Emergence of Tool Construction in an Articulated Limb Controlled by Evolved Neural Circuits

Randall Reams, Yoonsuck Choe
Department of Computer Science and Engineering
Texas A&M University
College Station, TX
{reams, choe}@tamu.edu

***Abstract*— Tool construction requires sophisticated cognitive function and is only observed in higher mammals and a few avian species. In this paper, we will examine the spontaneous emergence of tool construction during the simulated evolution of a two-degree-of-freedom articulated limb controller in a reaching task environment. The limb controller is a recurrent neural network with a topology evolved using the NeuroEvolution of Augmenting Topologies (NEAT) algorithm. First, we show how broad fitness criteria such as distance to target, number of successful reaches, number of steps to reach the target, and number of instances holding the correct length tool are enough to give rise to tool construction. Second, we analyze how the number of tools and their location in the environment during evolution affect the evolved neural circuits' ability to detect tool affordances and employ the optimal decision strategy. We expect our results to help us understand the implications of tool use capability and the environmental conditions that may facilitate its development.**

I. Introduction

A. Background

The use of tools in animals indicates a high level of cognitive capability, requiring a detailed understanding of the causal relations in the environment, complex planning, and learning through trial and error [1]. Consequently, tool use behavior is only observed in a small number of higher mammals and avian species [2][3][4][5]. Associative tool use, tool use and tool construction incorporating more than one tool to solve a task, is only seen among the great apes [6].

Chung and Choe demonstrated that simple neural circuits can be evolved to use the simplest form of tool, i.e. a "bread-crumb" dropped in the environment to serve as external memory [7]. In previous work, we evolved neural circuits controlling a simulated two-degree-of-freedom articulated limb to reach distant objects, having access to a simulated reaching tool to extend the range of the limb upon pickup [8]. We also showed that broad fitness criteria such as distance to target, number of steps to reach the target, number of successful reaches, and the difference between the number of required and actual tool pickups were sufficient to evolve networks that implement near-optimal decision strategy [9].

In this paper, we investigate how similarly broad fitness criteria can be used to evolve neural circuits capable of a simple form of associative tool use, combination-based tool construction. This behavior requires simultaneous or sequential access to two reaching tools capable of being combined to form one reaching tool of longer length. In either case, the tool that is first picked up can be used independently to reach the target, or it can be used as a secondary tool, a tool used to construct (structurally modify) another tool. In general, different modes of construction are possible, including detachment, material subtraction, material addition, combination, and reshaping. In this paper we are concerned with arguably the simplest of these modes, combination, in which the secondary tool is used to combine with another tool forming a tool with length equal to the sum of its constituent lengths. Because the secondary tool is used to reach and acquire as well as to combine with the constructed tool, it can also be considered a sequential tool.

Furthermore, we investigate how, during evolution, variation in the number of tools and their placement within the environment, and consequently whether they afford use as secondary tools, affects the evolved neural circuits' capability to detect tool affordances and employ the optimal decision strategy.

B. Related Work

Tool use has been extensively studied in artificial intelligence and robotics [10], and the existing work can be grouped into four categories:

1. Programmed, hard-coded behavior [11]
2. Learning through demonstration[12] [13][14][15][16]
3. Learning through random trial-and-error [17][18][19]
4. Evolved tool use behavior [20][7][8][9] and evolved body morphology [21]

Most of these works required some degree of designer knowledge regarding tool use and motor control, spanning fully hard-coded behavior, already attached tools, predefined tool features, and predefined motor primitives. The evolution-based approaches were relatively free of such constraints.

Note: This is a preprint. Some content may be slightly different from the final version. See https://ieeexplore.ieee.org/abstract/document/7965913 for the final version.

II. APPROACH

In this section, we introduce the NEAT neuroevolution algorithm, specify the variants of the reaching task environment in which our experiments took place, and detail how the neural circuits are integrated with the environment. Finally we define the fitness criteria used for evolution across all tested reaching task environment specifications.

A. Algorithm for Evolving Neural Circuits

Unlike standard neuroevolution algorithms, the NeuroEvolution of Augmenting Topologies (NEAT) algorithm developed by Stanley and Miikkulainen evolves both the connection weights and the network topology over time, achieving behavioral complexity through gradual topological complexification [22]. NEAT has been shown to be effective in evolving complex, non-trivial behavior in various learning environments [23][24].

The strength of NEAT is in the way it uses speciation to protect topological innovations. Unlike connection weight mutations, topological mutations such as the addition of a neuron in the middle of a connection or adding a connection between two neurons are unlikely to immediately confer a fitness advantage and are more likely to confer an initial disadvantage. To keep these innovations from being eliminated immediately during selection, they are protected through speciation and explicit fitness sharing, made possible by the labeling of genes with innovation numbers which provide a means for determining how compatible two chromosomes are by measure of how many topological structures they have in common.

The overall operation of the NEAT algorithm proceeds as follows: the recurrent neural network phenotype is instantiated from the genotype, it is tested on the task environment, and its fitness is calculated. When finished for all chromosomes, selection and reproduction occur, and the cycle repeats for the next generation.

B. Reaching Task Environment

The task is to control a two-limbed articulated arm to reach an object. It is a 2 degree of freedom arm that moves on a 2-D plane. The target object can appear both within and outside reach of the arm. The environment is equipped with one or two tools that can be picked up, and in the latter case combined, and used to reach objects beyond the reach of the arm.

