," in *Parallel Problem Solving from Nature – PPSN XI. Vol. 6239 of Lecture Notes in Computer Science, Krakow*, eds R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph (Berlin, Heidelberg: Springer), 71–80.

Green, C. (2003–2006). *SharpNEAT Homepage*. Available at: http://sharpneat.sourceforge.net/

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press.

Hornby, G. S., and Pollack, J. B. (2001). "The advantages of generative grammatical encodings for physical design," in *Proceedings of the 2001 Congress on Evolutionary Computation, Seoul* (Washington, DC: IEEE).

Hornby, G. S., and Pollack, J. B. (2002). Creating high-level components with a generative representation for body-brain evolution. *Artif. Life* 8, 223–246. doi:10.1162/106454602320991837

Kirschner, M., and Gerhart, J. (1998). Evolvability. *Proc. Natl. Acad. Sci. U.S.A.* 95, 8420–8427. doi:10.1073/pnas.95.15.8420

Kistemaker, S., and Whiteson, S. (2011). "Critical factors in the performance of novelty search," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation. GECCO '11, Dublin* (New York, NY: ACM), 965–972.

Krcah, P. (2010). "Solving deceptive tasks in robot body-brain co-evolution by searching for behavioral novelty," in *10th International Conference on Intelligent Systems Design and Applications (ISDA), Cairo* (Washington, DC: IEEE), 284–289.

Lehman, J., Risi, S., and Clune, J. (2016). "Creative generation of 3D objects with deep learning and innovation engines," in *Proceedings of the 7th International Conference on Computational Creativity*, Paris.

Lehman, J., and Stanley, K. O. (2008). "Exploiting open-endedness to solve problems through the search for novelty," in *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI), Winchester*, eds S. Bullock, J. Noble, R. Watson, and M. Bedau (Cambridge, MA: MIT Press).

Lehman, J., and Stanley, K. O. (2010). "Revising the evolutionary computation abstraction: minimal criteria novelty search," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO '10)* (New York, NY: ACM), 103–110.

Lehman, J., and Stanley, K. O. (2011a). Abandoning objectives: evolution through the search for novelty alone. *Evol. Comput.* 19, 189–223. doi:10.1162/EVCO_a_00025

Lehman, J., and Stanley, K. O. (2011b). "Evolving a diversity of virtual creatures through novelty search and local competition," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO '11), Dublin* (New York, NY: ACM), 211–218.

Lehman, J., and Stanley, K. O. (2013). Evolvability is inevitable: increasing evolvability without the pressure to adapt. *PLoS ONE* 8:e62186. doi:10.1371/journal.pone.0062186

Liapis, A., Martínez, H. P., Togelius, J., and Yannakakis, G. N. (2013a). "Transforming exploratory creativity with delenox," in *Proceedings of the Fourth International Conference on Computational Creativity, Sydney* (Sydney: University of Sydney).

Liapis, A., Yannakakis, G. N., and Togelius, J. (2013b). "Enhancements to constrained novelty search: two-population novelty search for generating game content," in *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, GECCO '13* (New York, NY: ACM), 343–350.

Mahfoud, S. W. (1995). *Niching Methods for Genetic Algorithms*. Ph.D. thesis, University of Illinois at Urbana-Champaign, Urbana, IL.

Martinez, Y., Naredo, E., Trujillo, L., and Galvan-Lopez, E. (2013). "Searching for novel regression functions," in *2013 IEEE Congress on Evolutionary Computation (CEC), Cancun* (Washington, DC: IEEE), 16–23.

Mitchell, T. M. (1997). *Machine Learning*. New York, NY: McGraw-Hill.

Morse, G., Risi, S., Snyder, C. R., and Stanley, K. O. (2013). "Single-unit pattern generators for quadruped locomotion," in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13), Amsterdam* (New York, NY: ACM), 719–726.

Mouret, J.-B. (2011). "Novelty-based multiobjectivization," in *New Horizons in Evolutionary Robotics*, eds S. Doncieux, N. Bredeche, and J.-B. Mouret (Berlin, Heidelberg: Springer), 139–154.

Mouret, J.-B., and Clune, J. (2015). Illuminating search spaces by mapping elites. arXiv preprint arXiv:1504.04909.

Mouret, J.-B., and Doncieux, S. (2009). "Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC-2009), Trondheim* (Washington, DC: IEEE), 1161–1168.

Mouret, J.-B., and Doncieux, S. (2012). Encouraging behavioral diversity in evolutionary robotics: an empirical study. *Evol. Comput.* 20, 91–133. doi:10.1162/EVCO_a_00048

Naredo, E., and Trujillo, L. (2013). "Searching for novel clustering programs," in *Proceeding of the fifteenth Annual Conference on Genetic and Evolutionary Computation Conference. GECCO '13, Amsterdam* (New York, NY: ACM), 1093–1100.

Nguyen, A., Yosinski, J., and Clune, J. (2015a). "Deep neural networks are easily fooled: high confidence predictions for unrecognizable images," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15), Boston* (Washington, DC: IEEE).

Nguyen, A., Yosinski, J., and Clune, J. (2015b). "Innovation engines: automated creativity and improved stochastic optimization via deep learning," in *Proceedings*

of the 17th Annual Conference on Genetic and Evolutionary Computation (GECCO '15) (New York, NY: ACM).

Nolfi, S., and Floreano, D. (2000). *Evolutionary Robotics*. Cambridge: MIT Press.

Pugh, J. K., Soros, L. B., Szerlip, P. A., and Stanley, K. O. (2015). "Confronting the challenge of quality diversity," in *Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation (GECCO '15)* (New York, NY: ACM).

Risi, S., Hughes, C., and Stanley, K. (2011). Evolving plastic neural networks with novelty search. *Adapt. Behav.* 18, 470–491. doi:10.1177/1059712310379923

Risi, S., and Stanley, K. O. (2013). "Confronting the challenge of learning a flexible neural controller for a diversity of morphologies," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2013)* (New York, NY: ACM).

Risi, S., Vanderbleek, S. D., Hughes, C. E., and Stanley, K. O. (2009). "How novelty search escapes the deceptive trap of learning to learn," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2009)* (New York, NY: ACM).

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). "Learning internal representations by error propagation," in *Parallel Distributed Processing*, Vol. 1 eds J. L. McClelland and D. L. Rumelhart (Cambridge, MA: MIT Press), 318–362.

Schwefel, H.-P. P. (1993). *Evolution and Optimum Seeking: The Sixth Generation*. New York, NY: John Wiley & Sons, Inc.

Simon, H. A. (1957). *Models of Man: Social and Rational – Mathematical Essays on Rational Human Behavior in a Social Setting*. New York, NY: Wiley.

Soltoggio, A., and Jones, B. (2009). "Novelty of behaviour as a basis for the neuro-evolution of operant reward learning," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation GECCO '09* (New York, NY: ACM), 169–176.

Standish, R. K. (2003). Open-ended artificial evolution. *Int. J. Comput. Intell. Appl.* 3, 167–175. doi:10.1142/S1469026803000914

Stanley, K. O. (2007). "Compositional pattern producing networks: a novel abstraction of development," in *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, Vol. 8 (Berlin, Heidelberg: Springer), 131–162.

Stanley, K. O. (2011). "Why evolutionary robotics will matter," in *New Horizons in Evolutionary Robotics*, eds S. Doncieux, N. Bredeche, and J.-B. Mouret (Berlin, Heidelberg: Springer), 37–41.

Stanley, K. O., and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evol. Comput.* 10, 99–127. doi:10.1162/106365602320169811

Stanley, K. O., and Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artif. Life* 9, 93–130. doi:10.1162/106454603322221487

Szerlip, P., and Stanley, K. O. (2013). "Indirectly encoded sodarace for artificial life," in *Proceedings of the European Conference on Artificial Life (ECAL-2013), Taormina* Vol. 12 (Cambridge, MA: MIT Press), 218–225.

Szerlip, P. A., Morse, G., Pugh, J. K., and Stanley, K. O. (2015). "Unsupervised feature learning through divergent discriminative feature accumulation," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-2015)* (Menlo Park, CA: AAAI Press).

Trujillo, L., Olague, G., Lutton, E., and De Vega, F. F. (2008). "Discovering several robot behaviors through speciation," in *Applications of Evolutionary Computing* (Berlin, Heidelberg: Springer), 164–174.

Trujillo, L., Olague, G., Lutton, E., De Vega, F. F., Dozal, L., and Clemente, E. (2011). Speciation in behavioral space for evolutionary robotics. *J. Intell. Robot. Syst.* 64, 323–351. doi:10.1007/s10846-011-9542-z

Velez, R., and Clune, J. (2014). "Novelty search creates robots with general skills for exploration," in *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation, GECCO '14* (New York, NY: ACM), 737–744.

Woolley, B. G., and Stanley, K. O. (2011). "On the deleterious effects of a priori objectives on evolution and representation," in *GECCO '11: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation* (Dublin, Ireland: ACM), 957–964.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

APPENDIX

Algorithm Descriptions

This section provides pseudocode for selected algorithms (particularly studies that were not published previously). Some algorithms represent trivial modifications to existing algorithms, in which case only the modifications are shown. In all cases, g represents an individual genome, B represents a batch of genomes (batch size: *bsize*), R represents the novelty archive (maximum archive size: *rmax*), and P represents the population or the MAP-Elites grid (population size: *psize*). Following NEAT (Stanley and Miikkulainen, 2002), genomes are initialized with minimal complexity; additional neurons and connections are added later through mutations.

Algorithm 1 | MAP-Elites + Novelty (MENOV)

```
1:  Generate and evaluate B random genomes
2:  for all g ∈ B do
3:      Add g to R
4:      Map g to its corresponding grid bin
5:      if g.fitness > bin.fitness then
6:          Save g in the bin (discard current occupant)
7:      else
8:          Discard g
9:      end if
10: end for
11: Calculate novelty scores for all g ∈ P (against R ∪ P)
12: loop
13:     Select bsize parents from P with chance proportional to novelty score
14:     Generate and evaluate B offspring as asexual mutations of selected
        parents
15:     for all g ∈ B do
16:         Add g to R
17:         Map g to its corresponding grid bin
18:         If g.fitness > bin.fitness then
19:             Save g in the bin (discard current occupant)
20:         else
21:             Discard g
22:         end if
23:     end for
24:     Calculate novelty scores for all g ∈ P (against R ∪ P)
25:     Calculate novelty scores for all g ∈ R (against R ∪ P)
26:     while R.count > rmax do
27:         Discard g ∈ R with lowest novelty score
28:     end while
29: end loop
```

Algorithm 2 | MAP-Elites + Passive Genetic Diversity (MEPGD)

```
1:  Generate and evaluate B random genomes
2:  for all g ∈ B do
3:      Map g to its corresponding grid bin
4:      if g.fitness > bin.fitness then
5:          Save g in the primary bin (discard current occupant)
6:      else if random: 30% chance then
7:          Save g in the secondary bin (discard current occupant)
8:      else
9:          Discard g
10:     end if
11: end for
12: loop
13:     Select bsize parents from P (uniform random chance)
14:     Generate and evaluate B offspring as asexual mutations of selected
        parents
15:     for all g ∈ B do
16:         Map g to its corresponding grid bin
17:         if g.fitness > bin.fitness then
18:             Save g in the primary bin (discard current occupant)
19:         else if random: 30% chance then
20:             Save g in the secondary bin (discard current occupant)
21:         else
22:             Discard g
23:         end if
24:     end for
25: end loop
```

Algorithm 3 | Multi-BC Novelty Search with Local Competition (NS-NSLC)

```
1:  Generate and evaluate B random genomes
2:  Add B to R1 (aligned BC)
3:  Add B to R2 (unaligned BC)
4:  Add B to P
5:  for all g ∈ P do
6:      Calculate novelty score g.nov1 (against R1 ∪ P with aligned BC)
7:      Calculate novelty score g.nov2 (against R2 ∪ P with unaligned BC)
8:      Calculate local competition g.lc (against R2 ∪ P with unaligned BC)
9:  end for
10: Pareto-rank all g ∈ P according to (g.nov1, g.nov2, g.lc)
11: loop
12:     Select bsize parents from P with chance inversely proportional to Pareto
        rank
13:     Generate and evaluate B offspring as asexual mutations of selected
        parents
14:     Add B to R1 (aligned BC)
15:     Add B to R2 (unaligned BC)
16:     Add B to P
17:     for all g ∈ P do
18:         Calculate novelty score g.nov1 (against R1 ∪ P with aligned BC)
19:         Calculate novelty score g.nov2 (against R2 ∪ P with unaligned BC)
20:         Calculate local competition g.lc (against R2 ∪ P with unaligned BC)
21:     end for
22:     Pareto-rank all g ∈ P according to (g.nov1, g.nov2, g.lc)
23:     Calculate novelty score g.nov1 for all g ∈ R1 (against R1 ∪ P with aligned
        BC)
24:     Calculate novelty score g.nov2 for all g ∈ R2 (against R2 ∪ P with
        unaligned BC)
25:     while P.count > psize do
26:         Discard g ∈ P with worst Pareto rank
27:     end while
28:     while R1.count > r1max do
29:         Discard g ∈ R1 with lowest novelty score g.nov1
30:     end while
31:     while R2.count > r2max do
32:         Discard g ∈ R2 with lowest novelty score g.nov2
33:     end while
34: end loop
```

Algorithm 4 | Multi-BC Novelty Search (NS-NS)

NS-NS works exactly the same as NS-NSLC, except the LC objective is removed. Modifications to Algorithm 3:

- Remove lines 8 and 20
- Change lines 10 and 22 to:
  Pareto-rank all $g \in P$ according to ($g.nov1$, $g.nov2$)

Algorithm 5 | Multi-BC MAP-Elites + Novelty (MENOV-MENOV)

```
1:  Generate and evaluate B random genomes
2:  for all g ∈ B do
```

3: Add g to $R1$ (aligned BC)
4: Add g to $R2$ (unaligned BC)
5: Map g to its corresponding grid bin in $P1$ (aligned BC)
6: **if** *g.fitness* > *bin.fitness* **then**
7: Save g in the bin (discard current occupant)
8: **else**
9: Discard g
10: **end if**
11: Map g to its corresponding grid bin in $P2$ (unaligned BC)
12: **if** *g.fitness* > *bin.fitness* **then**
13: Save g in the bin (discard current occupant)
14: **else**
15: Discard g
16: **end if**
17: **end for**
18: Calculate novelty scores *g.nov*1 for all $g \in P1$ (against $R1 \cup P1$ with aligned BC)
19: Calculate novelty scores *g.nov*2 for all $g \in P2$ (against $R2 \cup P2$ with unaligned BC)
20: **loop**
21: Select *bsize*/2 parents from $P1$ with chance proportional to novelty score *g.nov*1
22: Select *bsize*/2 parents from $P2$ with chance proportional to novelty score *g.nov*2
23: Generate and evaluate B offspring as asexual mutations of selected parents
24: **for all** $g \in B$ **do**
25: Add g to $R1$ (aligned BC)
26: Add g to $R2$ (unaligned BC)
27: Map g to its corresponding grid bin in $P1$ (aligned BC)
28: **if** *g.fitness* > *bin.fitness* **then**
29: Save g in the bin (discard current occupant)
30: **else**
31: Discard g
32: **end if**
33: Map g to its corresponding grid bin in $P2$ (unaligned BC)
34: **if** *g.fitness* > *bin.fitness* **then**
35: Save g in the bin (discard current occupant)
36: **else**
37: Discard g
38: **end if**
39: **end for**
40: Calculate novelty scores *g.nov*1 for all $g \in P1$ (against $R1 \cup P1$ with aligned BC)
41: Calculate novelty scores *g.nov*2 for all $g \in P2$ (against $R2 \cup P2$ with unaligned BC)
42: Calculate novelty scores *g.nov*1 for all $g \in R1$ (against $R1 \cup P1$ with aligned BC)
43: Calculate novelty scores *g.nov*2 for all $g \in R2$ (against $R2 \cup P2$ with unaligned BC)
44: **while** *R1.count* > *r1max* **do**
45: Discard $g \in R1$ with lowest novelty score *g.nov*1
46: **end while**
47: **while** *R2.count* > *r2max* **do**
48: Discard $g \in R2$ with lowest novelty score *g.nov*2
49: **end while**
50: **end loop**

Algorithm 6 | Multi-BC MAP-Elites (ME-ME)

ME-ME works similarly to MENOV-MENOV, except selection is uniform rather than proportional to novelty scores (thus there is no need for novelty archives $R1$ and $R2$). Modifications to Algorithm 5:

- Remove lines 3–4, 17–18, 25–26, and 40–49
- Change lines 21–22 to:
  Select *bsize*/2 parents from $P1$ (uniform random chance)
  Select *bsize*/2 parents from $P2$ (uniform random chance)

frontiers in Robotics and AI

PERSPECTIVE
published: 02 August 2016
doi: 10.3389/frobt.2016.00045

On the Critical Role of Divergent Selection in Evolvability

Joel Lehman[1], Bryan Wilder[2] and Kenneth O. Stanley[3]*

[1] Center for Computer Games Research, IT University of Copenhagen, Copenhagen, Denmark, [2] Department of Computer Science, University of Southern California, Los Angeles, CA, USA, [3] Department of EECS, University of Central Florida, Orlando, FL, USA

An ambitious goal in evolutionary robotics (ER) is to evolve increasingly complex robotic behaviors with minimal human design effort. Reaching this goal requires evolutionary algorithms that can *unlock* from genetic encodings their latent potential for evolvability. One issue clouding this goal is conceptual confusion about evolvability that often obscures important or desirable aspects of evolvability. The danger from such confusion is that it may establish unrealistic goals for evolvability that prove unproductive in practice. An important issue separate from conceptual confusion is the common misalignment between selection and evolvability in ER. While more expressive encodings can represent higher-level adaptations (e.g. sexual reproduction or developmental systems) that increase long-term evolutionary potential (i.e. evolvability), realizing such potential requires gradients of fitness and evolvability to align. In other words, selection is often a critical factor limiting increasing evolvability. Thus, drawing from a series of recent papers, this article seeks to both (1) clarify and focus the ways in which the term evolvability is used within artificial evolution and (2) argue for the importance of one type of selection, i.e. divergent selection, for enabling evolvability. The main argument is that there is a fundamental connection between divergent selection and evolvability (on both the individual and population level) that does not hold for typical goal-oriented selection. The conclusion is that selection pressure plays a critical role in realizing the potential for evolvability and that divergent selection in particular provides a principled mechanism for encouraging evolvability in artificial evolution.

Keywords: evolutionary robotics, evolvability, divergent selection, encodings, evolution of complexity

OPEN ACCESS

Edited by:
Stephane Doncieux, Pierre-and-Marie-Curie University, France

Reviewed by:
Jean-Baptiste Mouret, French Institute for Research in Computer Science and Automation, France
Anders Lyhne Christensen, University Institute of Lisbon (ISCTE-IUL), Portugal

****Correspondence:***
Joel Lehman
jleh@itu.dk

Specialty section:
This article was submitted to Evolutionary Robotics, a section of the journal Frontiers in Robotics and AI

Received: *30 March 2016*
Accepted: *08 July 2016*
Published: *02 August 2016*

Citation:
Lehman J, Wilder B and Stanley KO (2016) On the Critical Role of Divergent Selection in Evolvability. Front. Robot. AI 3:45. doi: 10.3389/frobt.2016.00045

1. INTRODUCTION

Natural evolution is an unguided process that has produced organisms with functionalities far exceeding the products of current human engineering. Although biological evolution is well studied, the abstract mechanisms through which cascades of increasingly complex functionalities evolve are not deeply understood. Supporting this claim, artificial evolutionary processes so far cannot reproduce biological levels of behavioral complexity.

This problem is well known within the evolutionary robotics (ER) community, given its aim to create complex robotic behaviors through algorithmic evolution. Note that while this paper focuses on ER, the insights likely generalize to many applications of evolutionary algorithms (EAs) beyond the black box setting (Doncieux and Mouret, 2014); for this reason, the term EA assumes domains in which observable *behavior* results from an evaluation. While ER's aims are ambitious,

in practice evolved behaviors have remained simple relative to biological organisms, and ER does not compete commercially with mainstream robotics approaches. Thus, it is natural to seek the cause of the qualitative gulf between natural organisms and ER's current products. One hypothesis is that *evolvability*, i.e. the speed of evolutionary innovation, stagnates in ER systems, whereas in biological evolution, second-order mechanisms (e.g. sexual reproduction or developmental systems) often evolve that *accelerate* innovation.

Accordingly, this paper argues that an important limiting factor in ER is bounded evolvability, which manifests on different structural levels. First, there are limits to the evolvability *possible* to attain within particular ER models, because the search space does not *contain* certain possibilities. For example, most encodings do not enable an organism to influence how its offspring are generated, even if such influence could increase evolvability. Second, there are limits *in practice* to what levels of evolvability will be attained, given how the chosen environment, form of selection pressure, and encoding interact. In particular, because the gradients of task performance and evolvability often fail to align conveniently, selection may prune hereditary pathways that lead to increased evolvability. This paper focuses on this latter factor, because while there exist many interesting proposals for more expressive encodings (Spector and Robinson, 2002; Risi et al., 2010; Lehman and Stanley, 2011b), in practice realizing their potential for evolvability remains difficult (Edmonds, 2001; Spector and Robinson, 2002; Clune et al., 2008). The position taken here is that this challenge largely results from applying selection pressure uncorrelated with evolvability.

A further issue is that the term evolvability is defined across ER studies in disjoint and conflicting ways, which conflates distinct concepts. The result is confusion over what challenges evolvability poses for ER and even over what outcomes are most desirable or possible. One particular source of confusion addressed here concerns the level of organization (e.g. the level of the individual or the population) in which it is most important to consider and encourage evolvability.

To address such confusion, this paper aggregates from an ongoing research agenda (Lehman and Stanley, 2011b; Wilder and Stanley, 2015) one coherent vision of evolvability. Rather than focus on an *individual's* hereditary potential to achieve particular goals, the idea is to encourage the *population's* hereditary potential for creative divergence. The argument is that this conception of evolvability is more realistic and productive than alternative visions, and that it aligns well with researchers' expectations of ER and intuitions about natural evolution's creativity.

Viewing evolvability from this perspective of accelerating creativity has consequences for algorithm design. In particular, this paper argues that *divergent* selection, i.e. selection that encourages simultaneous exploration of diverse solutions, aligns systematically with this type of evolvability; in contrast, more traditional *objective-based* selection, i.e. selection for improving objective performance, has no such consistent alignment. Thus, to create more prolific ER algorithms may require increasing the potential for evolvability in encodings, focusing on population-level evolvability instead of on individual-level evolvability, and guiding search through divergent selection.

2. EVOLVABILITY IN EVOLUTIONARY ROBOTICS

Because evolutionary computation (EC) as a whole struggles with evolvability (Wagner and Altenberg, 1996; Reisinger et al., 2005; Hu and Banzhaf, 2010), the subfield of ER naturally confronts the same issue (Lehman and Stanley, 2011b; Tarapore and Mouret, 2015). A distracting complication when discussing or quantifying evolvability is the lack of consensus on evolvability's definition across biology (Pigliucci, 2008), EC in general (Altenberg, 1994; Reisinger et al., 2005), or ER in particular (Lehman and Stanley, 2011b; Tarapore and Mouret, 2015).

Overall, evolvability definitions largely can be divided into two main families. One focuses on the ability to respond to particular adaptive challenges (Reisinger et al., 2005; Pigliucci, 2008; Clune et al., 2013), while the other focuses more generally on future creative potential, i.e. the variety of phenotypes reachable from an individual or population (Wagner and Altenberg, 1996; Dichtel-Danjoy and Félix, 2004; Lehman and Stanley, 2011b). Encompassing these narrower viewpoints, a larger-scale conception of evolvability is the ability to create major phenotypic, behavioral, or morphological breakthroughs (Pigliucci, 2008), e.g. the *de novo* evolution of a developmental system. A unifying point is that this kind of large-scale evolvability is evident in natural evolution and is what researchers ultimately aspire to recreate. The hope is that studying and learning how to encourage lower levels of evolvability may aid this latter pursuit.

Such plurality of definitions is not inherently problematic; evolvability as an overarching concept may simply take on different meanings when considered across different functional goals (e.g. whether privileging adaptation to a specific target or potential for founding new niches through creative divergence) or levels of organization (e.g. whether considered over individuals, populations, or species) (Pigliucci, 2008). Thus, rather than absolute truth, the choice of evolvability definition may instead reflect the aims of a particular research agenda. That is, if one hopes to create an ER algorithm that directly solves any ambitious problem through optimization, the most fitting characterization of evolvability may be one aligned with increasing the value of a static fitness function. Yet when adopting a particular definition and devising a measure of it for ER, it is important that the resulting measure still respects what is ultimately *realistic* or *possible*. The measure can then identify limitations in current algorithms and highlight possible improvements, in service of a particular research agenda.

However, some overarching aims of ER may be more realistic than others. For example, while the optimization-based abstraction of evolution has long dominated ER (Lehman and Stanley, 2010), growing evidence suggests that such an approach is misaligned with how ambitious objectives are actually achieved (Doncieux and Mouret, 2014; Stanley and Lehman, 2015) and does not scale in practice (Lehman and Stanley, 2011c; Lehman et al., 2013; Nguyen et al., 2015). Thus, viewing evolvability through the lens of optimization (i.e. evolvability as the potential for solving particular adaptive challenges) may draw focus to desirable but ultimately unrealistic goals, like the design of an algorithm that can successfully optimize toward any arbitrarily

complex robotic behavior. So while defining evolvability this way is valid, it may not fruitfully steer ER research.

In contrast, this paper instead frames evolvability by considering ER's most ambitious *creative* goals (i.e. evolvability as future creative potential) where creativity is defined here as the ability to create novel functionally complex artifacts. The argument is that framing evolvability this way draws focus to more realistic goals, because it better reflects how complex behaviors evolve in practice. That is, evolution does not craft highly complex functionalities because such behaviors are an explicit engineering objective, but instead cobbles them together through opportunistic tinkering (Jacob, 1977). Indeed, according to one mainstream biological opinion, complexity in biological evolution results from exploratory *diffusion* through the space of complexity rather than from any pervasive selective advantage that such complexity provides (Miconi, 2008; McShea and Brandon, 2010; Gould, 2011; Stanley and Lehman, 2015). In this view, complex behavior emerges as evolution creatively expands through a growing space of ecological niches (Kauffman, 2000; Schluter, 2000; Kelly, 2010), driven by opportunistic exaptation (Gould and Vrba, 1982) or ecological pressures toward diversity such as competitive exclusion (Hardin, 1960). Empirically, this understanding is supported by studies showing that complex functionalities fail to evolve without the scaffolding of functionally simpler niches (Lenski et al., 2003; Arthur and Polak, 2004), that bacteria colonies tend naturally to generate and maintain vast phenotypic diversity (Saint-Ruf et al., 2014), and that biodiversity begets further biodiversity (Jousset et al., 2016). Furthermore, viewing evolution as an open-ended creative process aligns with a similar understanding of cultural processes such as technological growth (Arthur, 2009; Kelly, 2010; Stanley and Lehman, 2015). Thus, if evolution's productivity largely derives from its capacity to diverge creatively, then *accelerating* such capacity is a logical aim for ER. This paper accordingly argues for measuring and encouraging evolvability as the potential for creative divergence, while acknowledging that other views may better fit alternative research agendas.

One representative such measure of divergent potential is to quantify the phenotypic variability of an individual (Dichtel-Danjoy and Félix, 2004; Pigliucci, 2008), i.e. the phenotypic diversity accessible within an individual's mutational neighborhood. This measure is used in practice (Lehman and Stanley, 2011b) and captures an important facet of an *individual's* potential for creative divergence. However, as explored in the next section, such individual-level focus neglects the greater possibilities enabled by considering the population as a whole (Wilder and Stanley, 2015).

3. EVOLVABILITY OF POPULATIONS AND INDIVIDUALS

When evolvability is measured and encouraged in ER, most often it is considered a property of an *individual* (Clune et al., 2013; Lehman and Stanley, 2013; Velez and Clune, 2014), often *aggregated* across the population by taking the mean or maximum value. However, simple aggregations obscure important properties unique to the population level, such as to what extent the population contains a *complementary* diversity of evolvable individuals. An alternative approach is to consider evolvability explicitly as a property of a *population* (Wilder and Stanley, 2015). For example, one such measure is the diversity of phenotypes accessible within the mutational neighborhood of all individuals in the population.

Both conceptions are logically consistent, but they imply different goals for ER. That is, benchmarking methods by their production of individual-level evolvability sets the implicit goal of producing *maximally* evolvable individuals. Even if individual evolvability is aggregated over the population by considering mean or maximum evolvability, both common aggregations can be maximized by a population converged to a *single* maximally evolvable individual. In contrast, benchmarking on population-level evolvability stresses the need for maximally evolvable populations, which may result from a diversity of individuals with complementary and specialized evolutionary potentials.

The fundamental problem with guiding ER by individual-level evolvability stems from the logical conclusion of its maximization. For evolvability measures focused on adapting to new goals, maximization requires an individual that can very quickly evolve to achieve any possible goal. For evolvability measures focused on creative potential, maximization requires an individual who can quickly lead to any possible phenotype. While such outcomes are certainly desirable, neither of these extreme visions are realistic nor do they reflect intuitions about terrestrial evolvability.

To highlight the benefits of population-level evolvability for ER, consider two controllers for a biped robot, one that locomotes with a one-legged hop and the other that uses both legs to achieve a smooth walking gait. To achieve their distinct gaits, both controllers contain specialized knowledge and thus are only *locally* evolvable, e.g. mutating the hopping controller yields many hopping gait variations, while mutating the walking controller likewise generates many variations of walking. Through the lens of individual-level evolvability, if both controllers had equivalent *quantities* of evolvability, there is no preference to preserve both in a population. In contrast, population-level evolvability recognizes the greater span of evolutionary potential when the hopper and the walker are both viable leaping-off points (see **Figure 1**).

In this way, population-level evolvability may be a more principled guiding light for ER than individual-level evolvability. As a result, increased focus on population-level evolvability may have implications for improving the design and creativity of ER algorithms and may illuminate promising research trajectories. In this spirit, the next section argues that divergent selection is an important algorithmic feature for cultivating such population-level evolvability.

4. DIVERGENT SELECTION AND EVOLVABILITY

One necessary condition for the evolution of evolvability is that selection must encourage (or at minimum, not oppose) trajectories through the search space leading to evolvable genotypes. Even if highly evolvable individuals exist within the search space, they are unlikely to be discovered if selection and evolvability are completely decorrelated. Just as degeneracy is over-represented relative to functionality within complex search spaces (i.e. a mutation more often causes catastrophe than novel functionality), there is

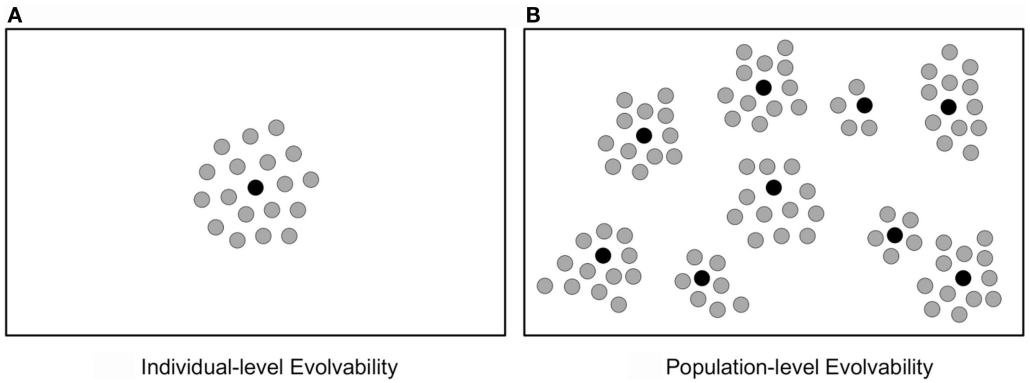


FIGURE 1 | **Encouraging evolvability on different levels of organization**. Both figures show an abstract space of phenotypes, where black circles indicate individuals and gray circles indicate phenotypes accessible within an individual's mutational neighborhood. Thus, a more expansive radius of gray circles indicates a more evolvable individual. When considering **(A)** individual-level evolvability, its maximization is achieved by a single highly evolvable individual. However, when considering **(B)** population-level evolvability, its maximization is achieved by accumulating a diversity of complementarily evolvable individuals. In this way, encouraging population-level evolvability is a more productive goal, because a broader set of phenotypes can be reached through mutations of the resulting population than through mutations of any singularly evolvable individual.

no reason to expect an arbitrary genome to be highly evolvable. Thus it is important to explore the relationship between different forms of selection pressure and evolvability. Note that a similar argument related to the evolution of complexity is provided by Miconi (2008).

The dominant form of selection pressure in ER is *objective-based* fitness: a heuristic measure of progress to some fixed objective. Search driven to optimize such a measure often *converges*, because as the population's fitness rises the set of admissible improvements shrinks. This objective-based perspective derives from abstracting natural evolution as an optimization process (Lehman and Stanley, 2011a), i.e. from considering optimization as the key ingredient that enables impressive evolutionary products.

One might intuitively assume that evolvability will naturally increase with objective-based selection, because increased evolvability will hitchhike as a byproduct of selecting the adaptations such evolvability enables. That is, an evolvable individual will be more likely than an unevolvable one to produce useful adaptions (i.e. increases in fitness); therefore, evolvability will consistently be favored as a result of selecting high-fitness individuals. However, this intuition does not hold up to closer scrutiny or empirical studies (Lehman and Stanley, 2011b).

First, there is no necessary logical connection between narrowly increasing fitness and long-term potential. For example, in ER there may be many genomes with varying evolvability that express the same high-fitness behavior (e.g. a low-evolvability memorized list of motor commands vs. a higher evolvability generic wall following policy); yet short-term fitness optimization cannot *distinguish* between them. For more evolvable lineages to reliably distinguish themselves requires competition between sufficiently *different* lineages with differing potentials. Yet maintaining such diversity is in direct tension with objective-based selection's tendency to converge. As a result, in most EAs an adaptation quickly propagates throughout the population, regardless if other reachable adaptations hold greater future potential.

A second more fundamental problem results from *deception*, i.e. the tendency in objective-based search for short-term fitness increases to themselves *anticorrelate* with long-term potential; such deception seems to increase in frequency and severity as problems scale in complexity (Zaera et al., 1996; Ficici and Pollack, 1998; Lehman and Stanley, 2011c; Stanley and Lehman, 2015). When objective-based fitness is itself a broken compass, then even if lineage-level selection could lead to evolutionary acceleration, the end effect would only be faster discovery of an ultimately unsatisfactory local optimum.

An alternative to objective-driven search is to explicitly abstract evolution as a creative process, reflecting that a key characteristic of biological evolution is its *divergent* accumulation of novelty. Practical examples of such selection in EAs are given by novelty search (Lehman and Stanley, 2011a), behavioral diversity (Mouret and Doncieux, 2012), and MAP-Elites (Mouret and Clune, 2015). In such algorithms, novelty or divergence is directly selected for, serving as a proxy for important creative processes in biology that are overlooked when evolution is treated only as an optimizer, e.g. negative frequency-dependent selection (Endler and Greenwood, 1988) and adaptive radiation (Schluter, 2000).

In contrast to objective-driven search, a consistent relationship often holds between divergent selection and evolvability (Lehman and Stanley, 2011b, 2013; Lehman and Miikkulainen, 2015; Wilder and Stanley, 2015; Mengistu et al., 2016). Importantly, unlike with *static* measures of progress, measures of divergence are *relative* to the current and past states of the search process. As a result, seeking divergence is self-defeating in a way that pursuing a fixed goal is not (see **Figure 2**). When objective-driven search is stuck within a local optimum, a high-fitness individual retains such fitness indefinitely; selection in such cases is *antagonistic* to evolvability, because phenotypic deviations are overwhelmingly unlikely to be adaptive. However, with divergent

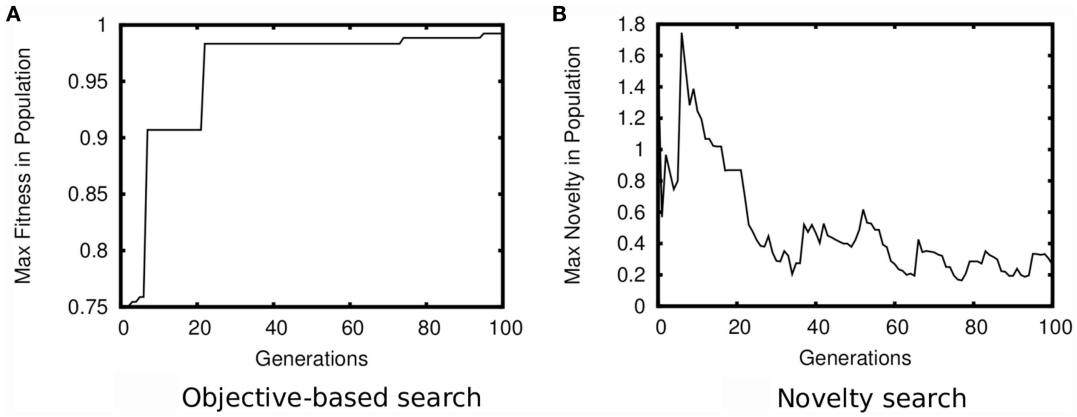


FIGURE 2 | **Contrast between optimizing absolute and relative measures**. Both figures show a plot of the maximum of an optimization measure present in the population over generations. When optimizing a static measure of progress, as in **(A)** objective-driven search, a high-fitness individual remains so indefinitely. As a result, adaptations quickly sweep through the entire population. Such convergence precludes lineages from distinguishing themselves by producing adaptations at differential rates. In contrast, when optimizing a measure of divergence, as in **(B)** novelty search, novel innovations cause only a temporary spike in the maximum novelty of the population. In this way, selection can maintain many diverging lineages that can distinguish themselves by more consistently producing novelty than the other ones. Plots are the characteristic result from running search in the medium maze domain of Lehman and Stanley (2011a).

selection, a lineage must continually innovate to remain novel. Thus there is consistent pressure to diverge, meaning mechanisms that enhance the ability to diverge can hitchhike along with the divergences they enable. Furthermore, because divergent algorithms like novelty search and MAP-Elites are designed to simultaneously maintain many distinct lineages (Lehman et al., 2012; Gomes et al., 2015; Nguyen et al., 2015), evolvable lineages can therefore distinguish themselves by consistently producing novelty. In this way, divergent selection systematically aligns with evolvability. Note that the choice of how to *measure divergence* will significantly impact the evolvability generated by a divergent EA, similarly to how such choice significantly impacts algorithmic performance (Mouret and Doncieux, 2012; Doncieux and Mouret, 2013); how to best align behavioral distance metrics with evolvability without relying on human domain expertise remains an open research question.

Importantly, this argument gracefully extends to the population level. Because divergence is the primary selection criteria, individuals are directly incentivized to diversify and spread over the space of possible phenotypes. The product is thus a diverse set of leaping-off points that enable a significant base level of population-level evolvability. At the same time, by the argument above, lineages are indirectly rewarded to consistently produce diversity, which encourages individual evolvability that is fit to an individual's local phenotypic neighborhood. In this way, divergent selection encourages a population of individuals with complementary evolvabilities, resulting in a much wider range of reachable phenotypes than that would result from considering any single individual.

Beyond theoretical arguments, empirical studies have demonstrated that divergent search often results in higher evolvability than objective-based search (Lehman and Stanley, 2011b, 2013; Lehman and Miikkulainen, 2015; Wilder and Stanley, 2015; Mengistu et al., 2016). Other studies have highlighted that objective-based search often cannot fully exploit features that enable greater *potential* for evolvability, e.g. allowing individuals to control aspects of mutation (Clune et al., 2008; Lehman and Stanley, 2011b) or of reproduction (Spector and Robinson, 2002). One important caveat for interpreting these results is that nearly all such studies focus on individual-level evolvability (Wilder and Stanley, 2015), meaning that further studies focusing on population-level evolvability are needed to validate the theory. But, taken as a whole, evidence is accumulating that divergent selection is a critical ingredient for encouraging evolvability in ER.

5. CONCLUSION

This paper addressed two connected aspects of ER: (1) what is a coherent vision that realistically relates evolvability to the goals of ER, and (2) what algorithmic considerations align with achieving that vision. The idea is that it is both realistic and desirable to create divergent EAs that indirectly optimize population-level evolvability, formalized as the potential for a population to realize phenotypic variety. Such an approach agrees with intuitions about and understanding of natural evolution, and respects limitations about what ultimately is algorithmically possible.

While divergent search methods seem more aligned with evolvability than convergent search, a vast gulf still remains between the dynamics of such algorithms and those of natural evolution. One hope is that an explicit focus on population-level evolvability may lead to algorithmic improvements that enhance evolvability. However, a more fundamental problem is that existing measures of divergence in EAs are relatively simplistic and do not enable the seemingly endless and interesting innovation observed in natural evolution. Interesting proposals for more sophisticated divergent selection pressure are beginning to emerge (Liapis et al., 2013;

Nguyen et al., 2015; Pugh et al., 2015), and are an important direction for future research.

AUTHOR CONTRIBUTIONS

JL, BW, and KS, each contributed to the conception and writing of this article.

FUNDING

This work was supported by the National Science Foundation under grant no. IIS-1421925. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

Altenberg, L. (1994). The evolution of evolvability in genetic programming. *Adv. Genet. Program.* 3, 47–74.

Arthur, W. B. (2009). *The Nature of Technology: What It Is and How It Evolves*. New York, NY: Simon and Schuster.

Arthur, W. B., and Polak, W. (2004). The evolution of technology within a simple computer model. *Complexity* 11, 23–31. doi:10.1002/cplx.20130

Clune, J., Misevic, D., Ofria, C., Lenski, R. E., Elena, S. F., and Sanjun, R. (2008). Natural selection fails to optimize mutation rates for long-term adaptation on rugged fitness landscapes. *PLoS Comput. Biol.* 4:e1000187. doi:10.1371/journal.pcbi.1000187

Clune, J., Mouret, J. B., and Lipson, H. (2013). The evolutionary origins of modularity. *Proc. Biol. Sci.* 280, 20122863. doi:10.1098/rspb.2012.2863

Dichtel-Danjoy, M.-L., and Félix, M.-A. (2004). Phenotypic neighborhood and micro-evolvability. *Trends Genet.* 20, 268–276. doi:10.1016/j.tig.2004.03.010

Doncieux, S., and Mouret, J.-B. (2013). "Behavioral diversity with multiple behavioral distances," in *2013 IEEE Congress on Evolutionary Computation (CEC)* (Washington, DC: IEEE), 1427–1434.

Doncieux, S., and Mouret, J.-B. (2014). Beyond black-box optimization: a review of selective pressures for evolutionary robotics. *Evol. Intell.* 7, 71–93. doi:10.1007/s12065-014-0110-x

Edmonds, B. (2001). Meta-genetic programming: co-evolving the operators of variation. *Turk. J. Elec. Engin.* 9, 13–29.