As depicted in Figure 1, all task variants involve an arm composed of an upper limb of length l_1 = 80 attached to a fixed shoulder joint around which it can be rotated from -150° to 150° and a lower limb of length l_2 = 100 hinged to the upper limb at an elbow joint rotatable from -150° to 150°

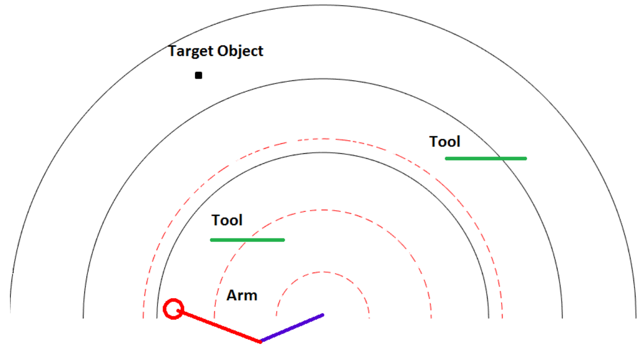


Fig. 1. Sample task environment. A two-degree-of-freedom articulated limb can move freely on a 2-D plane containing reaching tools and a target object. The three solid semicircles of increasing radii depict the farthest reachable distance with no tool, one tool, and both tools combined, respectively. The three dashed semicircles of increasing radii depict the closest reachable distance with no tool, one tool, and both tools combined, respectively.

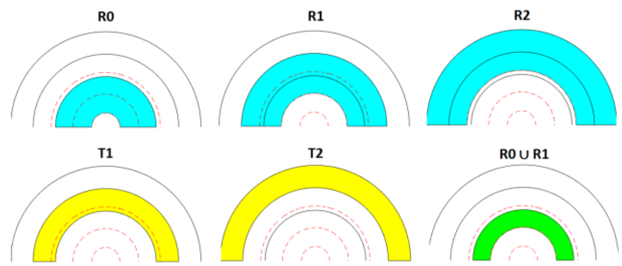


Fig. 2. Regions of interest. The regions of the environment that are reachable without a tool, with a tool, and with both tools combined are represented by the shaded areas labeled R0, R1, and R2, respectively. T1 is the region in which one tool is required to reach a target, and T2 is the region in which both tools are required to reach a target. Finally, the region that is reachable both with and without a tool is depicted in the bottom-right.

relative to the upper limb. The joint angles θ_1 and θ_2 (see Figure 4b) can be changed by up to 1.5° each time step, thereby moving the end effector. When the end effector reaches the base of a tool, it automatically picks up the tool or combines it with the tool it's already holding, collinearly extending the second limb by the length of the tool, l_t = 80.

We have defined regions *R0*, *R1*, and *R2* to be the areas in the environment that are reachable under each tool-holding condition, holding no tool, holding one tool, and holding two tools combined into one longer, constructed tool. Region *T1* is the area in which a tool or target could appear that would require holding one tool to be reached, including area also reachable while holding the constructed tool. Region *T2* is the area that is only reachable while holding the constructed tool. Finally, region *R0* ∩ *R1* is the area that is both reachable while not holding a tool and while holding a non-constructed tool.

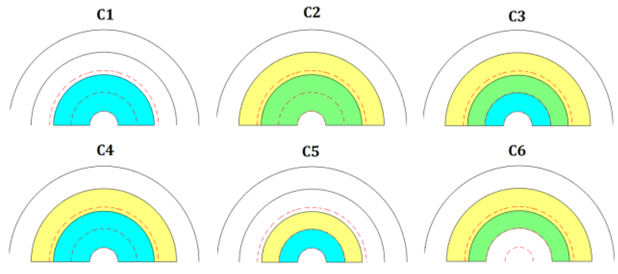


Fig. 3. Task variants. C1-C6 depict the six examined task variants. The cyan and yellow shaded areas depict where the first and second tools appear, respectively, with green depicting overlap.

The simplest task variant, condition *C1*, is defined such that the target object can appear in *R1* and one tool can appear in *R0*. Task variant *C1* can be solved given any initial conditions and does not require tool construction.

The remaining task variants differ in that the target object can appear anywhere in *R0* ∪ *T1* ∪ *T2* and in that two tools always appear in the environment. In task variant *C2*, one tool can appear anywhere in *R0*, and the other tool can appear anywhere in *R0* ∪ *T1*. *C2* is unique in that it is the only task variant that is not always solvable. For example, the target could appear in *T2* with both tools located in *R0* - *R0* ∩ *R1*. In this case, neither tool affords use as a secondary tool because its pickup makes the remaining tool out of reach. Task variant *C3* is defined such that every initial configuration is solvable, but it may become unsolvable if the neural circuit employs suboptimal strategy. In *C3*, one tool appears in *R0*, and the other tool appears in *R1*. Because of the overlap, there are some instances where the order in which the two tools are picked up determines whether the problem can be solved. In these cases, one tool affords use as a secondary tool whereas the other does not.

Task variant *C4* differs from *C1* and *C2* in that the order in which the tools may be picked up is fixed, so picking up one tool will never prohibit the arm from reaching the other tool. This is accomplished by having the first tool appear in *R0* and the second tool appear in *T1*.

Task variant *C5*, in which one tool appears in *R0* – *R0* ∩ *R1* and the other in *R0* ∩ *R1*. This configuration ensures that only one tool, the one appears in *R0* – *R0* ∩ *R1*, closer to the shoulder joint, affords use as a secondary tool. Thus if the target appears in *T2*, the order in which the tools are picked up determines whether it can be reached. Preliminary results from evolving circuits on *C5* suggested the tendency to adopt a strategy that performs well on *C5* but generalizes poorly to other task variants, namely avoiding picking up a tool in region *R0* ∩ *R1* if not holding a tool already. Therefore, *C5* was only used to test evolved circuits' ability to detect tool affordances.

Finally, task variant *C6*, in which one tool appears in *R0* ∩ *R1* and the other appears in *R1*, was evolved on, but evolved circuits' performance in general were not tested on it.

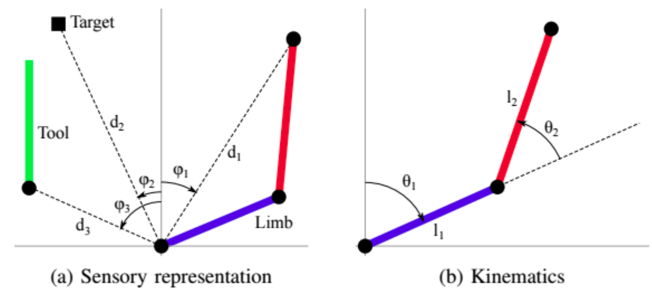


Fig. 4. Agent-centered sensory representation (AC) and kinematics of the articulated limb. (a) AC uses a polar coordinate system. The angles for the end effector, target, and tool are represented by φ_1, φ_2, and φ_3, respectively. Analogously, the distances are represented by d_1, d_2, and d_3. (b) The arm consists of two limbs, l_1 = 80 and l_2 = 100. The angle between the reference direction (up) and the first limb is represented by θ_1, and the angle between the first and second limb is represented by θ_2.

The two-tool task variants differ from one another by which regions in the environment the two tools can appear in. The dimensions of these regions are an artifact of the limb and tool lengths and the joint angle constraints. If those parameters were to have different values, the regions would change shape, but the affordance of the tools and consequently the optimal strategy, which are defined by the tool placement configuration within these regions, would not.

C. Input and Output for Neural Circuits

As in our previous work [7][8][9], we chose a relative agent-centered (RAC) environment representation using polar coordinates [25]. Consequently, environment configurations that are equivalent after translation or rotation are naturally characterized by the same coordinates. RAC produces the following coordinates:

$$\begin{aligned}
\mathrm{AC}_{\mathrm{end_eff}} &= (\varphi_1, d_1)\\
\mathrm{AC}_{\mathrm{target}} &= (\varphi_2, d_2)\\
\mathrm{AC}_{\mathrm{tool}} &= (\varphi_3, d_3)\\
\mathrm{RAC}_{\mathrm{end_eff\&target}} &= (\varphi_2 - \varphi_1, d_2 - d_1)\\
\mathrm{RAC}_{\mathrm{end_eff\&tool}} &= (\varphi_3 - \varphi_1, d_3 - d_1)
\end{aligned}$$

The input layer for the neural circuit includes both perceptual and proprioceptive neurons [11]. Eight input neurons are used: two joint angles θ_1 and θ_2, relative distance and angle to the target from the end effector $\varphi_2 - \varphi_1$ and $d_2 - d_1$, relative distance and angle to the tool that is closest to the end effector from the end effector $\varphi_3 - \varphi_1$ and $d_3 - d_1$, and joint limit detectors (υ_1, υ_2), where υ_i = 1 if θ_i = 150°, -1 if θ_i = -150°, and 0 otherwise.

The output layer consists of two neurons, the values of which determine the change in the two joint angles, with a maximum of 1.5° change per time step.

D. Fitness Criteria

Each neural circuit was evaluated on n = 100 random initial configurations for up to t_{max} = 600 time steps each. At the end of the evaluation period, the quantities representing the four fitness criteria were calculated and multiplied together to yield the individual's fitness value.

The distance factor of the fitness function is:

$$D = 1 - \frac{\sum_{k=1}^{n} \|\vec{o}_k - \vec{e}_k\|}{2nr_{max}}$$

where $\vec{o}_k$ is the location of the target object in the k-th trial, $\vec{e}_k$ is the location of the end effector at the end of the k-th trial, and $r_{max} = l_1 + l_2 + l_t$ or $l_1 + l_2 + 2l_t$, for one and two tools respectively. D is a measure of how close the end effector got to the target object relative to the total length of the arm, on average.

The step factor, a measure of how quickly the end effector reached the target object on average, is:

$$S = 1 - \frac{\sum_{k=1}^{n} s_k}{nt_{max}}$$

where s_k is the number of time steps required to reach the target in the k-th trial. If the end effector does not reach the target, S = t_{max}.

The reach factor, effectively the success rate, is:

$$R = \frac{\sum_{k=1}^{n} r_k}{n}$$

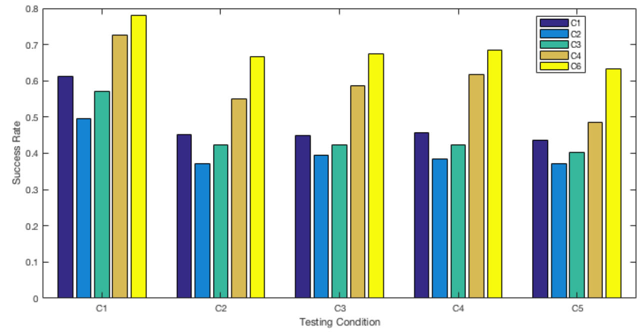
where r_k = 1 if the target was successfully reached in the k-th trial, and r_k = 0 otherwise.