Endler, J. A., and Greenwood, J. J. D. (1988). Frequency-dependent predation, crypsis and aposematic coloration [and discussion]. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 319, 505–523. doi:10.1098/rstb.1988.0062

Ficici, S., and Pollack, J. B. (1998). "Challenges in coevolutionary learning: arms-race dynamics, open-endedness, and mediocre stable states," in *Proceedings of the Sixth International Conference on Artificial Life* (Cambridge, MA: MIT Press), 238–247.

Gomes, J., Mariano, P., and Christensen, A. L. (2015). "Devising effective novelty search algorithms: a comprehensive empirical study," in *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference* (New York, NY: ACM), 943–950.

Gould, S. J. (2011). *Full House*. Cambridge, MA: Harvard University Press.

Gould, S. J., and Vrba, E. S. (1982). Exaptationa missing term in the science of form. *Paleobiology* 8, 4–15. doi:10.1017/S0094837300004310

Hardin, G. (1960). The competitive exclusion principle. *Science* 131, 1292–1297. doi:10.1126/science.131.3409.1292

Hu, T., and Banzhaf, W. (2010). Evolvability and speed of evolutionary algorithms in light of recent developments in biology. *J. Artif. Evol. Appl.* 2010, 1. doi:10.1155/2010/568375

Jacob, F. (1977). Evolution and tinkering. *Science* 196, 1161–1166. doi:10.1126/science.860134

Jousset, A., Eisenhauer, N., Merker, M., Mouquet, N., and Scheu, S. (2016). High functional diversity stimulates diversification in experimental microbial communities. *Sci. Adv.* 2, e1600124. doi:10.1126/sciadv.1600124

Kauffman, S. A. (2000). *Investigations*. Oxford: Oxford University Press.

Kelly, K. (2010). *What Technology Wants*. London: Penguin.

Lehman, J., and Miikkulainen, R. (2015). "Enhancing divergent search through extinction events," in *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference* (New York, NY: ACM), 951–958.

Lehman, J., Risi, S., and Stanley, K. O. (2012). "On the benefits of divergent search for evolved representations," in *Proceedings of the EvoNet 2012 Workshop at ALIFE XIII*. Cambridge, MA.

Lehman, J., and Stanley, K. O. (2010). "Revising the evolutionary computation abstraction: minimal criteria novelty search," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2010)* (New York, NY: ACM), 103–110.

Lehman, J., and Stanley, K. O. (2011a). Abandoning objectives: evolution through the search for novelty alone. *Evol. Comput.* 19, 189–223. doi:10.1162/EVCO_a_00025

Lehman, J., and Stanley, K. O. (2011b). "Improving evolvability through novelty search and self-adaptation," in *Proceedings of the 2011 Congress on Evolutionary Computation (CEC 2011)* (Washington, DC: IEEE), 2693–2700.

Lehman, J., and Stanley, K. O. (2011c). "Novelty search and the problem with objectives," in *Genetic Programming Theory and Practice IX*, eds T. McConaghy, E. Vladislavleva, and R. Riolo (Berling, Heidelberg: Springer), 37–56.

Lehman, J., and Stanley, K. O. (2013). Evolvability is inevitable: increasing evolvability without the pressure to adapt. *PLoS ONE* 8:e62186. doi:10.1371/journal.pone.0062186

Lehman, J., Stanley, K. O., and Miikkulainen, R. (2013). "Effective diversity maintenance in deceptive domains," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2013)* (New York, NY: ACM), 215–222.

Lenski, R. E., Ofria, C., Pennock, R. T., and Adami, C. (2003). The evolutionary origin of complex features. *Nature* 423, 139–144. doi:10.1038/nature01568

Liapis, A., Martinez, H. P., Togelius, J., and Yannakakis, G. N. (2013). "Transforming exploratory creativity with delenox," in *Proceedings of the Fourth International Conference on Computational Creativity* (Cambridge, MA: AAAI Press), 56–63.

McShea, D. W., and Brandon, R. N. (2010). *Biology's First Law: The Tendency for Diversity and Complexity to Increase in Evolutionary Systems*. Chicago: University of Chicago Press.

Mengistu, H., Lehman, J., and Clune, J. (2016). "Evolvability search: directly selecting for evolvability in order to study and produce it," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2016)* (New York, NY: ACM).

Miconi, T. (2008). Evolution and complexity: the double-edged sword. *Artif. Life* 14, 325–344. doi:10.1162/artl.2008.14.3.14307

Mouret, J.-B., and Clune, J. (2015). Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*.

Mouret, J.-B., and Doncieux, S. (2012). Encouraging behavioral diversity in evolutionary robotics: an empirical study. *Evol. Comput.* 20, 91–133. doi:10.1162/EVCO_a_00048

Nguyen, A., Yosinski, J., and Clune, J. (2015). "Innovation engines: automated creativity and improved stochastic optimization via deep learning," in *Proceedings of the Genetic and Evolutionary Computation Conference*. New York, NY.

Pigliucci, M. (2008). Is evolvability evolvable? *Nat. Rev. Genet.* 9, 75–82. doi:10.1038/nrg2278

Pugh, J. K., Soros, L., Szerlip, P. A., and Stanley, K. O. (2015). "Confronting the challenge of quality diversity," in *Proceedings of the Genetic and Evolutionary Computation Conference*. New York, NY.

Reisinger, J., Stanley, K. O., and Miikkulainen, R. (2005). "Towards an empirical measure of evolvability," in *Genetic and Evolutionary Computation Conference (GECCO2005) Workshop Program* (Washington, DC: ACM Press), 257–264.

Risi, S., Lehman, J., and Stanley, K. O. (2010). "Evolving the placement and density of neurons in the hyperneat substrate," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation* (New York, NY: ACM), 563–570.

Saint-Ruf, C., Garfa-Traoré, M., Collin, V., Cordier, C., Franceschi, C., and Matic, I. (2014). Massive diversification in aging colonies of *Escherichia coli*. *J. Bacteriol.* 196, 3059–3073. doi:10.1128/JB.01421-13

Schluter, D. (2000). *The Ecology of Adaptive Radiation*. Oxford: OUP Oxford.

Spector, L., and Robinson, A. (2002). Genetic programming and autoconstructive evolution with the push programming language. *Genet. Program. Evolvable Mach.* 3, 7–40. doi:10.1023/A:1020945110902

Stanley, K. O., and Lehman, J. (2015). *Why Greatness Cannot Be Planned*. Berlin, Heidelberg: Springer Science Business Media, 978–983.

Tarapore, D., and Mouret, J.-B. (2015). Evolvability signatures of generative encodings: beyond standard performance benchmarks. *Inf. Sci.* 313, 43–61. doi:10.1016/j.ins.2015.03.046

Velez, R., and Clune, J. (2014). "Novelty search creates robots with general skills for exploration," in *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation* (New York, NY: ACM), 737–744.

Wagner, G., and Altenberg, L. (1996). Complex adaptations and the evolution of evolvability. *Evolution* 50, 967–976. doi:10.2307/2410639

Wilder, B., and Stanley, K. (2015). Reconciling explanations for the evolution of evolvability. *Adapt. Behav.* 23, 171–179. doi:10.1177/1059712315584166

Zaera, N., Cliff, D., and Bruten, J. (1996). "(Not) evolving collective behaviours in synthetic fish," in *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior* (Cambridge, MA: MIT Press Bradford Books).

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 04 July 2016
doi: 10.3389/frobt.2016.00038

Behavioral Specialization in Embodied Evolutionary Robotics: Why So Difficult?

Jean-Marc Montanier[1], Simon Carrignon[1] and Nicolas Bredeche[2]*

[1] *Computer Application in Science and Engineering (CASE), Barcelona Supercomputing Center, Barcelona, Spain,*
[2] *CNRS, Institute of Intelligent Systems and Robotics (ISIR), Sorbonne Universités, UPMC Univ. Paris 06, Paris, France*

Embodied evolutionary robotics is an on-line distributed learning method used in collective robotics where robots are facing open environments. This paper focuses on learning behavioral specialization, as defined by robots being able to demonstrate different kind of behaviors at the same time (e.g., division of labor). Using a foraging task with two resources available in limited quantities, we show that behavioral specialization is unlikely to evolve in the general case, unless very specific conditions are met regarding interactions between robots (a very sparse communication network is required) and the expected outcome of specialization (specialization into groups of similar sizes is easier to achieve). We also show that the population size (the larger the better) as well as the selection scheme used (favoring exploration over exploitation) both play important – though not always mandatory – roles. This research sheds light on why existing embodied evolution algorithms are limited with respect to learning efficient division of labor in the general case, i.e., where it is not possible to guess before deployment if behavioral specialization is required or not, and gives directions to overcome current limitations.

Keywords: embodied evolution, evolutionary robotics, behavioral specialization, division of labor, distributed online learning, collective behavior

OPEN ACCESS

Edited by:
John Howard Long, Vassar College, USA

Reviewed by:
Eliseo Ferrante, Université Libre de Bruxelles, Belgium
Eduardo J. Izquierdo, Indiana University, USA

****Correspondence:***
Jean-Marc Montanier
montanier.jeanmarc@gmail.com

Specialty section:
This article was submitted to Evolutionary Robotics, a section of the journal Frontiers in Robotics and AI

Received: *30 March 2016*
Accepted: *17 June 2016*
Published: *04 July 2016*

Citation:
Montanier J-M, Carrignon S and Bredeche N (2016) Behavioral Specialization in Embodied Evolutionary Robotics: Why So Difficult? Front. Robot. AI 3:38. doi: 10.3389/frobt.2016.00038

1. INTRODUCTION

Embodied evolutionary robotics (EER) is defined as the design of on-line distributed evolutionary algorithms to be implemented in a population of robots with limited computation and local communication capabilities (Watson et al., 2002; Eiben et al., 2010). These algorithms can be deployed in *a priori* unknown and open environments, and aim at optimizing on-the-fly the individual's and (ideally) the group's performance with respect to a pre-defined objective. EER takes its root in Evolutionary Robotics (Nolfi and Floreano, 2000; Doncieux et al., 2015), but is also related to evolutionary swarm robotics (Trianni et al., 2008), as it is sometimes (though not always) concerned with the automated design of control architecture for large, swarm-like, population of robots.

In recent years, the on-line nature of such algorithms was shown to be very robust when conducting experiments with real robots (Watson et al., 2002; Prieto et al., 2010; Bredeche et al., 2012; Trueba et al., 2013): compared with more classic evolutionary robotics setup, the emphasis in EER is on the design of robust algorithms (i.e., design *while* already deployed) rather than on producing robust solutions (i.e., design *then* deploy) (Doncieux et al., 2015; Silva et al., 2016). However, the complexity of the tasks to be achieved has been quite limited so far, either resulting with each

individual maximizing its own benefit [e.g., phototaxis (Watson et al., 2002), foraging, exploration, etc.] or in a limited level of cooperation among individuals who all display the same typical behavior [e.g., energy sharing (Montanier and Bredeche, 2011)].

To go further, evolving more complex organizations such as division of labor is clearly part of the research agenda. However, in the context of embodied evolution, the amount few works have tackled the evolution of populations where individuals can split in two (or more) sub-groups with specific roles. In this paper, we are interested in the evolution of specialized behaviors as a key milestone toward tackling more complex problems in collective robotics.

Classic (as in *off-line*) evolutionary robotics has already been used as a tool to explore important issues such as the nature of self-organized regulation mechanisms (Waibel et al., 2006; Duarte et al., 2011, 2012a,b; Lichocki et al., 2012; Ferrante et al., 2015), the benefits of communication (Trianni et al., 2007; Goldsby et al., 2010), the importance of coordination (Bernard et al., 2016b), and the trade-off between evolving polymorphic and monomorphic populations (Waibel et al., 2009; Bernard et al., 2015, 2016a; Tuci and Rabérin, 2015). However, embodied evolutionary robotics poses a problem on its own as mating and reproduction are performed *in situ*, meaning that *how* and *where* interactions between individuals are performed actually influence the course of evolution.

So far, the embodied evolution of specialized behaviors has been studied in two contexts: whether sub-tasks are geographically separated or not. First, some works considered the evolution of specialized behaviors in structured environments, where separate regions call for specific skills [e.g., cleaning tasks requiring two different methods (Prieto et al., 2010), increasing reproductive success with either phototaxis or photophobis behaviors (Bredeche et al., 2012; Bredeche, 2014)]. In this context, geographical separation plays an important role as sub-populations can evolve without interacting with one another due to the limitation in terms of communication range, therefore favoring the acquisition and conservation of different skills.

Second, other works have considered whether specialized behaviors could be acquired without geographical separation. It has been shown that specialized foraging behaviors can co-exist in a population of individuals (Haasdijk et al., 2014): faced with two resources available in limited quantity, the population evolves into two sub-groups, each specialized to forage one particular type of resource. However, this work showed that balancing between the two resources is challenging and could only be achieved by introducing a market mechanism explicitly favoring the smallest sub-group.

Possibly the most advanced work on this topic is presented by Trueba et al. (2013). The authors conducted an in-depth empirical study of behavioral specialization within the same geographic location. The authors showed that very specific values for the frequency of replacement and a carefully tuned recombination operator (with low rate) could be used to enforce behavioral specialization in a population of foraging individuals. However, validation in a realistic robotic setup remains to be done as the problem used in this study was greatly simplified: each robot's genome contains a single-gene that can take only three possible values, each value accounting for predefined behaviors.

In light of the limited results obtained so far, we address the following question in this paper: in the absence of geographical separation, **what challenges are posed by the evolution of behavioral specialization in embodied evolutionary robotics?** Specifically, we aim at identifying the limiting constraints in the evolution of behavioral specialization, including a more general formulation of the limiting factors with respect to both setups studied so far that is with or without geographical separation.

Indeed, the challenge of evolving specialization without geographical separation in embodied robotics echoes with concerns in biological speciation, where the lack of reproductive isolation is known to be a major obstacle with respect to genotypic divergence (Coyne and Orr, 1998; Gavrilets, 2003; Nosil, 2012). In this paper, we explore how reproductive isolation (whether by geographical separation or any other means) can favor the emergence of specialization, and what are the other relevant mechanisms at play. In particular, we also explore how selection and population size may impact the evolution of specialized behavior.

In the following, we perform an experimental study using different flavors of embodied evolutionary algorithms in two variations of a task with autonomous virtual robots: a foraging task *with* and *without* geographical separation. Furthermore, an abstract model is presented and used to identify the conditions required for behavioral specialization to occur in the general case, i.e., without referring to any specific evolutionary mechanisms to artificially enforce specialization, such as dedicated evolutionary operators or environment-induced phenotypic plasticity.

2. METHOD

2.1. A Foraging Task with Mutually Exclusive Resources

In order to study the evolution of behavioral specialization, we devised two experimental setups where foraging resources are required to survive. In both cases, two resources are available, and located in a particular location. These locations may change through time, requiring the agents to move accordingly. In order to successfully get energy from a particular resource, one agent must be on top of the resource location *and* must be able to *synthesize* this particular resource into energy, which requires a particular genetic trait.

Both experimental setups are defined as a circular arena without obstacles. Resources R_0 and R_1 are set at a specific location, which initial locations may differ with respect to the environment considered (cf. **Figure 1**), and which regularly moves from one location to another through a total of 8 possible locations. In the first environment (termed *collocateEnv*), the two resources are located in the same area and will move to a similar new area on a regular basis. In the second environment (termed *seperateEnv*), the two resources are located on the opposite side to one another, and will also move on a regular basis, always remaining far from one another. In both environments, resource's locations will move counter-clockwise.

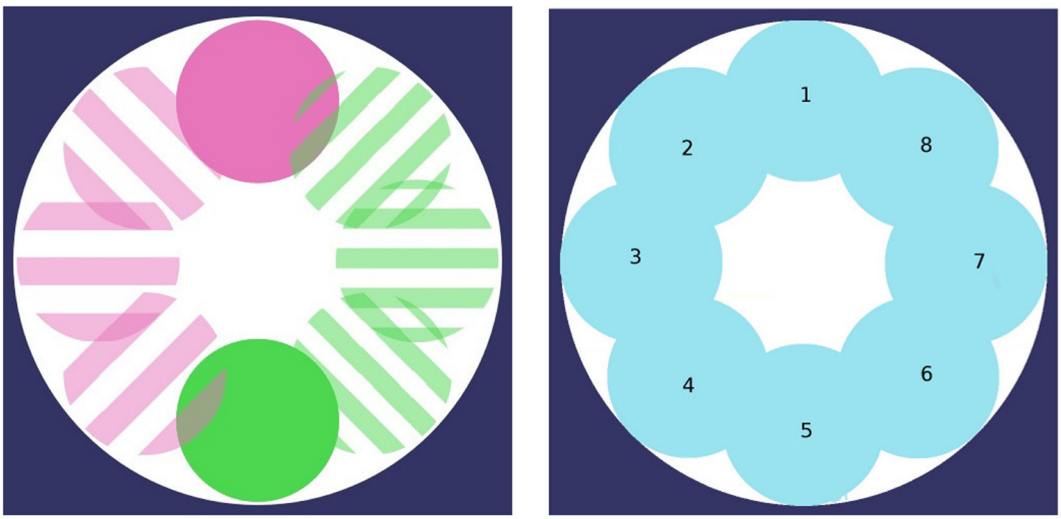


FIGURE 1 | The two foraging areas are changing locations through time (8 possible locations, moving counter-clockwise). Foraging areas may or may not share the same position. *Left*: both regions simultaneously move to the same position (*collocateEnv* setup); *right*: both regions simultaneously move, always on the opposite side from one another (*seperateEnv* setup).

The ability for an agent to synthesize energy from one resource is defined by one specific gene, termed g_{skill}. It is defined in (−1.0, +1.0), and conditions the amount energy that can be *automatically* extracted from one resource when located in its area. The energy synthesis function, F_{synth}, is shown in **Figure 2**. It illustrates that the function is designed so that an agent can get energy from *one* resource *only* (in addition to being located in the right area).

In order to account for the evolution of specialization, we introduce an additional constraint regarding the *carrying capacity* of the resources. Each resource area provides a limited amount of energy available at each time step, which is set so that only half the population can feed from a particular resource. Access to a resource is set according to a *first-come, first-served* basis: if an agent gets access to a resource, it may extract from it until it leaves the area (or until the resource area is relocated). As a result, the optimal survival strategy for the population of agents is to specialize half the agents on one resource (both in terms of tracking and synthesizing capability) and the other half on the other resource.

The fitness function for robot x at time t is defined as $fitness(x,t) = \sum_{i=t}^{t-w} f_i(x)$ where $f_i(.)$ is computed at time step i depending on the energy synthesis function F_{synth} with the value of g_{skill} as parameter *and* the availability of resources at this particular location. A sliding window of size w is used in order to get a reliable estimation of the agent performance throughout its lifetime, and no genome (nor fitness value) is broadcasted during the first w iterations. Whether the value of g_{skill} is negative or positive conditions the resource to be harvested ($g_{skill} < 0$ vs. $g_{skill} > 1$, means that resource R_o vs. R_1, will be harvested), and, *if the target resource is available at this location*, the exact value of g_{skill} determines the amount of energy to be harvested thanks to the F_{synth} function.

In this setup, *resource availability* is true if the robot is located close enough to the resource *and* if the resource has not yet reached its carrying capacity.

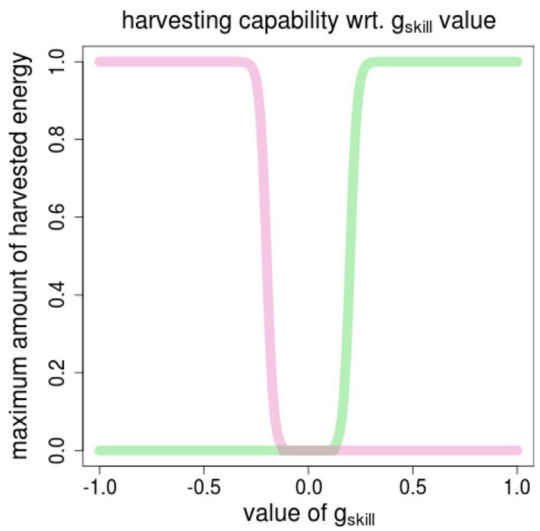


FIGURE 2 | The energy synthesis function, F_{synth} for computing the amount of foraged energy depending on the value of the g_{skill} gene. Pink (vs. green) curve shows the amount of energy of R_0 (vs. R_1) wrt. the value of g_{skill} [defined in (−1.0, +1.0)].

2.2. Algorithms

The control function for all agents is a perceptron without hidden layer and an hyperbolic tangent activation function, which maps sensory inputs (12 proximity sensors, ground detectors, energy level, angle and distance to energy sources, and a bias node) to motor outputs (left and right motor speed). All sensory and motor values are normalized in (−1, +1). This results in a total of 38 weights to evolve. The control architecture is illustrated in Supplementary Material 1.

For the sake of simplicity, we devised a simple implementation of embodied evolution, which we term *vanillaEE*. As with other embodied evolution algorithm, it is assumed that each agent is able to receive genomes and current fitness values from agents within a pre-defined range, as well as to send its own genome and current fitness value to these nearby agents. After a pre-defined duration, which corresponds to the time allowed for evaluation, one of the genomes received previously is selected, and mutated, to produce a new genome, which will be used to provide the parameters for the control function in the next evaluation period.

The pseudo-code for this algorithm is described as Algorithm 1. It starts with a randomly initialized genome, whose parameters are used to set up the neural network controller. At each time step, the agent moves according to its controller outputs and broadcasts its current genome (and possibly its current fitness). Reciprocally, it may receive incoming genome from other neighboring agents [lines 13–16 of the algorithm – the *ListeningQueue* is filled by a subroutine (not shown here) and *getListeningQueueContent*(.) (line 13) is a non-blocking call]. At the end of the evaluation time, the current genome is deleted and replaced, if available, by a genome build from the list of genomes previously received [the *select(.)* and *applyVariation(.)* functions]. Once this new genome's parameters are used to set up the new controller, the list of genomes is emptied. Being a template algorithm, the *vanillaEE* algorithm may yield many variations depending on the particular implementations of the functions used (see below).

It is important to note that selection pressure in embodied evolution acts at two levels: first, pure performance with relation to a task can be evaluated by a fitness function and used to select a particular genome; second, an agent can also boost the chance for survival of its own genome by spreading more copies of this genome than other agents do, especially if a stochastic selection operator is used.

In the following, we use two different variations over the canonical vanillaEE algorithm, instantiating a particular selection scheme for each:

- vanillaEE-elitist: the best genome out of the genomes available (i.e., received from other agents during the last evaluation session) is selected (cf. line 20 of Algorithm 1). This is a pure exploitation strategy.
- mEDEA (or vanillaEE-noFitness): selection is performed by choosing a random genome among the genomes available. Therefore, no selection pressure *wrt. fitness value* is applied, but selection pressure *wrt. ability to spread one's own genome* is still at work. The mEDEA algorithm has been extensively studied in previous works (Bredeche and Montanier, 2010; Bredeche et al., 2012; Bredeche, 2014) and provides a good baseline for embodied evolution. Efficient genomes are able to spread themselves and perform the necessary actions for survival, such as foraging, without requiring a fitness function to do so. However, the lack of selection pressure toward performance (as opposed to selection pressure toward survival only) can also lead to mitigate results with respect to foraging (i.e., no more than what is required to survive is foraged). Moreover, in some cases, the lack of selection pressure at the individual level can also ease up survival. With respect to Algorithm 1, the mEDEA algorithm does not implement the *select* operator (cf. line 20) nor the *computeFitness()* function call (cf. line 10), and does not broadcast a fitness value (cf. line 11) nor receives fitness values with incoming genomes (cf. line 13).

For both algorithms, genome initialization is performed by randomly picking weight values in $(-1.0, +1.0)$, and the variation operator is defined as a gaussian mutation with $\sigma = 0.1$. Evaluation time (or *lifetime*) for one genome is set to 600 iterations.

ALGORITHM 1 | The VanillaEE algorithm.

```
1:  genome.randomInitialize( )
2:  genomeList.empty( ) //set up a list which will be filled later with genomes
    received from neighbours.
3:  while forever do
4:    if genome! = NULL then
5:      load(genome) // set up the agent's controller wrt current genome
        parameters.
6:    end if
7:    for iteration = 0 to lifetime do
8:      if genome! = NULL then
9:        move( ) // execute the agent's controller for one step.
10:       fitness = computeFitness( )
11:       broadcast(genome,fitness) // broadcast current genome to
          neighbours (if any).
12:     end if
13:     incomingGenomes = getListeningQueueContent( ) // store any
        genome(s) (and fitness(es)) from neighbours received since last checked.
14:     if incomingGenomes != NULL then
15:       genomeList.add(incomingGenomes)
16:     end if
17:   end for
18:   genome = NULL
19:   if genomeList.size > 0 then
20:     genome = applyVariation(select(genomeList))
21:   end if
22:   genomeList.empty( )
23: end while
```

3. RESULTS

We devised a total of four setups testing all possible combinations of algorithms (mEDEA, VanillaEE-elitist) and environments (*collocateEnv*, *separateEnv*). For each setup, 50 independent runs are performed, each using the parameters described in **Table 1**. Experiments are implemented in the Roborobo 2*D* simulation tool (Bredeche et al., 2013) with 200 robotic agents.[1]

3.1. Evolution of Specialization

For each environment, two algorithms are tested: mEDEA (with *random* selection) and VanillaEE-elitist (with *elitist* selection). For each setup combining an algorithm and an environment,

[1]Code for all experiments in this Section and the next: http://pages.isir.upmc.fr/~bredeche/Experiments/Frontiers2016/

we classify each of the 50 independent runs depending on their outcomes: (1) all individuals display a similar harvesting pattern wrt. the resource harvested ("*one group*"), (2) two patterns are observed ("*two groups*"), and (3) individuals fail to harvest any resources as the population is extinct ("*extinct*"). For the first two outcomes, classification is made possible by looking at the values of g_{skill} in the population, i.e., whether there is *one* or *two* clusters of values. As an example, **Figure 3** shows all g_{skill} values in the population for two typical runs over time.

Results are shown in **Table 2**. For both algorithms, specialization fails to evolve in the *collocateEnv* environment, with the *elitist* algorithm also failing partly to even evolve any viable behaviors (two-third of the runs go extinct). The outcome is different in the *seperateEnv* environment as the mEDEA algorithm is actually able to evolve specialization in half of the runs, a result that is not observed with the VanillaEE-elitist algorithm. A statistical test (Pearson's χ^2-squared) confirms the obvious: strategies used in each environment with the random selection scheme produce significantly different results (*p-value* <0.01).

Figures 4A,B compare the outcome of runs using mEDEA algorithm in the *seperateEnv* environment. As expected, runs where two sub-groups evolve also display the highest survival rate, as specialization is the only way for the whole population to survive due to limited available amount of each resource. Results are identical with the VanillaEE-elitist algorithm.

Results show that adding a fitness function (VanillaEE-elitist) not only hinders the survival rate but also almost completely shuts down the possibility to evolve specialization. This sheds a negative light on the use of an explicit selection pressure defined through a fitness function over simply considering environmental selection pressure, as with the mEDEA algorithm. However, we can hypothesize that performing selection solely based on task-dependant fitness values does not leave much room for exploration of survival strategies.

TABLE 1 | Parameters used.

Parameter	Value
Arena width and length	1000 × 1000 space units
Duration	400 k timesteps
Evaluation duration	600 timesteps per generation
Population size	200 robots
Sensor and broadcast range	16 space units
Agent rotational velocity	30°/timestep
Agent translational velocity	2″/timestep
Genome length	37 real values
Energy diameter	140 space units
Energy per iteration	10 energy units
Agent energy consumption	1 energy unit/timestep
Agent maximum energy level	600 energy units
Agent initial energy level	400 energy units
Fitness window size	100 iterations

3.2. Investigating the Trade-off between Exploration and Exploitation

In order to explore the impact of using a fitness function, we posit that there is a trade-off between exploration, which in this case balance toward environmental selection pressure, and exploitation, i.e., selection pressure provided by using a fitness function. We introduce the VanillaEE-tournament-k algorithm, similar to what has been proposed by Fernandez Pérez et al. (2015): this algorithm uses a tournament selection of size k to regulate the trade-off between the *exploitation* of a fitness function and the *exploration* of solutions allowing the survival of robots. Tournament selection selects the best genome out of k randomly picked genomes among those received during the last

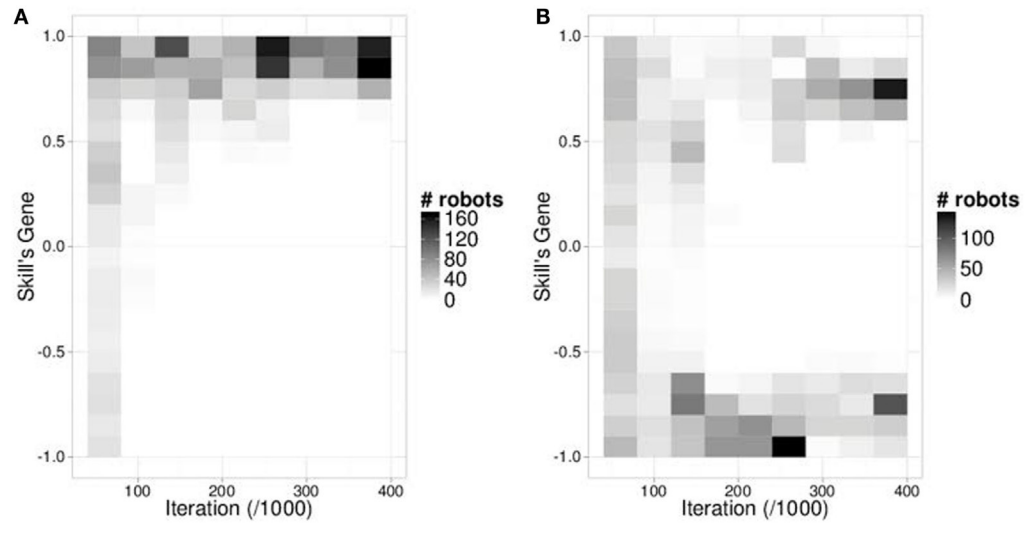


FIGURE 3 | Skill's gene heatmap representative of two situations. **(A)** heatmap of a run with one group. **(B)** heatmap of a run with two groups. The number of iteration is shown on the *x*-axis and the values of g_{skill} are shown on the *y*-axis. Darker cells' values correspond to more robots using a g_{skill} value. A run receives the label *one group* if it relies on either positive values [such as in **(A)**] or only negative values. If two groups are observed in one run (one using positive g_{skill} values and the other using negative g_{skill} values), it is tagged with the *two groups* label [such as in **(B)**].

evaluation session. Tournament sizes used are $k = 5$ and $k = 20$. Large tournament sizes tends to converge toward elitism selection (favoring exploitation) while small tournament sizes tends to favor exploration. It should be noted that the mEDEA algorithm is identical to a VanillaEE-tournament algorithm with $k = 1$, as using such a value implies random selection.

We design 4 new setups, testing all possibilities between the two tournament sizes and the two environments (*collocateEnv*, *separateEnv*). For each setup 50 independent runs are performed and classified as before. Results shown in **Tables 3** and **4** reveal that as the pressure toward exploitation increases, (a) the number of runs with extinctions also increases (in both setups) and (b) the number of runs where two specialized groups evolve decreases, at least when resources are seperated. A χ^2 statistical test is used for significance.

The smaller the tournament's size, the closer the results are to the results obtained with random selection. Reciprocally, larger size of tournament size produce results close to results obtained with the elitist selection scheme. We conclude that increasing the pressure toward the exploitation of genomes with higher fitnesses leads to sub-optimal solutions. While this mechanism allows us to mitigate the pressure from task-driven fitness function, the pressure from the environment keeps a strong influence, and the question remains open as to why specialization is (nearly) impossible when resources are *not* spatially separated.

TABLE 2 | Classification of the outcome of runs using random selection (i.e., the mEDEA algorithm) and elitist selection.

Selection	Environment	# Runs		
		Two groups	One group	Extinct
Random	*SeparateEnv*	18	32	0
Elitist	*SeparateEnv*	1	31	18
Random	*CollocateEnv*	2	42	6
Elitist	*CollocateEnv*	0	18	32

Classes are determined using the value of the skill's gene. Fifty runs per experiment. Population size is 200 robots.

3.3. Discussion

As expected from Trueba et al. (2013), we show that when resources are collocated, it is very difficult to evolve specialization. While Trueba et al. (2013) was successful at finding a very precise set of parameter values to achieve specialization, this was done under very specific conditions: either an abstract model or a toy problem (a genome with one parameter that can take one among three possible values). Our results confirm that in the general case, evolving specialization is challenging at the least, and unlikely if resources are collocated.

Similarly, and in accordance with Prieto et al. (2010), Bredeche et al. (2012), and Bredeche (2014), we show that geographical

TABLE 3 | Classification of the outcome of runs where resources are separated.

Selection	Environment	# Runs		
		Two groups	One group	Extinct
Random	*SeparateEnv*	18	32	0
Tournament-5	*SeparateEnv*	8	37	5
Tournament-20	*SeparateEnv*	3	35	12
Elitist	*SeparateEnv*	1	31	18

Classes are determined using the value of the skill's gene. Fifty runs per experiment. Population size is 200 robots. Random and elitist selection methods are copied from ***Table 2*** *for clarity.*

TABLE 4 | Classification of the outcome of runs where resources are collocated.

Selection	Environment	# Runs		
		Two groups	One group	Extinct
Random	*CollocateEnv*	2	42	6
Tournament-5	*CollocateEnv*	1	33	16
Tournament-20	*CollocateEnv*	0	16	34
Elitist	*CollocateEnv*	0	18	32

Classes are determined using the value of the skill's gene. Fifty runs per experiment. Population size is 200 robots. Random and elitist selection methods are copied from ***Table 2*** *for clarity.*

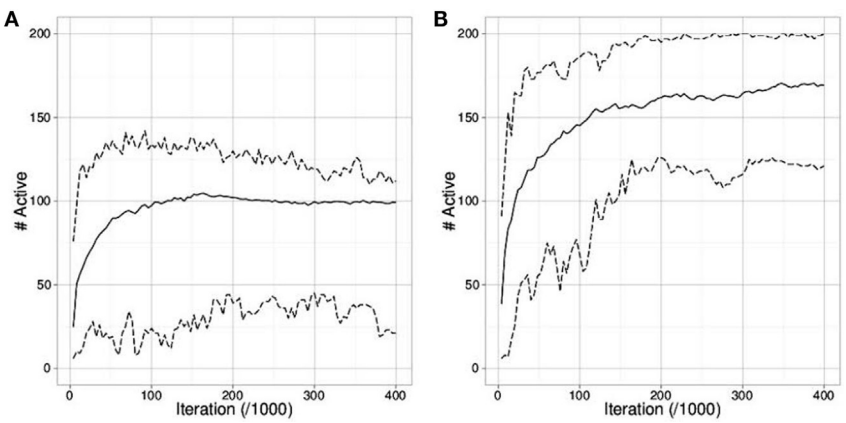


FIGURE 4 | Number of active robots (mean, min and max from the 50 runs) in the *separateEnv* where (A) only one group is evolved (i.e., one cluster of values for the g_{skill} gene) and (B) two groups are evolved (i.e., two clusters of values). The maximum number of active robots is 200.

separation promotes the evolution of specialization. However, specialization is not always evolved here (at best 36% of the runs for the best setting), while it was always the case in previously cited works. This is actually not a surprise as we face a more challenging set-up: we consider *extinction*, that is, the possibly to engage in evolutionary dead-ends. As a consequence, not only it is risky for the population to switch from one equilibrium (e.g., foraging without specialization) to another (e.g., foraging the two resources) but also historical contingencies can lead to early extinction, if foraging is not evolved in the very first generations.

From the results we obtained so far, we now question the current claim stating that one needs either geographical separation and/or very specific evolutionary operators as a necessary condition. In fact, bipolar crossover (Prieto et al., 2010), high replacement frequency (Trueba et al., 2013), low recombination rate (Trueba et al., 2013), market mechanism (Haasdijk et al., 2014), and geographical separation (Bredeche et al., 2012; Bredeche, 2014) can all be seen as coming from the same origin: a mean to achieve reproductive isolation.

Therefore, we posit this new hypothesis: **reproductive isolation is a key factor in the evolution of specialization**, whether the population is geographically dispersed or not. In order to investigate this assumption, we explore in the next Section an abstract model to study the impact of reproductive isolation independently from how it is implemented in practical, i.e., through geographical separation *or any other means*, and reveal the critical conditions for evolving specialization.

4. ANALYSIS

In order to identify the necessary conditions required to evolve specialization, we introduce an abstract model to perform computationally intensive experiments. In this model, each agent is located on a node within a graph, and each node hosts one agent only. Edges between two nodes indicate that genetic material is exchanged by the agents. Each agent lives for four iterations, has a battery that consumes one unit per iteration, and forage from resources R_0 or R_1 depending on the value of its g_{skill} gene (as before, a value close to zero means no foraging), just as in the setup used in the previous Section (except that a genome contains now a single g_{skill} gene). As before, each resource enables the survival of one half of the population, meaning that specialization into two groups is mandatory for the whole population to survive. We do not consider extinction: an agent, which runs short of energy is deactivated for 4–14 iterations (random), then listen to its neighbors during 4 iterations, and is finally reactivated using one of the received (and mutated) genome.

For each run, we randomly generate graphs, fixing only the number of nodes and the average number of edges for each node.

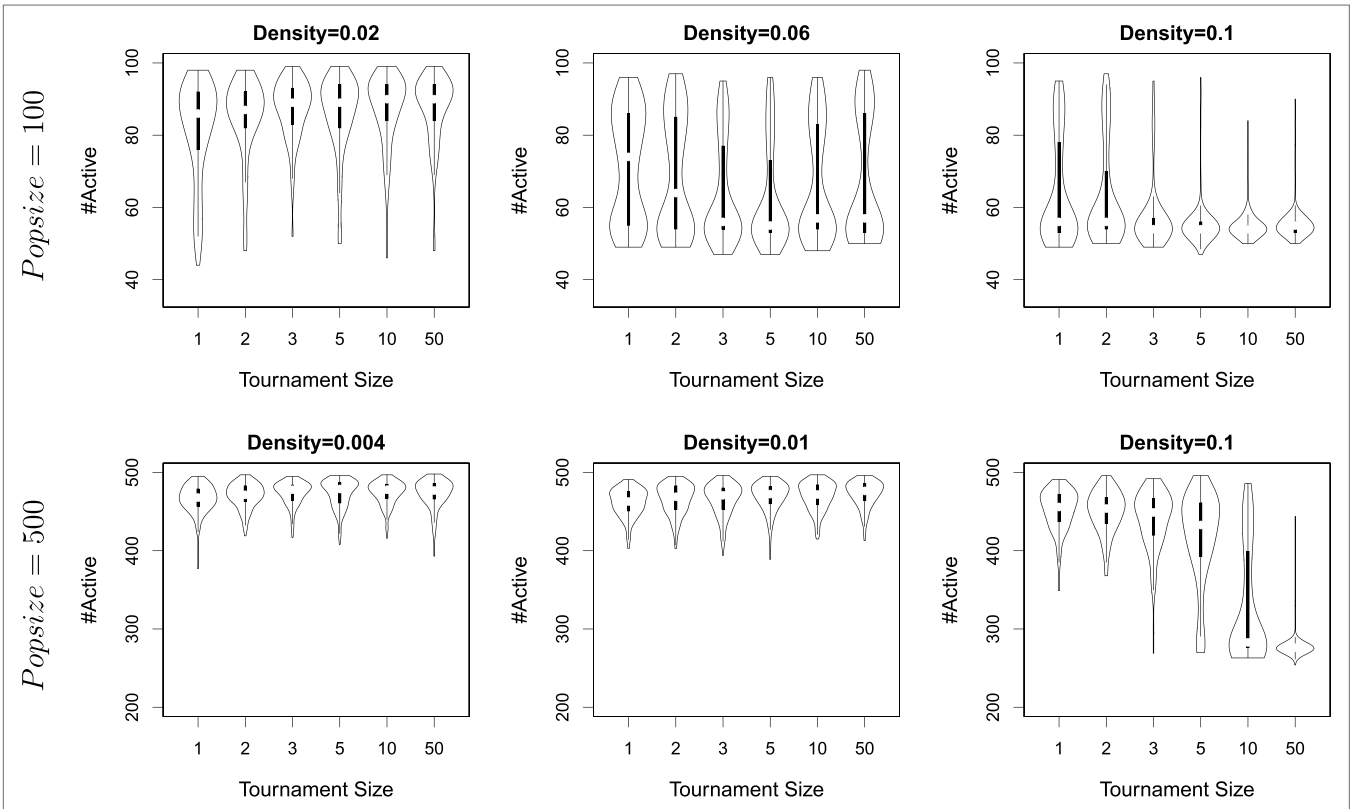


FIGURE 5 | The number of agents alive at the end of a simulation for different densities, different tournament sizes (from 1 to 50), and population sizes (100 and 500). Each violon plots is built from the result of 160 independent runs.

The number of edges acts as a proxy for the study of reproductive isolation. In order to do so, we have devised a method, which generates random connex graphs with a desired *density* of edges between nodes. The minimal density is $\frac{2}{n}$ (i.e., a ring), with n the number of agents and the maximal density is 1.0 (i.e., a complete graph). Supplementary Material 2 provides a formal definition of density, and Supplementary Material 3 provides the pseudo-code for the graph generation algorithm.

This simplified model corresponds to the *collocateEnv* environment used earlier, where all agents may access any of the two resources at all time, but with interactions between agents being determined by the selected density. In this Section, we first explore what are the critical parameters and parameter values that makes specialization possible (Subsection 4.1). Then, we investigate whether the non-homogeneous availability of resources impacts (or not) the possibility to evolve specialization (Subsection 4.2).

4.1. Interaction between Reproductive Isolation, Population Size, and Selection Pressure

We identify three candidate hypotheses that, if true, may lead to the evolution of specialization:

1. increasing *reproductive isolation* may act as a protection for groups with different skills to co-exist. This will be tested by varying the graphs' *density*, i.e., the mating opportunities for each agent;
2. increasing *population size* may reduce the stochastic effect known to occur in small populations, which could otherwise hinder the fixation of beneficial mutations. Two different population sizes will be tested, using graphs with 100 and 500 nodes.
3. increasing *exploration* (over *exploitation*) may help to escape local minima, defined here as the convergence for the whole population to one efficient, yet sub-optimal, behavior (e.g., foraging only one resource). Tournament selection with various tournament sizes (k) will be tested, from $k = 1$ (i.e., MEDEA, emphasizing exploration) to $k = 50$ (i.e., selection largely favoring exploitation).