Finally, the tool factor, a measure of optimal tool use is:

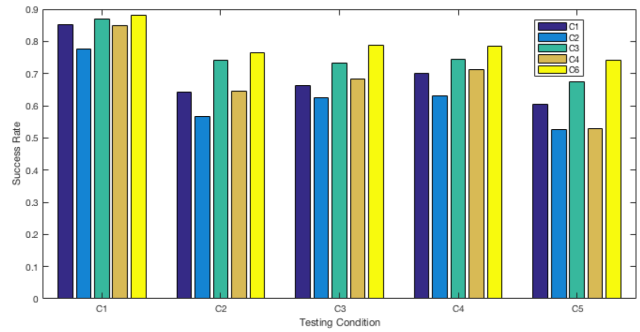
$$T = \frac{\sum_{k=1}^{n} t_k}{n}$$

where t_k = 1 if the target region and number of tools picked up were respectively *R0* and zero, *T1* and one, or *T2* and two, and t_k = 0 otherwise. Previous work [8] demonstrated that T is not required to evolve tool use behavior, but it facilitates its evolution.

The fitness factors, D, S, R, and T, were normalized to the interval [0.5;1] and multiplied together to yield the overall fitness score.



(a) Average success rate on each task variant by evolution condition



(b) Highest success rate on each task variant by evolution condition

Fig. 5. Comparison of success rates. Controllers were evolved for five task variants, *C1*, *C2*, *C3*, *C4*, and *C6* (see color-coded inset boxes), and they were evaluated on five task variants *C1*, *C2*, *C3*, *C4*, and *C5* (x-axis labels). Evolving on task variant *C6*, in which one tool is placed in *R0* ∩ *R1* and the other is placed in *R1*, led to the best performance on all task variants. High performance on *C5* requires accurate detection of tool affordances

III. Experiment (Simulation)

Each neural circuit controller was evaluated on n = 100 randomly instantiated trials, with joint angles assigned initial values in the range [-150°;150°] according to a uniform distribution on that interval. The target object and tool(s) are placed randomly according to a uniform random distribution on the semicircle from -90° to 90° around the shoulder joint location. The distance range in which the target and tool(s) vary by task variant as was described in the previous section. Each trial consists of up to t_{max} = 600 time steps, ending early when the target is reached.

Four times for each of the five task variants *C1*, *C2*, *C3*, *C4*, and *C6*, a population of 100 controllers was evolved for 100 generations. The highest fitness controller was kept from each evolutionary run, yielding four high performing neural circuits evolved for each task variant. Each of these controllers was evaluated five times on the same random 300 target object locations, 100 from each of *R0*, *T1*, and *T2*, on each of the five task variants *C1*, *C2*, *C3*, *C4*, and *C5*.

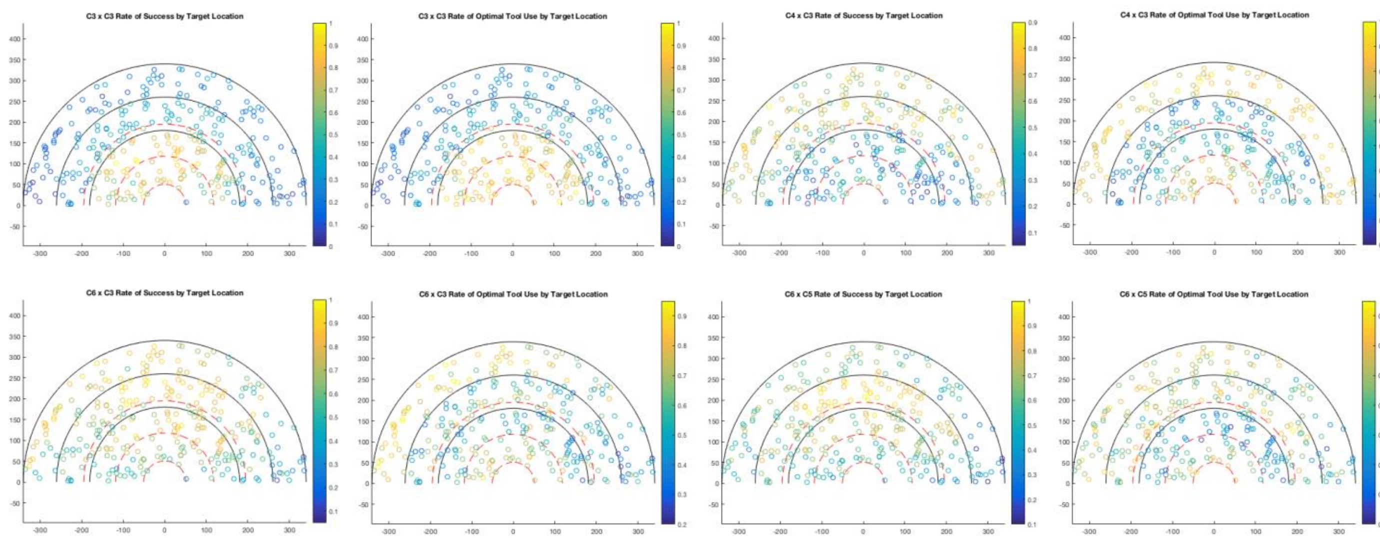


Fig. 6. Rates of success and optimal tool use by target location for select conditions. The two plots in the top-left depict success rate and rate of optimal tool use by neural circuits evolved on task variant *C3* when evaluated on task variant *C3*. The two plots in the top-right depict the same for circuits evolved on *C4*, evaluated on *C3*. The bottom row depicts the performance of circuits evolved on *C6*, evaluated on *C3* (left) and on *C5* (right).