Figure 5 shows the results obtained with different tournament sizes ($k = 1, 2, 3, 5, 10, 50$), population size (100 and 500) and densities (starting from the minimal density wrt. population size). For each parameter sets, 160 independent runs are conducted (i.e., a total of 5760 runs). Results are compiled from the data of the last generation for each run and shown as violin plots to capture the details of possibly non-uniform distributions (i.e., full histograms, rather than box-plots).

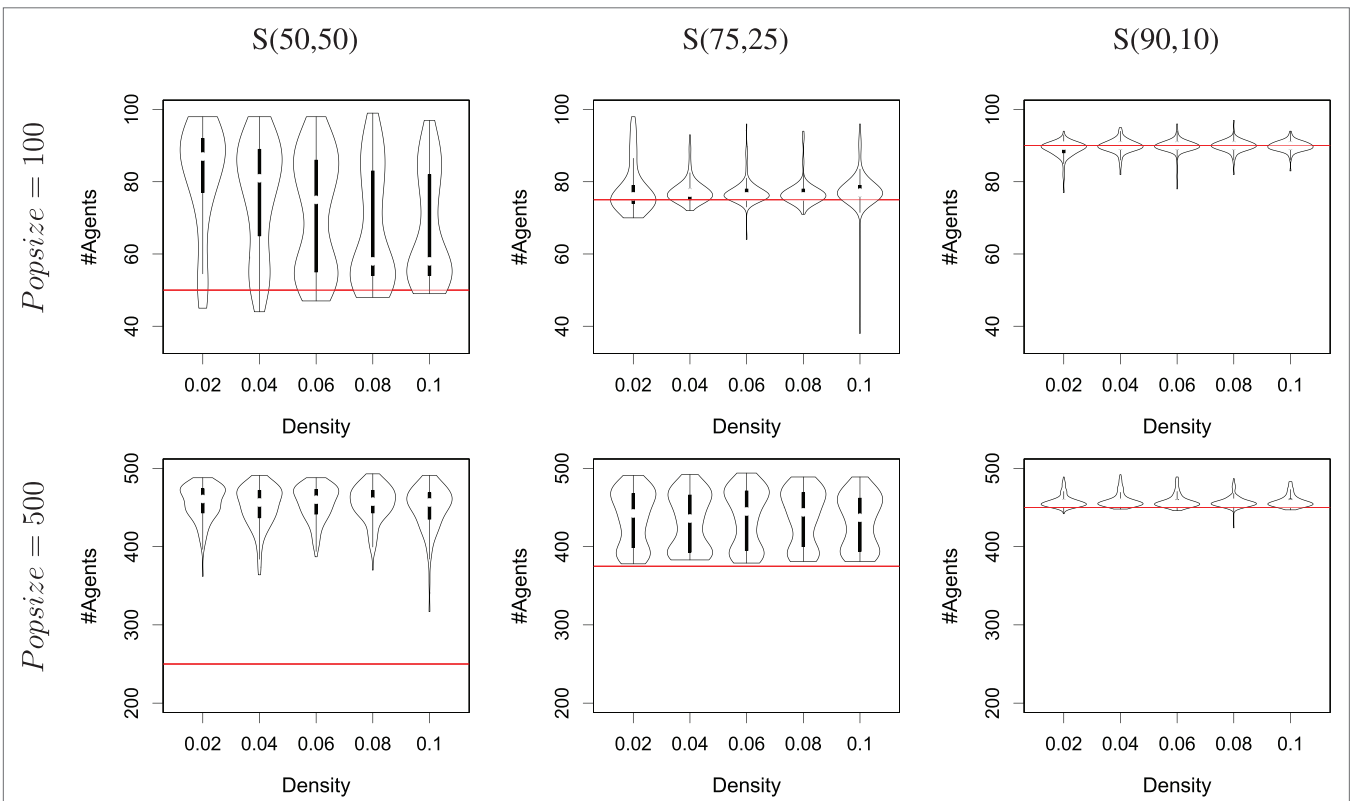


FIGURE 6 | The number of agents alive at the end of a simulation depending on the distribution of resources available for different densities and population sizes (100 and 500), using the mEDEA algorithm. Each violon plots is built from the result of 160 independent runs. The column marked *S*(50,50) is a recap from **Figure 5** for ease of comparison. The column marked *S*(75,25) show results in an environment where resource R_0 (vs. R_1) provides 75% (vs. 25%) of the amount of energy required for the whole population to survive. The column marked *S*(90,10) does the same for a 90/10 balance between the two resources. The horizontal line, drawn in *red*, shows the theoretical upper limit in terms of number of agents alive if only one resource is foraged.

All three hypotheses are validated, though their importance varies. Lower densities *always* lead to specialization, whatever the population size or the selection pressure. A large population size favors specialization, and selection favoring exploitation (i.e., larger tournament sizes) turns to be detrimental as density increases. Actually, the lack of detrimental effect of large tournament sizes when density is low can be explained that for the lowest densities considered, tournament sizes is quickly far superior to the actual number of neighbors (e.g., for the lowest densities, the number of neighbors for one node is 2) – in other words, tournament sizes with $k > 2$ do not impact further the outcome of selection for such low densities.

As a conclusion to our original question, reproductive isolation is a key factor for evolving specialization, with a larger population size and a selection scheme favoring exploration rather than exploitation as secondary factors. Results from this Section also shed light on the negative results obtained in Section 3: the failure of all algorithms to achieve specialization in the *collocateEnv* environment is explained by the lack of restrictions on mating opportunities (all individuals are mixed together).

4.2. Deleterious Effects of Non-Homogeneous Resources Availability

So far, we have considered situations where both resources provide the same *quantity* of energy. We now depart from our initial question to consider the possible impact of resources being available in different amounts. We use the same setup as in the previous Subsection, with only $k = 1$ as tournament size (i.e., the MEDEA algorithm), and consider two environments [termed $S(75,25)$ and $S(90,10)$] with different resource distributions: one where R_0 (vs. R_1) provides 75% (vs. 25%) of the amount of energy required to sustain the whole population, and another where R_0 (vs. R_1) provides 90% (vs. 10%) of the amount of energy.

Results are shown in **Figure 6**, compiled from 160 runs for each set of parameters (i.e., a total of 3200 runs). Specialization can be observed whenever a run displays more active agents that can be sustained with one resource only (cf. the *red* lines in the graphs, which mark the maximum level of sustainability by foraging only the largest resource). When the distribution of

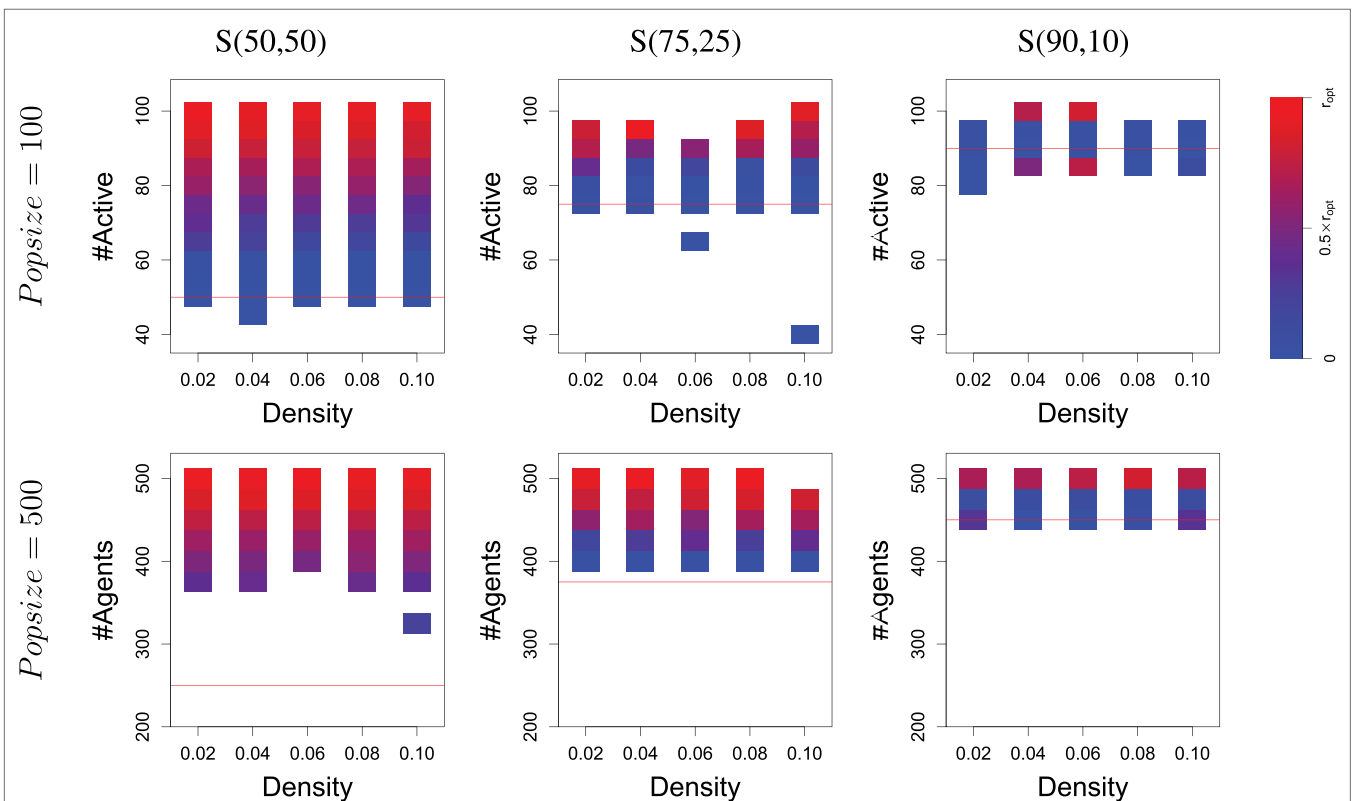


FIGURE 7 | The level of specialization (at the end of evolution) within populations depending on the number of active agents. Results are similar to **Figure 6** but display the average specialization level observed in runs, rather than the number of runs. For each cell (i.e., rectangle covering a small interval in the number of active agents), the level of specialization is calculated as the distance between the ideal distribution over the two resources and the observed distribution (averaged over all runs considered in this cell). Blue means no specialization (i.e., only one resource foraged); red means specialization (i.e., population is split in two groups, each with an optimal size wrt. resources availability). A detailed explanation of how specialization is computed can be found in Supplementary Material 4. Note that the two cells colored red which are below the threshold line in the upper-left graph are due to successful specialization but only within a sub-part of the population (i.e., some agents carry a g_{skill} value around zero). It is important to keep in mind that cells may correspond to very different number of runs: this Figure provides *complementary information* to what is shown in **Figure 6**.

resources follows a 75/25 distribution, more than half of the runs with a population of 100 ends up with specialization, as well as *all* runs with a population of 500. Specialization is confirmed by looking at the middle column of **Figure 7**, which displays the same results as the previous, but over-emphasizes on the (possibly different) values of g_{skill} rather than the number of runs. The distribution of the g_{skill} gene values show that all runs that display a survival success to be high above the threshold (the red lines in the graphs) are explained by the occurrence of two groups of individuals, one specialized to forage R_0 and the other to forage R_1.

Results are different when resources follow a 90/10 distribution, as **Figure 6** displays a survival rate around the threshold for runs with a population size of 100 and slightly above for most runs with a population size of 500. A Wilcoxon rank-sum test confirms that using a population of 500 yields significantly better results than a population of 100. Again, **Figure 7** provides a more precise analysis. Runs with a population of 100 almost never display specialization while runs with a population of 500 and a high survival rate are always displaying specialization (the top red boxes in the bottom-right graph).

Heterogeneous distributions of resources do have a negative effect on the ability to evolve specialization, though it can be mitigated – to a limited extent – by increasing the population size. This poses yet another challenge about ensuring the evolution of specialization even with a small population and unbalanced distribution of resources availability.

5. CONCLUSION AND PERSPECTIVES

In this paper, we explored why evolving behavior specialization remains an important challenge in embodied evolutionary robotics. We defined a foraging task where two resources are available in limited quantity to identify the critical parameters at work in the evolution of specialization. We implemented this task in both a pseudo-realistic robotic simulation and an abstract graph-based model.

The take-home messages from this work are threefold. First, reproductive isolation is mandatory for the evolution of specialization, whether such isolation is due to geographic constraints or particular mating strategies. This may open ways toward defining new mechanisms and/or operators to reduce the amount of mating interactions between individuals, such as preferential choice.

Second, larger population sizes also help, leading toward an important remark: a significant amount of works in embodied evolutionary robotics are concerned with small populations (approximately 10 robots), and face problems that are possibly unique to such population sizes. To some extent, embodied evolution with either small or large populations may well be to two different classes of problems, each with their own issues, and we ought to be cautious not to generalize conclusions obtained with larger populations to smaller populations, and reciprocally.

Third, a selection method should leave room to exploration, to be understood as performing a trade-off between environment-driven selection versus task-driven fitness function selection. The benefit of such a trade-off has already been explored elsewhere (Haasdijk et al., 2014), but mechanisms favoring exploration explicitly could also be explored [e.g., applying novelty measures for evolutionary swarm robotics (Gomes et al., 2013)]. Here lies an important aspect of embodied evolution: mating is evolved as a strategy, and is not given for free as an algorithmic feature as would be the case with a more classic evolutionary algorithm.

As for future works, a natural extension of this work is to consider the evolution of specialization into more than two subgroups, as well as to consider behaviors that are substantially different. So far, we have considered specialization as being the product of few skills (ability to forage one resource, and possibly to track its location), which may imply a limited distance in the genotypic space between the two genetic codes. Things may be very different if two (or more) behaviors exist in very different locations of the search space: except for very specific historical contingencies or dedicated operators it might be very difficult to co-evolve both behaviors simultaneously.

Another extension of this work is to consider setups where both generalists and specialists can evolve. Even if a population of generalists may provide a suboptimal solution, it is not clear that specialists could still evolve, even when the conditions discussed throughout this paper are met. Finally, it is also not clear what would be the respective advantages and drawbacks to achieving behavioral specialization through evolutionary adaptation (as explored here) versus lifetime adaptation (e.g., learning or memory mechanisms).

AUTHOR CONTRIBUTIONS

Conceived and designed the experiments: J-MM, SC, and NB. Analysed the data: J-MM, SC, and NB. Performed the experiments: J-MM and SC. Coordinated the writing: NB.

ACKNOWLEDGMENTS

This work is supported by the European Unions Horizon 2020 research and innovation programme under grant agreement No 640891, and the ERC Advanced Grant EPNet (340828). Part of the experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER, and several Universities as well as other funding bodies (see https://www.grid5000.fr). The other parts of the simulations have been done in the supercomputer MareNostrum at Barcelona Supercomputing Center – Centro Nacional de Supercomputacion (The Spanish National Supercomputing Center).

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at http://journal.frontiersin.org/article/10.3389/frobt.2016.00038

REFERENCES

Bernard, A., André, J.-B., and Bredeche, N. (2015). "Evolution of cooperation in evolutionary robotics: the tradeoff between evolvability and efficiency," in *European Conference on Artificial Life (ECAL 2015)* (Cambridge, MA: MIT press), 495–502.

Bernard, A., André, J.-B., and Bredeche, N. (2016a). "Evolving specialisation in a population of heterogeneous robots: the challenge of bootstrapping and maintaining genotypic polymorphism," in *Artificial Life 15: Proceedings of the Fifteenth International Conference on the Synthesis and Simulation of Living Systems*, 1–8.

Bernard, A., André, J.-B., and Bredeche, N. (2016b). To cooperate or not to cooperate: why behavioural mechanisms matter. *PLoS Comput. Biol.* 12:e1004886. doi:10.1371/journal.pcbi.1004886

Bredeche, N. (2014). "Embodied evolutionary robotics with large number of robots," in *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, Vol. 1 (New York: The MIT Press), 272–273.

Bredeche, N., and Montanier, J.-M. (2010). "Environment-driven embodied evolution in a population of autonomous agents," in *Parallel Problem Solving from Nature (PPSN)* (Berlin; Heidelberg: Springer), 290–299.

Bredeche, N., Montanier, J.-M., Liu, W., and Winfield, A. F. T. (2012). Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents. *Math. Comput. Model. Dyn. Syst.* 18, 101–129. doi:10.1080/13873954.2011.547660

Bredeche, N., Montanier, J.-M., Weel, B., and Haasdijk, E. (2013). Roborobo! a fast robot simulator for swarm and collective robotics. *CoRR* 2, 2.

Coyne, J. A., and Orr, H. A. (1998). The evolutionary genetics of speciation. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 353, 287–305. doi:10.1098/rstb.1998.0210

Doncieux, S., Bredeche, N., Mouret, J.-B., and Eiben, A. E. (2015). Evolutionary robotics: what, why, and where to. *Front. Robot. AI* 2:4. doi:10.3389/frobt.2015.00004

Duarte, A., Pen, I., Keller, L., and Weissing, F. J. (2012a). Evolution of self-organized division of labor in a response threshold model. *Behav. Ecol. Sociobiol.* 66, 947–957. doi:10.1007/s00265-012-1343-2

Duarte, A., Scholtens, E., and Weissing, F. J. (2012b). Implications of behavioral architecture for the evolution of self-organized division of labor. *PLoS Comput. Biol.* 8:e1002430. doi:10.1371/journal.pcbi.1002430

Duarte, A., Weissing, F. J., Pen, I., and Keller, L. (2011). An evolutionary perspective on self-organized division of labor in social insects. *Ann. Rev. Ecol. Evol. Syst.* 42, 91–110. doi:10.1146/annurev-ecolsys-102710-145017

Eiben, A. E., Haasdijk, E., and Bredeche, N. (2010). "Chapter 5.2: embodied, on-line, on-board evolution for autonomous robotics," in *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*, eds P. Levi and S. Kernbach (Berlin; Heidelberg: Springer), 361–382.

Fernandez Pérez, I., Boumaza, A., and Charpillet, F. (2015). "Decentralized innovation marking for neural controllers in embodied evolution," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (New York: ACM), 161–168.

Ferrante, E., Turgut, A. E., Duéñez-Guzman, E., Dorigo, M., and Wenseleers, T. (2015). Evolution of self-organized task specialization in robot swarms. *PLoS Comput. Biol.* 11:e1004273. doi:10.1371/journal.pcbi.1004273

Gavrilets, S. (2003). Perspective: models of speciation: what have we learned in 40 years? *Evolution* 57, 2197–2215. doi:10.1111/j.0014-3820.2003.tb00233.x

Goldsby, H. J., Knoester, D. B., and Ofria, C. (2010). "Evolution of division of labor in genetically homogenous groups," in *GECCO 2010 – Proceedings of the 2010 Genetic and Evolutionary Computation Conference*, 135–142.

Gomes, J., Urbano, P., and Christensen, A. (2013). Evolution of swarm robotics systems with novelty search. *Swarm Intell.* 7, 115–144. doi:10.1007/s11721-013-0081-z

Haasdijk, E., Bredeche, N., and Eiben, A. E. (2014). Combining environment-driven adaptation and task-driven optimisation in evolutionary robotics. *PLoS ONE* 9:e98466. doi:10.1371/journal.pone.0098466

Lichocki, P., Tarapore, D., Keller, L., and Floreano, D. (2012). Neural networks as mechanisms to regulate division of labor. *Am. Nat.* 179, 391–400. doi:10.1086/664079

Montanier, J.-M., and Bredeche, N. (2011). "Surviving the tragedy of commons: emergence of altruism in a population of evolving autonomous agents," in *European Conference on Artificial Life* (Cambridge, MA: MIT press), 550–557.

Nolfi, S., and Floreano, D. (2000). *Evolutionary Robotics: The Biology, Intelligence, and Technology*. Cambridge, MA: MIT Press.

Nosil, P. (2012). *Ecological Speciation*. Oxford: Oxford University Press.

Prieto, A., Becerra, J., Bellas, F., and Duro, R. (2010). Open-ended evolution as a means to self-organize heterogeneous multi-robot systems in real time. *Rob. Auton. Syst.* 58, 1282–1291. doi:10.1016/j.robot.2010.08.004

Silva, F., Duarte, M., Correia, L., Oliveira, S. M., and Christensen, A. L. (2016). Open issues in evolutionary robotics. *Evol. Comput.* 24, 205–236. doi:10.1162/EVCO_a_00172

Trianni, V., Ampatzis, C., Christensen, A. L., Tuci, E., Dorigo, M., and Nolfi, S. (2007). "From solitary to collective behaviours: decision making and cooperation," in *Advances in Artificial Life, ECAL 2007* (Berlin; Heidelberg: Springer), 575–584.

Trianni, V., Nolfi, S., and Dorigo, M. (2008). "Evolution, self-organisation and swarm robotics," in *Swarm Intelligence* (Berlin; Heidelberg: Springer), 163–191.

Trueba, P., Prieto, A., Bellas, F., Caamaño, P., and Duro, R. (2013). Specialization analysis of embodied evolution for robotic collective tasks. *Rob. Auton. Syst.* 61, 682–693. doi:10.1016/j.robot.2012.08.005

Tuci, E., and Rabérin, A. (2015). On the design of generalist strategies for swarms of simulated robots engaged in a task-allocation scenario. *Swarm Intell.* 9, 267–290. doi:10.1007/s11721-015-0113-y

Waibel, M., Floreano, D., Magnenat, S., and Keller, L. (2006). Division of labour and colony efficiency in social insects: effects of interactions between genetic architecture, colony kin structure and rate of perturbations. *Proc. Biol. Sci.* 273, 1815–1823. doi:10.1098/rspb.2006.3513

Waibel, M., Keller, L., and Floreano, D. (2009). Genetic team composition and level of selection in the evolution of cooperation. *IEEE Trans. Evol. Comput.* 13, 648–660. doi:10.1109/TEVC.2008.2011741

Watson, R. A., Ficici, S. G., and Pollack, J. B. (2002). Embodied evolution: distributing an evolutionary algorithm in a population of robots. *Rob. Auton. Syst.* 39, 1–18. doi:10.1016/S0921-8890(02)00170-7

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 07 October 2016
doi: 10.3389/frobt.2016.00059

Morphological Modularity Can Enable the Evolution of Robot Behavior to Scale Linearly with the Number of Environmental Features

Collin K. Cappelle[1], Anton Bernatskiy[1], Kenneth Livingston[2], Nicholas Livingston[3] and Josh Bongard[1]*

[1] Department of Computer Science, University of Vermont, Burlington, VT, USA, [2] Department of Cognitive Science, Vassar College, Poughkeepsie, NY, USA, [3] Robotics Laboratory, Vassar College, Poughkeepsie, NY, USA

In evolutionary robotics, populations of robots are typically trained in simulation before one or more of them are instantiated as physical robots. However, in order to evolve robust behavior, each robot must be evaluated in multiple environments. If an environment is characterized by f free parameters, each of which can take one of n_p features, each robot must be evaluated in all n_p^f environments to ensure robustness. Here, we show that if the robots are constrained to have modular morphologies and controllers, they only need to be evaluated in n_p environments to reach the same level of robustness. This becomes possible because the robots evolve such that each module of the morphology allows the controller to independently recognize a familiar percept in the environment, and each percept corresponds to one of the environmental free parameters. When exposed to a new environment, the robot perceives it as a novel combination of familiar percepts which it can solve without requiring further training. A non-modular morphology and controller however perceives the same environment as a completely novel environment, requiring further training. This acceleration in evolvability – the rate of the evolution of adaptive and robust behavior – suggests that evolutionary robotics may become a scalable approach for automatically creating complex autonomous machines, if the evolution of neural and morphological modularity is taken into account.

Keywords: evolutionary robotics, modularity, evolvability, evolutionary algorithms, embodied cognition

OPEN ACCESS

Edited by:
Stephane Doncieux, UPMC, France

Reviewed by:
Jean-Baptiste Mouret, Inria, France
Sylvain Cussat-Blanc, University of Toulouse, France

****Correspondence:***
Collin K. Cappelle
collin.cappelle@uvm.edu

Specialty section:
This article was submitted to Evolutionary Robotics, a section of the journal Frontiers in Robotics and AI

Received: *30 March 2016*
Accepted: *20 September 2016*
Published: *07 October 2016*

Citation:
Cappelle CK, Bernatskiy A, Livingston K, Livingston N and Bongard J (2016) Morphological Modularity Can Enable the Evolution of Robot Behavior to Scale Linearly with the Number of Environmental Features. Front. Robot. AI 3:59. doi: 10.3389/frobt.2016.00059

1. INTRODUCTION

Matarić and Cliff (1996) pointed out that the time necessary to evolve robots grows with the number of environments in which the robot should behave correctly. Following their work, let f be the number of free parameters in the environmental set and n_p be the number of features for each of these free parameters. So, the total number of environments is n_p^f. (For example, if a robot must behave appropriately in environments containing two objects ($f = 2$), and each object may be small, medium, or large ($n_p = 3$), then there are $n_p^f = 3^2 = 9$ possible environments in which the robot must perform correctly.) Thus, in order to evolve robots to perform complex behavior (which means increasing n_p, f, or both) the number of environments the robot needs to be evolved in scales exponentially.

Pinville et al. (2011) presented one way to reduce the number of environments evaluated while still obtaining robust and generalized controllers for evolved robots. Using the *ProGAb* approach, they were able to successfully obtain robust and successful controllers with better generalization abilities in less time than other top methods. However, their work did neither look specifically at the structure of the controller and morphology as methods to reduce the necessary number of environments nor did it categorize which environments should be trained on.

The work presented here demonstrates that morphological and neural modularity is one possible way to reduce the number of environments needed for evolving robust behavior.

Modularity is ubiquitous at all levels of biological organization, from cells to distinct species. Explaining why such modularity exists, and how it evolved, remains an important question in biology. Much work has focused on how modularity evolves in non-embodied systems, but relatively little work has focused on the impact of modularity in evolving embodied systems. The work presented here contributes to this latter aim.

1.1. Non-Embodied Modularity

Wagner (1996) argued that a combination of directional and stabilizing selection, acting on different parts of the organism's phenotype, should lead to modular developmental programs. Such modularity would enable evolutionary changes to that part of the phenotype experiencing directional selection while retaining the structure and function of the other parts of the phenotype under stabilizing selection.

This theoretical argument was confirmed by a number of computational experiments. Lipson et al. (2002) showed that environmental change can be a catalyst for the evolution of modularity. That work was followed by experiments in which non-embodied Boolean networks (Espinosa-Soto and Wagner, 2010) or neural networks (Kashtan and Alon, 2005; Clune et al., 2013) were evolved to perform various tasks. The tasks and fitness functions were chosen in such a way as to favor networks that computed partial results using separate genetic or neural modules; changes to the fitness function over evolutionary time favored networks that could rapidly change how those partial results were combined. Thus, stabilizing selection came to bear on the partial results, while directional selection acted on how those partial results were combined.

More recently, it has been shown that selecting sparse networks helps to favor the evolution of modular networks. Espinosa-Soto and Wagner (2010) accomplished this by formulating a biased mutation operator that favors low in-degree network nodes. Clune et al. (2013) used a multi-objective approach, in which one objective was to minimize the number of edges in the network. Bernatskiy and Bongard (2015) showed that this relationship between sparsity and modularity can be exploited to enhance the evolution of modular networks by seeding the initial population with sparse, rather than random, networks.

Modularity is a desirable property of artificial systems for a number of reasons, beyond just the desire to create biologically inspired artifacts. First, modular systems possess a form of robustness: modular systems can more rapidly adapt to certain kinds of changes in their environments, compared to non-modular systems. Second, modular neural networks are better able to avoid catastrophic forgetting than non-modular networks (Ellefsen et al., 2015). Catastrophic forgetting (French, 1999) is a common problem in machine learning, whereby a learner must forget something in order to learn something new. Third, complex predictive models and dense, non-modular networks can suffer from the pathology of overfitting: they fail to generalize to novel environments (Kouvaris et al., 2015). Modular networks can avoid overfitting by internally reflecting the modularity in its environment: it responds appropriately in a "new" environment, which is actually just an unfamiliar combination of familiar percepts.

1.2. Embodied Modularity

A modular robot may likewise be robust and avoid catastrophic forgetting and overfitting, but there are additional challenges that arise when evolving embodied agents compared to non-embodied networks and morphologies.

Embodied cognition is a particular approach to the understanding of intelligence, which holds that the body must necessarily be taken into any account of adaptive behavior (Brooks, 1990; Clark, 1998; Pfeifer and Bongard, 2006). One repercussion of the embodied cognitive stance is that if neural controllers are evolved for artificial embodied agents (i.e., robots), a given robot body plan may facilitate or hinder the evolution of desirable traits. In the context of modularity, previous work showed that there do exist body plans in which modular neural controllers will evolve (Bongard, 2011).

Follow-on work demonstrated that, given appropriate conditions, evolution will find such body plans (Bongard et al., 2015). However, in Bongard et al. (2015), the morphology itself was not modular, only the neural networks that evolved to control it.

Here, we investigate another aspect of the relationship between morphology and modularity: for a given task environment, must both the body and neural controller be modular, and if so, in what way? Before addressing these issues, however, we must define both neural modularity and morphological modularity.

1.3. Neural Modularity and Morphological Modularity

In this work, we investigate robots controlled by artificial neural networks. A common approach to measuring the amount of modularity in a network is to investigate its connectivity: a network that has dense connectivity within subsets of nodes, and relative sparsity between those subsets, is said to be modular (Newman, 2006). Following this approach, we here investigate modular neural controllers in which subsets of sensor, internal, and motor neurons are connected, but there are no synaptic connections between these subsets. We compare these to non-modular networks in which any sensor can influence any motor.

In a neural controller in which sensor information flows from sensor neurons to internal neurons to motor neurons, this structural approach to modularity implies a functional repercussion. If subsets of sensors and motors are completely structurally independent, they will be functionally independent as well: changes to a subset of sensors will only have an influence on a subset of motors.

Thus, we here define neural modularity in the following manner.

1.3.1. Neural Modularity

A neural network with i sensor neurons $S = \{s_1, s_2, \ldots, s_i\}$ and j motor neurons $M = \{m_1, m_2, \ldots, m_j\}$ is defined to be modular if every possible change to less than i of the sensors results in changes to less than j of the motors.

Conversely, in a non-modular neural controller, it is possible for a change to fewer than i sensors to influence the new values of all j motors. It is possible that a non-modular neural controller may internally extinguish certain sensor dynamics from reaching some motors, but we disregard this case in the present work. This results in a simplified, binary definition of modularity: either a neural controller is modular or not. Here, we investigate robots with both types of neural controller.

This approach to defining the modularity of robot neural controllers suggests a similar approach for defining the modularity of a robot's body plan:

1.3.2. Morphological Modularity

A robot is defined to be morphologically modular if a change in less than j of its motors results in a change in the state of the world registered by at least one and strictly less than i of the robot's sensors.

One common definition of morphology is any agent subsystem that mediates between its controller and its environment. More specifically, when an agent acts, it alters its relationship with its environment. If it is equipped with sensors, it can register this change. The above definition of morphological modularity captures the intuition that structural independence of the body, like structural independence of a neural network, implies functional independence: if a robot moves one part of its body that is independent of the rest of its body, local sensors will register the action, but more sensors on other morphological modules will not.

Armed with these two definitions, one can investigate four classes of robots:

1. those that are morphologically and neurally non-modular;
2. those that are morphologically modular but neurally non-modular;
3. those that are morphologically non-modular but neurally modular; or
4. those that are morphologically and neurally modular.

In this study, we evolve robots belonging to the first, second, and fourth class. One can deduce that robots which belong to the third class are functionally equivalent to those which belong to the first class: if a morphologically non-modular robot moves, its motion will affect all of its sensors. These sensors will then affect all motors, regardless of whether its neural controller is modular or not. Further, for this instance of the treebot, there is no design of a robot of the third class with a completely modular controller where both leaf sensors influence the motor neuron. If the controller was modular, only one or none of the leaf sensors would influence the motor neuron.

Although modular robots have been the focus of a number of studies (Yim et al., 2007; Fitch et al., 2014), here we compare morphologically modular and non-modular robots to investigate a specific and new question: if modular and non-modular robots are evolved in an increasing number of environments, are the robots with modular controllers able to detect familiar percepts combined in unfamiliar ways, and, with a modular morphology, respond appropriately?

This question brings to light a challenge for modular, embodied agents that modular, non-embodied systems do not experience. Even if an embodied agent has a modular neural controller with which it detects novel combinations of familiar percepts in a new environment, once it moves, its perceptions will change, and the environment may no longer "look" modular. We show here that movement in a new environment continues to appear modular from the robot's point of view only if it also has a modular morphology: it is free to move in response to independent percepts as it did previously, without disrupting the sensory signals arriving at other morphological modules.

The methods employed for investigating this issue are described in the next section. Section 3 reports our results, while Sections 4 and 5 provide some analysis and concluding remarks, respectively.

2. MATERIALS AND METHODS

This section describes the body plans of the simulated robots (Section 2.1), their various controllers (Section 2.2), the task environments they operated within (Section 2.3), the evolutionary algorithm used to optimize their controllers in those environments (Section 2.4), and the experimental design (Section 2.5).

2.1. The Robot Morphologies

Two robot morphologies were considered: one which is modular and one which is non-modular. **Figures 1A,B** represent robots with modular morphologies, while **Figure 1C** represents the non-modular one.

Robots were instantiated as trees composed of hierarchically branching segments. Both robot morphologies considered here were composed of one root branch and two leaf branches. Each branch had length 1, and the leaf branches were placed at 45° angles from the base. The robot contained three joints: one connecting the base branch to the environment itself (the base joint), and two that connect the base of each leaf branch to the tip of the root branch (the leaf joints). In the modular robot, the leaves were free to move independently of one another and the root was fixed, whereas in the non-modular robot, the leaves were fixed and the root was free to move.

In the non-modular robot, this was accomplished by instantiating the base joint as a rotational hinge joint and the two leaf joints as fixed joints. In the modular robot, the base joint was fixed and the two leaf joints were rendered as rotational hinge joints. The base hinge joint movement was restricted to rotations of [−120°, 120°] and the leaf hinge joints restricted movement to rotations of [−45°, +45°] around the vertical axis. These angles are relative to the initial angle of the joint, which is treated as 0°.

The robots were designed in this way such that a single parameter could dictate how modular the robot's body plan

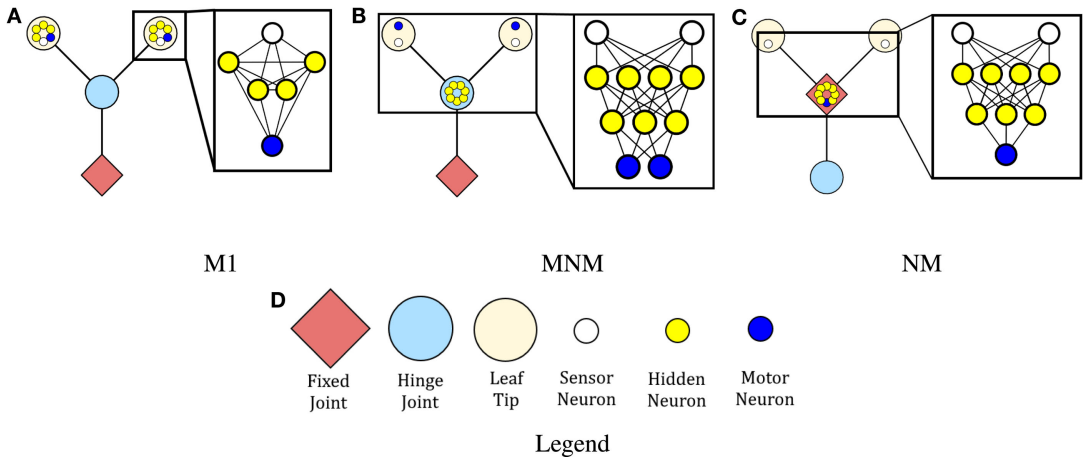


FIGURE 1 | The controllers and morphologies for each of the three robots. The red diamonds indicate that branches connected at that position are fixed relative to one another. Large blue circles indicate that the branches rooted at the circles are free to move independently of one another. Beige circles represent leaf tips. The small circles represent neurons: blue neurons represent motor neurons, white neurons represent sensor neurons, and yellow neurons represent hidden neurons. The modular robot is represented by **(A)** on the left, the mod-non-modular **(B)** is in the middle, and the non-modular robot is represented by **(C)** on the right. Blow ups of the network structure are included. All hidden neurons and motor neurons have recurrent self-connections, which are not depicted. Further, all connections except those with the sensor neurons are two-way. **(D)** is a legend showing the various types of joints and neurons.

was. If we define $[-45\alpha°, +45\alpha°]$ as the range of rotation of the leaf joints, $[-120(1-\alpha)°, +120(1-\alpha)°]$ as the range of rotation of the base joint, and restrict α to [0,1], then higher values of α create more modular robots, with $\alpha = 0$ and $\alpha = 1$ corresponding to the maximally non-modular and maximally modular robots investigated here. The robot with $\alpha = 0$ is considered morphologically non-modular according to the definition above, and any robot with $\alpha > 0$ is considered morphologically modular.

2.2. The Robot Controllers

Three robot controllers were considered in this work. The first makes the robot neurologically modular (**Figure 1A**), while the second and third make the robot neurologically non-modular (**Figures 1B,C**). All controllers contain two distance sensors (the small blue circles in **Figure 1**), one in each of the two branches of the robot's body plan. These sensors emit a beam that enables the robot to sense the distance from a branch to any objects in the environment. The value returned by this sensor is the length of the beam. The maximum length of the beam, if unobstructed, was set to 10 U, so the largest value the sensor neuron could have is 10.

Controller M (**Figure 1A**) consists of a sensor neuron, a motor neuron, and four hidden neurons in each leaf branch. The sensor feeds into all of the hidden neurons, which are completely interconnected with each other. All of the hidden neurons also have connections to the motor neuron, which also is connected back to all of the hidden neurons. Finally, all of the hidden neurons and the motor neuron are self connected, giving the M robot a total of 12 neurons and 50 synapses.

Controller MNM (**Figure 1B**) consists of two sensor neurons, seven hidden neurons, and two motor neurons. The hidden neurons are in a two-layer structure. The input from the sensors is passed into each of the four neurons in the first hidden layer. They, in turn, feed forward into the second hidden layer. The second layer has synapses connected back to the first one and also forward to the motor neurons. The motor neurons are also connected back to the second hidden layer. Finally, all of the hidden neurons and the motor neurons are self-connected. Therefore, MNM has 11 neurons and 53 synapses.

Controller NM (**Figure 1C**) consists of two sensor neurons, seven hidden neurons, and one motor neuron. The hidden neurons are organized in a two-layer structure. The sensor values input into the four neurons in the first layer, which then feed forward into the three neurons in the second layer. The second layer has synapses going back to the first layer and forward to the motor neuron. The motor neuron is also connected back to the second layer. Finally, all of the hidden neurons and the motor neuron are self-connected. Therefore, NM has 10 neurons and 46 synapses.

During evaluation, each sensor neuron received the raw distance value from its sensor. The hidden and motor neurons were updated using

$$n_i = tanh\left(\sum_{n_j \in I_{n_i}} w_{ji} n_j\right) \quad (1)$$

where I_{ni} is the set of incoming synapses to neuron n_i and w_{ji} is the weight of the synapse from neuron n_j to neuron n_i. The hyperbolic tangent function limits the hidden and motor neurons to floating point values in $[-1, +1]$.

Movement was controlled using proportional difference control. The values output by the motor neurons were scaled to the range $[-45, +45]$ and treated as desired angles. The rotational velocity of a branch at each time step was thus determined by the difference between the desired angle determined by the value of the motor neuron in that branch (or at the root) and the current angle of that branch (or root).

2.3. The Task Environments

The robots were evolved for a simple embodied categorization task: the robots were evolved to "point at" Type **A** spheres and "point away" from Type **B** spheres (**Figure 2**). Each environment that a robot was placed in contained a pair of spheres. Following Matarić and Cliff (1996), this corresponds to two free parameters ($f = 2$): the object on the left and the object on the right.

Three environment spaces were considered.

The first was the simplest consisting of a 2×2 environment space, giving four separate environments (**Figure 3A**). Each sphere could be Type **A** or Type **B** ($n_p = 2$). For this environment, the type *A* sphere had a radius of 3.5, and the type *B* sphere had a radius of 0.5.

The second environment space contained 3×3 environments, meaning nine total environments to consider (**Figure 3B**). A sphere could be one of two instances of Type **A** (either *A* or *a*) or Type **B**. For this environment, space *A* had a radius of 3.5, *a* had a radius of 0.5, and *B* was in the middle with a radius of 2.0. Thus, for this environment space, $n_p = 3$.

Finally, the last environment space considered contained $4 \times 4 = 16$ different environments (**Figure 3C**). A sphere could be one of two instances of Type **A** (*A* or *a*) or one of two instances of Type **B** (*B* or *b*). For this environment space, spheres of type *A*, *B*, *a*, and *b* had radii of 3.5, 2.5, 1.5, and 0.5, respectively. Therefore, $n_p = 4$ for this environment space.

Open Dynamics Engine was used to simulate the robots and the environment. A time step size of 0.05 was used.

2.4. Evolutionary *Optimization*

The robots were trained using Age-Fitness Pareto Optimization [AFPO; Schmidt and Lipson (2011)]. AFPO is a multi-objective optimization algorithm, which is designed to maintain diversity in an evolving population by periodically injecting new random individuals into the population and restricting the ability of older individuals to unfairly compete against younger individuals. In all of the experiments reported herein, a population size of 40 was employed.

Mutations in the population occurred in the form of choosing a new weight for a synapse from a normal distribution with mean of the current weight and a SD proportional to the absolute value of the current weight. This mutation operator enables evolution to rapidly incorporate high magnitude weights if required while also being able to fine tune weights with low magnitude. Mutation rates were set to be the reciprocal of the number of synapses, thus yielding an average of one synapse change per mutation.

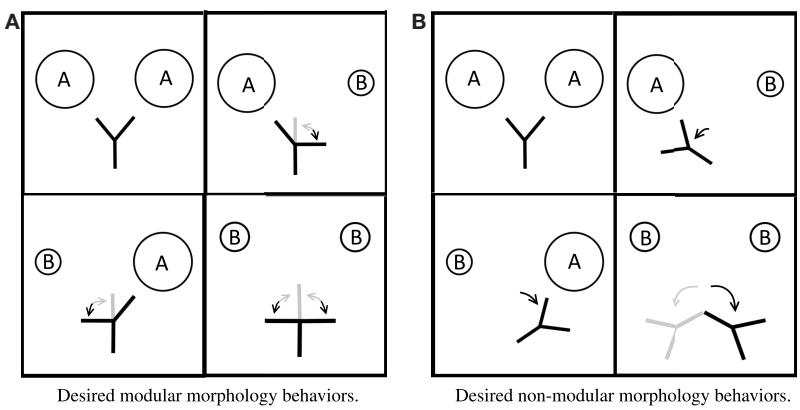


FIGURE 2 | Drawings of desired behavior for the modular morphology (A) and non-modular morphology (B) in each of the four environments in the 2×2. Arrows indicate desired movement away from the base position (the base position is shown in the top left panels). Gray segments and arrows indicate other acceptable behaviors.

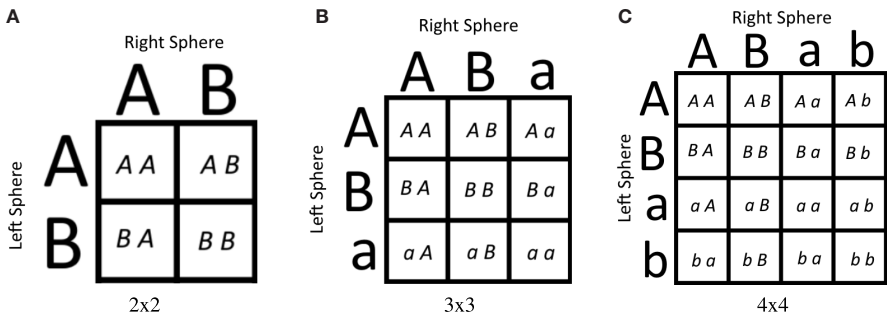


FIGURE 3 | The three environment spaces considered. *A*, *B*, *a*, and *b* represent spheres positioned on the left or right of the robot. Uppercase letters represent bigger spheres than their lowercase counterparts. Robots were evolved to "point" at A = {*A*, *a*} spheres and away from B = {*B*, *b*} spheres. **(A)** represents the 2×2 environment space, **(B)** represents the 3×3 environment space, and **(C)** represents the 4×4 environment space.

The optimization function used was an error function, which averaged the error of the robot when exposed to each environment in the environment list $E_{\{\}}$:

$$\mathrm{Err}(E_{\{\}}) = \frac{1}{||E_{\{\}}||} \sum_{o_\ell o_r \in E_{\{\}}} \frac{e(o_\ell o_r) - e_{\min}(o_\ell o_r)}{e_{\max}(o_\ell o_r) - e_{\min}(o_\ell o_r)} \quad (2)$$

$$e(o_\ell o_r) = \frac{g(o_\ell) + g(o_r)}{2} \quad (3)$$

$$g(\mathbf{A}) = \begin{cases} 1, \text{ if } d(\mathbf{A}) > d_{\max}(\mathbf{A}) \\ \frac{d(\mathbf{A}) - d_{\min}(\mathbf{A})}{d_{\max}(\mathbf{A}) - d_{\min}(\mathbf{A})}, \text{ otherwise,} \end{cases}$$
$$g(\mathbf{B}) = \begin{cases} 0, \text{ if } d(\mathbf{B}) > d_{\max}(\mathbf{B}) \\ \frac{d(\mathbf{B}) - d_{\min}(\mathbf{B})}{d_{\max}(\mathbf{B}) - d_{\min}(\mathbf{B})}, \text{ otherwise} \end{cases} \quad (4)$$

where

- $E_{\{\}}$ is a single environment, e.g., (AA, bA, Bb, etc.);
- $o_l \in \{\mathbf{A},\mathbf{B}\}$ indicates the type of the object on the left. Either ($o_l = \mathbf{A}$) or ($o_l = \mathbf{B}$);
- $o_r \in \{\mathbf{A},\mathbf{B}\}$ indicates the type of the object on the right. Either ($o_r = \mathbf{A}$) or ($o_r = \mathbf{B}$);
- $e(o_l o_r)$ indicates the robot's error incurred in environment $o_l o_r$ during the last time step;
- $e_{\min}(o_l o_r)$ and $e_{\max}(o_l o_r)$ indicate the minimum and maximum possible error the robot can incur in environment $o_l o_r$ during any one-time step, respectively. These were calculated based on the environment present and the geometry of the robot;
- $g(o_l)$ and $g(o_r)$ denote the errors incurred as a result of the left-hand and right-hand objects, respectively;
- $g(\mathbf{A})$ and $g(\mathbf{B})$ denote the errors incurred as a result of the objects of each type.
- $d(\mathbf{A})$ and $d(\mathbf{B})$ denote the distances from the midpoint of the closest leaf to the center of the object considered.
- $d_{\max}(\mathbf{A})$, $d_{\min}(\mathbf{A})$, $d_{\max}(\mathbf{A})$, and $d_{\min}(\mathbf{B})$ denote the maximum and minimum distance values for the **A** and **B** environments.

Because the motion range of the modular and non-modular robots is inherently different, these values are necessarily different. Further, the $d_{\max}(\mathbf{A})$ and $d_{\max}(\mathbf{B})$ values could be set artificially lower than the actual maximums in order to create weighting which more heavily considered $g(\mathbf{A})$ term over $g(\mathbf{B})$. $d_{\min}(\mathbf{A})$ and $d_{\min}(\mathbf{B})$ represent the actual observable minimums depending on the geometry of the robot. The values are presented in **Table 1**.

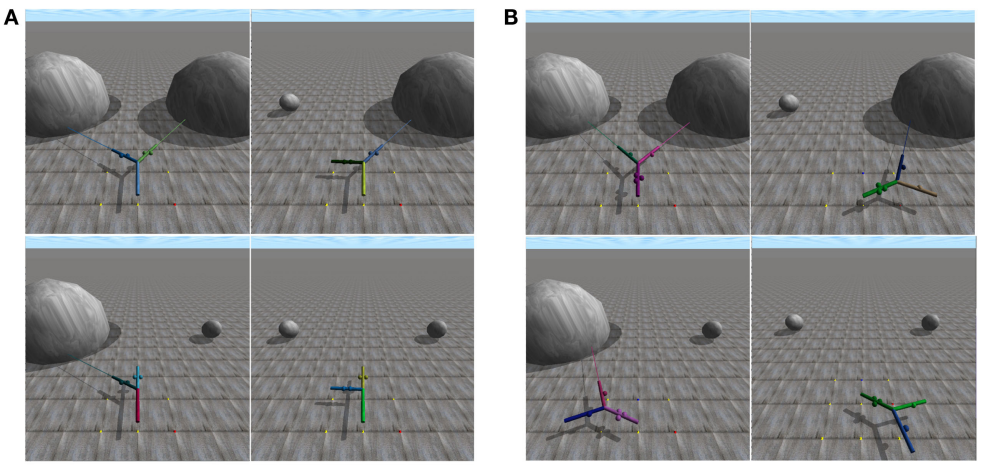
For the modular morphologies, $d_{\max}$ was set to the actual limit of motion of the branch. For the non-modular morphologies, $d_{\max}$ was set to less than the actual range of the motor to produce the desired behavior. By setting $d_{\max}$ less than the actual range, any robot that goes past a certain distance away from the **A** sphere would have an error of 1 for that object. Similarly, in the **B** sphere, if the robot moved far enough away to be past d_{max}, it was considered to have 0 error. This effectively created a weighting to the influences between the **A** and **B** spheres, which corresponded to the robot learning the desired behavior as seen in **Figures 2** and **4**.

2.5. Experimental Design

The first set of experiments consisted of evolutionary trials made up of fixed length epochs in the 2×2 environment space. The robot starts by training on one environment for the duration of the epoch. At the end of each epoch, a new environment is added for the robot to be trained on. By the last epoch, the robot is

TABLE 1 | Table of maximum and minimum distance values for each morphology type.

	Actual maximum distance	$d_{max}(B)$	$d_{max}(S)$	$d_{min}(B)$	$d_{min}(S)$
Modular morphology	5.315	5.315	5.315	5.157	5.157
Non-modular morphology	7.596	6.002	7.200	5.028	5.028



An evolved controller for the M robot. An evolved controller for the NM robot.

FIGURE 4 | The behaviors generated by two controllers that evolved to succeed in each of the four environments in the 2×2 environment space. Lines emanating from the leaf branches represent the distance sensors embedded in each. Video of the robots can be found at https://www.youtube.com/watch?v=t4gjv5nYeAA. **(A)** depicts an evolved robot with a modular morphology in each of the four environments in the 2×2 environment space. **(B)** depicts an evolved robot with a non-modular morphology in each of the four environments in the 2×2 environment space.

trained on all four environments. So, from the robots' perspective, when each new environment was introduced, the environment space changes by becoming more complex. The epoch length was set to 100 generations; thus, each evolutionary run lasted for 400 generations. If a robot survived from the last generation of one epoch into the first generation of the next epoch, its fitness was recomputed against this expanded set of environments.

In the second set of experiments, the robots were evolved in a predetermined subset of the environment space. Unlike the previous experiment, the robot is introduced to all of the environments in the subset at the same time instead of sequentially. After the best robot in the population achieved a prespecified error threshold in all of the environments in the chosen subset, it was tested in the remaining environments, not in the subset, without any further evolution to see how well it performed.

3. RESULTS

Experiment 1, described in Section 2.5, was run 50 times for all three robots in four environments in the 2×2 environment space, yielding a total of $50 \times 2 = 100$ independent evolutionary runs. The order of the environments was *AA*, *BB*, *AB*, and *BA*. **Figure 5** shows that at the start of each epoch, there is a spike in the error in the case of both the MNM and NM robots. In the case of the M robot, there is no spike in error when the third (*AB*) and fourth (*BA*) epochs are introduced.

Experiment 2, described in section 2.5, was also run 50 times for the 2×2, 3×3, and 4×4 environments on all of the robots. Thus, there were $50 \times 2 \times 3 = 300$ independent trials. For the first set of trials, only the "diagonal" of the environment space was considered. For the 2×2 environment space, this consisted of {*AA*, *BB*}. For the 3×3 environment space, the diagonal was {*AA*, *BB*, *aa*}. Finally, for the 4×4 environment space, the diagonal was {*AA*, *BB*, *aa*, *bb*}. The error threshold was set to 0.15. **Figure 6** shows the results for these trials.

The next test using this experimental setup considered another subset other than the diagonal, which had the same number of elements as the diagonal. Specifically, the "corner" of the environment space was considered **Figure 7**. All the three environment spaces were considered. For the 2×2 environment space, the corner was designated to be the top row of the environment space {*AA*, *AB*}. For the 3×3 environment space, the corner was set as {*AA*, *AB*, *BA*}. Finally, for the 4×4 environment space, the corner was {*AA*, *AB*, *BA*, *BB*}. Fifty trials of each robot in each environment space were performed, yielding $50 \times 2 \times 3 = 300$ independent trials. Again, the error threshold was set to 0.15.

The last test performed using this experimental setup looked at how well the MNM and NM robots respond to an unseen environment in the 2×2 case when evolved in three out of the four environments. The robots were evolved in three different subsets: {*AA*, *AB*, *BA*}, {*AA*, *AB*, *BB*}, and {*AB*, *BA*, *BB*}. Because of the inherent symmetry in the problem, {*AA*, *AB*, *BB*} is the same as {*AA*, *BA*, *BB*}; so, only one was chosen to be tested. Results are presented in **Table 2**.

4. DISCUSSION

When the modular robot is presented with a new environment, it is able to break down that environment into a combination of percepts. If the robot has seen those percepts before, even if the combination of those percepts is unfamiliar, it is able to act appropriately. Evidence for this is shown in **Figure 5**. There is no spike in error in the modular case at the start of the third and fourth epochs when the *AB* and *BA* environments are introduced. In contrast, the non-modular robots cannot see the environment in this manner, as is shown by the presence of error spikes at each new epoch.

Figure 6 shows that when the modular robot is evolved along the diagonal of the environment space, it is able to achieve acceptable error levels, that is at or below the predetermined cut off threshold (0.15), in the remaining environments in the environment space. This suggests that for this specific task, the number of environments needed to evolve a robot with a modular morphology and controller scales with the size of the diagonal

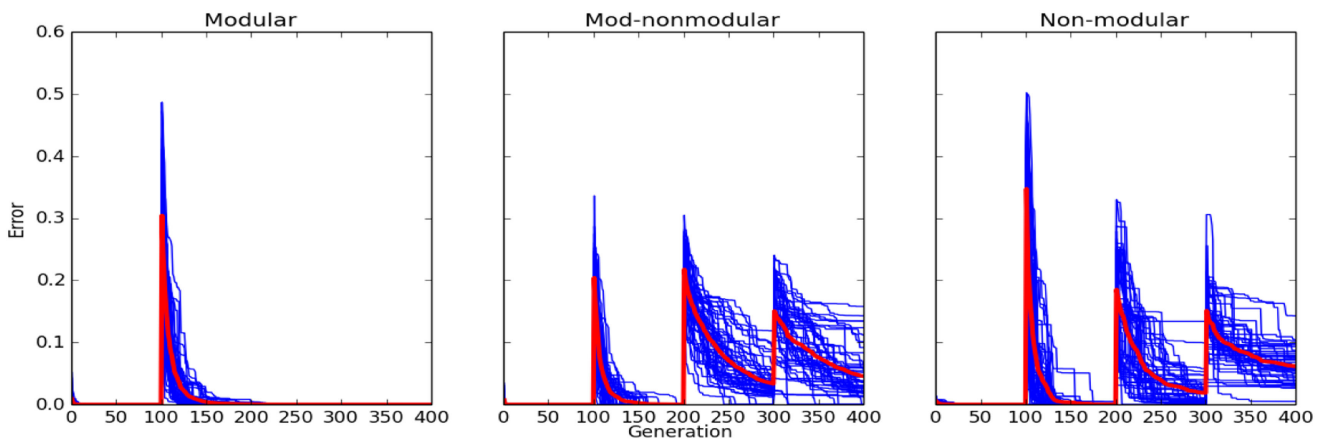


FIGURE 5 | Errors of controllers evolved for the M robot (left column), MNM robot (middle column), and the NM robot (right column) in fixed epoch training (Experiment 1 as described in Section 2). New environment regimes occurred every 100 generations. Robots were evolved along the diagonal of the environment space meaning the order presented to the robot was *AA*, *BB*, *AB*, and *BA*. Each blue curve corresponds to an individual evolutionary run: it reports, at each generation, the controller with the lowest error in the population at that time. The red curve reports the average of these runs.

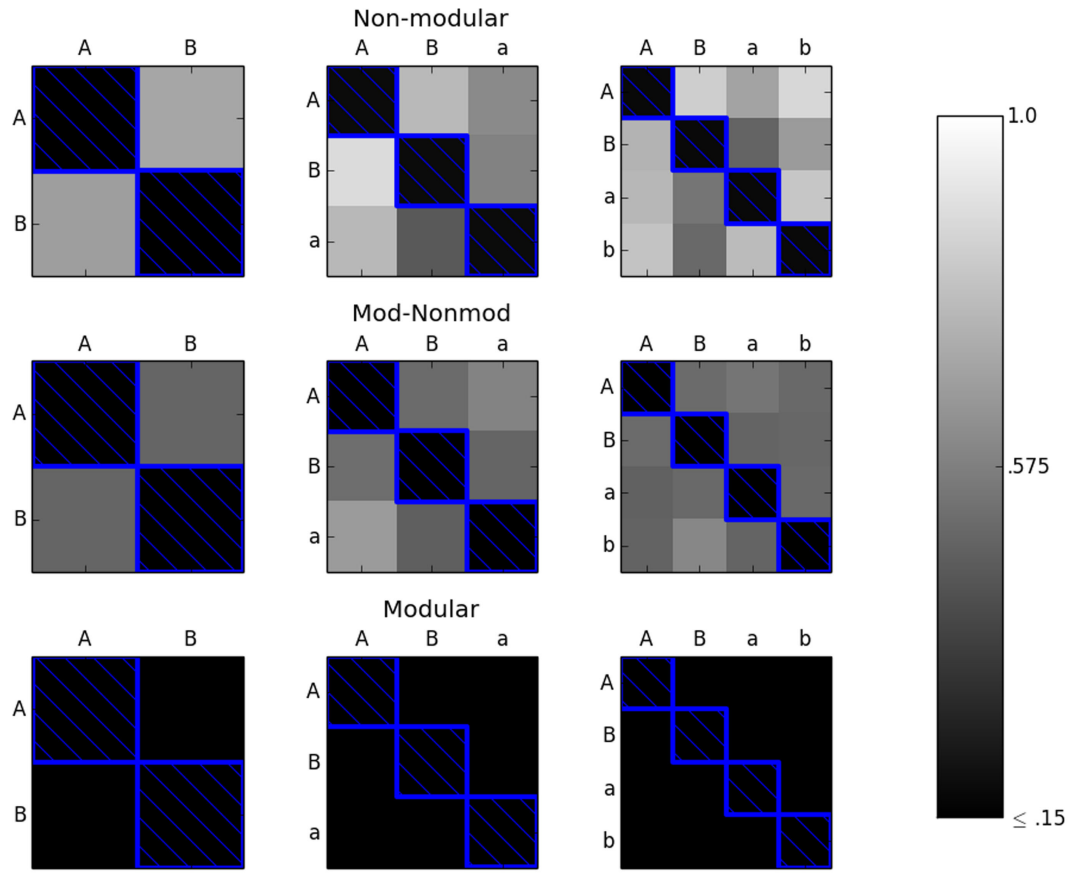


FIGURE 6 | The M (bottom row), MNM (middle row), and NM (top row) robots were evolved along the diagonal (environments with blue boxes around them) until they achieved an error of less than or equal to 0.15 in each environment in the subset considered. The robots were then tested in the remaining environments. The color of the box of each environment represents the average reported error in that environment. The lighter the color, the greater the average error with white representing an error of 1.0 and black representing an error of $\leq$0.15. In the modular case, every robot achieved an error of less than or equal to 0.15 on the off-diagonal environments. Over the 50 trials, both the non-modular and mod-non-modular robots averaged an error greater than 0.15 in all of the off-diagonal environments.

of the environment space. Therefore, the necessary number of environments for the modular robot seems to scale linearly with n_p, where n_p is equal here to the number of variations in the size of the spheres.

Conversely, the robots with the non-modular morphologies or controllers do not achieve acceptable, at or below 0.15, errors in the other environments in the space by simply evolving along the diagonal, as seen in **Figure 6**. This means that for this task, the number of environments the robot needs to be evolved in before achieving adequate fitness for the whole environment space is greater than the number of environments along the diagonal.

Table 2 shows that even when either of the non-modular robots is presented with three out of the four environments in the 2×2 environment space, they cannot use what it has seen in previous environments to help them in the unseen environment. Thus, at least for the 2×2 environment space case, the non-modular robots need to be evolved in each environment in the entire space in order to achieve adequate fitness.

Figure 7 indicates that just choosing any subset of environments to evolve in does not guarantee adequate fitness in the remaining unseen environments. Specifically, the results point to choosing a subset of environments in which each environment is completely independent from every other environment in the subset. In this context, completely independent environments are those which do not share the same row or column. For example, *AB* would be completely independent from *aa* since both the right ($A \neq a$) and left ($B \neq a$) spheres are different. As a converse example, *AB* and *Aa* are not completely independent since the left sphere is the same in both environments, namely, *A*. These results further suggest that a modular robot can recognize familiar precepts from previous environments and respond appropriately to them, even when they are presented in an unfamiliar combination. This is seen in the result from **Figure 7**, which shows that in the 3×3 environment space case, when the robot is tested in the *BB* environment, it reacts appropriately without requiring further evolution.

Figure 7 also shows the side result that evolution will generally find the simplest action to solve the problem at hand. In the 4×4 environment space case, both the modular and non-modular robots evolve to act on any sphere of size *B* or smaller (the *a* or *b* sizes) as an instance of the *B* sphere. Thus, the robots do well in the remaining environments comprised of *b* spheres and

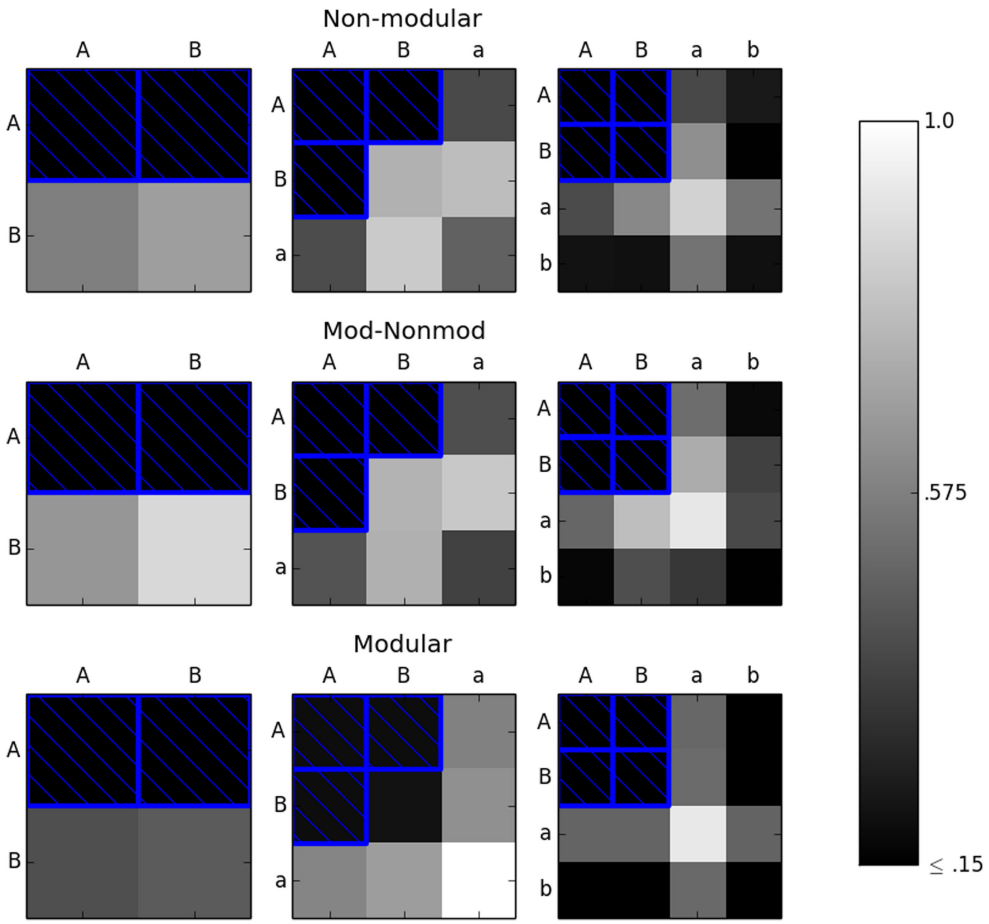


FIGURE 7 | The M (bottom row), MNM (middle row), and NM (top row) robots were evolved in the corners (environments with blue boxes around them) until they achieved an error of less than or equal to 0.15 in each environment in the subset considered. The robots were then tested in the remaining environments. The color of the box of each environment represents the average reported error in that environment. Here, we see that it is necessary to use completely independent subsets of environment to ensure linear scaling in the modular case.

TABLE 2 | Mean values of the error for the non-modular robot in the unseen environment after achieving an error of at most 0.15 in the three seen environments.

	{*AA*, *AB*, *BA*}	{*AA*, *AB*, *BB*}	{*AB*, *BA*, *BB*}
Non-modular	*BB*: 0.465 (±0.0445)	*BA*: 0.593 (±0.0385)	*AA*: 0.388 (±0.0232)
Mod-non-modular	*BB*: 0.665 (±0.00614)	*BA*: 0.580 (±0.0168)	*AA*: 0.586 (±0.0269)

Values in the parenthesis represent 1 SEM.

poorly in the environments containing *a* spheres since the action desired for *B* sizes is the same as *b* and different than the action desired for *a*.

5. CONCLUSION

This paper has shown that a modular morphology, combined with a modular neural control, can enable a robot to break down seemingly novel environments into combinations of familiar percepts. Moreover, if robots possess both this morphological and neural modularity, these robots are also likely to move in a similar manner in these environments, thus continuing to perceive the environment as a combination of familiar percepts. Assuming that the robot should always react the same way to each of these local percepts, it follows then that such a robot is likely to exhibit a successful behavior in this novel environment without requiring further training.

Robots with either non-modular morphologies or non-modular neural controllers cannot easily exhibit this phenomenon and, as a result, are likely to require additional training even in environments that contain individually familiar percepts. Given this, we have shown that for this task, robots with a modular morphology, combined with a modular neural controller, need to be evolved only in a linearly growing number of environments, whereas the number of environments non-modular robots require grows superlinearly. Our results indicate that it is likely that non-modular robots will require evolution in all of the possible environments in the space.

In future work, we would like to investigate specifically how the amount of evolutionary time necessary to evolve adequately fit robots scales for both the modular and non-modular robots.

We plan to accomplish by completely evolving both the modular and non-modular robots in the 2×2, 3×3, and 4×4 environment spaces. Further, we will look into scaling both f and n_p instead of just n_p, as was presented in this work.

If we consider our entire environment space to be a hypercube composed of n_p^f hypervoxels representing each individual environment, then there will be n_p voxels along the diagonal of the hypercube. If it is sufficient for a modular robot to simply evolve along this diagonal, then it is possible for time complexity, in this case the number of evolutionary time steps, necessary to evolve a given robot in an n_p^f-sized environment space to decrease from $O(n_p^f)$ to $O(n_p)$. However, this ideal case holds only if the robots are already morphologically and neurologically modular.

If robots begin with little or no morphological or neural modularity, it follows from Kashtan and Alon (2005) that if environments are added in a modularly varying way, more modular robots should evolve. This can be accomplished in this framework by ensuring that each newly added environment contains just one new feature of one of the free parameters describing the environments, while the other free parameters hold to a feature against which the robots have already been trained. This would require environments to be added to the training set along each of the edges of the environment hypercube in sequence, thus reducing $O(n_p^f)$ to $O(n_p f)$. Determining whether this theoretical result holds in practice, and under what conditions, is another worthy target of future investigation.

There are many other problems to investigate, including how these results here can be generalized to more complex and realistic robots and task environments; furthermore, under what conditions would the evolved modularity be maintained when the evolved robots are instantiated as physical robots.

Ultimately, this work thus suggests that there may exist a relationship between morphology, modularity, evolvability, and scalability, which may in future enable the automated optimization of increasingly complex robots that perform appropriately in increasingly complex environments.

AUTHOR CONTRIBUTIONS

CC – coding, data analysis, and experimental setup. JB – experimental design and inspiration; heavy review and editing. AB – heavy review and editing; contributions to simulations. KL – heavy review and editing; experimental design. NL – heavy review and editing; contributions to simulations.

ACKNOWLEDGMENTS

This work was supported by National Science Foundation awards INSPIRE-1344227 and PECASE-0953837. The authors would also like to thank John Long, Jodi Schwarz, and Marc Smith of Vassar College for innumerable discussions that contributed indirectly to this work.

CODE

Source code can be found at https://github.com/ccappelle/TreebotFrontiers

REFERENCES

Bernatskiy, A., and Bongard, J. C. (2015). "Exploiting the relationship between structural modularity and sparsity for faster network evolution," in *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference* (Madrid: ACM), 1173–1176.

Bongard, J. C. (2011). "Spontaneous evolution of structural modularity in robot neural network controllers," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation* (Dublin: ACM), 251–258.

Bongard, J. C., Bernatskiy, A., Livingston, K., Livingston, N., Long, J., and Smith, M. (2015). "Evolving robot morphology facilitates the evolution of neural modularity and evolvability," in *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference* (Madrid: ACM), 129–136.

Brooks, R. A. (1990). Elephants don't play chess. *Rob. Auton. Syst.* 6, 3–15. doi:10.1016/S0921-8890(05)80025-9

Clark, A. (1998). *Being There: Putting Brain, Body, and World Together Again*. Cambridge: MIT press.

Clune, J., Mouret, J.-B., and Lipson, H. (2013). The evolutionary origins of modularity. *Proc. R Soc. B Biol. Sci.* 280, 20122863. doi:10.1098/rspb.2012.2863

Ellefsen, K. O., Mouret, J.-B., and Clune, J. (2015). Neural modularity helps organisms evolve to learn new skills without forgetting old skills. *PLoS Comput. Biol.* 11:e1004128. doi:10.1371/journal.pcbi.1004128

Espinosa-Soto, C., and Wagner, A. (2010). Specialization can drive the evolution of modularity. *PLoS Comput. Biol.* 6:e1000719. doi:10.1371/journal.pcbi.1000719

Fitch, R., Stoy, K., Kernbach, S., Nagpal, R., and Shen, W.-M. (2014). Reconfigurable modular robotics. *Rob. Auton. Syst.* 7, 943–944. doi:10.1016/j.robot.2013.08.015

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends Cogn. Sci.* 3, 128–135. doi:10.1016/S1364-6613(99)01294-2

Kashtan, N., and Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proc. Natl. Acad. Sci. U.S.A.* 102, 13773–13778. doi:10.1073/pnas.0503610102

Kouvaris, K., Clune, J., Kounios, L., Brede, M., and Watson, R. A. (2015). How evolution learns to generalise: principles of under-fitting, over-fitting and induction in the evolution of developmental organisation. arXiv preprint arXiv:1508.06854.

Lipson, H., Pollack, J. B., and Suh, N. P. (2002). On the origin of modular variation. *Evolution* 56, 1549–1556. doi:10.1554/0014-3820(2002)056[1549:OTOOMV]2.0.CO;2

Matarić, M., and Cliff, D. (1996). Challenges in evolving controllers for physical robots. *Rob. Auton. Syst.* 19, 67–83. doi:10.1016/S0921-8890(96)00034-6

Newman, M. E. (2006). Modularity and community structure in networks. *Proc. Natl. Acad. Sci. U.S.A.* 103, 8577–8582. doi:10.1073/pnas.0601602103

Pfeifer, R., and Bongard, J. (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT press.

Pinville, T., Koos, S., Mouret, J.-B., and Doncieux, S. (2011). "How to promote generalisation in evolutionary robotics: the progab approach," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation* (Dublin: ACM), 259–266.

Schmidt, M., and Lipson, H. (2011). "Age-fitness pareto optimization," in *Genetic Programming Theory and Practice VIII*, eds R. Riolo, T. McConaghy, and E. Vladislavleva (New York: Springer), 129–146.

Wagner, G. P. (1996). Homologues, natural kinds and the evolution of modularity. *Am. Zool.* 36, 36–43. doi:10.1093/icb/36.1.36

Yim, M., Shen, W.-M., Salemi, B., Rus, D., Moll, M., Lipson, H., et al. (2007). Modular self-reconfigurable robot systems [grand challenges of robotics]. *Rob. Autom. Mag. IEEE* 14, 43–52. doi:10.1109/MRA.2007.339623

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 14 December 2016
doi: 10.3389/frobt.2016.00075

Modularity and Sparsity: Evolution of Neural Net Controllers in Physically Embodied Robots

*Nicholas Livingston[1], Anton Bernatskiy[2], Kenneth Livingston[1,3], Marc L. Smith[1,4], Jodi Schwarz[1,5], Joshua C. Bongard[2], David Wallach[1,4] and John H. Long Jr.[1,3,5]**

[1] Interdisciplinary Robotics Research Laboratory, Vassar College, Poughkeepsie, NY, USA, [2] Department of Computer Science, University of Vermont, Burlington, VT, USA, [3] Department of Cognitive Science, Vassar College, Poughkeepsie, NY, USA, [4] Department of Computer Science, Vassar College, Poughkeepsie, NY, USA, [5] Department of Biology, Vassar College, Poughkeepsie, NY, USA

OPEN ACCESS

Edited by:
John Rieffel,
Union College, USA

Reviewed by:
Sylvain Cussat-Blanc,
University of Toulouse, France
Takashi Ikegami,
University of Tokyo, Japan
Amine Boumaza,
Lorraine Research Laboratory in Computer Science and its Applications (CNRS), France

***Correspondence:**
John H. Long Jr.
jolong@vassar.edu

Specialty section:
This article was submitted to Evolutionary Robotics, a section of the journal Frontiers in Robotics and AI

Received: 20 April 2016
Accepted: 15 November 2016
Published: 14 December 2016

Citation:
Livingston N, Bernatskiy A, Livingston K, Smith ML, Schwarz J, Bongard JC, Wallach D and Long JH Jr. (2016) Modularity and Sparsity: Evolution of Neural Net Controllers in Physically Embodied Robots. Front. Robot. AI 3:75. doi: 10.3389/frobt.2016.00075

While modularity is thought to be central for the evolution of complexity and evolvability, it remains unclear how systems bootstrap themselves into modularity from random or fully integrated starting conditions. Clune et al. (2013) suggested that a positive correlation between sparsity and modularity is the prime cause of this transition. We sought to test the generality of this modularity–sparsity hypothesis by testing it for the first time in physically embodied robots. A population of 10 Tadros – autonomous, surface-swimming robots propelled by a flapping tail – was used. Individuals varied only in the structure of their neural net controller, a $2 \times 6 \times 2$ network with recurrence in the hidden layer. Each of the 60 possible connections was coded in the genome and could achieve one of three states: −1, 0, and 1. Inputs were two light-dependent resistors and outputs were two motor control variables to the flapping tail, one for the frequency of the flapping and the other for the turning offset. Each Tadro was tested separately in a circular tank lit by a single overhead light source. Fitness was the amount of light gathered by a vertically oriented sensor that was disconnected from the controller net. Reproduction was asexual, with the top performer cloned and then all individuals entered into a roulette wheel selection process, with genomes mutated to create the offspring. The starting population of networks was randomly generated. Over 10 generations, the population's mean fitness increased twofold. This evolution occurred in spite of an unintentional integer overflow problem in recurrent nodes in the hidden layer that caused outputs to oscillate. Our investigation of the oscillatory behavior showed that the mutual information of inputs and outputs was sufficient for the reactive behaviors observed. While we had predicted that both modularity and sparsity would follow the same trend as fitness, neither did so. Instead, selection gradients within each generation showed that selection directly targeted sparsity of the connections to the motor outputs. Modularity, while not directly targeted, was correlated with sparsity, and hence was an indirect target of selection, its evolution a "by-product" of its correlation with sparsity.

Keywords: modularity, sparsity, selection, evolution, robot

INTRODUCTION

The evolution of modularity is a central concern for biologists, neuroscientists, and roboticists alike, as modularity has been found to positively correlate with a number of desirable features of adaptive systems. These features include rapid response to environmental change (Lipson et al., 2002; Kashtan and Alon, 2005), specialization without forfeiting generally useful subfunctions (Espinosa-Soto and Wagner, 2010), avoidance of catastrophic forgetting in neural networks (Ellefsen et al., 2015), and evolvability (Wagner, 1996; Rorick and Wagner, 2011; Clune et al., 2013). To date, efforts to model and test hypotheses about the evolution of modularity have focused on using non-embodied systems to test ideas drawn from genetic regulatory networks and artificial neural networks (ANNs) (Voordeckers et al., 2015). However, some work has been dedicated to the specific challenges of evolving modularity in embodied systems (Bongard, 2011, 2015; Bernatskiy and Bongard, 2015; Bongard et al., 2015).

Contrary to the prediction that modularity evolves in response to selection on performance alone, it has been shown to evolve, instead, as a by-product of selection for enhanced performance and reduced connection costs (Clune et al., 2013). As the number of connections is reduced, the network's sparsity increases and drives the increase in modularity. Testing the importance of initial conditions, Bernatskiy and Bongard (2015) found that modularity evolved more rapidly under selection for enhanced performance and reduced connection costs when populations were seeded with sparse networks compared to those seeded with dense networks. These computer simulations support Clune et al. (2013) hypothesis of a causal linkage between the evolution of modularity and sparsity. We call this the "modularity–sparsity hypothesis." Because this hypothesis has only been tested in digital simulation, our aim is to test its generality by evolving neural net controllers in physically embodied robots. We predict that if an initial, randomly generated population contains some sparse networks, both modularity and sparsity will increase under selection for enhanced behavioral performance.

The networks used in this study were recurrent ANN controllers. Each ANN had two light-dependent resistors (LDRs) for inputs and two motor outputs to control the frequency and turning of a flapping propulsive tail of a swimming robot. A population of robots was created, each individual initially having a randomly generated ANN. Individuals were then subjected to artificial selection, testing their ability to detect, navigate toward, and collect energy at a light source, a behavior called phototaxis. Those individuals with better phototaxis relative to other individuals preferentially transmitted their genetic information, which represented the connections in their ANN, to the next generation.

Sparsity, S, of the ANN was measured simply as the difference between one and the ratio of actual to possible connections. Ranging from 0 to 1, higher values of S indicate fewer connections, with $S = 1$ meaning no connections. Since $S = 1$ is an impossible state for a controller that links inputs with outputs, we expect a high but non-unity level of S to provide the best controller performance.

Modularity of the controller was measured using the algorithm of Blondel et al. (2008), yielding a number, Q, from 0 to 1. When $Q = 0$, all possible connections among nodes are made and the network is fully integrated. The value of Q grows as connections are lost, provided that the remaining connections partition the network into groups that have nodes that are more densely connected to each other than they are to other nodes. Algorithmically, Q is determined as the difference between the fraction of connections that fall within the given groups and such fractions if the connections were distributed at random, maximized over all possible decompositions of the node set into groups. It is important to note that at a given intermediate value ($0 < Q < 1$), Q may correspond to many different networks.

In an ANN serving as a controller for a mobile robot, we expect a Q-mediated trade-off between the simplicity and complexity of sensorimotor control. If we imagine a high-Q controller in which two sensors are connected independently to two motors, then those two sensorimotor modules cannot combine information or calculations. Control of each module is simple, but completely separate modules cannot be coordinated by the controller. In general, larger values of Q indicate greater independence, and less interference, among the modules (Wagner and Altenberg, 1996), a feature that impacts both sensorimotor circuit function and the ability to evolve and differentiate multiple circuits within a single network.

At the other end of the spectrum, a low-Q ANN serving as a controller combines sensor inputs and shares calculations among motor output nodes. This allows for more complex patterns of operation. If the Q of the controller arises directly from the Q of the genetic regulatory network, then low Q also increases pleiotropic effects during evolution (Wagner and Altenberg, 1996). In animals, the value of Q that balances the trade-offs between simplicity and complexity of motor control likely depends on behavioral and ecological circumstances. Kim and Kaiser (2014) found Q values of 0.15 and 0.26 in the neural connectomes of the round worm, *Caenorhabditis elegans*, and the human, *Homo sapiens*, respectively.

Structural measures such as S and Q are not intended to measure the functional dynamics of the network. To examine the function of an ANN operating within a body that interacts with an environment, physically embodied robots are particularly useful, since the combined system often produces unanticipated behavior. For example, in this study we were surprised to find high variance in the behavior of our evolved robots. Upon investigation, we discovered that recurrence in the hidden layer of the ANN caused integer overflow in the calculations of those nodes. Thus the direct output to the two motor control nodes oscillated wildly, varying every few time steps across the whole range of the signed 16-bit integer. Yet the robots functioned and their behavior improved under selection for enhanced performance.

We test the Q–S hypothesis using a population of physically embodied, behaviorally autonomous, and surface-swimming robots called Tadros (tadpole robots). We selected this system for four reasons. First, this is the first time, to our knowledge,

that the *Q*–*S* hypothesis has been tested in physically embodied robots. Second, Tadros are capable of helical klinotaxis (HK), a type of gradient climbing that requires only a single LDR and a single motor control variable (Long et al., 2004). This sensorimotor simplicity increases the probability, relative to more complex systems, that working behaviors will be evolved by increasing *S* after controller networks have been generated randomly in the starting population. Third, since the HK circuit requires a single connection from LDR to motor, this allows for multiple parallel circuit modules and hence allows *Q* to evolve in a system with, for example, two LDRs and two motor control variables. Fourth, we have extensive experience testing and evolving Tadros.

Previously, we evolved Tadros to test hypotheses regarding the evolutionary dynamics of morphology. Under selection for enhanced phototaxis, the tail morphology of a population of Tadros evolved (Long et al., 2006). With the addition of a predator, a second eye, and a predator-detection system, constant selection pressure on Tadros for enhanced phototaxis and predator avoidance yielded variable evolutionary patterns, a combination of directional, random, and correlated ("by-product") effects on morphology of the flapping tail and the sensitivity of the predator-detection system (Doorly et al., 2009; Long, 2012; Roberts et al., 2014). In this study, we keep morphology constant and permit the connections of the controller to evolve under selection for enhanced phototaxis.

The framework for the controller of Tadro is a three-layer ANN with recurrence in the hidden layer (**Figure 1**). While the structure of each individual's ANN may vary, 60 connections between 10 nodes are possible. The connections may have weights of −1 (inhibitory), 0 (no connection), or 1 (excitatory). At each time step, a node sums the values from the nodes that feed into

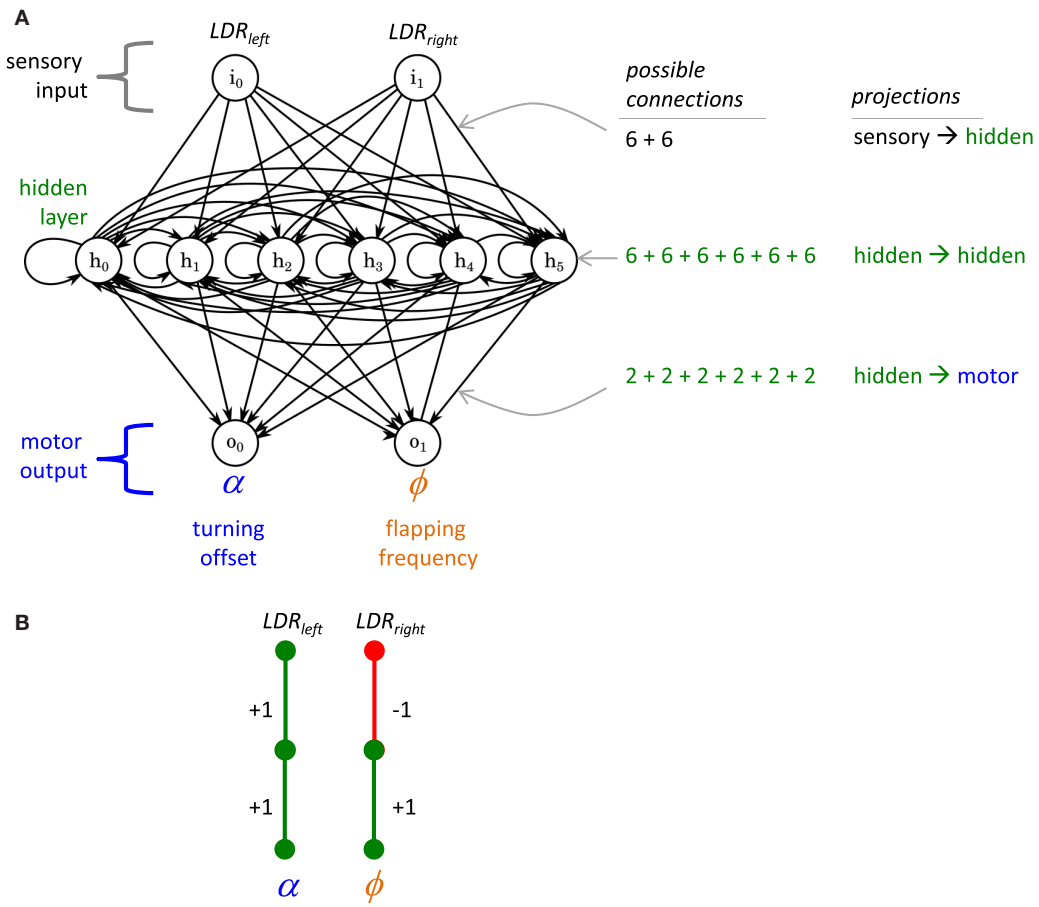


FIGURE 1 | **Framework for Tadro's artificial neural network (ANN)**. **(A)** The basic framework is a 2 × 6 × 2 recurrent network instantiated in software. Two inputs are the light-dependent resistors (LDRs), left and right, mounted on the perimeter of the hull. Each input may have up to six connections to the nodes of the hidden layer. Each of the six nodes of the hidden layer may have up to six recurrent connections, one with itself and five with other hidden layer nodes (represented by the horizontal line connecting the nodes). The nodes of the hidden layer may have up to two connections to the nodes of the motor output layer. The weights of the connections may be −1, 0, or 1, inhibitory, missing, or excitatory, respectively. At each time step, each node sums inputs and generates a single activation. **(B)** Engineered network. This is the simplest network that solves the task of phototaxis in this environment using both sensors and both motor outputs. It has a modularity, *Q*, value of 0.5, a sparsity, *S*, value of 0.93 for the whole ANN, and a fitness value, ω, of 52.6×10^6. Since this engineered solution represents a local optimum in the *Q*–*S* fitness landscape, we expect a starting population of randomly generated networks to evolve toward this structure.

it, where the value from an upstream node is the product of its connection weight and its activation. If the sum is outside of the limits of the integer type we use to represent the node state (16-bit signed), overflow occurs. Since it is a signed integer that is overflowing, the behavior is undefined. Note that the microcontroller we used is a deterministic device and in practice the resulting value is a function of the microcontroller's state preceding the overflow.