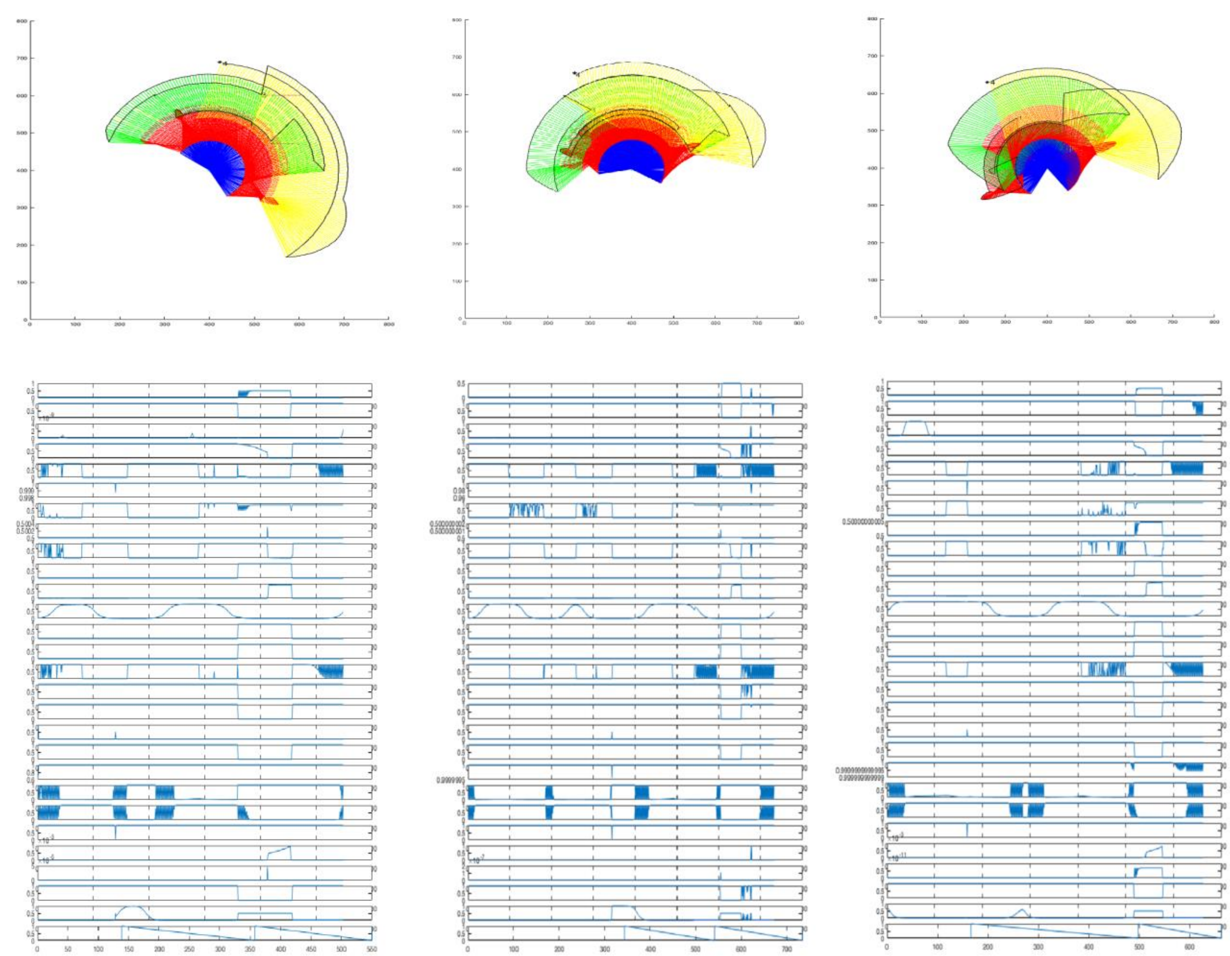


Fig. 7. Limb trajectories and associated hidden neuron activation patterns for three instances of the task requiring tool construction. For the trajectories, blue depicts the upper limb, red the lower limb, green the secondary tool, and yellow the constructed tool. The plot at the bottom of each of the three hidden neuron activation patterns depicts the times of the tool pickup events in those runs. The activations of several groups of hidden neurons are correlated with one another, and many hidden neurons change mode in anticipation of or in response to tool pickup and in anticipation of reaching the target.

Note: This is a preprint. Some content may be slightly different from the final version. See https://ieeexplore.ieee.org/abstract/document/7965913 for the final version.

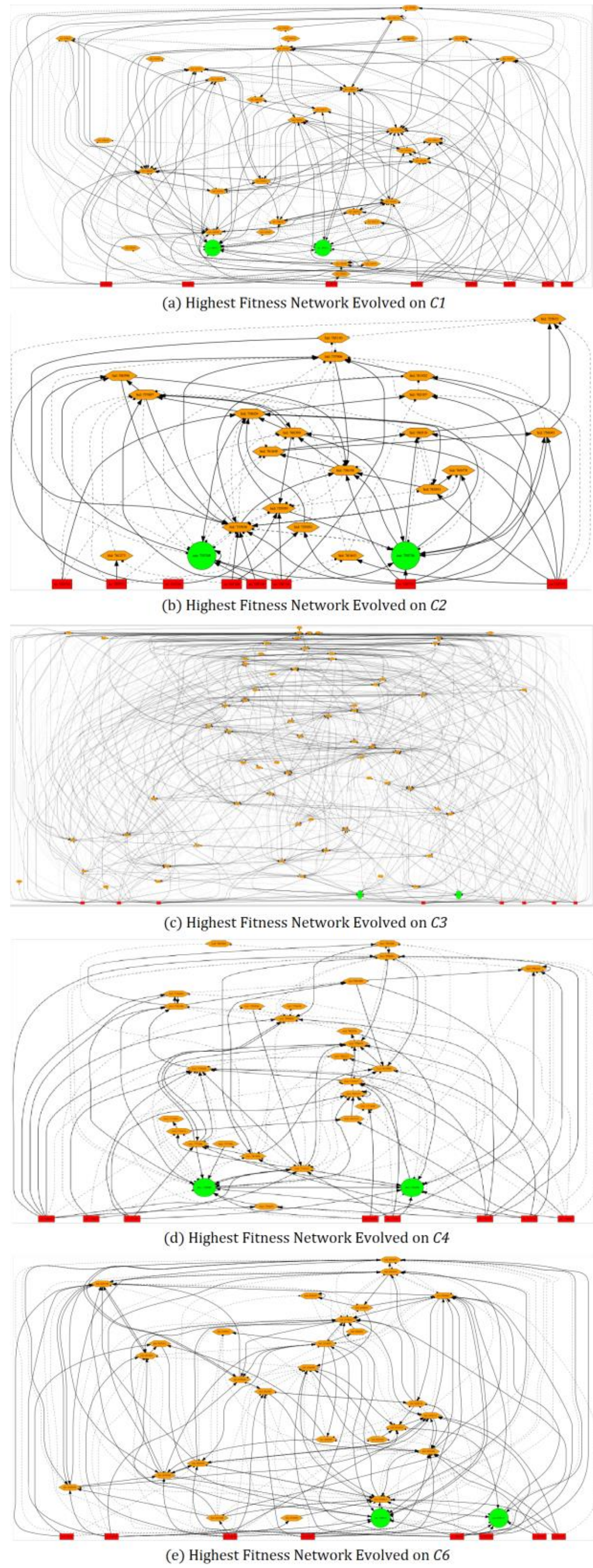
IV. Results

The four chosen fitness criteria, DSRT, proved sufficient to evolve a capacity for tool construction behavior, even when tool construction was not possible during evolution. Figure 5 shows how neural circuit controllers evolved on each task variant performed on each task variant. A few patterns are apparent. First, all neural circuit controllers performed better on *C1*, the task variant only requiring tool use, than they did on all other task variants, which require tool construction. Second, the controllers evolved on task variant *C2* performed worst on all tasks, suggesting that the guaranteed failure of a portion of trials during evolution does not confer any performance advantage. Third, although evolving under condition *C4* produced neural circuits that perform better "on average" than those evolved under condition *C3*, *C3* produced better "highest success rates" than *C*4-evolved neural circuits. The *C3*-evolved circuits are better able to detect tool affordances as evidenced by their much higher performance on tasks *C2* and *C5*, the latter being the most challenging task variant. Finally, neural circuits evolved under condition *C6* outperformed all others on all task variants both "on average" and by "highest success rates".

Figure 6 depicts the relationship between target location and likelihood of success and optimal tool use for different evolution conditions. The top row contrasts the C3 performance of neural circuits evolved on *C3* and *C4*. Controllers evolved on *C3* appear to prioritize reaching for the target over tool pickup, whereas those evolved on *C4* appear to employ the opposite strategy. Interestingly, *C4* controllers appear to know not to pick up a tool when the target is close, but they still usually fail to reach said target. It's possible that, since these controllers appear optimized for reaching the target when no tools are available for pickup, the sensory input received from a tool interferes with their ability to reach the target.

The bottom row of Figure 6 depicts the performance of neural circuits evolved on *C6* on task variants *C3* (left) and *C5* (right). Unlike the controllers evolved on *C3* and *C4*, those evolved on *C6* display much greater uniformity of success across target locations on task *C3*, and they ever perform better than chance at choosing which tool to pick up first on task *C5*. This suggests that evolution on task variant *C6* confers the ability to detect whether a tool affords use as a secondary tool for reaching and combining with the remaining tool. Additionally, because the neural circuit is receiving sensory input from one tool at a time, exploration behavior must be employed should the sensed tool not afford reaching the target.

Figure 7 shows three limb trajectories involving tool construction and the controller's associated hidden neuron activation patterns. During reaching behavior, the controller tends to move the end effector to the correct distance before rotating to the correct angle. The very bottom three plots display the timing of the tool pickup events, allowing for comparison with changes in the hidden state of the network.



(a) Highest Fitness Network Evolved on *C1*

(b) Highest Fitness Network Evolved on *C2*

(c) Highest Fitness Network Evolved on *C3*

(d) Highest Fitness Network Evolved on *C4*

(e) Highest Fitness Network Evolved on *C6*

Fig. 8. Network topologies for highest fitness individuals on each task variant. Red depicts input neurons, green output neurons, and orange hidden neurons. Solid arrows are feedforward connections, and dashed arrows are recurrent connections. (a), (b), (c), (d), and (e), are networks evolved on tasks *C1, C2, C3, C4,* and *C6*, respectively. (e) notably depicts the best performing network overall.

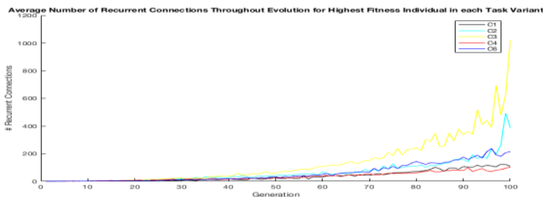


Fig. 9. Growth in number of recurrent connections in highest fitness individual over the course of an average evolutionary run.