Each of the two input nodes can make up to six connections with each of the six nodes in the hidden layer for a total of 12 possible downstream connections. Each hidden layer node can have up to six recurrent connections, one with itself ("recurrent self-connection"), and up to five with the other five nodes in the hidden layer, for up to 36 possible recurrent connections. In addition, each hidden layer node can connect to each of the two nodes of the output layer, for a maximum of 12 downstream connections. The two output nodes provide signals to the servo motor that flaps Tadro's tail. One output node controls the tail's flapping frequency, ϕ, while the other controls the tail's turning offset angle, α.

According to the *Q*–*S* hypothesis, we expect that evolution by selection for enhanced phototaxis will create behaviors where the Tadro swims directly and quickly to the light and then slows down, thus maximizing energy harvested by remaining directly under the light for as long as possible. We expect that ANNs with the best phototaxic behavior would have evolved a highly modular controller with a navigational module that links the LDRs to α and a propulsion module that links the LDRs to ϕ (**Figure 1B**). *Q* and *S* would work in concert to keep these modules separate and thus avoid functional interference between the two. Thus the simple task of phototaxis, in conjunction with a morphological framework of two sensors and two motor outputs, is sufficient to permit the evolution of both *Q* and *S*.

MATERIALS AND METHODS

Artificial Neural Network

The basic framework of the ANN was a 2 × 6 × 2 recurrent network (**Figure 1**). The raw data from the two LDRs fed to the input nodes were constrained to the range 0–250 by truncating values above 250. This high value was achieved when the Tadro was stationed directly under the light in the experimental pool. At the perimeter of the pool, values approached 0 if the Tadro was oriented away from the light, toward the wall. Input values were passed to hidden layer nodes *via* connections that would multiply the input value by −1, 0, or 1. Thus, the value of a given hidden layer node was the sum of products of the input node(s) connected to it and the connection weight linking the input node to the hidden node.

Recurrence occurred within the hidden layer. Each hidden node was updated using a sum of values from other nodes in the previous time step multiplied by the respective connection weights, −1, 0, or 1, from each of the other hidden nodes to the node being calculated. Finally, this same summing of products was performed with every hidden node connecting to a given output node. The final sum of products for a given output node was then constrained to the minimum and maximum values calculated for that node in that ANN during the program's setup routine. This constrained value was then mapped onto the relevant ranges for the tail's flapping frequency, ϕ, and turning offset angle, α: 1.7–5.0 Hz and 10–170°, respectively.

Software

The base code for the Tadro was implemented in C (Arduino IDE version 1.6.0) on a TinyDuino microcontroller (ASM2001, Rev.8, http://Tiny-Circuits.com). For each different individual ANN, a header file contained the 60 connection weights. Within the base code, a setup routine initiated communication with the micro SD shield so that data could be logged during the experiment, and it scaled the range of raw output values by calculating the minimum and maximum values possible for each of the two output nodes, which varied for each ANN.

The main loop of the base code consisted of four parts: (1) reading sensor pins, (2) executing the ANN, (3) recording sensor, output, and timestamp data, and (4) executing the tail-beat. Three sensor pins are read: the two navigational LDRs and the LDR "light mouth" that is centrally located on the top of the Tadro and keeps track of how much light (which is our proxy for energy) the robot harvests during a trial. The navigational LDR values become the ANN input node values. The first of the output values is mapped to the range 10–170° for α, while the second is mapped to the range 5–15 ms to calculate ϕ (see next paragraph).

The tail flap function sends a PWM (pulse-width modulated) signal to the servo motor. The range of motion of the motor is limited to ±90°, with 0° as the midline. The servo receives two inputs: α and ϕ, both of which can be adjusted once each flapping cycle. During swimming, α turns the Tadro. When the tail flaps, its amplitude is ±10° relative to the α. Hence, α is limited to ±80° relative to the midline. The flapping function sends the servo to a position of α −10° and then steps in 1° increments to α +10°. Each half-tail flap thus has 20 steps. Each step has a duration of 1/20th of half of the period, T, where $T = 1/f$, where f is frequency (Hz). Without a connection from the ANN to the ϕ node, the tail flaps at its maximum f, 5 Hz.

After completing data collection our analysis of logged motor output revealed that the values of the nodes within the hidden layer were overflowing. Because of the many recursive connections, the node values quickly exceeded the maximum possible for a signed 16-bit integer. At that point the output of a node began oscillating wildly.

The Tadro controllers exhibit oscillations whenever any recurrent self-connections are present (**Figure 1**). The probability that any ANN will lack recurrent self-connections is $(1/3)^6 = 1.4 \times 10^{-3}$, where 1/3 is the probability of a connection weight of 0 and each of the six nodes in the hidden layer may connect to itself. Given that only 100 networks were considered in our experiment, it is unlikely that any non-oscillating networks have participated in this study. Given its likely presence, our concern was that this behavior would eliminate or substantially attenuate information passed between the sensory inputs and motor outputs.

To assess whether useful signal was reaching motors in spite of the oscillations, we investigated a pair of controllers, one randomly created and the other evolved. Individual T_0_9 was randomly created in generation 0 and had five recurrent self-connections. It had the best performance of any individual in that generation and was the most successful over the course of the evolutionary runs, leaving seven descendants in the final population. Individual T_9_9 was one of those descendants, a member of the final generation, possessing the highest fitness score of any individual over the entire evolutionary run. Like T_0_9, it also had five recurrent self-connections. We sought to understand whether the relatively high fitness of either T_0_9 or T_9_9 could reasonably be attributed to the reactivity of the controller.

We modeled the state of each node of the ANN as a random variable and measured the mutual information between pairs of these variables. We passed the values of the sensory inputs recorded during the embodied experiment through a simulator,

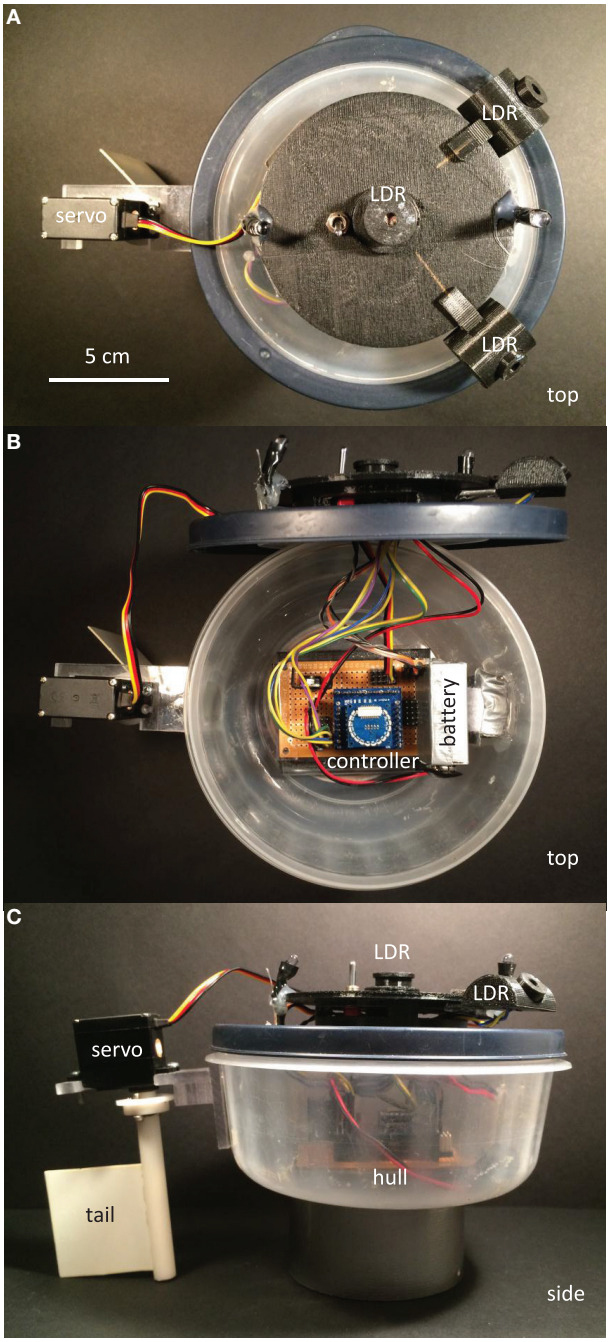


FIGURE 2 | Tadro, mechanical design. **(A)** The two light-dependent resistors (LDRs) on the perimeter, angled at 45° to the horizontal, serve as inputs to the artificial neural network (ANN). The central LDR is the "light mouth," independent of the net that logs the exposure of the Tadro to light. The servo motor drives the flapping tail. **(B)** The controller runs the ANN, logs sensory inputs and motor outputs, and logs light intensity from the light mouth. Power is supplied by a 9 V rechargeable lithium battery. **(C)** The tail is thin, rigid plastic positioned below the bottom of the hull.

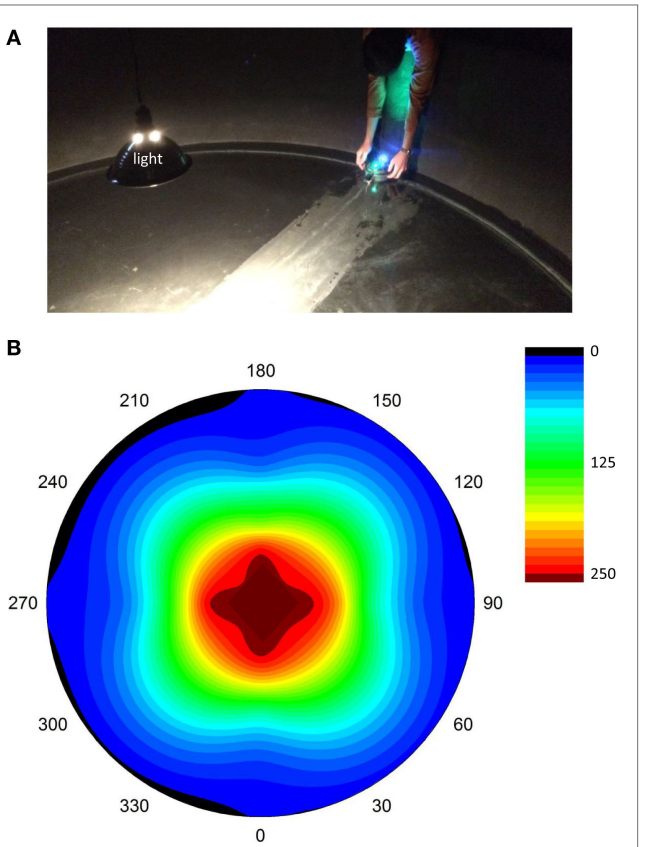


FIGURE 3 | Selection experiments. **(A)** Launch of Tadro in the "head out" starting position. A single light source is present, and reflections from the walls are reduced by the black matte finish of the tank, which is 3.05 m in diameter. Each trial lasted 5 min; each of the 10 individual ANNs were tested twice (starting position head-in and head-out) in random order within that generation. **(B)** Light gradients in the tank, as measured by the LDR "light mouth" on the top of the Tadro. Measurements were made by manually moving the Tadro along transects running on radii from 0° to 180° and 90° to 270°. Intermediate values were interpolated between measuring points on the radii and between radii, a process that distorted the isoclines from a circular shape expected with exhaustive spatial coverage. Cool colors are low light intensity; hot colors are high light intensity. The units are arbitrary, scaled to account for the input range of the LDRs.

the ANNalyzer that runs the ANN implementation used in the embodied experiments. This resulted in 1067 states of the network. For each node of the network, the full range of possible values was divided into bins and each state of the node was labeled with the number of the bin to which the state belonged. Based on the labels, we computed an estimate of the normalized mutual information using a contingency matrix as a proxy for the bivariate joint probability distribution (implementation *via* SciKit by Pedregosa et al., 2011). Since in the case of both T_0_9 and T_9_9 the output o_1 is disconnected from the rest of the network, we knew *a priori* that the mutual information between it and any other node in the network should be 0. Thus we could use the behavior of output o_1 as our baseline reference.

Hardware

Tadro (**Figure 2**) was constructed with a hull of a round plastic food storage container, 800 mL volume, 14.2 cm diameter on top, tapering to 12.9 cm diameter at bottom with a depth of 6.3 cm.

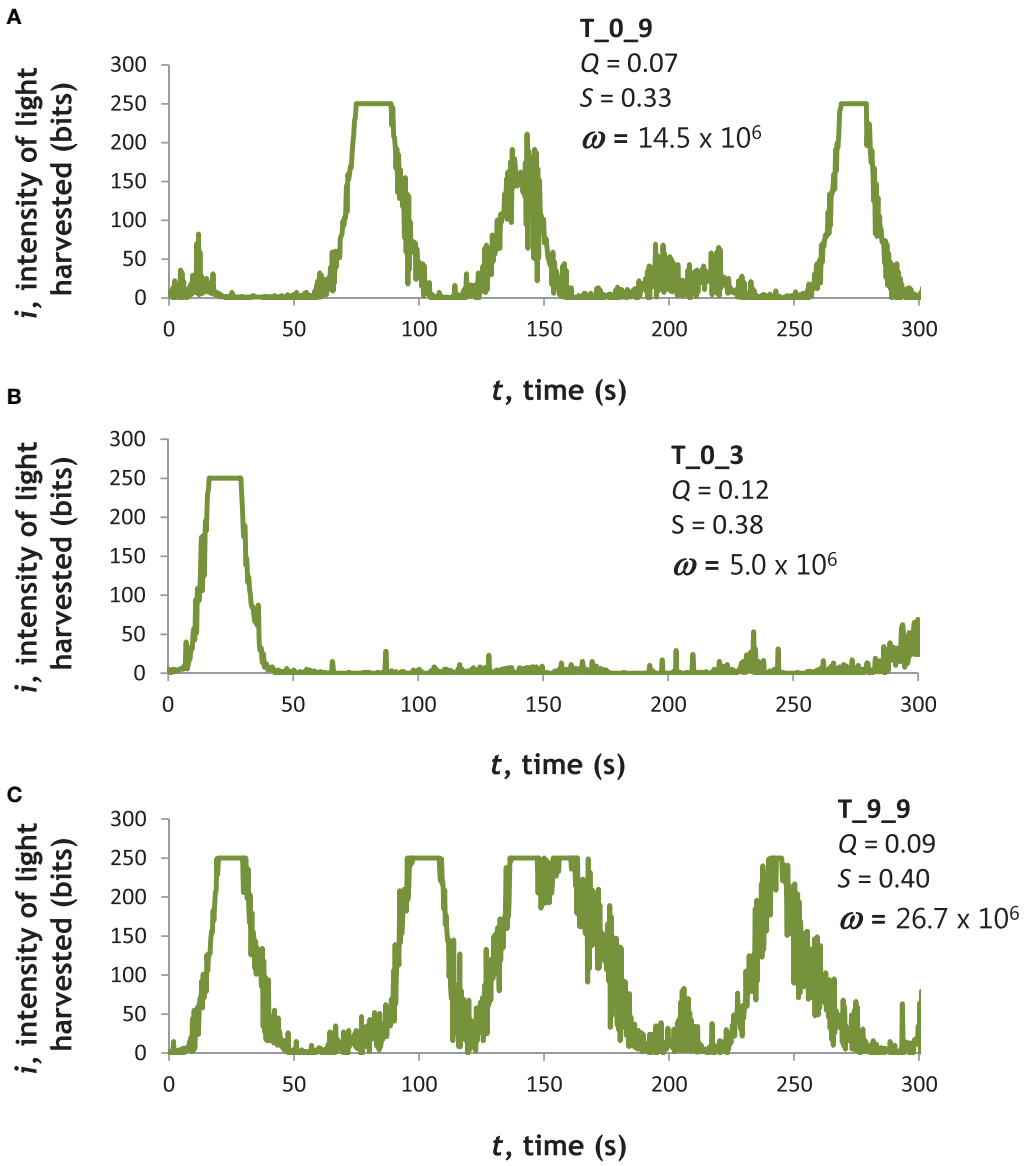


FIGURE 4 | **Light-harvesting performance varied among Tadros**. Examples of three different ANNs, two **(A,B)** from the first generation of randomly generated ANNs (T_0_9 and T_0_3) and one **(C)** from the final generation (T_9_9). Naming convention: T, Tadro; first digit, generation; and second digit, rank order of individual based on fitness on ascending scale. These examples were chosen to highlight variance in the first generation and then the best individual in the last generation. Moreover, T_0_9 had the highest fitness in generation 0 and produced more descendants, including T_9_9, than any other genotype. T_0_3 had an intermediate fitness in generation 0 but produced no offspring and was selected as an example of an evolutionary dead end.

The servo was mounted such that its center was 4.4 cm from the edge of the hull. The Delrin™ drive shaft attached to the servo was 8.0 cm long, holding under water a 5.0-cm square rigid plastic tail of 0.1 cm thickness. Mass of the Tadro was 314 g, including its 9 V battery.

The three LDRs were cadmium sulfide (model 161, http://Adafruit.com) with a resistance of approximately 200 kΩ in the dark and 10 kΩ in bright light. The servo motor was a model HS-225BB, from Hi-Tec.

Selection and Reproduction

Selection trials were run in a 3.05 m diameter indoor tank with a single light source overhead at its center (**Figure 3**). A gradient of light intensity was centered in the tank, with values, measured by the sensor on top of Tadro, at or near 0 near the perimeter and maximal values in the center (**Figure 3B**). In each generation, there were 10 genotypically unique individuals with correspondingly unique neural networks. Robots were tested one at a time in random order. Two trials were run for each individual, one starting with the LDRs toward the light and one starting with LDRs away from the light. The starting position was at the perimeter of the tank. Each trial was 5 min long.

The proxy for the amount of energy logged in each trial was the sum of the products of the interval's light intensity, recorded by the central LDR light mouth, and the duration of each time step. Because the light intensity values were uncalibrated, the units of energy were arbitrary but constant across individuals, trials, and generations. The energy values from both trials for an individual were averaged to calculate the individual's fitness, ω. We calculated two variables related to changes in ω. The selection differential, d_{parents}, was the difference in the mean ω of the parents selected to reproduce and the mean ω of the entire parental population, including the parents. The evolutionary response, R, was the difference between the mean ω of the offspring and the mean ω of the parents.

The first generation of ANNs was created by randomly assigning values to the 60 genes in each individual. Each gene can have one of three states: −1, 0, and 1. With an equal probability of being in each state, the initial population started with an average of 40 connections per individual, where a connection is said to be present if the gene has a value of ±1.

To create the ANNs for the second and all subsequent populations, we ranked the individuals by ω. The ANN with the highest ω was cloned. The next nine offspring were produced by mutation of a parental ANN, with each parent chosen with a roulette wheel method, with the probability of reproduction proportional to their relative fitness in that generation. For each of the 60 ANN connections in each individual, the probability of mutation of any connection was 0.03 with equal probability of switching from one state to another, −1, 0, and 1.

Modularity, Sparsity, and Selection Gradients

The modularity of each ANN was quantified using the Q metric introduced by Clauset et al. (2004) and expanded by Blondel et al. (2008). The Q can be described as the difference between the fraction of connections that fall within given groups and the fraction if the same number of connections were distributed at random while preserving the nodes' degree distribution. Quantitatively, Q is as follows:

$$Q(\vec{c}) = \frac{1}{2m}\sum_{i,j}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j).$$

Here, c_i denotes the label of the module to which node i is assigned, giving $\vec{c}$ the meaning of a complete assignment of the nodes into modules; A_{ij} is one if nodes i and j are connected and 0 otherwise; $m = \frac{1}{2}\sum_{i,j} A_{ij}$ is the total number of edges; $k_i = \sum_j A_{ij}$ is the number of edges attached to the vertex i; $\delta(c_i, c_j)$ is one if nodes i and j are assigned into the same module and 0 otherwise.

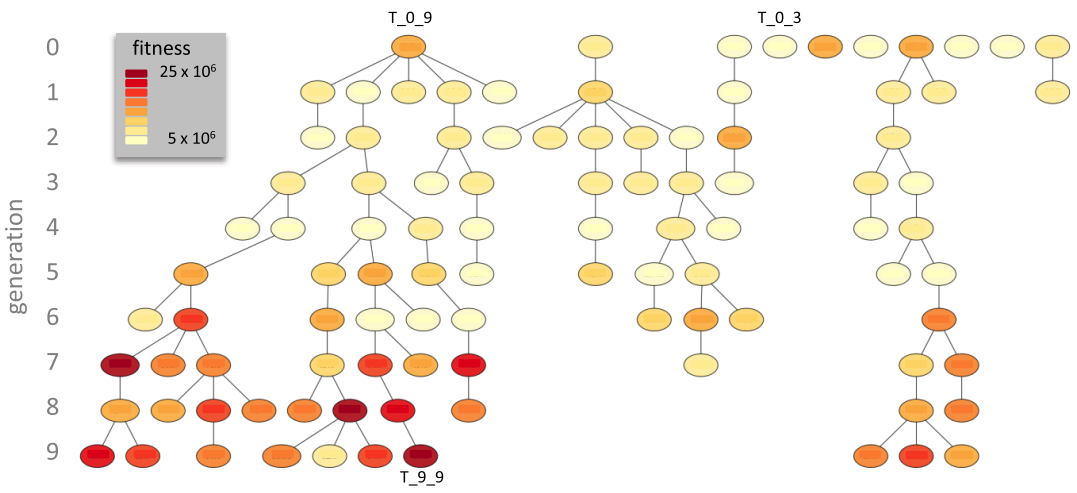


FIGURE 5 | Descent of the Tadros. In this genealogy fitness, ω, is color coded and the three exemplar individuals (see **Figure 4**) are indicated. Over 10 generations of evolution, T_0_9 left the most descendants, with seven in generation 9.

The metric $Q(\vec{c})$ depends on the assignment of nodes into modules; to obtain an assignment-independent metric we look for an assignment which maximizes the metric:

$$Q = \max_{\vec{c}} Q(\vec{c}).$$

This optimization problem is computationally hard. We use an approximate optimization method by Blondel et al. (2008) to estimate Q.

Sparsity, S, of the network is measured as follows:

$$S = 1 - \frac{C}{n},$$

where C is the total number of connections with weights of ± 1 and n is the total number of possible connections. For the whole ANN, $n = 60$ and S is indicated as S_w. The possible connections within the hidden layer, connections from the hidden layer to the α output, and connection from the hidden layer to the ϕ output, were 36, 6, and 6, respectively. The S for each is indicated as S_h, S_α, and S_ϕ.

To measure how selection is targeting traits, one may calculate selection gradients, β, the standardized coefficients from a multivariate regression of ω onto the traits:

$$\omega_j = a + \beta_Q Q + \beta_w S_w + \beta_h S_h + \beta_\alpha S_\alpha + \beta_\phi S_\phi,$$

where j indexes the generation and a is a regression constant. A larger β relative to other β values indicates that that trait is a target of selection, correlated strongly with ω. We also tested the hypothesis that ω, Q, and the various types of S change over generational time using multiple one-way analyses of variance (ANOVAs), with generation as the ordinal factor and *a priori* effects tests to conduct pairwise generational comparisons.

Statistical Design and Analysis

This experiment was designed to test the Q–S hypothesis by testing several predictions that stem logically from it. In accordance with Fisherian statistical methods, we adopt the modus tollens logic of negation: falsifying the prediction falsifies the hypothesis. Failure to refute the predictions thus constitutes tentative support for the hypothesis.

The Q–S hypothesis (Clune et al., 2013) proposes a causal linkage between the evolution of modularity and sparsity; specifically, the evolution of S facilitates the evolution of Q, and not *vice versa*. Accordingly, we predict the following: S rather than Q of the ANN will be the target of selection acting on the phototactic behavior of the Tadros. In contrast to previous studies on Q and S, note that connection costs in the ANN are not part of the fitness function: ω is solely the integral of light collected by the Tadro through its "light mouth" over time (see Selection and Reproduction).

The population was evolved under selection for 10 generations, producing a total sample size of 100 individuals (10 individuals in each of 10 generations). Within the population, each individual was statistically independent; hence, ANOVA was appropriate. To test the prediction that the population would undergo adaptive evolution from its starting condition of randomly generated ANNs, we conducted a one-way ANOVA with ω as the dependent variable and generation as the independent. To test for statistical difference between generations, *a priori* contrasts were conducted. The identical statistical model was also used to examine the evolution of the ANN, specifically the measures of Q and S defined in the Section "Modularity, Sparsity, and Selection Gradients." All statistical analyses were conducted using JMP software (v. 12, SAS Institute Inc., Cary, NC, USA, 1989–2016). The significance level for all tests was 0.05.

RESULTS

Tadros with different ANNs showed differences in light-harvesting behavior (**Figure 4**). T_0_9 (**Figure 4A**) had the highest fitness, ω, in generation 0. T_0_3 (**Figure 4B**) had an intermediate value of ω in generation 0. T_9_9 (**Figure 4C**), from generation 9, had the maximum ω of any individual at any time.

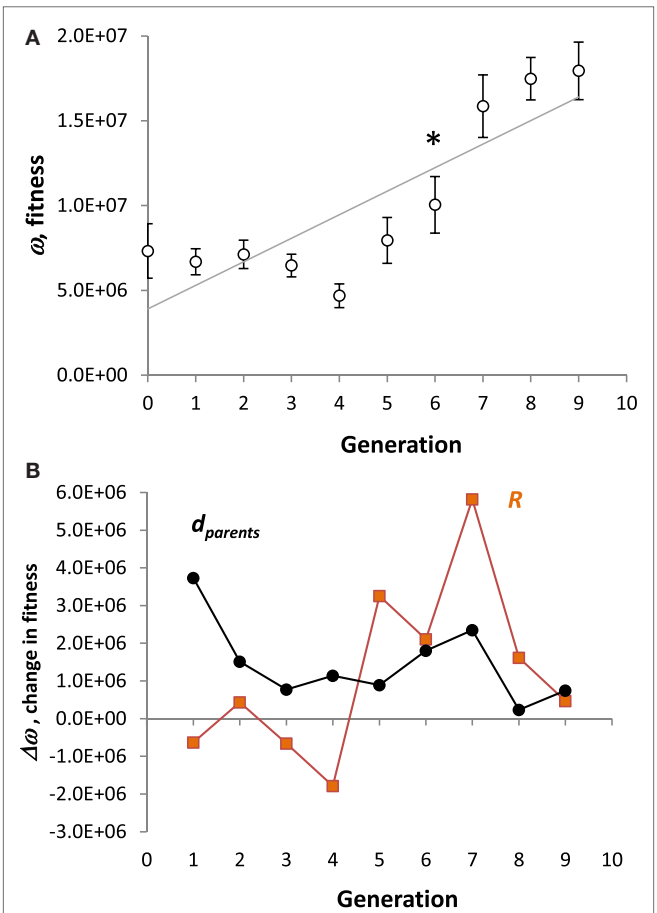


FIGURE 6 | Evolution of Tadros. (A) Fitness. All values are means ($N = 10$) of the population ± SE. A strong directional trend ($p < 0.001$, ANOVA) is present, represented by the regression line. In spite of the overall linear trend, the pattern is more complicated: *a priori* contrasts between generations detect a significant saltation between generations 6 and 7 ($p < 0.05$, denoted by *). **(B)** Change in fitness. The selection differential, $d_{parents}$, is the difference in the mean fitness between parents selected to reproduce and the mean fitness of the parental population. The evolutionary response, R, is the difference between the mean fitness of the offspring and the mean fitness of the parents.

To understand the variability of an individual's behavior, we compared the variation in ω of an individual to that of the population as a whole. We selected T_9_9 because its light-harvesting behavior was highly variable: it spent most of its time away from the wall of the tank, moving in, out, and around the light source, in a steep portion of the light gradient, as evidenced by the time-course data from the LDR "light mouth" (**Figure 4C**). By comparison, the behavior of individuals like T_0_3 resulted in lower fitness as a result of moving along the wall of the tank, a region with a very low level of light (**Figure 4B**). After the evolutionary trials, we performed 20 trials on T_9_9 (10 starting head toward the light and 10 starting head away from the light). We calculated the coefficient of variation (CV), the ratio of SD to mean, in ω, and compared that to the CV for all 20 trials (2 for each of 10 individuals) from generation 0 and generation 9. The CV values for ω were 0.738, 0.500, and 0.520, respectively.

To test whether the high variability of T_9_9 was caused by superior light-harvesting *per se* or the oscillatory behavior of the hidden layer of the ANN, we engineered by hand a different ANN. This engineered ANN connected the left LDR with a weight of −1 to a single hidden node; that hidden node was connected with a weight of +1 to the α output node. The right

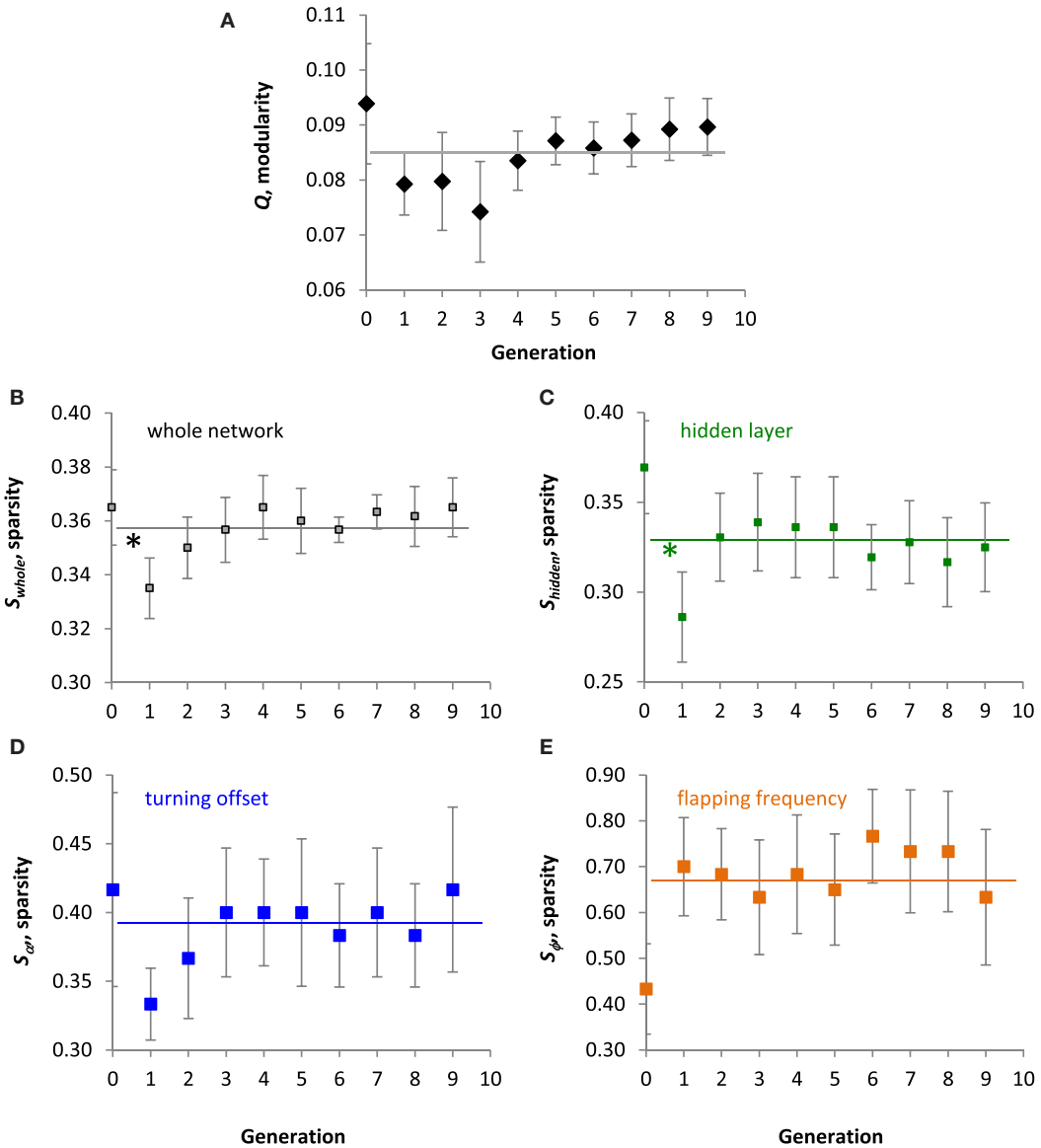


FIGURE 7 | Evolution of the ANNs. All values are means ($N = 10$) of the population ± SE. **(A)** Modularity, Q. No significant linear trend was detected over 10 generations. The grand mean of 0.085 is indicated by the horizontal line. **(B–E)** Sparsity of the whole ANN, the hidden layer, projections of the hidden layer to the turning offset, and projections of the hidden layer to flapping frequency output, respectively. No significant linear trends were detected over 10 generations. Grand means of 0.358, 0.329, 0.390, and 0.665, respectively, are indicated by the horizontal lines. For both the whole network and the hidden layer, *a priori* contrasts showed a significant decrease in the mean values between generations 0 and 1 (*$p < 0.05$).

LDR with a weight of +1 was connected to a different hidden node; that hidden node was connected with a weight of +1 to the ϕ output node. Neither hidden node had connections to itself or to the other hidden node. Thus this ANN lacked recurrence and was similar to the ideal agent outlined in the Section "Introduction." We tested the engineered ANN 20 times, 10 trials starting head toward the light and 10 starting head away from the light. It achieved a mean ω of 52.7×10^6, nearly twice that of the best trial of T_9_9 (**Figure 4**). Its CV for ω was the lowest of all groups tested, 0.174. This was indirect evidence that recurrence in the hidden layer was creating oscillating signals that created the variability of the behavior of T_9_9.

T_0_9 left seven descendants in the final generation while T_0_3 did not reproduce (**Figure 5**). T_0_8 left the other three final descendants. T_9_9, a descendant of T_0_9, achieved the maximum ω of any individual in any generation. By generation 7, most individuals had evolved relatively high levels of ω.

On average, the population of Tadros evolved greater ω over 10 generations (**Figure 6A**). A strong positive directional trend ($p < 0.001$, ANOVA) was present. In addition to the overall trend, *a priori* contrasts between adjoining generations detected a significant saltation event between generations 6 and 7 ($p < 0.05$). The selection differential, d_{parents}, was always positive, but the evolutionary response, R, was not (**Figure 6B**). The largest values of R occurred in the transitions from generations 4 to 5 and 6 to 7.

In the population of Tadros, structure of the ANN, as measured by modularity, Q, and the different types of sparsity, S, did not evolve directionally overall (**Figure 7**). No significant linear trends were detected by ANOVA for Q (**Figure 7A**), sparsity of the whole network, S_{whole} (**Figure 7B**), sparsity of the hidden layer, S_{hidden} (**Figure 7C**), sparsity of the projections to the turning offset, S_α (**Figure 7D**), or sparsity of the projections to the flapping frequency of the motor output layer, S_ϕ (**Figure 7E**). The mean S_ϕ of 0.665 was highest of the S means, which were 0.358, 0.329, and 0.390, respectively, for the others. For both the whole network and the hidden layer, *a priori* contrasts showed a significant saltational decrease in the mean values of S between generations 0 and 1 ($p < 0.05$). Over all 10 generations, the range of S_{whole} and Q was 0.300–0.433 and 0.016–0.133, respectively. Over 10 generations S_{whole} and Q were positively and significantly correlated ($r = 0.407$, $p < 0.001$).

To examine the detailed correlational structure between Q and the various measures of S, we used stepwise linear regression (mixed direction, $p = 0.25$ to enter or leave, JMP, v. 12). Over all 100 individuals and 10 generations, S_α and S_ϕ predicted Q in the linear regression ($p < 0.0001$, $r^2 = 0.293$), yielding statistically significant coefficients of 0.051 ($p = 0.0029$) and −0.014 ($p = 0.0314$), respectively.

In spite of the lack of overall trends in the evolution of the structure of the ANNs, selection gradients, β, detected the effect of selection over smaller time scales. The strongest directional selection pressure acted on S_α and S_ϕ in generations 7 and 8 (**Figure 8**); selection switched from strongly positive to strongly negative. The switch in the sign of the selection pressure indicates stabilizing selection that can be seen most clearly for S_ϕ in the jump and plateau in magnitude over generations 6, 7, and 8 (**Figure 7E**). The positive selection pressure corresponded to the significant evolutionary jump in fitness from generation 6 to 7 (see **Figure 6A**). In no generation is Q under selection, negative or positive, that is of greater magnitude than a measure of S.

The structure of the ANNs may be visualized and compared using a connectome diagram: T_0_9 and T_9_9 have an identical pattern of inputs from the sensory to the hidden layer (**Figure 9**); they have a nearly identical pattern of inputs to the motor layer, with only a difference in sign of one node and a complete lack of any nodes controlling the ϕ node. Contrast this pattern with that of T_0_3, which has a motor output layer dominated by connections to ϕ and has only one connection to the α node. T_0_9 and T_0_3 have a connectome similarity of 0.20, where similarity is the ratio of shared connections and weights to the total possible. T_0_9 and T_9_9, ancestor and descendent, have a connectome similarity of 0.90; T_0_3 and T_9_9 have a connectome similarity of 0.20.

The network structures of these individuals show clearly the differences in projections from the hidden layer to the motor output layer (**Figure 10**). Given the number of recurrent self-connections in the hidden layer, these networks have oscillatory behavior. Despite the oscillations, evolution by selection was able to improve the fitness of T_9_9, which has five recurrent self-connections, over that of its ancestor and the population as a whole.

To further probe the impact of the oscillatory behavior of the ANN, we measured the mutual information between nodes of T_9_9. First, we examined the disconnected output node o_1, that of the flapping frequency, ϕ. When paired with the input nodes, the estimates for o_1 were negative and on the order of 10^{-6} (**Table 1**). Estimates of mutual information between the inputs

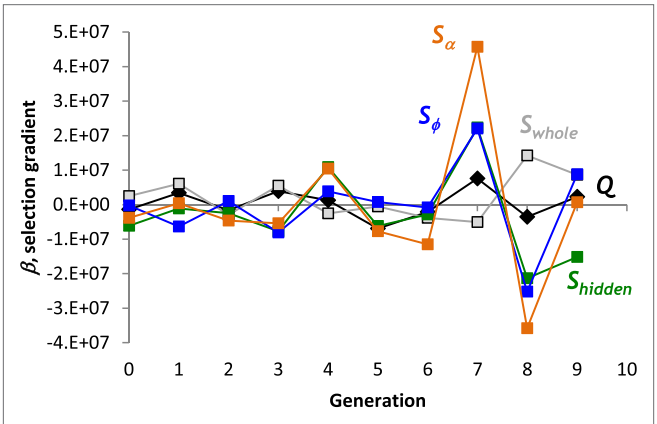


FIGURE 8 | Selection gradients, β, for structural properties of the ANN. The largest positive and negative selection gradients occur in generation 7 and 8, respectively, for sparsity, S, of the projections from the hidden layer to offset and frequency output nodes. Linear, directional selection gradients measure the effect of each trait on fitness. Scaled coefficients allow comparisons among different properties.

and the connected offset, α, output o_0 (**Table 1**) were found to be positive and of the order of 10^{-2}–10^{-3} (**Table 1**). This mutual information in the connected channel, o_0, was greater than that in the disconnected one, o_1 (see also T_0_9, **Table 2**). The estimates of the mutual information between input and hidden neurons and between hidden neurons and the offset output neuron were of the order of 10^{-3} (**Table 1**; **Figure 11A**).

These results suggest that the controller of the individual T_9_9 was indeed reactive, in spite of the variance in light-harvesting behavior and the initial impression of the oscillatory behavior of the controller (**Figure 11B**). Information analysis shows that output of the controller, namely the motor control, was not independent from the input, the sensor readings. Non-zero mutual information between input and hidden neurons suggests that the inputs influence the oscillating part of the network, which in turn influences the outputs. When we ran the sensory inputs logged during an experiment through our simulator (see Materials and Methods) and removed the recurrence, the controller delivered a turning signal that was tightly correlated with the inputs (**Figure 11C**).

DISCUSSION

The modularity–sparsity hypothesis (Clune et al., 2013) proposes that sparsity, *S*, enhances the evolution of modularity, *Q*. We tested this hypothesis, which was based on work in digital simulation, in a population of 10 physically embodied robots, Tadros, evolved over 10 generations from a population generated randomly. When Tadros were selected for improved phototaxis, selection, as measured by linear selection gradients (**Figure 8**), acted to a greater degree on the *S* of the ANN than on *Q*. But *S* and *Q* were positively correlated across generations, indicating an underlying functional relationship. Thus, as predicted by the modularity–sparsity hypothesis (Clune et al., 2013), selection on *S* does appear to influence the evolution of *Q*, indirectly, in physically embodied robots.

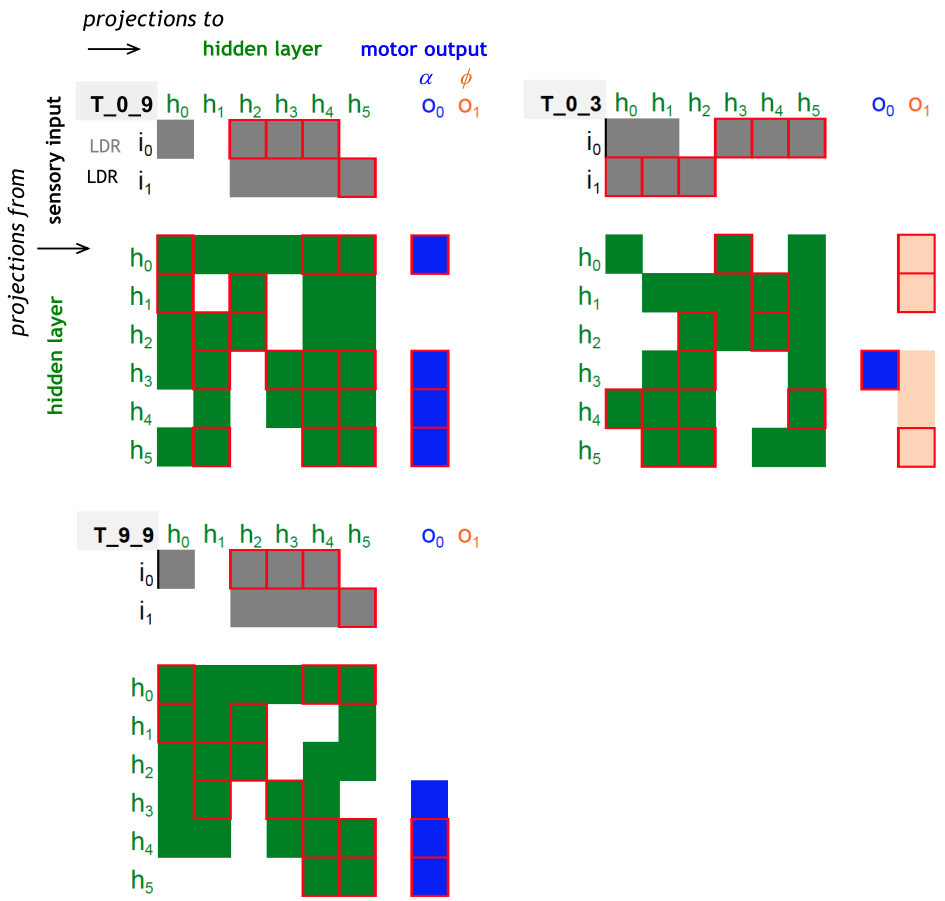


FIGURE 9 | Connectomes of random and evolved Tadros. Same examples used in **Figures 4** and **5**. Projections from one node to another are indicated by the colored cells in the matrix, with positive (1) or negative (−1) connection weights with our without a red outline, respectively; lack of a connection is indicated by a blank space. Sensory projections to the hidden layer are coded in gray; hidden layer projections are coded in green when recurrent and in blue and orange for the motor output layer. T_0_9 and T_9_9 are related by descent, have an identical pattern of inputs from the sensory to the hidden layer, and have a nearly identical pattern of inputs to the motor layer, with only a difference in sign of one node and a complete lack of any nodes controlling the flapping frequency, ϕ. Contrast that pattern with the unrelated and randomly generated connectome of T_0_3.

The evolution of S and Q is complicated, even in this simple system. While the fitness, ω, of the population increases linearly over generational time (**Figure 6A**), S and Q do not (**Figure 7**). Only within a generation did we detect a strong relationship between S and ω, in the form of linear selection gradients, and then only in some generations (**Figure 8**). In no generation is the positive or negative magnitude of the selection gradient for Q greater than that for any aspect of S. On this evidence alone, it appears that S rather than Q is the direct target of selection in this system, as predicted from results in digital simulation (Clune et al., 2013).

As we have shown previously in Tadros (Roberts et al., 2014), phenotypes not directly targeted by selection may evolve as "by-products" yoked to targeted traits by functional correlation. Selecting for both enhanced performance and increased sparsity, Clune et al. (2013) found evidence for the by-product evolution of Q in digital simulation. This appears to be the case for Q in this population of physically embodied Tadros, as well. The S_{whole} and

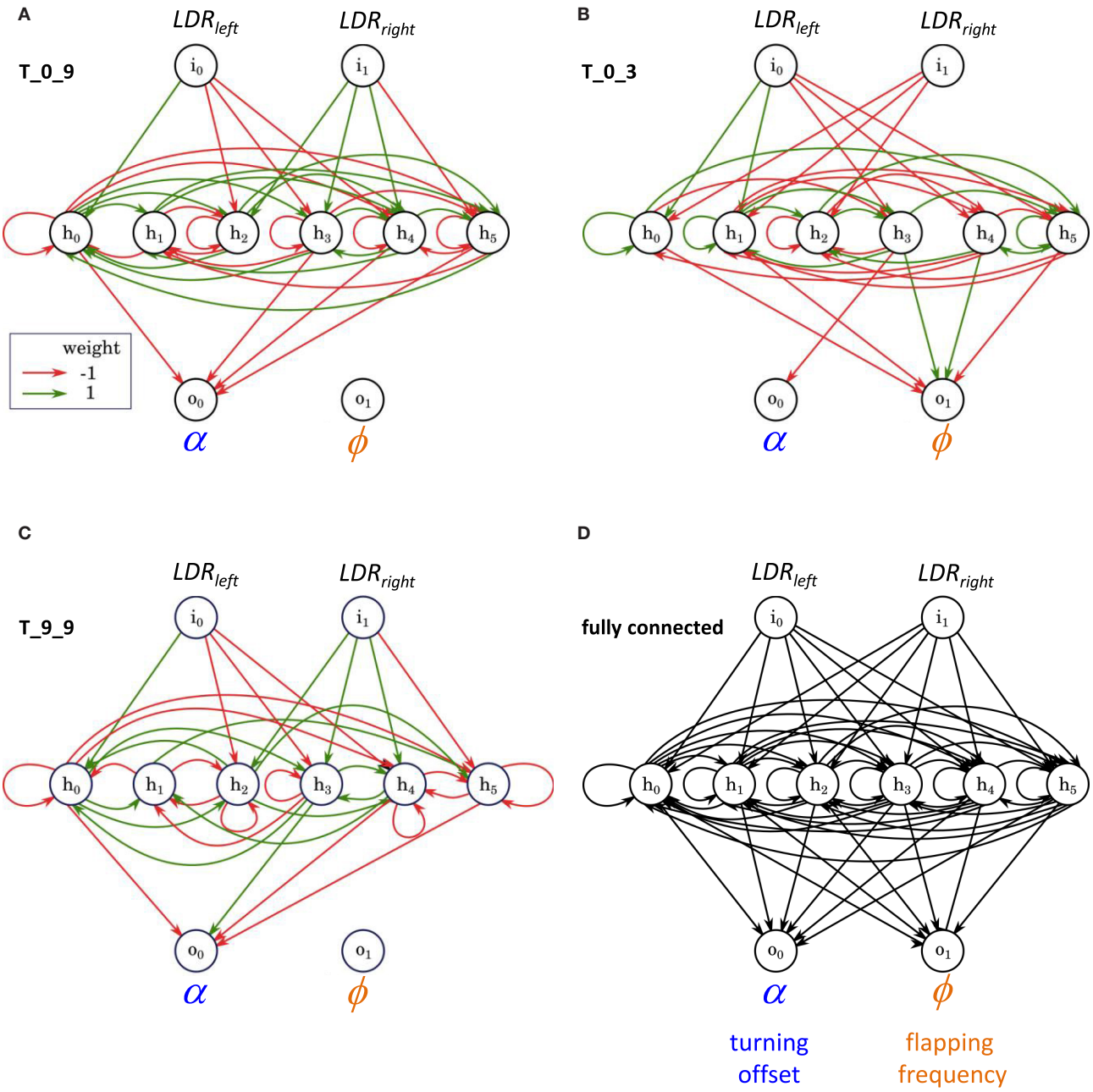


FIGURE 10 | The structure of random and evolved ANNs, examples. (A–C) Same individuals used in **Figures 4**, **5** and **9**. **(D)** A fully connected ANN for comparison.

Q of all 100 of the ANNs were positively and significantly correlated ($r = 0.407$, $p < 0.001$). Moreover, when all of the metrics of S are considered at the same time, S_α and S_ϕ, are significantly correlated with Q, positively and negatively, respectively.

In addition to the difference in physical embodiment, a subtle but important difference between this paper and the simulations of Clune et al. (2013) and Bernatskiy and Bongard (2015) is that selection on Tadros resulted from a fitness function that did not explicitly represent S. This experimental approach allowed for either, both, or neither S and Q to be targeted by selection. That S emerged as a direct target and Q and an indirect by-product is thus strong evidence in support of the Q–S hypothesis.

How are S and Q related functionally? The ANN that had the highest overall fitness (T_9_9, **Figure 5**) appeared in the final generation, and it had a S_{whole} of 0.400, near the high end of the range. The relatively sparse T_9_9 controller worked to guide the robot under and in close orbit about the light source four times (**Figure 4**), in spite of the oscillatory behavior of its ANN (**Figure 11**). This effective phototaxis nearly doubled the fitness, ω, of T_9_9's ancestor from generation 0, T_0_9, the best of that randomly generated generation. T_0_9 had a S_{whole} of 0.33, near the low end of the range. For this comparison, Q is less indicative of the differences in ω, with Q for T_9_9 and T_0_9 being 0.09 and 0.07, respectively.

But simple pairwise comparisons hide informative variance. For example, T_0_3 does not conform to the pattern suggested by the comparison of T_9_9 and T_0_9. Like T_0_9, T_0_3 is from the first, random generation. But it performed poorly, with a ω only one-third that of T_0_9, in spite of having an intermediate value of S_{whole}, 0.38, and a high value of Q, 0.12. Clearly, features of the ANN matter that are not captured in the high-level structural metrics of S_{whole} and Q.

The success of T_0_9 and T_9_9 may be linked, in part, to the sparsity of the projections from their ANN hidden layers to the motor output layer and the functional consequences of that pattern. Both T_0_9 and T_9_9 have 0 connections, $S_\phi = 1$, from the hidden layer to the node controlling the flapping frequency, ϕ (**Figures 10** and **11**). On the other hand, T_0_3 has connections

TABLE 1 | Mutual information between the states of nodes of the ANN of T_9_9.

(A)	5 bins per variable		10 bins per variable	
	o_0	o_1	o_0	o_1
i_0	1.5×10^{-3}	-7.1×10^{-6}	1.7×10^{-3}	-8.0×10^{-6}
i_1	6.9×10^{-4}	-5.6×10^{-6}	1.3×10^{-3}	-7.9×10^{-6}

(B)	5 bins per variable						10 bins per variable					
	h_0	h_1	h_2	h_3	h_4	h_5	h_0	h_1	h_2	h_3	h_4	h_5
i_0	3.7×10^{-3}	4.3×10^{-3}	4.7×10^{-3}	3.0×10^{-3}	3.7×10^{-3}	4.0×10^{-3}	1.1×10^{-2}	1.5×10^{-2}	1.3×10^{-2}	1.0×10^{-2}	8.9×10^{-3}	1.3×10^{-2}
i_1	4.3×10^{-3}	4.9×10^{-3}	4.0×10^{-3}	4.7×10^{-3}	6.4×10^{-3}	5.0×10^{-3}	1.7×10^{-2}	1.7×10^{-2}	1.5×10^{-2}	1.9×10^{-2}	2.1×10^{-2}	1.5×10^{-2}

(C)	5 bins per variable						10 bins per variable					
	h_0	h_1	h_2	h_3	h_4	h_5	h_0	h_1	h_2	h_3	h_4	h_5
o_0	1.3×10^{-3}	1.3×10^{-3}	2.0×10^{-3}	1.4×10^{-3}	1.4×10^{-3}	1.9×10^{-3}	2.1×10^{-3}	1.7×10^{-3}	3.4×10^{-3}	1.9×10^{-3}	3.7×10^{-3}	3.3×10^{-3}
o_1	-8.9×10^{-6}	-8.9×10^{-6}	-3.3×10^{-6}	-3.9×10^{-6}	-3.3×10^{-6}	-6.7×10^{-6}	-4.4×10^{-6}	-4.7×10^{-6}	-5.0×10^{-6}	-5.6×10^{-6}	-4.4×10^{-6}	-5.3×10^{-6}

(A) Between the input nodes (i_0, i_1) and the output nodes (o_0, o_1). (B) Between input nodes and hidden nodes (h_0–h_5). (C) Between hidden nodes and output nodes.

TABLE 2 | Mutual information between the states of nodes of the ANN of T_0_9.

(A)	5 bins per variable		10 bins per variable	
	o_0	o_1	o_0	o_1
i_0	1.4×10^{-3}	5.3×10^{-6}	1.7×10^{-3}	-8.0×10^{-6}
i_1	1.3×10^{-4}	0.0	1.3×10^{-3}	-7.9×10^{-6}

(B)	5 bins per variable						10 bins per variable					
	h_0	h_1	h_2	h_3	h_4	h_5	h_0	h_1	h_2	h_3	h_4	h_5
i_0	4.5×10^{-3}	5.4×10^{-3}	3.5×10^{-3}	4.9×10^{-3}	4.8×10^{-3}	3.2×10^{-3}	1.1×10^{-2}	1.5×10^{-2}	1.3×10^{-2}	1.0×10^{-2}	8.9×10^{-2}	1.3×10^{-2}
i_1	4.9×10^{-3}	5.0×10^{-3}	7.2×10^{-3}	9.6×10^{-3}	7.4×10^{-3}	3.2×10^{-3}	1.7×10^{-2}	1.7×10^{-2}	1.5×10^{-2}	1.9×10^{-2}	2.1×10^{-2}	1.5×10^{-2}

(C)	5 bins per variable						10 bins per variable					
	h_0	h_1	h_2	h_3	h_4	h_5	h_0	h_1	h_2	h_3	h_4	h_5
o_0	2.8×10^{-3}	1.8×10^{-3}	9.7×10^{-4}	8.5×10^{-4}	2.8×10^{-3}	1.5×10^{-3}	3.4×10^{-3}	1.9×10^{-3}	2.6×10^{-3}	1.9×10^{-3}	6.7×10^{-3}	4.9×10^{-3}
o_1	-1.7×10^{-6}	3.9×10^{-6}	-2.2×10^{-6}	1.7×10^{-6}	1.1×10^{-6}	0.0	-3.3×10^{-6}	-8.3×10^{-7}	-3.6×10^{-6}	-1.7×10^{-6}	4.7×10^{-6}	-3.6×10^{-6}

(A) Between the input nodes (i_0, i_1) and the output nodes (o_0, o_1). (B) Between input nodes and hidden nodes (h_0–h_5). (C) Between hidden nodes and output nodes.

to both motor output nodes, the turning offset *a* and ϕ, with most of them to ϕ. Here is where the unintended oscillatory behavior of the hidden layer comes into play, adding noise to the motor control outputs. A simple way to reduce total noise is to eliminate one of the output channels. We have shown that a single channel can behave reactively, even with noise, as determined by mutual information analysis (**Figure 11**). Controlling only α, T_0_9 and most of its descendants, including T_9_9, were the most successful lineage (**Figure 5**). High S_ϕ at the level of projections from the hidden layer to the motor output layer improved phototaxis.

Understanding the functional importance of S_α and S_ϕ allows us to understand the evolutionary relationship between the two types of sparsity and *Q*. During the evolutionary jump in fitness from generation 6 to 7 (**Figure 6A**), the selection gradients measured in generation 7 showed strong positive selection on S_α and S_ϕ without selection on *Q* (**Figure 7**). The trend reversed in the next generation, suggesting that over those generations oscillating selection acted to stabilize these phenotypes. As selection acts directly on S_α and S_ϕ, it acts indirectly on *Q* because of the correlation of *Q* with both types of *S*. Interestingly, when the selection

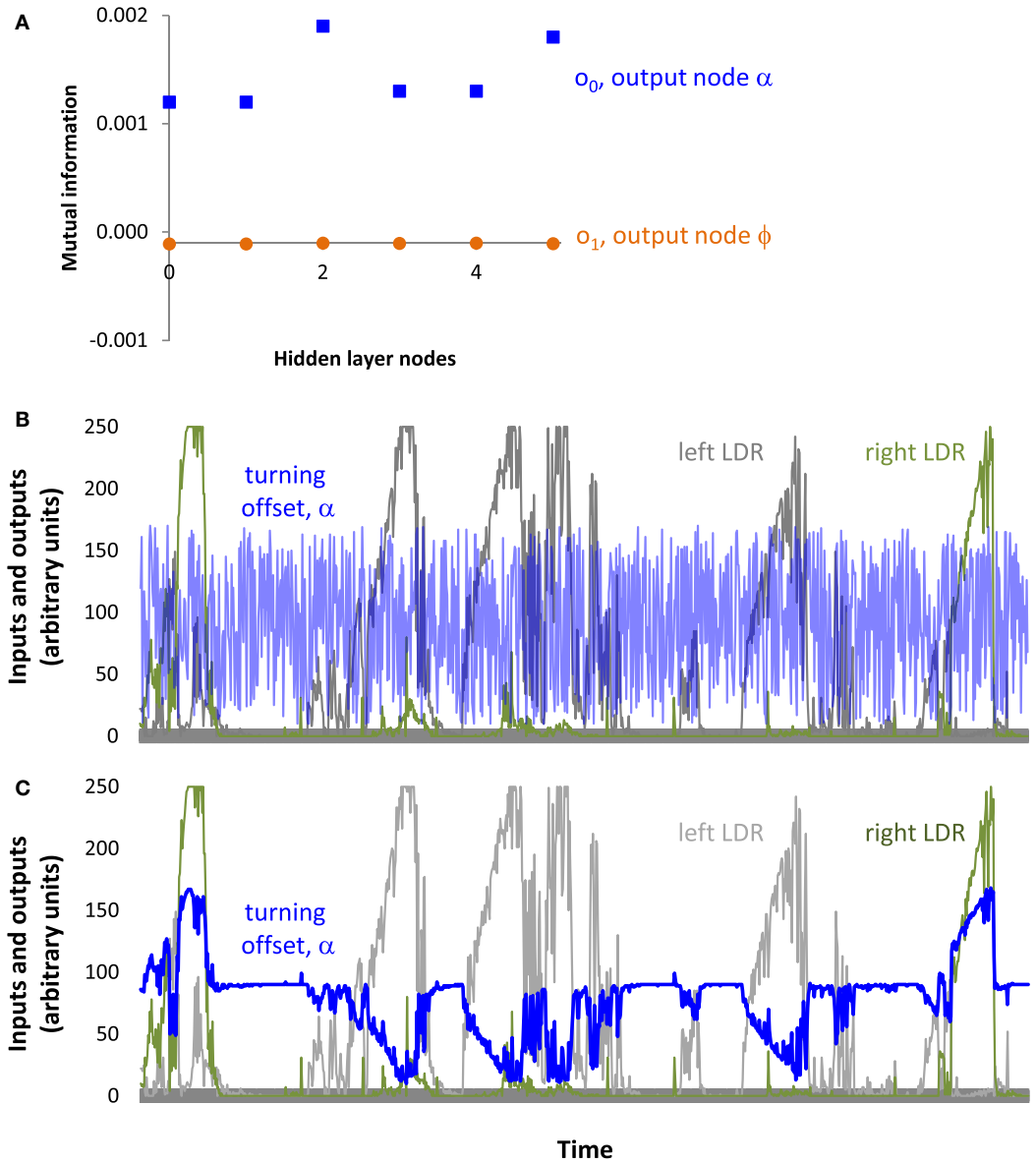


FIGURE 11 | Information pass-through for the ANN of T_9_9. **(A)** Mutual information is greater in the output node, α, connected to the ANN. Since the output node for ϕ is disconnected, it serves as a reference. Note that only hidden nodes 0, 3, 4, and 5 connect directly to output node 0; the passing of information from hidden nodes 1 and 2 occurs through the other hidden nodes, to which they are connected. **(B)** Oscillatory output. Inputs to the two light-dependent resistors (LDRs) and the resulting output to the turning offset of the tail, α. Inputs and outputs are the actual values logged by T_9_9 during a trial. **(C)** Without recurrence in the hidden layer, the signals at the sensors would produce a slowly changing turning signal that would steer Tadro left and right, depending on which LDR was stimulated. These are the same inputs as above but the output is hypothetical.

on S_α and S_ϕ is of the same sign for both, the indirect effects on Q counteract, since Q is positively and negatively correlated with S_α and S_ϕ, respectively. But since the coefficient for S_α (0.051) is approximately triple the magnitude of that for S_ϕ (−0.014), selection of the same magnitude will result in indirect selection of Q in phase with S_α.

Understanding the interactions of S, Q, and ω, we are now in a position to make predictions about the Q–S fitness landscape. Because S_{whole} is positively correlated with S_α ($r = 0.699$), we can meaningfully talk about S of the whole ANN as it relates to Q and ω. The Q–S fitness landscape contains three important landmarks (**Figure 12**). The first landmark is T_9_9, the best of the evolved Tadros. The second is the engineered ANN, a mobile Tadro akin to the ideal agent in this situation, with two modules ($Q = 0.50$), one acting to slow the tail flapping as it approaches the light and the other acting to turning more tightly as it approaches the light. The ANN of this Tadro is very simple, without recurrence in the hidden layer, and $S_{whole} = 0.933$. It also has ω nearly double that of T_9_9. But the highest possible "fitness" is actually represented by the third landmark, an immobile squatter that we simply placed directly under the light for 5 min. Lacking any connections in its ANN, the squatter has $Q = 0$ and $S = 1$. But this position in Q–S space is impossible for a Tadro to achieve, since mobile Tadros start in the dark and must move under the light. Moreover, the Tadros as built cannot stop when they get under the light, since they are programmed with a default flapping frequency. For these reasons, we predict that the population of Tadros, given sufficient time and genetic variation, would evolve up the fitness gradient that leads to the engineered Tadro.

In order for this population of Tadros to climb the entire fitness gradient to the position of the engineered Tadro, one critical feature must evolve: the reduction of recurrent connections in the hidden layer. When that increase in S occurs, the population will be free to take advantage of more complex motor control by then connecting to the tail flap output. This prediction may be specific to the Tadro system with the oscillatory controllers. Were the Tadros to evolve without the oscillatory behavior, recurrence might allow for internal memory and the more complex processing and behaviors that this capacity would allow. Indeed, this possibility motivated our original decision to include recurrence in the hidden layer.

While the oscillatory behavior of the controller was an accident, it proved to be an informative one. The information from controller to motor output was sufficient to create reactive behavior, but at a cost: high variability in performance. That high variability adds noise to the relationship between selection

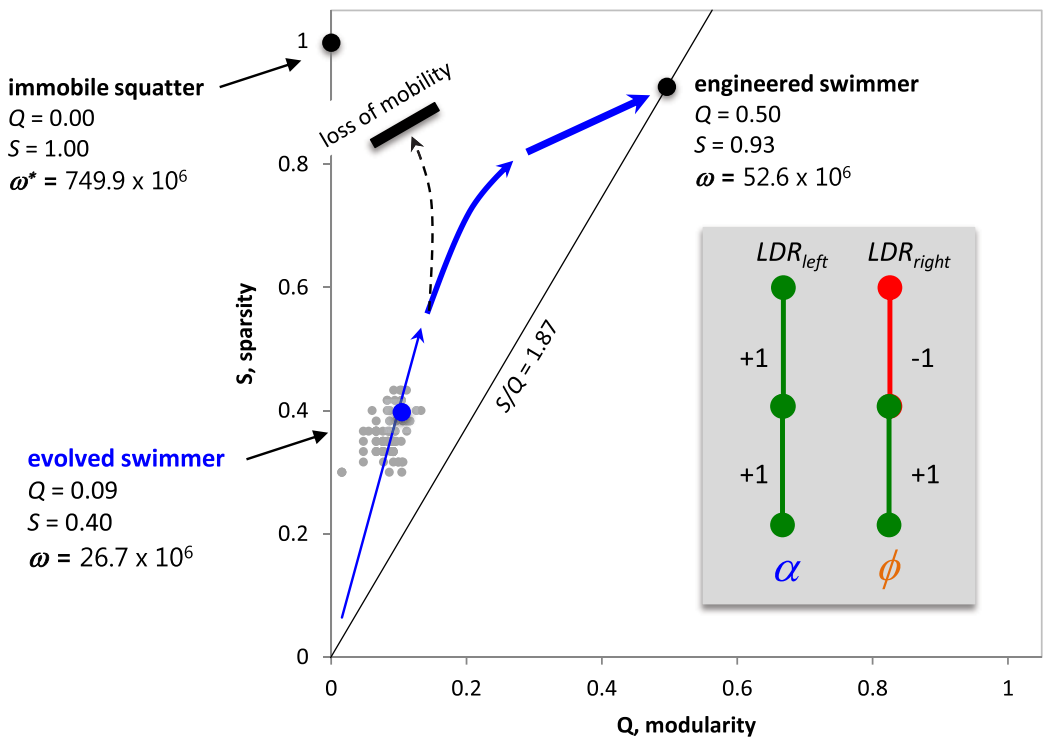


FIGURE 12 | Predicted *Q*–*S* fitness landscape for Tadros under selection for enhanced phototaxis in a world with a single light source. While the immobile squatter has the highest light-gathering possible, the evolutionary path to that spot is blocked because of the need for mobility in the behavioral task of phototaxis. Hence, we predict that the population, given more time, will follow a path (blue arrows) toward the higher fitness of the engineered swimmer, evolving an ANN with high S_{whole}, intermediate Q, and no recurrence. For the experimental data (gray points), over 10 generations S_{whole} and Q were positively and significantly correlated ($r = 0.407$, $p < 0.001$). Inset: engineered ANN from **Figure 1**.

and behavior. Thus, once recurrence is reduced by evolution, we expect evolution to accelerate. This behavior, should it occur, speaks to evolvability. The random integration of controllers, represented here by a low *Q* and low *S* in our early populations, puts brakes on evolution by spreading and propagating noise. Thus increases in *Q* and *S* are expected, up to a point, to increase mutual information between inputs and outputs. Finding systems that can autonomously evolve along complex and undulating pathways in rugged fitness landscapes continues to be a central challenge in evolutionary robotics.

AUTHOR CONTRIBUTIONS

NL, KL, AB, MS, JS, JB, and JL conceived of and designed the experiments. NL built and programed the Tadro and ran the experiments. AB programed the reproduction and selection algorithm. MS and DW programed the neural network simulator for desktop analysis. All the authors analyzed the data and contributed to the writing of the manuscript.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the students who contributed important work to this project. Sabrina Kleman helped run the evolutionary experiments. Robot Fellows Evan Altiero, Nick Burka, John Loree, Jessica Ng, Josh Ridley, and Meghan Willcoxon helped in early robot design, prototyping, validation, and analysis of the environment and behavior. Larry Doe of Vassar College was instrumental with electronics.

FUNDING

This work was funded by the U.S. National Science Foundation (grant no. 1344227, INSPIRE, Special Projects).

REFERENCES

Bernatskiy, A., and Bongard, J. C. (2015). "Exploiting the relationship between structural modularity and sparsity for faster network evolution," in *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference* (New York: ACM), 1173–1176.

Blondel, V. D., Guillaume, J. L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *J. Stat. Mech.* 2008, 10008. doi:10.1088/1742-5468/2008/10/P10008

Bongard, J. (2015). Using robots to investigate the evolution of adaptive behavior. *Curr. Opin. Behav. Sci.* 6, 168–173. doi:10.1016/j.cobeha.2015.11.008

Bongard, J. C. (2011). "Spontaneous evolution of structural modularity in robot neural network controllers: artificial life/robotics/evolvable hardware," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation* (New York: ACM), 251–258.

Bongard, J. C., Bernatskiy, A., Livingston, K., Livingston, N., Long, J., and Smith, M. (2015). "Evolving robot morphology facilitates the evolution of neural modularity and evolvability," in *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference* (New York: ACM), 129–136.

Clauset, A., Newman, M. E., and Moore, C. (2004). Finding community structure in very large networks. *Phys. Rev. E* 70, 066111. doi:10.1103/PhysRevE.70.066111

Clune, J., Mouret, J. B., and Lipson, H. (2013). The evolutionary origins of modularity. *Proc. Biol. Sci.* 280, 20122863. doi:10.1098/rspb.2012.2863

Doorly, N., Irving, K., McArthur, G., Combie, K., Engel, V., Sakhtah, H., et al. (2009). "Biomimetic evolutionary analysis: robotically-simulated vertebrates in a predator-prey ecology," in *Artificial Life, 2009. ALife'09. IEEE Symposium* (New York: IEEE), 147–154.

Ellefsen, K. O., Mouret, J. B., and Clune, J. (2015). Neural modularity helps organisms evolve to learn new skills without forgetting old skills. *PLoS Comput. Biol.* 11:e1004128. doi:10.1371/journal.pcbi.1004128

Espinosa-Soto, C., and Wagner, A. (2010). Specialization can drive the evolution of modularity. *PLoS Comput. Biol.* 6:e1000719. doi:10.1371/journal.pcbi.1000719

Kashtan, N., and Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proc. Natl. Acad. Sci. U.S.A.* 102, 13773–13778. doi:10.1073/pnas.0503610102

Kim, J. S., and Kaiser, M. (2014). From *Caenorhabditis elegans* to the human connectome: a specific modular organization increases metabolic, functional and developmental efficiency. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 369, 20130529. doi:10.1098/rstb.2013.0529

Lipson, H., Pollack, J. B., and Suh, N. P. (2002). On the origin of modular variation. *Evolution* 56, 1549–1556. doi:10.1554/0014-3820(2002)056[1549:OTOOMV]2.0.CO;2

Long, J. (2012). *Darwin's Devices: What Evolving Robots Can Teach Us About the History of Life and the Future of Technology*. New York: Basic Books.

Long, J. H. Jr., Koob, T. J., Irving, K., Combie, K., Engel, V., Livingston, N., et al. (2006). Biomimetic evolutionary analysis: testing the adaptive value of vertebrate tail stiffness in autonomous swimming robots. *J. Exp. Biol.* 209, 4732–4746. doi:10.1242/jeb.02559

Long, J. H. Jr., Lammert, A. C., Pell, C. A., Kemp, M., Strother, J. A., Crenshaw, H. C., et al. (2004). A navigational primitive: biorobotic implementation of cycloptic helical klinotaxis in planar motion. *J. IEEE Oceanic Eng.* 29, 795–806. doi:10.1109/JOE.2004.833233

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* 12, 2825–2830.

Roberts, S. F., Hirokawa, J., Rosenblum, H. G., Sakhtah, H., Gutierrez, A. A., Porter, M. E., et al. (2014). Testing biological hypotheses with embodied robots: adaptations, accidents, and by-products in the evolution of vertebrates. *Front. Rob. AI* 1:12. doi:10.3389/frobt.2014.00012

Rorick, M. M., and Wagner, G. P. (2011). Protein structural modularity and robustness are associated with evolvability. *Genome Biol. Evol.* 3, 456–475. doi:10.1093/gbe/evr046

Voordeckers, K., Pougach, K., and Verstrepen, K. J. (2015). How do regulatory networks evolve and expand throughout evolution? *Curr. Opin. Biotechnol.* 34, 180–188. doi:10.1016/j.copbio.2015.02.001

Wagner, G. P. (1996). Homologues, natural kinds and the evolution of modularity. *Am. Zool.* 36, 36–43. doi:10.1093/icb/36.1.36

Wagner, G. P., and Altenberg, L. (1996). Perspective: complex adaptations and the evolution of evolvability. *Evolution* 50, 967–976. doi:10.2307/2410639

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 24 January 2017
doi: 10.3389/frobt.2017.00001

Epigenetic Operators and the Evolution of Physically Embodied Robots

Jake Brawer[1,2,3], Aaron Hill[1,4], Ken Livingston[1,2], Eric Aaron[1,4], Joshua Bongard[5] and John H. Long Jr.[1,2,6]*

[1] Interdisciplinary Robotics Research Laboratory, Vassar College, Poughkeepsie, NY, USA, [2] Department of Cognitive Science, Vassar College, Poughkeepsie, NY, USA, [3] Department of Computer Science, Yale University, New Haven, CT, USA, [4] Department of Computer Science, Vassar College, Poughkeepsie, NY, USA, [5] Department of Computer Science, University of Vermont, Burlington, VT, USA, [6] Department of Biology, Vassar College, Poughkeepsie, NY, USA

The genetic operators (GOs) of recombination, mutation, and selection are commonly included in studies of evolution and evolvability, but they are not the only operators that can affect the genotype-to-phenotype (G → P) map and thus the outcomes of evolution. In this paper, we present experiments with an *epigenetic operator* (*EO*), interactive wiring of a circuit, alongside common GOs, investigating both epigenetic and GO effects on the evolution of both simulated and physically embodied Braitenberg-inspired robots. As a platform for our experiments, we built a system that encoded the genetics for the physical circuitry of the analog robots and made explicit rules for how that circuitry would be constructed; phenotypic expression consisted of the placement of wires to form the circuitry and thus govern robot behavior. We then varied the presence of gene interactions across populations of robots, studying how the EO—and its effects on G → P maps—affected the results of evolution over several generations. Additionally, a variant of these experiments was run in simulation to provide an independent test of the evolutionary impact of this EO. Our results demonstrate that robot populations with the EO had quantitatively different and potentially less adaptive evolution than populations without it. For example, selection increased the rate at which functional circuitry was lost in the population with the EO, compared to the population without it. In addition, in simulation, EO populations were significantly less fit than populations without it. More generally, results such as these demonstrate the interaction of genetic and EOs during evolution, suggesting the broad importance of including EOs in investigations of evolvability. To our knowledge, our work represents the first physically embodied EO to be used in the evolution of physically embodied robots.

Keywords: epigenetic operators, evolutionary robotics, development, genotype–phenotype mapping, physically embodied robots

OPEN ACCESS

Edited by:
Geoff Nitschke, University of Cape Town, South Africa

Reviewed by:
Evert Haasdijk, VU University Amsterdam, Netherlands
Julien Hubert, VU University Amsterdam, Netherlands

****Correspondence:***
Jake Brawer
jake.brawer@yale.edu

Specialty section:
This article was submitted to Evolutionary Robotics, a section of the journal Frontiers in Robotics and AI

Received: *21 April 2016*
Accepted: *03 January 2017*
Published: *24 January 2017*

Citation:
Brawer J, Hill A, Livingston K, Aaron E, Bongard J and Long JH Jr. (2017) Epigenetic Operators and the Evolution of Physically Embodied Robots. Front. Robot. AI 4:1. doi: 10.3389/frobt.2017.00001

INTRODUCTION

In biology, understanding how development—the mapping of genotype-to-phenotype (G → P)—shapes the creation of phenotypic variation has created a paradigmatic shift in evolutionary theory (Wagner and Altenberg, 1996; Pigliucci, 2010). The long-standing "adult transformation" paradigm treated development as if it were absent, invariant, or instantaneous, in spite of

Garstang's (Garstang, 1922) early hypothesis that ontogenies, not fully formed adults, evolve (Northcutt, 2002). Implementing Garstang's approach required the tools of modern molecular biology to find and track (1) the expression of genes, (2) the epigenetic processes that convert gene products into working molecular machinery, and (3) the feedback among developing cells and tissues in a system that bootstraps its own manufacture while interacting autonomously with its local ecology. With the understanding that the organism faces and responds to selection pressures throughout its life, evolutionary developmental biology was born ("evo-devo"; Amundson, 2005; Carroll, 2008). For roboticists, the evo-devo challenge is to create physically embodied systems that incorporate the three scales of time and the processes inherent in each: behavior, development, and evolution (Pfeifer and Bongard, 2006). Because of the complexity of building and evolving physical robots, this is a daunting challenge in the quest for the "evolution of things" (Eiben and Smith, 2015). As an initial step toward this goal, in this paper we create a physically embodied system that allows us to examine systematically how developmental and evolutionary processes interact.

An explicit evo-devo approach has proven invaluable in the evolution of artificial neural networks ("ANNs"; Kitano, 1990; Floreano et al., 2008; Mattiussi et al., 2008; for a review). Development serves as a new type of evolutionary driver—alongside the genetic operators (GOs) of mutation, recombination, and selection—facilitating evolvability in embodied agents (Bongard, 2002; Bongard and Pfeifer, 2003; Pfeifer et al., 2007). Since developmental processes, like genetic processes, are complex and varied, we recognize them as a class of operators—*epigenetic operators* (EOs).

By our restricted definition, EOs alter the phenotypic expression of a genome. Recognizing that EOs work in conjunction with GOs, our goal is to create a conceptual and physical methodology that allows investigators to manipulate the interaction of GOs and EOs in physically embodied robots. For starters, if one compares two genetically identical populations, one with and one without an EO, the evolutionary impact of the EO can be substantial. One can manipulate this EO effect by changing the developmental rules of the EO directly, altering the fitness landscape, and changing the initial position of the populations within that landscape (**Figure 1**).

A G → P map without EOs is the equivalent of Northcutt's (Northcutt, 2002) adult transformation paradigm: adults appear in final form on the evolutionary stage without explicit recognition of how they were created. In evolutionary robotics (ER), the construction of physically embodied robots has never been, to our knowledge, manipulated directly as an experimental variable. We do so here, creating an embodied EO that recognizes that connections between sensors and motors may interact during a developmental process in which the genome's instructions are enacted to sequentially wire a circuit board with a limited set of connection pins.

The value of incorporating developmental processes into evolutionary computational models has long been known to the AI community. Gruau's (Gruau, 1994) proposed model evolved cellularly encoded ANNs: the G → P map does not directly represent aspects of the phenotype but rather encodes the rules for how neural "cells" split and connect to their daughter cells. These rules are ordered in a collection of binary trees, which evolve through the application of the GOs. Each ANN begins as a single cell, but develops into a fully fledged network through execution of the rules in each tree during the G → P mapping process. The model's strength lies in the fact that the terminal nodes of trees are allowed to point to the root of other trees, allowing for potentially useful substructures to repeat in the completed networks.

This idea of reuse underlies a more recent evo-devo approach to designing ANNs, HyperNEAT (Stanley et al., 2009). HyperNEAT builds on *neuro-evolution of augmenting topologies* (NEAT, a method for evolving ANNs that does not require the topology of the network to be set *a priori*) by incorporating a generative and developmental encoding scheme. This encoding scheme, called *compositional pattern producing network* (CPPN), indirectly encodes the weights between nodes in the ANN (Stanley, 2007). This is done by treating the NEAT-evolved ANN as existing in an n-dimensional Cartesian space. The CPPN, which is essentially a composition of geometric functions, takes the coordinates of every pairwise set of nodes as input, and for each returns the output of the functions, which represents the connection weight between the two nodes. Because the network's weights are determined by repeated application of the same set of functions, the resultant connectivity network is often highly regular and symmetrical, much like biological brains (cf. Gilbert and Wiesel, 1992). Additionally, given that weights are determined as a function of the nodes' positions in space, geometric relationships between inputs could be autonomously exploited.

The main advantage of these developmental systems is that they efficiently encode phenotypes by doing so indirectly (Eiben and Smith, 2015). Genes code for processes that build structures rather than for the structures themselves. This efficiency is enhanced when genetic structures are reused and redeployed. These developmental models were designed with the understanding that the process by which a phenotype is constructed is as critical to an individual's fitness as the information coded in its genome. However, in biological systems, this construction process is not fully specified in the genome (Pigliucci, 2010). Instead, epigenetic processes, together forming the genetic regulatory network (GRN), alter how the genotype is expressed through physical interactions. For example, pleiotropic interactions spread the expression of one gene to many phenotypes. In epistasis, two or more genes interact to alter one phenotype. The dynamic GRN is the enacted G → P map.

Heeding the call for morphogenetic robotics (Jin and Meng, 2011) and morphogenetic engineering (Doursat et al., 2012), we note that what is missing from ER is not development *per se* but rather physically embodied development (PED). We take the first simple steps toward combining the two here by examining the interactions of EOs and GOs in the evolution of physically embodied and simulated robots. First, we provide a simple conceptual framework for the evolutionary impact of EOs on populations (**Figure 1**). Second, given the large number of possible EOs in any system, we use the constraints imposed by a physical, analog robot to motivate the

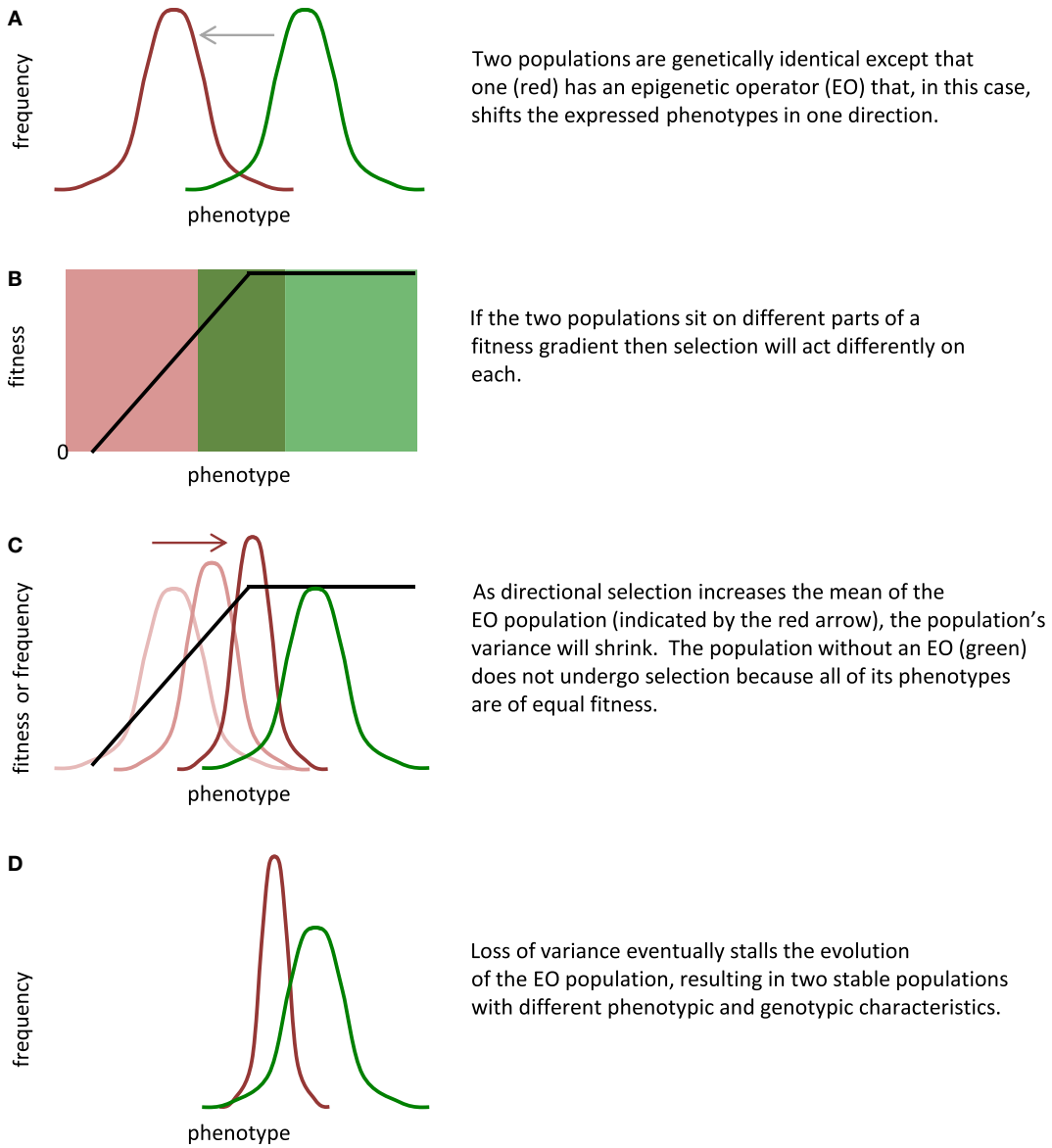


FIGURE 1 | **Epigenetic operators (EOs) and selection interact to alter evolution**. By design, EOs alter the expression of phenotypes by the genome. In this general scenario, the evolution of two populations that are initially genetically identical **(A)** is controlled by three main factors: (1) type of phenotypic difference created by the EO; (2) shape of the fitness landscape; and (3) location of the populations on the fitness landscape **(B)**. Assumptions of this model include the following: (1) the fitness gradient is stable over generational time; (2) the rate of mutation is constant but insufficient to replace genetic variance lost by selection; and (3) no gene flow exists between the populations. In this example, selection increases the mean and decreases the variance of the EO population **(C)**; the loss of variance stalls evolution by selection **(D)**. Given the ability to adjust the factors and the assumptions, the scenario shown here is one of many possible.

creation of a physically embodied EO. The embodied EO is the destructive physical interaction of two or more gene expression pathways, which we call *threads*. The instantiation of the threads is the physical wiring between sensors and motors (see Materials and Methods for details), a G → P mapping process that we call "interactive thread development." Third, we test the fundamental hypothesis that an EO will alter the evolutionary trajectory of a population. Using physical and simulated robots, we compare the evolutionary dynamics of populations with and without this EO.