Fig. 10. Average (top) and best (bottom) fitness trajectories for each task variant

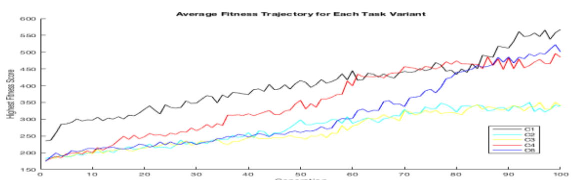


Fig. 11. Average (top) and best (bottom) fitness trajectories for each task variant

It's evident that several hidden neurons change their mode of activation in anticipation of, or in response to, tool pickup as well as in anticipation of reaching the target. This suggests active planning is being done by the neural network controller. Additionally, several groups of neurons' activation patterns appear to correlate with one another, and neurons output both constant and oscillatory signals, as was seen previously [8].

Figures 8 and 9 show how network complexity varies across the different evolution task variants. The number of recurrent connections, which have been shown to be necessary for tool use [8], and consequently the network complexity, grows at different, roughly quadratic rates, depending on evolution task variant. Networks evolved on *C1*, *C4,* and *C6* tend to maintain relatively simple topologies over the course of their evolution, which leads to evolutionary runs on those task variants being much faster than on other task variants.

Figure 10 shows the fitness trajectories averaged over all runs on the five evolution task variants and the fitness trajectories for the best individual on each task variant. Notably, controllers evolved on *C4* tend to experience early increases in fitness, whereas those evolved on *C6* experience them later. *C4* controllers tend to be optimized for reaching targets in area *T2*, requiring both tools to be picked up, whereas *C6* controllers have more well-rounded performance. It's likely that the early fitness gains for *C4* controllers may come from learning to always pick up both tools before trying to reach the target, which is a good, sub-optimal strategy because most targets are in *R2*. *C6* controllers take longer but evolve the optimal strategy.

V. Discussion

This main contribution of this paper is two-fold: (1) we showed that combination-based tool construction, a form of associative tool use, can be evolved with broad fitness criteria, and (2) we found that small variations in the task definition, namely the possible locations of tools, led to significant differences in evolved strategy and capability of detecting tool affordances.

Our approach has several limitations. For instance, world properties such as target location and tool location were heavily encoded in the inputs themselves. In future work, object recognition and grounding [26][27] and efficient codes for motor encoding [28] can be incorporated into our system for increased realism. Even without such additions, though, our approach allows for distinguishing between different types of associative tool use and can serve as a general method for investigating the difficulty and nuances of evolving such distinct behaviors. Other types of associative tool use worth investigating include the use of a tool set, more than one tool used sequentially in different modes, the use of a tool composite, more than one tool used simultaneously in different modes, and tool crafting, construction requiring multiple steps. Other modes of construction could be tested as well. Although preliminary work involving the addition of noise during evolution did not confer an advantage, it still may be worth investigating further.

It would be beneficial to add a pick up or drop output so as to eliminate any need for path avoidance behavior due to pick up occurring automatically and to allow for interaction with another arm. This would allow for the evolution of cooperative behaviors such as coordinated reaching and sharing of tools, and a second arm would allow for more complex tool construction such as variable angle tool connection. Instantiating these controllers in social environments may demonstrate more complex dynamics such as preferential association between agents due to information provided by each one's behavior relative to the environment.

Future research will attempt to use a generative adversarial network (GAN) as an inference network, trained in a wake sleep cycle, equipped with auxiliary policy networks evolved to perform basic tool use tasks. This inference net would be trained in an unsupervised manner on the experiences of the evolved policy networks with the purpose of forming a posterior over policies given perceptual input while performing some, possibly not seen before, task. This posterior would provide a weighting of the outputs of the individual policy networks, determining the extent to which each is applied in that time step. This might allow for more complex tool use and construction to emerge via the combination of more fundamental behaviors. Furthermore, the search of behavior space by the inference net may be constrained in a supervised manner with the use of a parameter defining the extent to which the previous superposition of policies is used versus the one inferred from the current percepts. Continuing to use the same weighting of policies could be seen as exploitation, whereas using the weighting based only on current percepts might be seen as exploration. Learning this parameter as a function of environmental or perceptual features may provide an intrinsic motivation for the agent.

VI. Conclusion

In this paper, we investigated how the capability of combination-based tool construction behavior can spontaneously emerge in an evolved neural controller for a two-degree-of-freedom articulated limb in a target-reaching task. The neural circuit evolution algorithm NEAT was used for evolution of the controllers, permitting the evolution of network topology in addition to weights. We found that small changes in the task definition, namely tool locations, lead to significant differences in strategy, particularly impacting the evolved controllers' ability to detect tool affordances, and we found that one such task variant yielded the best performance along all metrics. These findings will inform future investigation of the origin of associative tool use, helping use to understand what types of neural circuits and what environmental characteristics enabled such a powerful capability.

Acknowledgement

Neural circuit evolution experiments was performed using ANJI (Another NEAT Java Implementation, anji.sourceforge.net), an open-source Java implementation of the algorithm by Stanley and Miikkulainen [22].

References

[1] A. Whiten, J. Goodall, W. C. McGrew, T. Nishida, V. Reynolds, Y. Sugiyama, C. E. G. Tutin, R. W. Wrangham, and C. Boesch, "Cultures in chimpanzees," Nature, vol. 399, no. 6737, pp. 682–685, 1999.