MATERIALS AND METHODS

Physical Robot

The Ana BBot from Johuco Ltd. (http://johuco.com) is a physical, analog robot inspired by Braitenberg's (Braitenberg, 1986) vehicles (**Figure 2**). While a robot with a digital microcontroller could be used, we chose an analog robot so that the genome—which codes for the connections on the circuit—has an actual, not simulated, expression in the physical world. This physical expression of the genome guided our creation of an EO (see EO: Interactive

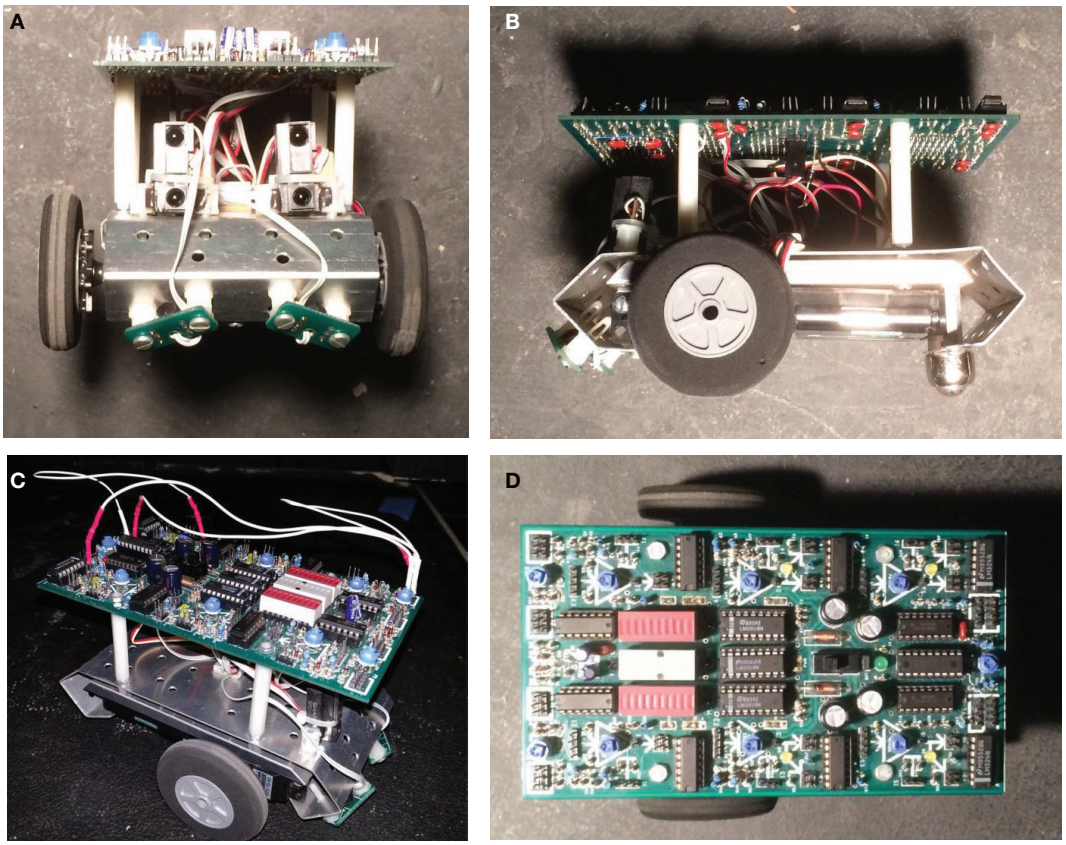


FIGURE 2 | **Ana BBot, a mobile robot that is programmable using jumper wires to connect sensors and motors**. **(A)** Front view, with photocells and IR range detectors. **(B)** Lateral (left) view, showing drive wheels. **(C)** Robot wired using jumpers. **(D)** Top view, showing circuit board unwired.

Thread Development) as the physical interaction of the wires that connect components on the circuit board.

The Ana BBot has four sensors, two IR proximity detectors and two photosensors, mounted on the front. The Ana BBot also has two motors that differentially drive the robot, with an unpowered posterior caster wheel to maintain balance. The open circuit board (**Figure 2D**) allows different components to be wired together with jumper wires (**Figure 2C**) that connect to headers ("pins"). The robot is programed by changing the wiring of its circuit board.

Each of the components on the circuit board has a corresponding group of functionally equivalent pins ("pin groups"). For a given input pin group, each pin may receive an electrical signal that comes from a variety of sources; for a given output pin group, each pin produces an identical electrical signal ranging from 0 to 1. While the signal between pin groups can be easily modified *via* alteration to the wiring, the strength of the signals themselves can be modulated if they pass through one of the robot's six "neurons." Each neuron has both excitatory and inhibitory input pin groups. The neuron outputs the sum of its inputs to a pin group directly or *via* an adjustable threshold for firing. Sensors may bypass the neurons and connect directly to the motors. Additional signal modulation is possible *via* the associated gain trimpots for the sensors, neurons, and motors and through alterations to the internal resistance of the wires. For the purposes of our experiment, trimpots were all centered and only wires with a 470 kΩ internal resistance (1× multiplication factor) were used.

Genome

The genome represents possible wirings of the Ana BBot's circuit board (**Figure 3**). The genome itself consists of a fixed number of objects called bases, containing two binary values: a bit value and a crossover point value. The bit serves as a basic unit of genetic expression analogous to biological nucleobases, while the crossover point is used in reproduction to signal a potential stop to the copying of data from a parental genome to its offspring. Both bits and crossover points may be either 1 or 0. Within the initial populations, each bit has an equal chance of being either 1 or 0. Each of the initial genomes in a population has two crossover points (value of 1) assigned randomly to a respective number of bases. In these experiments, crossover points and bits each have a 1/2,000 probability of being altered by the mutation operator.

The genome is split into genetic modules that encode for "threads." A thread specifies a series of wires that form a connection across the circuit, and each genome can code for 0 or more threads (**Figure 4**). We specified that wires cannot form self-recurrent connections at input pin groupings of neurons.

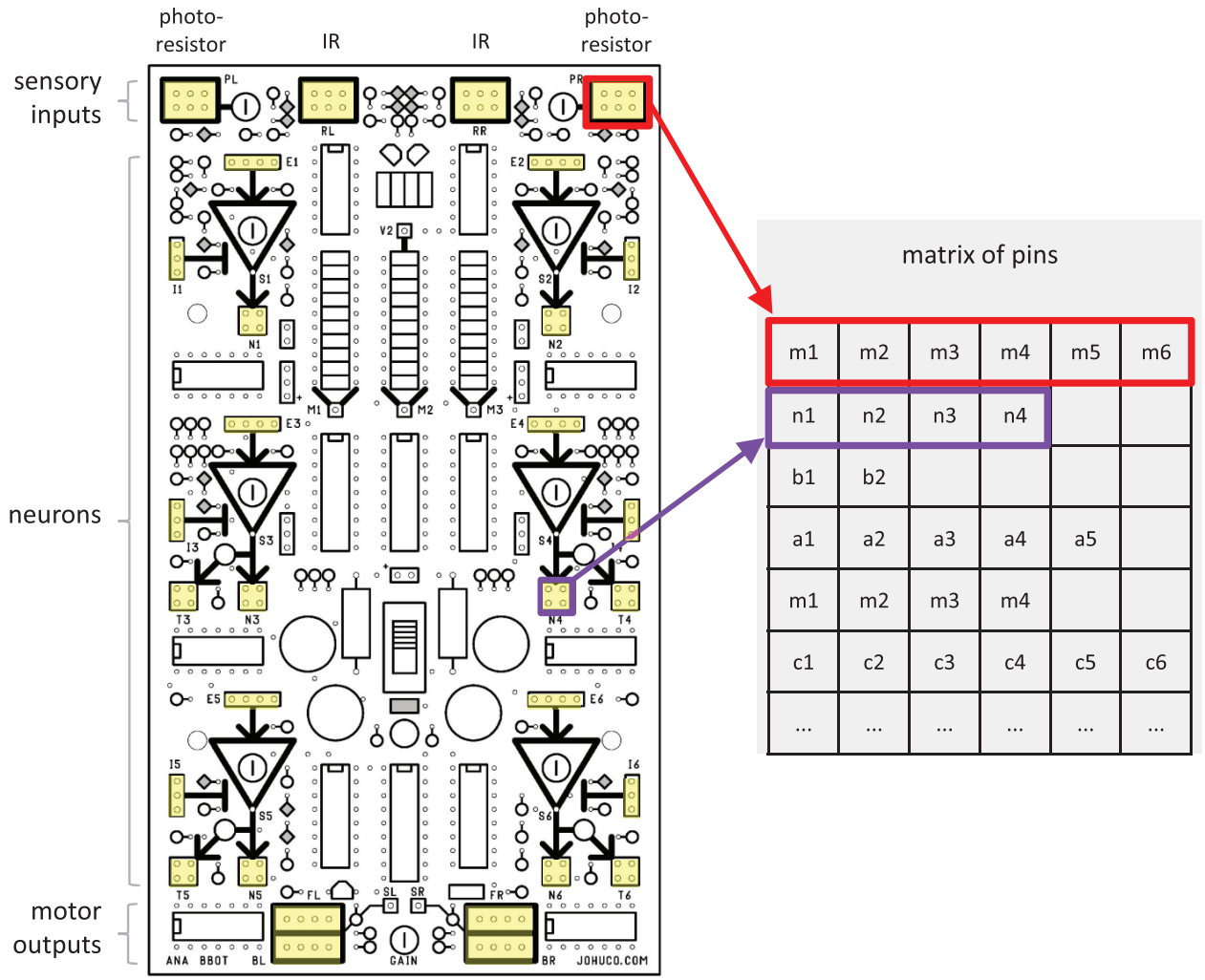


FIGURE 3 | The genome represents the possible wirings of the Ana BBot. The pins on the circuit board of the Ana BBot are represented in a matrix, with each pin group forming a row and the pins within that group forming its elements. Pin groups used are highlighted in yellow. There are 122 pins. Schematic used with permission from Johuco, Ltd. (http://johuco.com).

Self-recurrent connections were allowed between output pins of the neurons; a positive feedback loop that increases signals can be achieved through such a configuration. In total, 7,255 unique wire connections are possible, as calculated:

$$x = p(p-1)/2, \tag{1}$$

with p being the total number of pins on the circuit board, 122. We subtracted the disallowed self-recurrent connections,

$$y = x - ab(b-1)/2, \tag{2}$$

where $a = 6$, the number of neurons on the circuit, and $b = 7$, the number of input pins in each.

In order to map from a binary genome to the physical threads, a decoder operates on the genome, treating the genome as a concatenation of 4-bit strings and translating each 4-bit component into its equivalent decimal number (e.g., 0010 translates to 2, 0101 translates to 5); although 4-bit strings can represent any number from 0 to 15, the decoder only translates components with binary values in the decimal range 0–9, reducing the genetic search space and thus the computational time required to generate viable populations. The resulting decimal digits are then treated as a program for generating the resulting wiring—they can be viewed as encoding instructions for the movement of an "automaton" that traverses the circuit matrix (**Figure 3**), describing the pattern in which wires are added to create a thread (**Figure 4**). The first two decoded decimal digits are treated as the starting (X, Y) coordinates for the first wire; the next two digits determine the direction (eight possible, four cardinal, and four inter-cardinal) and distance of the jump to the ending position of that wire. If a thread contains more than one wire, the origin of the second wire will be a free pin in the pin group of the terminus of the previous wire connection.

It is possible for the decimal numbers to specify a coordinate position that is outside of the bounds of the pin matrix, which

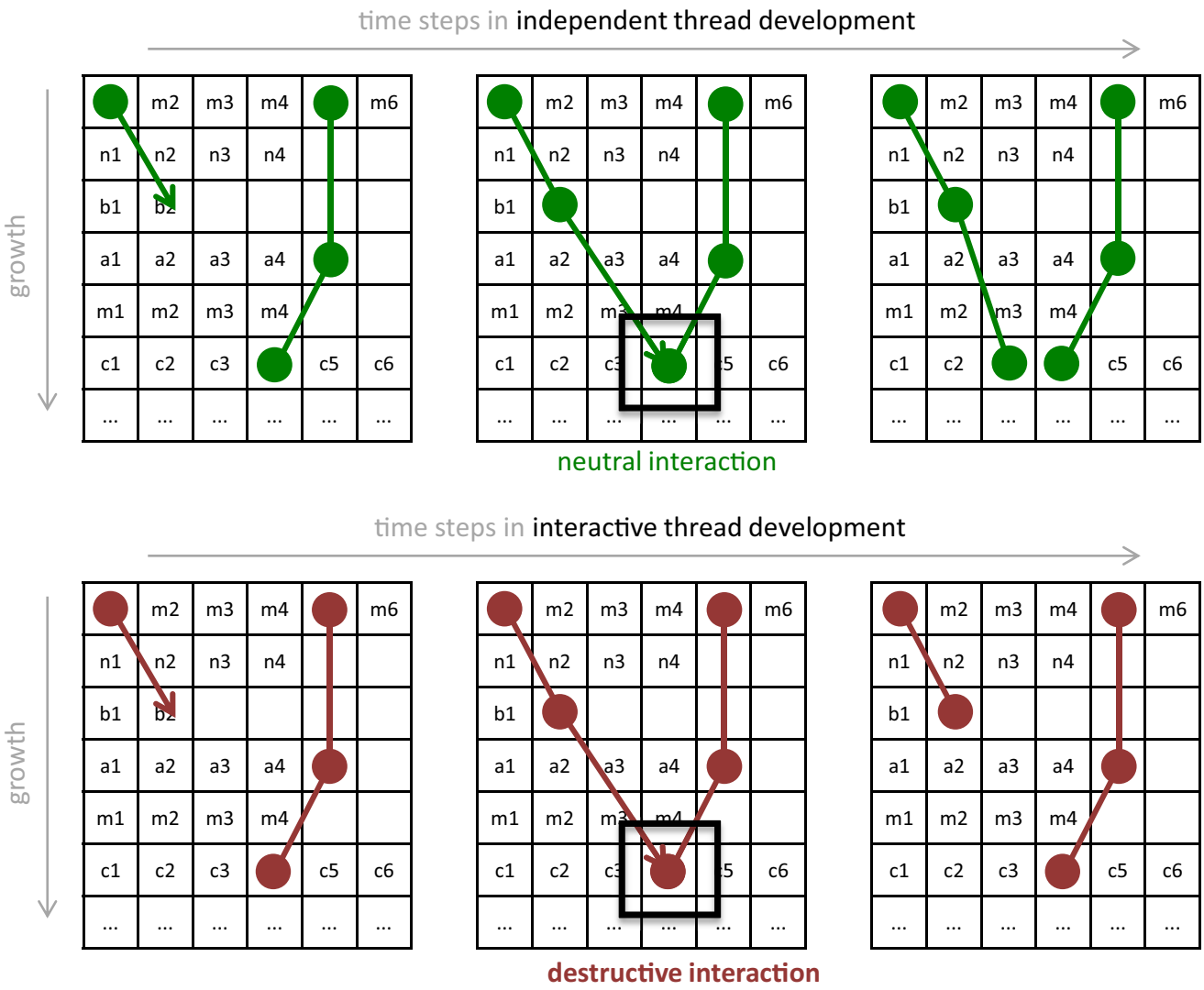


FIGURE 4 | **Development of threads with and without the epigenetic operator (EO) that causes threads to interact destructively**. **(A)** Without an EO, independent thread development allows threads to grow as a series of connections that share rows on the pin matrix, as genetically determined. If one thread attempts to attach onto an occupied pin, then it wires on an adjacent pin. **(B)** With the interactive thread EO, development prevents some threads from forming. If one thread attempts to attach to an occupied pin, then this interaction destroys that wire. The thread retreats to the previous pin and terminates its growth. In either process, once development is completed, the wiring diagram is used to wire the physical Ana BBot. Under the rules of development, the 122 pins may form 7,255 possible unique connection patterns.

would not correspond to an actual pin on the circuit. In this event, the thread simply terminates, leaving the previous wire connections, if they exist, intact on the circuit board. It is also possible for multiple genes to express threads that code for connection to the same pin on the matrix.

EO: Interactive Thread Development

Because the expression of genetically encoded threads involves the sequential connection of wires to pins on the circuit board, the beginning or end coordinates of wires specified in the second thread or later could in principle already be occupied by a previously specified wire. We treat this kind of physical interaction among wire endpoints as an EO: when interaction occurs, the forming thread is terminated, hence altering the expression of the genotype. We call this EO-driven process *interactive thread development* (**Figure 4**).

To test the hypothesis that this EO will alter the evolution of a population of Ana BBots (see **Figure 1**), we considered two populations: one with interactive thread development; and one with *independent thread development* (**Figure 4**), which allows full phenotypic expression of genes whose threads would be otherwise terminated by the EO of interactive threads. The process of gene expression in independent thread development is described in Section "Genome": a genome that specifies the same pin for

two different wires expresses both by shifting one of the wires to the next sequential unoccupied pin in that pin group.

Mating and Reproduction

Because of the time-consuming nature of experiments with physically embodied robots, we chose a population size of 10. While small, this population size is larger than that of previous work on evolving physical robots where selection for phototaxis could clearly act in spite of the presence of genetic drift (Long et al., 2006; Roberts et al., 2014; Livingston et al., 2016). Because of the concern that the small size of the population might eliminate or under-represent fit genomes in a standard roulette wheel mating algorithm, we used a simple ranking algorithm for choosing mating pairs.

Pairs of individuals are placed into five ranks by order of their fitness. Once the pair in the first rank has been crossed, these two individuals are moved to the second rank, where they join the two individuals there to form a mating pool of four. From that set of four individuals, two are randomly chosen to mate. Leaving the two unmated individuals in that rank, the two parents are then moved to the third rank, creating another pool of four individuals. This mate-and-move process continues until the fourth pair to mate moves to the fifth and final rank. After the fifth mating, those two parents are removed from the gene pool. Returning to the second rank, the highest remaining with individuals, the process continues with the mating of that pair and their movement to the third rank. This process continues until 10 offspring have been created.

The sexual reproduction algorithm recombines the two parental genomes (**Figure 5**). One parent is randomly chosen to start the process of replication. Its genome is copied until a crossover point shifts copying to the genome of the other parent. Once the recombined genome is produced, each of the 540 bases is subjected to mutation, with a 1/2,000 chance of mutating either the bit or crossover value of each base.

Evolutionary Trials

Three independent variables were manipulated. The first was the type of development: with or without an EO (see EO: Interactive Thread Development in this section). The second was selection: present or absent. The third was the nature of the crossing over: unconstrained (occurring anywhere in the genome) or constrained to positions between the genes that encode for threads. Our primary hypothesis was that an EO will alter the evolutionary trajectory of a population under selection (see **Figure 1**).

The task was phototaxis with obstacle avoidance. Each individual circuit phenotype was tested on an Ana BBot in a rectangular arena with a single light source and three barriers that prevented the robot from traveling straight to the light from its starting position (**Figure 6**). The robot carried a light data logger (Onset model HOBO) on its front, on top of the circuit board. The amount of light gathered over 2 minute was used as a direct measure of fitness for each individual. To control for degradation of hardware over the course of evolution, individuals of the two populations were tested in register with respect to generation and randomly within a generation. In addition, we normalized fitness using performance values from a hand-coded circuit, derived from Braitenberg's vehicle IIB (Braitenberg, 1986), that we ran each generation.

The task, environment, morphology, and fitness function combine to create a complex fitness landscape (**Figure 7A**). One virtue of simple Braitenberg-type vehicles is that their maximal performance is easy to predict, at least for circuits with just a few threads (**Figure 7B**). Since our primary goal was to examine the evolutionary impact of EOs, we sought to position our populations at a critical point on the fitness landscape; hence, populations with independent thread development (no EO) and with interactive thread development (EO) had their identical initial distributions of the number of threads with a mode of two threads and a range of one to three threads (**Figure 7C**). These genomes were randomly generated with a *post hoc* condition: at least one thread must connect a sensor to a motor. This screen was imposed because many genetically possible threads do not create a functional circuit; hence, we gave the populations, initially, mobility. As early as the third generation in both populations, some individuals lacked functional threads (details in Section "Results"). Because our mating algorithm (see Mating and Reproduction) kept these low-fitness individuals in the gene pool, they have an opportunity to mate. Thus, the experiments were run until mobility was lost. It is important to note that our goal was not to show adaptive evolution *per se* but rather to test the hypothesis that an EO can alter the evolutionary dynamics of a population of physically embodied robots.

During analysis of these selection experiments, five phenotypes in the second generation of the interactive thread development population were found to have threads prematurely terminated due to errors in the decoding process. Two of the five phenotypes lacked an additional motor connection, meaning their behavior, and thus their fitness, were possibly affected. This error was fixed in subsequent generations. We do not think that the error altered the evolutionary trajectory substantially, since the same trend is seen in the independent "no selection" trials.

We also ran trials without selection to test the effect of the EO without selection and to test the importance of constraining the crossover points to intergenic positions. Since these trials require no information about fitness (because there is no selection), they were run using the algorithms for mating, reproduction, and development.

For each statistical analysis, we ran a fully factorial analysis of variance (ANOVA) on SPSS (IBM, version 23). For all tests, the significance level was 0.05.

Simulated Robot

To complement the experiments on physical robots, we created and evolved simulated Ana BBots to further test the fundamental hypothesis that an EO will alter the evolutionary trajectory of a population. As with the physical robots, we select for enhanced phototaxis with object avoidance. We compare the evolutionary dynamics of populations of simulated robots with and without the interactive thread EO.

The simulated robots operated in a rectangular, walled enclosure modeled after the enclosure of its physical counterpart (**Figure 8**). The enclosure contained three obstacles between the robot and the light source. The robot was equipped with four

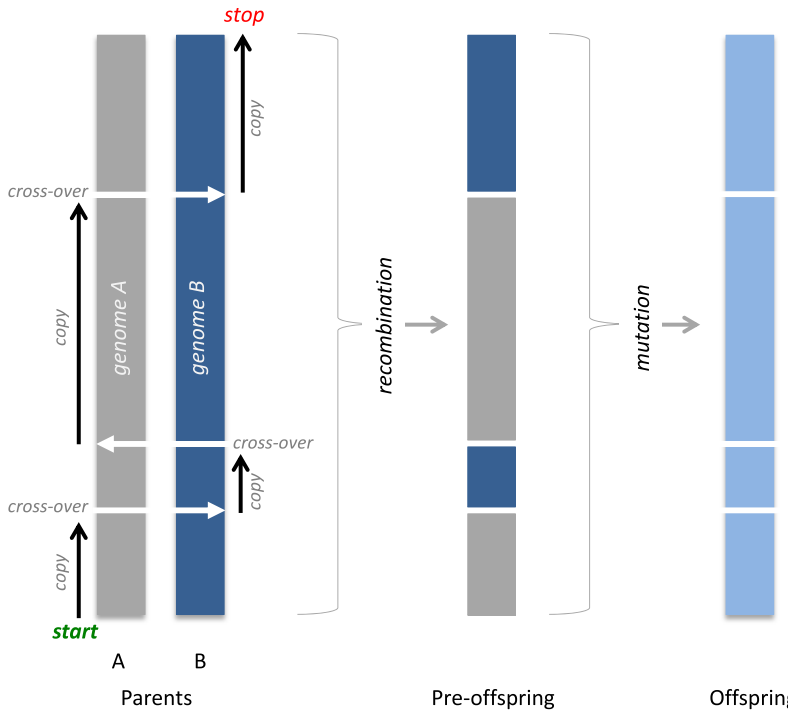


FIGURE 5 | Reproduction. After two parents are chosen for a given pairing (see text for details), the genome of one is randomly picked to start the process of replication with recombination and mutation. The genome is copied until a crossover point is reached. Copying then switches to the genome of the other parent, and so forth, until the end of one of the genomes is reached. In the pre-offspring genome, each of the 540 bases is put through a mutation operator that gives a 1/1,000 chance of change. The final genome of each offspring is a mosaic of two parental genomes further modified by mutation. As crossover points are genetically coded and hence transferable to the offspring and susceptible to mutation, they may be eliminated by evolution. Zero crossover points would, by this method of reproduction, create asexual reproduction, where only one parent's genome is copied as the template for the offspring.

sensors: two light sensors (placed at front left and front right) and two proximity sensors (placed at front left and front right). The intensity that a light sensor reads is $1/d^2$, where d is the distance from the sensor to the light source. Shadows of the obstacles were not simulated. The intensity that a proximity sensor reads is e, where e represents the length of a ray emitted by the sensor, determined by the first collision of the ray with an obstacle or the inside boundary of the arena. The robot had two wheels controlled by differential drive and an additional third caster wheel, used for balance.

Controller

The ANN created to control the simulated robot modeled that of the Ana BBot, with four sensor nodes as inputs, four hidden nodes, analogous to the Ana BBot's neurons, and two motor output nodes. Like the Ana BBot's wires, connections in the ANN could occur from the sensor inputs to the motor outputs, the sensor inputs to the hidden nodes, the hidden nodes to each other, and the hidden nodes to the motor outputs. At each time step, the sensor nodes were set to the raw values of the sensors without normalization or thresholding. The hidden and motor nodes were updated according to the following:

$$y_i^{(t)} = \tanh\left(y_i^{(t-1)} + \tau_i\left(\sum_j w_{ij} y_j^{(t-1)}\right)\right), \quad (3)$$

where $y_i^{(t)}$ denotes the value of the ith node at the t-th time step, τ_i denotes the time constant controlling the rate of change of the ith node (here all $\tau_i = 0.3$ following previous work), and w_{ij} denotes the weight of the connection from node j to node i.

Genotype-to-Phenotype Mapping

The G → P mapping scheme used in the physical experiment was re-implemented in simulation as faithfully as possible. Genomic parameters were maintained across the physical and simulated experiments. Genomes were encoded as strings of 560 bits. These genomes dictated where connections should be added to the ANN. The procedure for building threads was the same as used for the Ana BBot (see Genome), with connections in the ANN equivalent to wires in the Ana BBot.

Similar to the process in the Ana BBot, development can fail in the following conditions: (1) its target location is outside the boundary of the pins; (2) the target pin is already occupied; or (3) the target location equals the starting position, in which case no movement need occur. In trials with the EO and without (labeled "GO"), the process terminates if it experiences condition (1). In the EO trials, it also terminates when it experiences conditions (2) and (3). In the GO trials, the developmental process will attempt to find an empty pin on that row. If it can, it attaches there. If it cannot, it terminates.

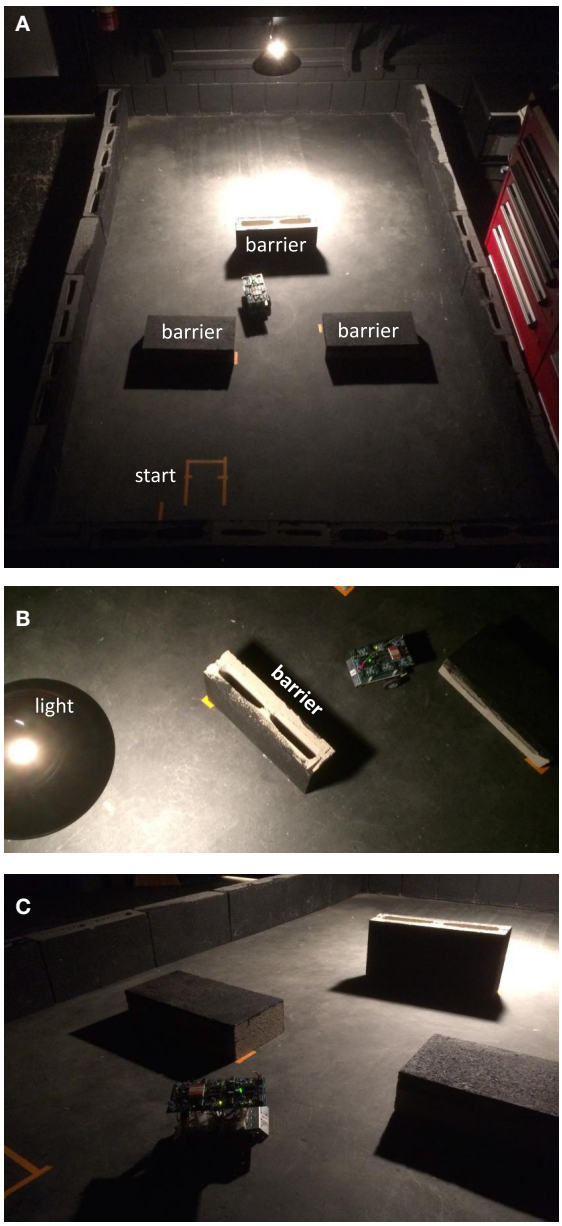


FIGURE 6 | Selection environment for phototaxis with obstacle avoidance. **(A)** The environment consisted of a rectangular arena (1.8 × 2.8 m) with the perimeter ringed with cinder blocks (0.2 m height) painted matte black. Barriers inside were likewise painted cinder blocks. A single 100 W incandescent light hung 70 cm above the floor over one end of the arena. The starting position was in the dark area, away from the light. **(B)** Barriers were positioned to prevent any straight path from the start to the light. **(C)** Barriers cast a shadow. From the perspective of the robot, the light gradient includes the shadows associated with obstacles.

The weight of a connection from neuron j to neuron i is set to the following:

$$w_{ji} = \sum_k p_{ik}, \tag{4}$$

if there are one or more wires traveling from pin row j to pin row i, and $w_{ij} = 0$ otherwise. p_{ik} denotes the weight of a wire that originates on pin row j and pin column k, and $p_{ik} = 0$, if there is no wire emanating from pin row j and pin column k. Only valid pin row pairs are considered, where a valid pin row pair is one that connects a sensor pin row to a hidden pin row, a sensor pin row to a motor pin row, a hidden pin row to another hidden pin row (including its own pin row), or a hidden pin row to a motor pin row.

Evolutionary Algorithm and Trials

To increase the generality of these findings, we employed the standard AFPO algorithm (Schmidt and Lipson, 2011) to evolve the controllers. Each evolutionary trial began with a population of 50 random bitstrings. Each was converted into a controller and embedded in the simulated robot. The robot was then evaluated four times from four different starting positions (**Figure 15**). Each evaluation lasted 300 time steps. After evaluation, fitness was calculated as follows:

$$f = \sum_{e=1\ldots4} \sum_{t=1\ldots300} \left(\mathrm{LP}_{et} + \mathrm{RP}_{et}\right), \tag{5}$$

where LP_{et} and RP_{et} denote the values of the left and right photosensors in the e-th environment at the t-th time step, respectively.

After all 50 controllers were evaluated, the dominated individuals were deleted using fitness and age as the two objectives (fitness is maximized while age is minimized). The population was filled back up to 49 individuals by randomly choosing a non-dominated individual, copying it, mutating it, and placing it in the population. The 50th slot was filled with a random bitstring and assigned an age of 0. The next generation was then conducted and continued until 50 generations had elapsed.

Two sets of 30 independent evolutionary trials consisting of 500 generations each were conducted using the EO operator and GO operator, respectively.

RESULTS

Physical Robots

Evolution of Fitness

The fitness of both populations of Ana BBots decreased significantly ($p = 0.009$) over generational time (**Figure 9A**) as determined by a 2 × 7 [Development Type (interactive, independent), Generation (1–7)] repeated-measures ANOVA. *A priori* contrasts detected a large and significant ($p = 0.012$) drop in fitness between generations 4 and 5; this drop in fitness may correspond to the predicted fitness "cliff" (see **Figure 7**). Counter to the hypothesis that the EO population should evolve differently under selection than the non-EO (also referred to as "GO") population, there was no significant difference in fitness or the rate of change in fitness between the two (**Figure 9B**). To determine whether selection was present, we measured the selection differential, S, as the difference in average fitness between those parents chosen to mate and the average fitness of the parental generation. Selection was present but decreasing in magnitude over generational time (**Figure 9C**). The variance in fitness also decreased over generational time (**Figure 9D**), indicating that the convergence of performance onto less mobile robots

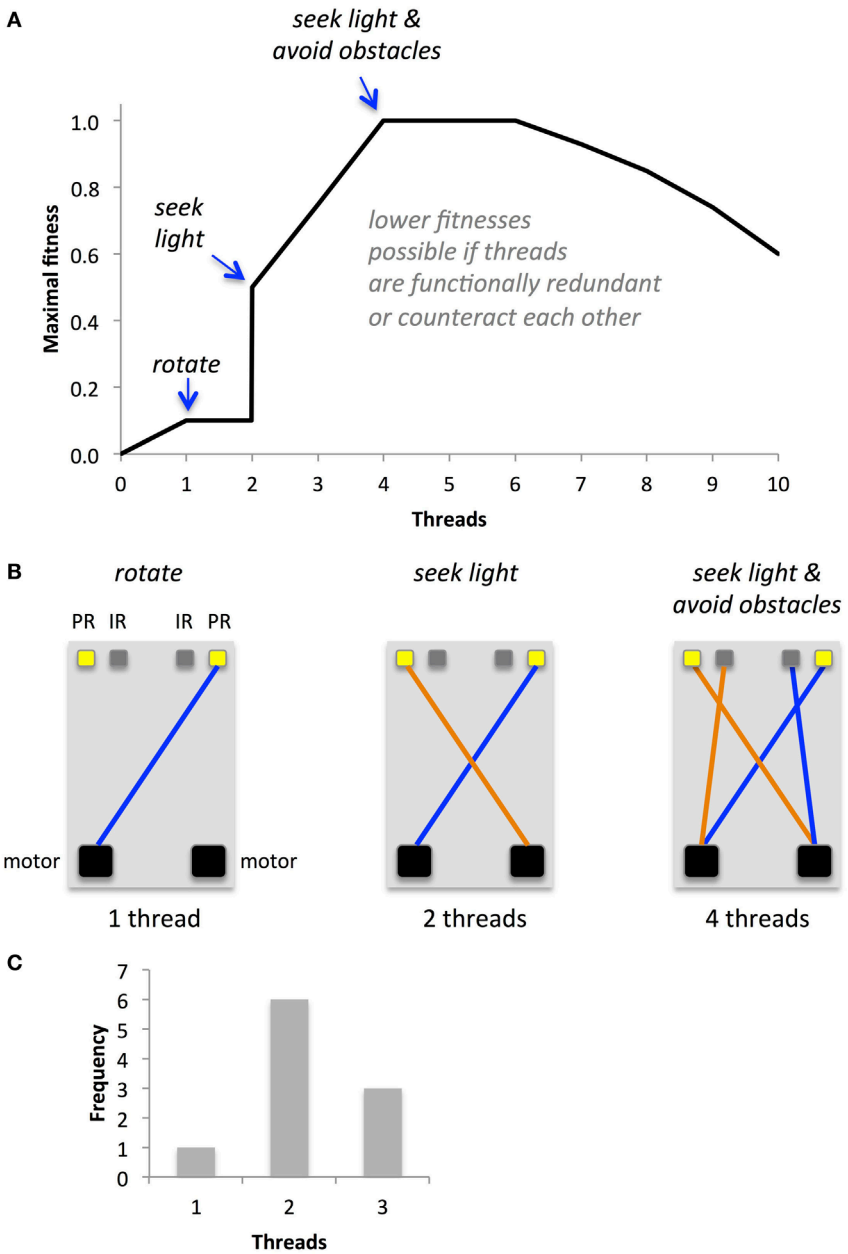


FIGURE 7 | **Predicted fitness landscape for Ana BBots**. Because threads connect sensors to motors, they should be directly proportional to fitness, which is measured as the amount of light gathered. **(A)** The maximal fitness (relative units) depends in a non-linear way on the number of threads. **(B)** Optimal thread configurations for maximal fitness at a given thread number. A single thread connecting a photoresistor (PR) to the motor may allow the robot to rotate toward the light source, allowing for a small increase in fitness. Two threads connecting the photoresistors may create the equivalent of a Braitenberg vehicle IIB; the robot will move forward and orient toward the light until, in this environment, it encounters an obstacle. Four threads connecting the both photoresistors and IR sensors (IR) to the motors may allow the robot to move toward the light and avoid obstacles along the way. For thread numbers of two or higher, lower than maximal fitnesses may occur if threads are redundant in their function or if their functions counteract each other. **(C)** The starting distribution of operational threads for the two populations under selection, with and without an epigenetic operator, was identical.

was collapsing the possibilities upon which selection could act. Variance was highly correlated with S for both the independent thread ($r = 0.982$) and interactive thread ($r = 0.974$) groups. The experiment was terminated when none of the robots showed mobility.

Evolution of Phenotypes

The hypothesis that the EO population should evolve differently under selection than the non-EO population was tested by examining four phenotypes that had a genetic basis: (1) the number of thread interactions, (2) the number of wire connections, (3) the

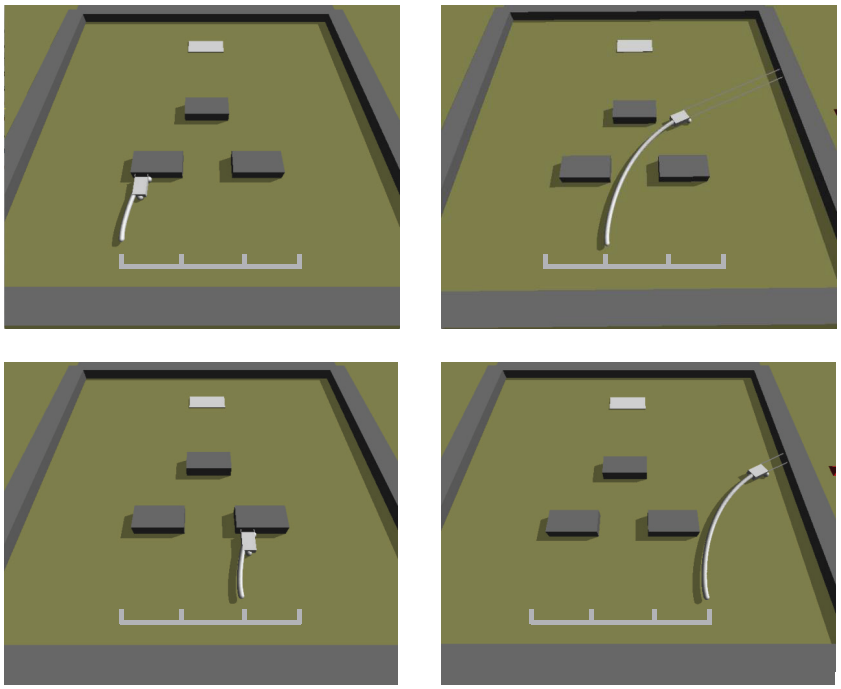


FIGURE 8 | Simulated Ana BBots. As with the physical Ana BBots, the selection environment is phototaxis with obstacle avoidance. Dark gray objects represent the obstacles; the light gray object in the background represents the light source. The curved white lines represent the trajectory of the robot (small gray rectangle) from its starting position to where it stopped. The grid in the foreground indicates the four possible starting positions. In selection experiments, each controller was evaluated four times from four different starting conditions, as shown here for the single best controller evolved without the interactive thread epigenetic operator.

number of threads, and (4) the number of crossover points. We tested the effects of selection, type of development, and generation with a 3 × 7 [Selection (selection, no-selection), Development Type (interactive, independent), Generation (1–7)] repeated-measures factorial analysis of variance (ANOVA). While 11 generations were run in the no-selection simulations, these data were truncated to 7 to facilitate comparison with the robot group.

In contrast with the results for fitness (see Evolution of Fitness), the phenotypes showed clear evidence of the impact of the EO development on evolution (**Figure 10**). For the number of thread interactions, there was a significant main effect of Selection (**Figures 10A,B**; $p < 0.001$), with the no-selection group having fewer interactions (M = 0.136) than the selection group (M = 0.750).

For the number of wires, there was a significant three-way interaction among Selection, Generation, and Development Type ($p < 0.05$; **Figures 10C,D**). This effect is due to the fact that there was a two-way interaction between Development Type and Generation for the no-selection condition ($p = 0.003$), but not one for the selection condition. Additionally there was a two-way interaction between Selection and Development Type on number of wires ($p < 0.001$) due to the interactive group having significantly smaller mean number of wires (M = 1.743, SD = 0.138) than the independent group (M = 3.257*)* in the no-selection condition, but there being no significant differences in the selection condition. There was a main effect of Development Type ($p = 0.003$) explained by the independent group having significantly more wires (M = 2.714) than the interactive group (M = 1.850) collapsed across Selection. Generation also had an effect on wire count ($p < 0.001$), with the tendency being that wire count decrease with successive generations. Additionally there was a main effect of Selection ($p < 0.001$), with the no-selection condition having significantly more wires (M = 2.500, SD = 2.064) than the selection condition (M = 2.064, SD = 0.097).

For the number of threads, there was a significant interaction between Selection, Generation, and Development Type ($p < 0.05$), with a significant interaction between Development Type and Generation in the no-selection condition (**Figure 10E**; $p < 0.001$), but not in the selection condition (**Figure 10F**; $p = 0.095$). Additionally there was a significant interaction between Generation and Development Type ($p = 0.025$) indicating that the trend of the interactive thread group's means was to decrease with increasing generation, whereas the independent thread group's means showed no clear trend in either direction. Development Type was also found to interact significantly with Selection ($p < 0.001$), explained by the interactive group's mean thread count being higher in the selection (M = 1.665) condition than in the no-selection condition (M = 1.000), whereas the independent group's means were lower in the selection condition (M = 1.771) than in the no-selection condition (M = 2.443). There was also a main effect of Development Type ($p < 0.001$), with the independent group on average having more threads (M = 2.107) than the interactive group (M = 1.329).

For the number of crossover points, there was a two-way interaction between Selection and Development Type (**Figures 11A,B**;

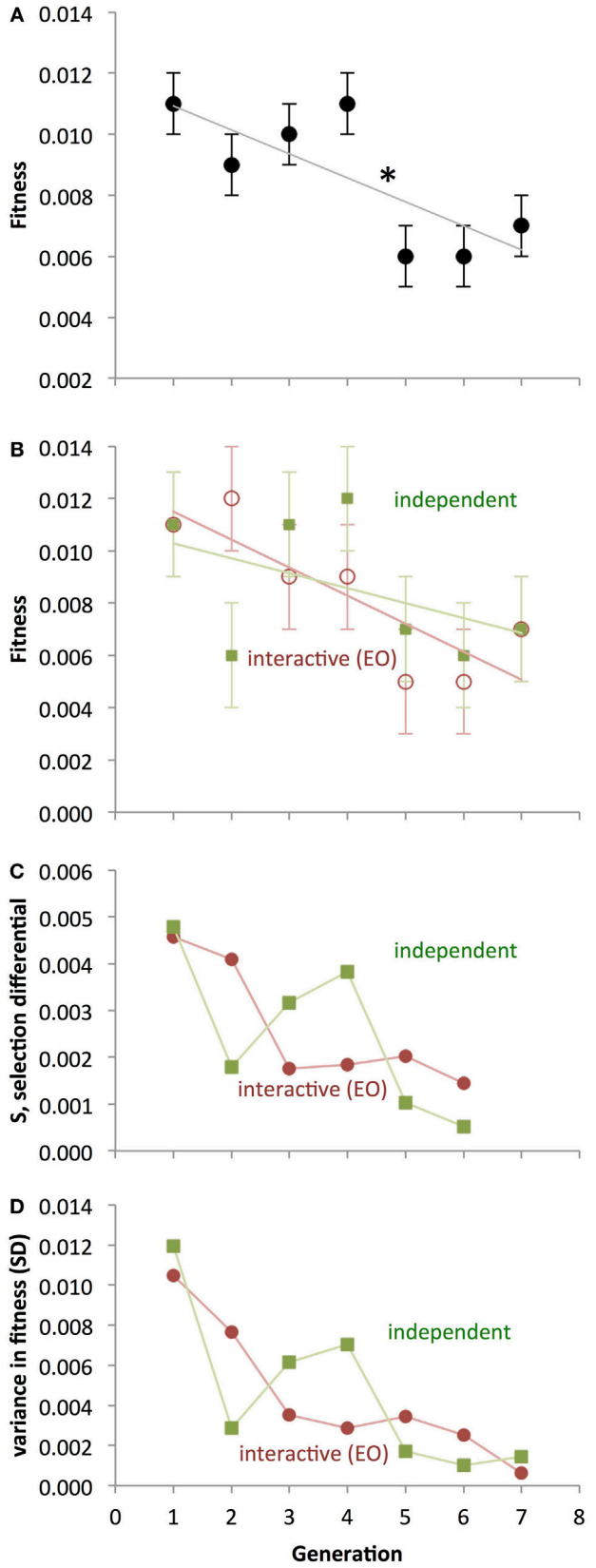


FIGURE 9 | Continued

FIGURE 9 | Continued
Evolution of physical Ana BBots under selection for enhanced phototaxis and obstacle avoidance. **(A)** As detected by ANOVA ($p < 0.05$), fitness decreases over generational time. A significant one-generation decrease between generations 4 and 5 (asterisk) is present as determined by *a priori* contrasts. Points are estimated marginal means ± 1 SE, with interactive and independent development pooled. **(B)** Interactive and independent thread developments are shown separately, even though they are not statistically distinct. **(C)** Selection differential, showing positive selection on fitness decreasing over generational time. The differential shown in a given generation is that applied to the next. **(D)** Variance in the populations, measured by SD, decreases by an order of magnitude over generational time. For the interactive development, the correlation between *S* and variance is 0.982; for independent development, the correlation is 0.974.

$p = 0.023$); also, there was a significant difference between the interactive and independent groups for the selection condition ($p < 0.001$), but not for the no-selection condition ($p = 0.715$). Selection and Generation also interacted significantly on number of crossover points, $p = 0.019$, with crossover points increasing more rapidly with Generation in the selection condition than in the no-selection condition. Additionally, Development Type had a significant effect on crossover ($p < 0.001$), explained by the independent group having significantly higher means (M = 4.407) than the interactive group (M = 3.786). Generation also had a significant effect ($p < 0.001$), with crossover points tending to increase with increasing generations. There was also a main effect of Selection ($p < 0.001$), with the selection group having more crossover points (M = 4.464) than the no-selection group (M = 3.729).

Given the importance of threads to the function of the robot, we had predicted a fitness landscape (see **Figure 7**). We see a precipitous decline in the number of threads in the EO population (interactive thread development) under selection from generations 2 to 3 (**Figure 10F**) that corresponds to a drop in fitness in the EO population under selection from generations 2 to 3 (**Figure 9B**). We do not see a similar drop in the number of threads or fitness under selection in the non-EO population (independent thread development) until generations 4 to 5. Also note that without selection the number of threads in the EO population plummets (**Figure 10E**). While these are qualitative results, they are important for three reasons: (1) the number of threads appears to be related to fitness, (2) the EO population responds differently than the non-EO population, and (3) selection changes the behavior of the EO population markedly.

To examine the evolution of threads in more detail, we examined the changes in their distribution patterns over generational time (**Figure 12**). Under selection, the distribution of the EO population changes more quickly than that of the non-EO population, with the two populations having overlapping distributions but different modes and skew after seven generations. Without selection, we see a similar rapid response of the EO population and different final distributions.

Crossover Point Constraints

To understand the importance of our decision to allow crossover points to be anywhere in the genome, we constrained crossover

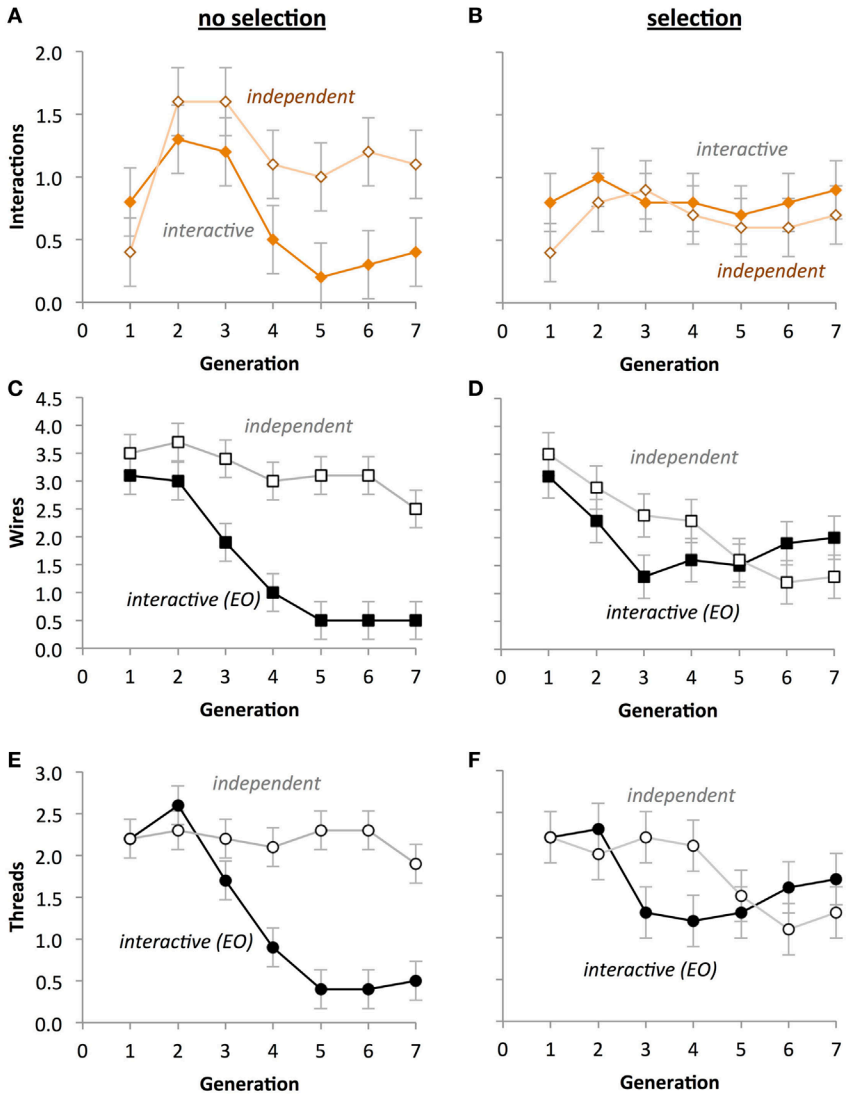


FIGURE 10 | Selection impacts the evolution of circuit phenotypes in physical Ana BBots. **(A,B)** A significant two-way interaction ($p < 0.05$) between the type of evolution and the type of development ($p < 0.05$) indicates that under selection **(B)**, the differences between developmental processes are eliminated. **(C,D)** A significant three-way interaction indicates that under selection, the number of wires in both types of development is not different and that the number of wires decreases over generational time. **(E,F)** A significant three-way interaction ($p < 0.05$) indicates that under selection the number of threads in both types of development is not different and that the number of threads decreases over generational time. A univariate three-way fully factorial ANOVA was run on each phenotype. Scale of the ordinate is identical across rows. Points are estimated marginal means ± 1 SE.

points to positions between genes. These trials were run without selection. A 2 × 2 × 11 [Crossover Placement (constrained, unconstrained), Development Type (interactive, independent), Generation (1–11)] repeated-measures factorial ANOVA was run on the following measures: normalized performance, mean number of thread interactions, mean number of wires, mean number of crossover points, and mean number of threads. The genomic parameters of the "constrained" population were identical to those of the "unconstrained" population with the exception that crossover points were constrained to the intergenic regions in the former and not the latter.

For the number of interactions between threads, a significant Development Type by Generation interaction ($p < 0.001$) indicated that interactions generally decreased across generational time under interactive thread development, but tended to increase with generational time under independent thread development (**Figures 13A,B**). A Crossover Placement by Development Type interaction ($p < 0.001$) indicates that independent development populations have significantly more interactions (M = 1.300) than interactive development populations (M = 0.600), but only under unconstrained crossover placement. In general, interactive development populations had significantly more interactions (M = 1.159) than interactive development populations (M = 0.841; $p < 0.001$).

For the number of wires, a significant three-way interaction between Crossover Placement, Development Type, and

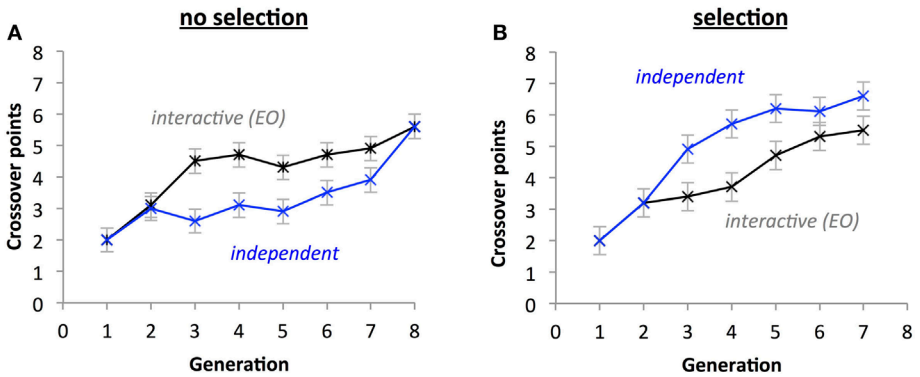


FIGURE 11 | Selection impacts the evolution of the number of crossover points. A significant three-way interaction ($p < 0.05$) indicates that under selection **(B)** and independent thread development, the number of crossover points increases faster than without selection **(A)** and with interactive development. Points are estimated marginal means ± 1 SE.

Generation on number of wires ($p < 0.05$) indicates that with constrained crossover placement the number of wires decreased more slowly for interactive thread development populations and increased less rapidly for independent thread populations (**Figure 13C**) when compared with unconstrained crossover placement (**Figure 13D**). There was a significant Development Type by Generation interaction ($p < 0.001$), indicating that wires tended to attenuate under interactive development, but stayed relatively stable under independent development. A significant Crossover Placement by Development Type interaction ($p < 0.05$) revealed a significantly larger gap in mean wires between independent development (M = 3.182) and interactive development (M = 1.218) under unconstrained crossover placement versus under constrained crossover placement (M = 3.364 and M = 2.400, respectively). The number of wires decreased with generation ($p < 0.05$), the independent group had significantly more wires (M = 3.273) than the interactive group (M = 1.809; $p < 0.05$), and that the mean number of wires was higher with constrained crossover placement (M = 2.882) than with unconstrained (M = 2.200; $p < 0.05$).

For the number of threads, a significant three-way interaction between Crossover Placement, Development Type, and Generation ($p < 0.001$) indicates that with interactive thread development, threads attenuated more rapidly under unconstrained crossover placement (**Figure 13F**) than constrained crossover placement (**Figure 13E**). A significant Development Type by Generation interaction ($p < 0.001$) reflects the tendency for thread count to decrease with generational time under interactive development, whereas thread count parabolically decreased then increased under independent development. There was also a Crossover Placement by Development Type interaction ($p < 0.001$), indicating that thread count is significantly higher under interactive development (M = 1.709) than independent development (M = 1.591) with constrained crossover placement, but significantly lower under unconstrained crossover placement (M = 1.055, M = 2.291, respectively). There was also a main effect of Generation ($p < 0.05$), reflecting the general upward parabolic change in thread count. Additionally, the independent development populations had significantly more threads (M = 1.941) than the interactive development populations (M = 1.382; $p < 0.05$).

For number of crossover points, a significant three-way interaction between Crossover Placement, Development Type, and Generation ($p < 0.05$) indicates that under unconstrained crossover, the number of crossover points generally increases faster with the interactive development group than with the independent group (**Figure 14A**), but this relationship switches with constrained crossover placement (**Figure 14B**). A significant two-way interaction between Development Type and Generation ($p < 0.05$) indicates that the independent development populations accrued crossover points more quickly than the interactive group. A significant Crossover Placement by Generation interaction ($p < 0.05$) indicates that populations with unconstrained crossover points accrue crossover points more rapidly than populations with constrained crossover points. In general, the number of crossover points increased across generations ($p < 0.05$) and populations with unconstrained crossover had more crossover points (M = 4.682) than constrained crossover populations (M = 3.041; $p < 0.05$).

Simulated Robots

A 2 × 500 [Development (EO, GO), Generation (1–500)] ANOVA revealed a main effect of Development ($p < 0.001$) and Generation ($p < 0.001$) on the fitness of the best individual in each population. There was no interaction effect ($p = 0.175$). These results (**Figure 15**) suggest that while the fitness of both types of simulated populations increased with successive generations, the EO significantly reduced fitness in comparison to the GO (non-EO) condition. Finally, note that while the fitness in the simulated populations increased over time, the fitness decreased over time in the physical populations (compare **Figures 9** and **15**).

DISCUSSION

In addition to the standard GOs of ER, EOs are a complementary class of mechanisms that alter the expression of the genome.

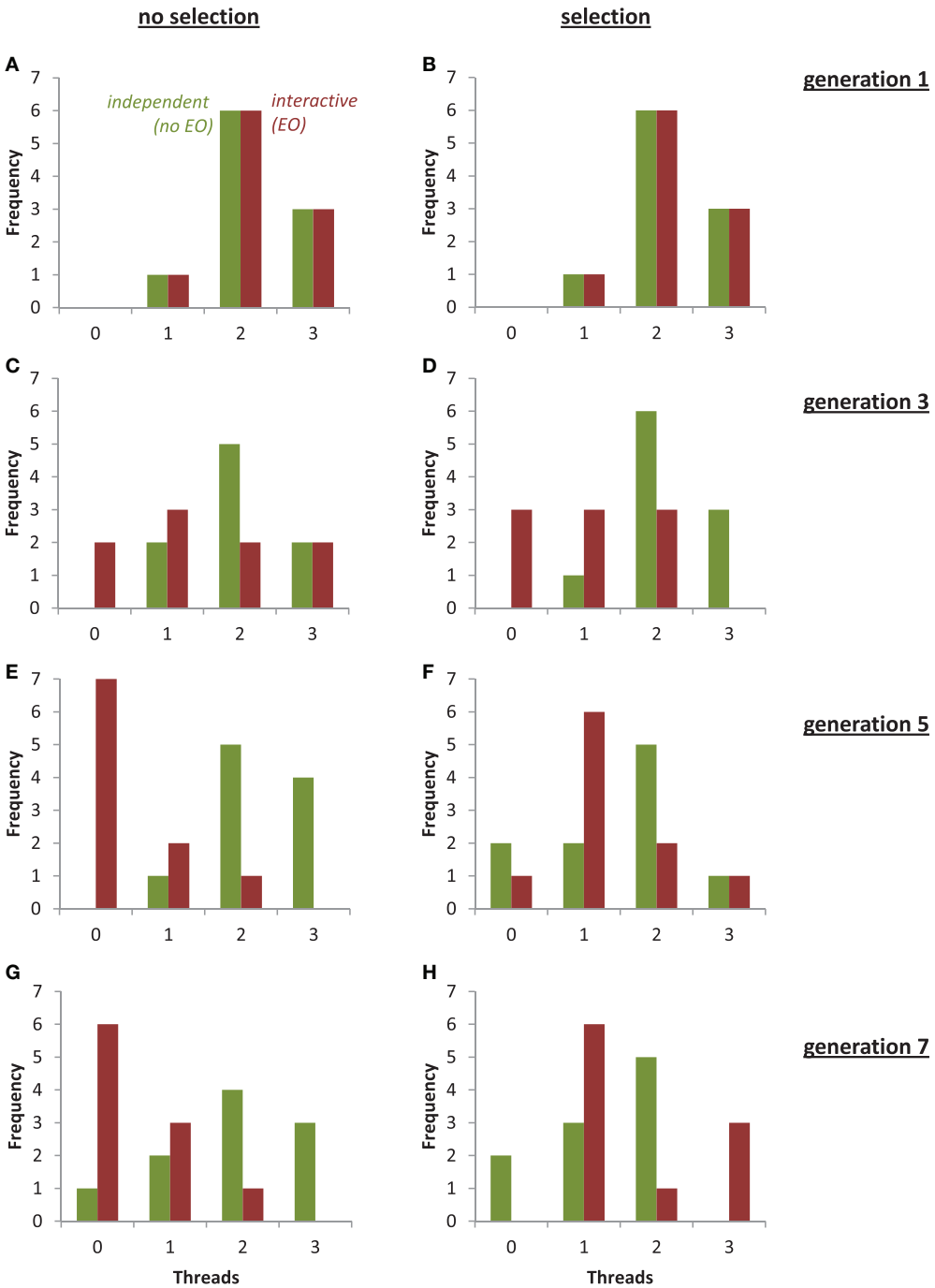


FIGURE 12 | Evolution of threads in the populations of physical robots under selection and with two different types of development. From generations 1 to 7, the mode of the population with independent development and no epigenetic operator (no EO, green) remained stable at two threads, with or without selection. From generations 1 to 7, the mode of the population with interactive development and an EO (red) changed from 2 to 0 without selection and from 2 to 1 with selection. For clarity, an individual with four threads in the EO population was omitted from **(D)**.

Physically embodied EOs, as we model them, may have important evolutionary consequences, as determined by the specific effect of the EO, the shape of the fitness landscape, and the position of the population on that landscape (**Figure 1**). With this model in mind, we hypothesized that an EO will alter the evolutionary trajectory of a population. This hypothesis is supported by the results of preliminary experiments in two populations of physically embodied robots, one with and one without an *interactive thread development* EO, under selection for enhanced phototaxis and obstacle avoidance.

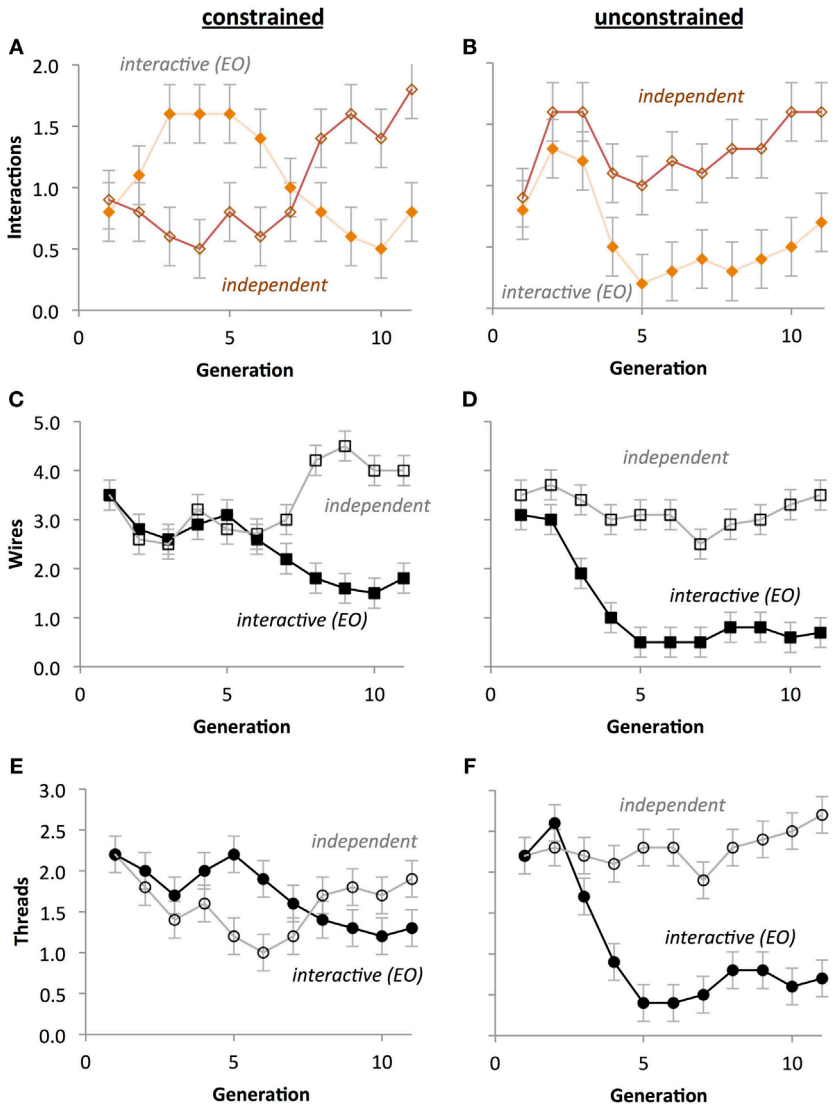


FIGURE 13 | Crossover placement impacts the evolution of circuit phenotypes without selection. **(A,B)** Compared to unconstrained crossover points, constrained crossover points cause the number of interactions to evolve in anti-phase oscillation with respect to type of development. **(C,D)** Compared to unconstrained crossover points, constrained crossover points cause the number of wires in both types of development to evolve in concert and then rapidly diverge. **(E,F)** Compared to unconstrained crossover points, constrained crossover points cause the number of threads to evolve in anti-phase oscillation with respect to type of development. Scale of the ordinate is identical across rows. Points are estimated marginal means ± 1 SE.

The generality of this result is extended by experiments run in simulation. The difference between the physical and simulated robots in terms of the direction of the change in fitness (decrease for physical, increase for simulated, **Figures 9** and **15**, respectively) may reflect different effective algorithms for information transmission in the neural networks. In the physical robots, signaling is implemented in hardware, while in the simulated case we used standard neural network updating models rather than trying to simulate the electronic components on the Ana BBot. However, the key point of comparison is between the EO (a.k.a. interactive thread development) and non-EO (a.k.a. independent thread development or GO) conditions; for both simulated and physical systems we find that the EO condition degrades the effect of selection on the evolution of fitness and phenotypes, respectively. In this critical comparison, the two approaches produce consistent results.

While the changes in mean fitness are statistically indistinguishable in the EO and non-EO populations of physical robots (**Figure 9**), mean phenotypic values diverge quickly (**Figures 10** and **11**). Because of its functional role, the key phenotype is the number of threads, where threads are the genetically encoded wiring patterns that may connect sensors to motors on the robot's physical circuit board (**Figure 4**). The distributions of thread number within each population are identical initially, in the first generation, but then they diverge rapidly (**Figure 12**). This rapid divergence occurs in EO populations both with and without

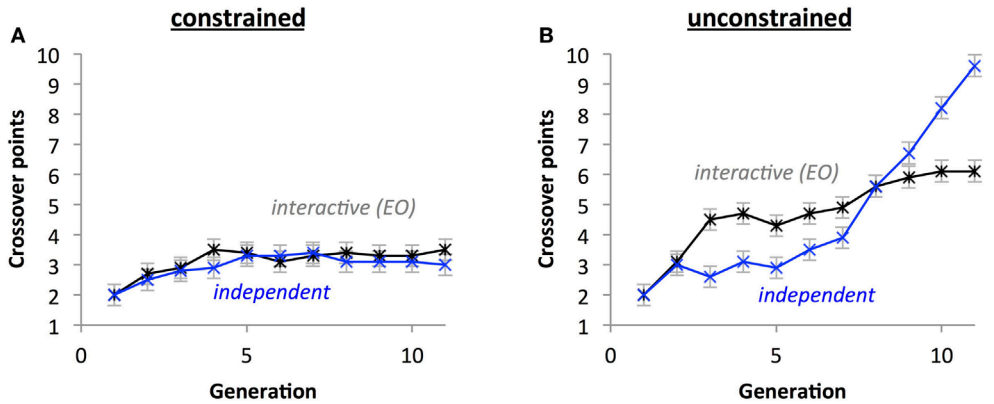


FIGURE 14 | Crossover placement impacts the evolution of the crossover phenotypes without selection. With crossover placement constrained **(A)**, the differences between types of development are eliminated and the growth of points over time is attenuated compared crossover placement being unconstrained **(B)**. Points are estimated marginal means ± 1 SE.

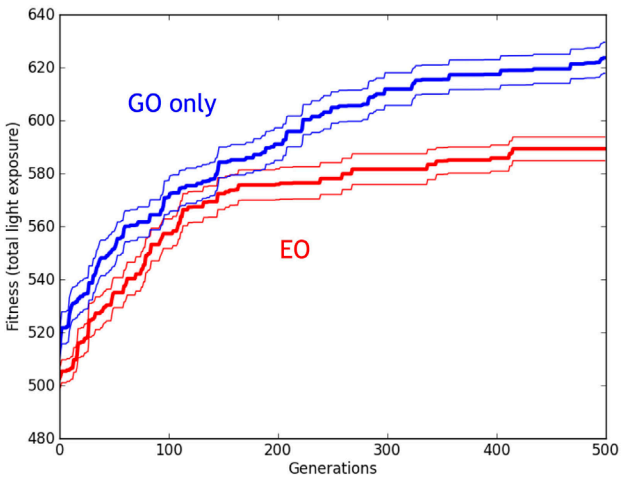


FIGURE 15 | Evolution of simulated Ana BBots with and without epigenetic operators (EOs). Relative performance of the robots with genetic operator (GO) (blue lines) and EO (red lines). Thirty evolutionary trials of GO and EO were performed, for a total of 60 runs, each lasting for 500 generations. While populations with both types of development increased in fitness over time, the GO populations had fitnesses that were significantly greater than the EO populations, as detected by ANOVA ($p < 0.05$). Thick lines indicate the mean fitness of the best individual in the population, averaged across the trials. Thin lines indicate ±1 SE.

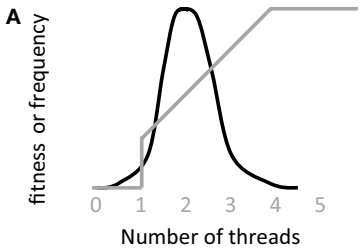
selection, which indicates that the EO effect is not selection in disguise; moreover, the effects of EO and selection interact, in a statistical sense, which provides additional evidence supporting the main hypothesis (compare **Figure 10E** and **Figure 10F**). In this instance, differences in the evolution under selection of two otherwise identical populations appear to be caused by the embodied EO of interactive thread development (**Figure 16**).

From simulations, we have a complementary perspective from populations that are larger and evolve for much longer than those in the physical robots. Importantly, the hypothesis that an EO will alter the evolutionary trajectory of a population is upheld. In simulation, otherwise identical populations of Ana BBots with EOs evolve fitness more slowly and with lower magnitudes of fitness than those lacking them (**Figure 15**).

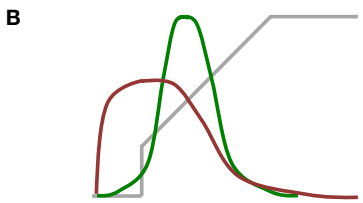
The populations of simulated Ana BBots with interactive thread EOs—a destructive process predicted to reduce the number of threads (**Figure 1**)—are less evolvable than populations without them. Although both the EO and GO treatments allow for large amounts of neutral mutation, which has been cited as a contributor to increased evolvability (Smith et al., 2001; Wagner, 2008), the GO treatment may allow for more connections to be constructed between the sensor and motor layers, or perhaps for more efficient, and less self-interfering, networks. This may in turn provide more raw materials for subsequent evolutionary change. By contrast, the EO treatment may produce fewer overall connections between sensor and motor layers, which may in turn make any subsequent mutations that change the nature of this path more disruptive. Future work will involve more detailed analysis of how such pathways in both treatments do change—or fail to change—over evolutionary time.

We note that the evolutionary impact of the interactive thread EO depends on the shape of the fitness landscape and the location of the populations on that landscape (**Figure 7**). For example, if we shifted the starting populations to the right on the fitness landscape, more threads result in a loss of fitness as threads are redundant in their function or if their functions counteract each other. In this situation, the interactive thread EO would increase the fitness of the populations by pruning threads. Thus the identical EO can have opposite effects on evolution depending on where a population sits in a particular fitness landscape.

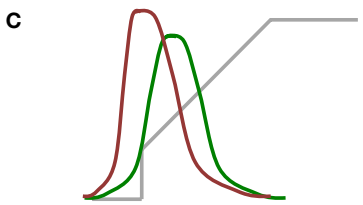
As we have narrowly defined it, an EO may be any mechanism inherent to an agent's developmental system that alters the expression of the genome. This leaves investigators with a daunting array of EOs from which to choose. To avoid an arbitrary decision, we let the physical embodiment of the Ana BBot (**Figure 2**) guide us. This is an analog robot that is programed using jumper wires to connect sensors to motors (**Figure 3**). With this in mind, we created a genome that encodes genes that governed multiple, separate wiring patterns called threads (**Figure 4**). Since only



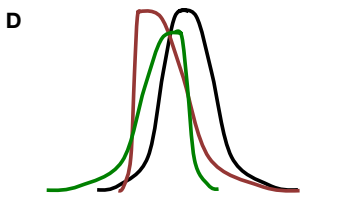
Generation 1. Populations with and without EO start with identical distributions and at the same position in the fitness landscape (gray line). The mode is 2, with a range of 1 to 3.



Generation 3. After two generations of selection, the mode of the EO population (red) has decreased to 1 and evolved to have a range of 0 to 4 (mode = 1). The non-EO population (green) remains unchanged.



Generation 5. Both populations have evolved. The EO population has increased the number of individuals with a modal value of 1. The range of the non-EO population has increased to 0 to 3 while the mode is unchanged at 2.



Generation 7. The modes of the population differ, with a value of 1 and 2 for the EO and non-EO populations, respectively. The distributions are skewed in opposite directions. No individuals possess 0 threads in the EO population. Both populations are different from the original (black curve).

FIGURE 16 | The epigenetic operator (EO) of interactive thread development alters evolution. In this qualitative summary of the experiments with physically embodied Ana BBots, the two genetically identical populations start with identical distributions in terms of the number of threads **(A)**. Exposed to different parts of the fitness landscape (gray line), the two populations quickly diverge from each other **(B–D)**. See **Figure 12** for actual distributions.

a limited set of pins is available for threads, we created an EO, interactive thread development, that recognized this physical constraint: if different genes code for the same pin, then only the first gene expressed may use it and the expression of the other gene is left incomplete. In development without this EO, all genes are fully expressed by allowing threads that call for the same pin to switch to an alternate and functionally equivalent pin.

We recognize that this developmental system is extremely simple, particularly when compared to one that changes full-body morphology in simulated, embodied mobile robots (Bongard, 2011). But to our knowledge, interactive thread development is the first physically embodied EO used in the evolution of physically embodied robots. Thus the developmental engine of ER (Eiben et al., 2010) has its first physical instantiation, albeit a simple one; physical instantiation is a necessary condition for the complete life cycle of the evolution of things (Eiben and Smith, 2015) and in the physical evolution of ontogenies (Northcutt, 2002).

Embodied development means different things in different contexts. For researchers interested in social and socially assistive robots (Tapus et al., 2007), for example, development focuses on learning and interactive changes in an agent's cognition (Asada et al., 2001). Others focus on life cycle changes in an agent's morphology (Jin and Meng, 2011; Doursat et al., 2012). For researchers interested in ER, changes in morphology during a digitally simulated embodied agent's lifespan dramatically alter the impact of selection on the evolution of behavior (Bongard, 2011). To bridge the reality gap between the simulation of morphological changes and the physical instantiation of those changes, we can incorporate the methods of reconfigurable robots (Levi et al., 2014), self-assembling swarms (Rubenstein et al., 2014), and programmable matter (Toffoli and Margolus, 1991; Felton et al., 2014) to create PED. Combining PED with ER creates an approach, ERPED, exemplified in a preliminary and simple manner in this study.

CONCLUSION

We have shown that an EO alters the evolution of populations of physical and simulated embodied robots under selection for enhanced phototaxis and object avoidance. While we must be cautious in drawing general conclusions from this preliminary result, the specific method employed is easily extended to other physically embodied robotic systems. Necessary to this extension is to make development explicit, genetic, and physical. When the expression of genes is altered by the physical rules and interactions governing the agent's physical construction, the genotype-to-phenotype mapping process becomes available as a creative tool to ER.

AUTHOR CONTRIBUTIONS

JBr, AH, and KL conceived of and designed the experiments. JBr and AH programed the genetic and epigenetic systems, and validation of the code was overseen by EA and JL. JBr and AH ran the experiments and reduced the data on the physical robots. JL analyzed the results. JBr and JBo designed, built, ran, and analyzed the simulation. All the authors contributed to the writing of the manuscript.

ACKNOWLEDGMENTS

Larry Doe was invaluable to the maintenance and modification of the Ana BBot. John Connell of Johuco Ltd. developed the Ana BBot and provided high-quality schematics. Nick Livingston provided invaluable assistance in the development of the project and revision of the manuscript. Anton Bernatskiy provided critical assistance in developing the simulation.

FUNDING

This work was funded by the U.S. National Science Foundation (grant no. 1344227, INSPIRE, Special Projects).

REFERENCES

Amundson, R. (2005). *The Changing Role of the Embryo in Evolutionary Thought: Roots of Evo-Devo*. Cambridge, UK: Cambridge University Press.

Asada, M., MacDorman, K. F., Ishiguro, H., and Kuniyoshi, Y. (2001). Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Rob. Auton. Syst.* 37, 185–193. doi:10.1016/S0921-8890(01)00157-9

Bongard, J. C. (2002). "Evolving modular genetic regulatory networks," in *Proceedings of the 2002 Congress on Evolutionary Computation*, Honolulu, HI.

Bongard, J. C. (2011). Morphological change in machines accelerates the evolution of robust behavior. *Proc. Natl. Acad. Sci. U.S.A.* 108, 1234–1239. doi:10.1073/pnas.1015390108

Bongard, J. C., and Pfeifer, R. (2003). "Evolving complete agents using artificial ontogeny," in *Morpho-Functional Machines: The New Species*, ed. F. Hara (Japan: Springer), 237–258.

Braitenberg, V. (1986). *Vehicles: Experiments in Synthetic Psychology*. Cambridge: MIT Press.

Carroll, S. B. (2008). Evo-devo and an expanding evolutionary synthesis: a genetic theory of morphological evolution. *Cell* 134, 25–36. doi:10.1016/j.cell.2008.06.030

Doursat, R., Sayama, H., and Michel, O. (eds). (2012). *Morphogenetic Engineering: Toward Programmable Complex Systems*. Berlin: Springer.

Eiben, A. E., Haasdijk, E., and Bredeche, N. (2010). "Embodied, on-line, on-board evolution for autonomous robotics," in *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*, Vol. 7, 361–382.

Eiben, A. E., and Smith, J. (2015). From evolutionary computation to the evolution of things. *Nature* 521, 476–482. doi:10.1038/nature14544

Felton, S., Tolley, M., Demaine, E., Rus, D., and Wood, R. (2014). A method for building self-folding machines. *Science* 345, 644–646. doi:10.1126/science.1252610

Floreano, D., Dürr, P., and Mattiussi, C. (2008). Neuroevolution: from architectures to learning. *Evol. Intell.* 1, 47–62. doi:10.1007/s12065-007-0002-4

Garstang, W. (1922). The theory of recapitulation: a critical re-statement of the biogenetic law. *J. Linn. Soc. Lond. Zool.* 35, 81–101. doi:10.1111/j.1096-3642.1922.tb00464.x

Gilbert, C. D., and Wiesel, T. N. (1992). Receptive field dynamics in adult primary visual cortex. *Nature* 356, 150–152. doi:10.1038/356150a0

Gruau, F. (1994). Automatic definition of modular neural networks. *Adapt. Behav.* 3, 151–183. doi:10.1177/105971239400300202

Jin, Y., and Meng, Y. (2011). Morphogenetic robotics: an emerging new field in developmental robotics. *IEEE* 41, 145–160. doi:10.1109/TSMCC.2010.2057424

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex Syst. J.* 4, 461–476.

Levi, P., Meister, E., and Schlachter, F. (2014). Reconfigurable swarm robots produce self-assembling and self-repairing organisms. *Rob. Auton. Syst.* 62, 1371–1376. doi:10.1016/j.robot.2014.07.001

Livingston, N., Bernatskiy, A., Livingston, K., Smith, M., Schwarz, J., Bongard, J., et al. (2016). Modularity and sparsity: evolution of neural net controllers in physically embodied robots. *Front. Robot. AI* 3:75. doi:10.3389/frobt.2016.00075

Long, J. H., Koob, T. J., Irving, K., Combie, K., Engel, V., Livingston, N., et al. (2006). Biomimetic evolutionary analysis: testing the adaptive value of vertebrate tail stiffness in autonomous swimming robots. *J. Exp. Biol.* 209, 4732–4746. doi:10.1242/jeb.02559

Mattiussi, C., Marbach, D., Dürr, P., and Floreano, D. (2008). The age of analog networks. *AI Mag.* 29, 63.

Northcutt, R. G. (2002). Understanding vertebrate brain evolution. *Integr. Comp. Biol.* 42, 743–756. doi:10.1093/icb/42.4.743

Pfeifer, R., and Bongard, J. (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. Cambridge: MIT Press.

Pfeifer, R., Lungarella, M., and Iida, F. (2007). Self-organization, embodiment, and biologically inspired robotics. *Science* 318, 1088–1093. doi:10.1126/science.1145803

Pigliucci, M. (2010). Genotype-phenotype mapping and the end of the 'genes as blueprint' metaphor. *Biol. Sci.* 365, 557–566. doi:10.1098/rstb.2009.0241

Roberts, S. F., Hirokawa, J., Rosenblum, H. G., Sakhtah, H., Gutierrez, A. A., Porter, M. E., et al. (2014). Testing biological hypotheses with embodied robots: adaptations, accidents, and by-products in the evolution of vertebrates. *Front. Robot. AI* 1:12. doi:10.3389/frobt.2014.00012

Rubenstein, M., Cornejo, A., and Nagpal, R. (2014). Programmable self-assembly in a thousand-robot swarm. *Science* 345, 795–799. doi:10.1126/science.1254295

Schmidt, M., and Lipson, H. (2011). "Age-fitness pareto optimization," in *Genetic Programming Theory and Practice VIII* (New York: Springer), 129–146.

Smith, T., Husbands, P., and O'Shea, M. (2001). "Neutral networks and evolvability with complex genotype-phenotype mapping," in *European Conference on Artificial Life* (Berlin, Heidelberg: Springer), 272–281.

Stanley, K. O. (2007). Compositional pattern producing networks: a novel abstraction of development. *Genet. Programm. Evol. Mach.* 8, 131–162. doi:10.1007/s10710-007-9028-8

Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artif. Life* 15, 185–212. doi:10.1162/artl.2009.15.2.15202

Tapus, A., Mataric, M. J., and Scassellati, B. (2007). Socially assistive robotics. *IEEE Robot. Auto. Mag.* 14, 35. doi:10.1109/MRA.2007.339605

Toffoli, T., and Margolus, N. (1991). Programmable matter: concepts and realization. *Phys. D* 47, 263–272. doi:10.1016/0167-2789(91)90296-L

Wagner, A. (2008). Neutralism and selectionism: a network-based reconciliation. *Nat. Rev. Genet.* 9, 965–974. doi:10.1038/nrg2473

Wagner, G. P., and Altenberg, L. (1996). Perspective: complex adaptations and the evolution of evolvability. *Evolution* 50, 967–976. doi:10.2307/2410639

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Advantages of publishing in Frontiers

OPEN ACCESS
Articles are free to read for greatest visibility and readership

FAST PUBLICATION
Around 90 days from submission to decision

HIGH QUALITY PEER-REVIEW
Rigorous, collaborative, and constructive peer-review

TRANSPARENT PEER-REVIEW
Editors and reviewers acknowledged by name on published articles

Frontiers
Avenue du Tribunal-Fédéral 34
1005 Lausanne | Switzerland

Visit us: www.frontiersin.org
Contact us: info@frontiersin.org | +41 21 510 17 00

REPRODUCIBILITY OF RESEARCH
Support open data and methods to enhance research reproducibility

DIGITAL PUBLISHING
Articles designed for optimal readership across devices

FOLLOW US
@frontiersin

IMPACT METRICS
Advanced article metrics track visibility across digital media

EXTENSIVE PROMOTION
Marketing and promotion of impactful research

LOOP RESEARCH NETWORK
Our network increases your article's readership