[2] C. Boesch and H. Boesch, "Tool use and tool making in wild chimpanzees," Folia Primatologicarimatologica, vol. 54, no. 1-2, pp. 86–99, 1990.

[3] P. Foerder, M. Galloway, T. Barthel, D. E. Moore, III, and D. Reiss, "Insightful problem solving in an asian elephant," PLoS ONE, vol. 6, no. 8, p. e23251, 2011.

[4] G. R. Hunt and R. D. Gray, "The crafting of hook tools by wild new caledonian crows," Proceedings of the Royal Society of London B: Biological Sciences, vol. 271, no. Suppl 3, pp. S88–S90, 2004.

[5] A. Streri and J. Feron, "The development of haptic abilities in very ´ young infants: From perception to cognition," Infant Behavior and Development, vol. 28, no. 3, pp. 290–304, 2005.

[6] Shumaker, R. W., Walkup, K. R., & Beck, B. B. (2011). Animal tool behavior: the use and manufacture of tools by animals. JHU Press.

[7] J. R. Chung and Y. Choe, "Emergence of memory in reactive agents equipped with environmental markers," Autonomous Mental Development, IEEE Transactions on, vol. 3, no. 3, pp. 257–271, 2011.

[8] Q. Li, J. Yoo, and Y. Choe, "Emergence of tool use in an articulated limb controlled by evolved neural circuits," in Neural Networks (IJCNN), 2015 International Joint Conference on, July 2015, pp. 1–8.

[9] Michael Freitag and Yoonsuck Choe. Analysis of tool use strategies in evolved neural circuits controlling an articulated limb. In *Proceedings of the International Joint Conference on Neural Networks*, 2016.

[10] R. S. Amant and A. B. Wood, "Tool use for autonomous agents," in Proceedings of the 20th National Conference on Artificial Intelligence - Volume 1. AAAI Press, 2005, pp. 184–189.

[11] R. R. Murphy, Introduction to AI robotics. MIT Press, 2000, vol. 108.

[12] D. Lee, H. Kunori, and Y. Nakamura, "Association of whole body motion from tool knowledge for humanoid robots," in 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2008, pp. 2867–2874.

[13] A. Arsenio, "Learning task sequences from scratch: applications to the control of tools and toys by a humanoid robot," in Control Applications, 2004. Proceedings of the 2004 IEEE International Conference on, vol. 1, 2004, pp. 400–405.

[14] R. Saegusa, G. Metta, G. Sandini, and L. Natale, "Developmental perception of the self and action," IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 1, pp. 183–202, 2014.

[15] Y. Wu and Y. Demiris, "Learning Dynamical Representations of Tools for Tool-Use Recognition," in Proceedings of the 2011 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2011, pp. 2664–2669.

[16] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," {IEEE} International Conference on Robotics and Automation, 2009. {ICRA} '09, pp. 763–768, 2009.

[17] S. Nishide, J. Tani, T. Takahashi, H. G. Okuno, and T. Ogata, "Tool– Body Assimilation of Humanoid Robot Using a Neurodynamical System," Autonomous Mental Development, IEEE Transactions on, vol. 4, no. 2, pp. 139–149, 2012.

[18] A. Stoytchev, "Behavior-grounded representation of tool affordances," Proceedings - IEEE International Conference on Robotics and Automation, vol. 2005, no. April, pp. 3060–3065, 2005.

[19] D. Katz and O. Brock, "Manipulating articulated objects with interactive perception," IEEE International Conference on Robotics and Automation (ICRA), pp. 272–277, 2008.

[20] B. Schafer, N. Bergfeldt, M. Riveiro Carballa, and T. Ziemke, "Evolution of tool use behavior," in Artificial Life, 2007. ALIFE '07. IEEE Symposium on, April 2007, pp. 31–38.

[21] K. Sims, "Evolving 3D Morphology and Behavior by Competition," Artificial Life, vol. 1, no. 4, pp. 353–372, 1994.

[22] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," Evolutionary computation, vol. 10, no. 2, pp. 99–127, 2002.

[23] K. O. Stanley, B. Bryant, I. Karpov, and R. Miikkulainen, "Real-time evolution of neural networks in the NERO video game," Proceedings of the Twenty-First National Conference on Articial Intelligence, pp. 1671–1674, 2006.

[24] J. Gauci and K. O. Stanley, "A Case Study on the Critical Role of Geometric Regularity in Machine Learning." Proceedings of the TwentyThird AAAI Conference on Artificial Intelligence, pp. 628–633, 2008.

[25] T. Mann and Y. Choe, "Prenatal to postnatal transfer of motor skills through motor-compatible sensory representations," in Development and Learning (ICDL), 2010 IEEE 9th International Conference on, Aug 2010, pp. 185–190.

[26] C. Yu and D. H. Ballard, "On the Integration of Grounding Language and Learning Objects," in Proceedings of the National Conference on Artificial Intelligence, no. Quine, 2004, pp. 488–494.

[27] J. Modayil and B. Kuipers, "Autonomous development of a grounded object ontology by a learning robot," in Proceedings of the National Conference on Artificial Intelligence, vol. 2, 2007, pp. 1095–1101.

[28] L. Johnson and D. H. Ballard, "Efficient codes for inverse dynamics during walking," in Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.

Note: This is a preprint. Some content may be slightly different from the final version. See https://ieeexplore.ieee.org/abstract/document/7965913 for the final version.