Replicating the Stroop Effect Using a Developmental Spatial Neuroevolution System

Amit Benbassat(✉) and Avishai Henik

Ben-Gurion University of the Negev, Beer-Sheva, Israel
amitbenb@post.bgu.ac.il, henik@bgu.ac.il

Abstract. We present an approach to the study of cognitive phenomena by using evolutionary computation. To this end we use a spatial, developmental, neuroevolution system. We use our system to evolve ANNs to perform simple abstractions of the cognitive tasks of color perception and color reading. We define these tasks to explore the nature of the Stroop effect. We show that we can evolve it to perform a variety of cognitive tasks, and also that evolved networks exhibit complex interference behavior when dealing with multiple tasks and incongruent data. We also show that this interference behavior can be manipulated by changing the learning parameters, a method that we successfully use to create a Stroop like interference pattern.

1 Introduction

Much research in cognitive psychology has been devoted to goal directed behavior or to the mental processes involved in focusing on relevant information and declining or ignoring irrelevant information. One of the paradigmatic tasks in cognitive psychology is the Stroop task in which people are presented with words in color (e.g., RED in green) and asked to pay attention to the color and ignore the meaning of the word. The current work applies evolutionary algorithms (EAs) to study the mechanisms involved in the Stroop task.

1.1 The Stroop Effect

In his original work Stroop [14] presented participants with lists of stimuli on a card and asked them to name the color of the ink as fast as possible. He measured the time to name 100 stimuli on each card. Stroop used two conditions, incongruent (e.g., RED in green) and neutral (i.e., patches of colors). Responding was slower to the incongruent condition than to the neutral condition. Stroop suggested that the difference between incongruent and neutral conditions was an indication for the automaticity of word reading. Importantly, when he asked participants to read the words and ignore their color, word reading was not hampered by the incongruent colors.

J. Handl et al. (Eds.): PPSN XIV 2016, LNCS 9921, pp. 602–612, 2016.
DOI: 10.1007/978-3-319-45823-6_56

With the introduction of computers to psychology laboratories, the task changed to a trial-by-trial task. Vocal response time of participants was measured in milliseconds. These single trial experiments enable experimenters to include congruent trials (i.e., Green in green). As computer presentations mix all conditions, participants are unable to predict the appearance congruent inputs and cannot adopt a reading strategy for those inputs.

Research on goal directed behavior use not only the Stroop task but other tasks also [2,5,12].

1.2 Neuroevolution

Neuroevolution is the subfield of evolutionary computation concerned with growing *Artificial Neural Networks* (ANNs) via artificial evolution. The field has attracted much research effort. Stanly and Miikkulainen [13] created the NEAT system for the explicit purpose of evolving complex networks from simple initial networks. NEAT uses direct encoding where evolved genes relate to specific parts of the network. HyperNEAT is a neuroevolution system created on the basis of NEAT that uses indirect encoding and is widely used to evolve ANNs that perform various tasks [10,11,15]. HyperNEAT works by evolving Meta-networks with NEAT, that in turn decide on edge weights in the ANN that is meant to perform the task. Some neuroevolution systems take a cue from nature and use a developmental scheme. For example, Kitano [9] presented a method of evolving grammars that generate ANN connectivity maps. Another example is Gruau [4]. Gruau suggested the concept of *Cellular Encoding* (CE) where the individual neurons act as cells during the developmental process. In Gruau's system development of ANNs is dictated by the genome (a tree genome in [9]'s case. A linear genome in ours). Many other implementations are presented in a review by Floreano et al. [3]

1.3 The Current Work

In this work we present an evolutionary learning based approach to the study of cognitive phenomena. We employ an EA on populations of randomly generated ANNs in order to evolve networks that perform cognitive tasks. This allows us to explore the specific conditions under which certain phenomena may occur. Specifically, in the first phase of the project we wanted to create a neuroevolution system that features natural qualities. Next, we attempted to generate a Stroop effect. Specifically, we aimed to generate interference and facilitation and also the asymmetry between word reading and color naming. That is, significant interference and facilitation in color naming with small or null effects in word reading.

2 Spatial Developmental Neuroevolution System

As suggested earlier, we designed our evolutionary system with an eye towards nature. We cannot emulate all natural traits but we focused on three important

traits which we integrated as design features. These important attributes of our system are listed below:

1. ANN based: We chose neuroevolution because the artificial neuron is an abstraction of the biological neuron. Though the two are by no means identical an ANN is similar to brain systems in being a decentralized computation system made of simple computation units that share some of the same attributes.
2. Developmental: A gene in the individual's genome does not map directly to a specific simple element in the final network. Rather, it is seen as a command that is to be performed by the developing network or a subset of its artificial neurons, during the developmental process.
3. Spatial: Every artificial neuron in our system is located in some point in a virtual space. Developmental steps are space related, placing, moving, and connecting neurons to each other using spatial coordinates.

2.1 The Spatial ANN

The ANNs in our system consist of three distinct layers: An input layer, an output layer, and a hidden layer. Each one of the layers exists in its own space defined by the user. The user defines the number of dimensions each layer has and the size of each dimension. The spaces contain a discrete grid of locations where a neuron may reside (e.g. A 3D layer of size $3 \times 4 \times 5$ contains exactly $3 \times 4 \times 5 = 60$ possible neuron locations).

In the input and output layers every location contains an artificial neuron. The hidden layer's content depends on the individual's genome. The genome defines all hidden neurons as well as all network edges. The input values are real numbers in the [−1,1] range (we typically use the extreme values −1 and 1 but the system supports using other values as well). Outputs are limited to the [−1,1] range.

2.2 Genome Structure

Our encoding is based on a genome in the form of a linear array of genome atoms (or genes). Each gene is a set of integers and real numbers denoted by

Table 1. Fields in the genome atom

Field name	Field type
`Read Mode`	Integer
`New Read Mode`	Integer
`Opcode`	Integer
`Weight`	Real
`Threshold`	Real
`Location Offset`	Integer array

field names. Table 1 presents the names of the fields. The last line shows the Location offset array of fields. This is an integer array that signifies a location offset in one of the network grids (the identity of which depends on the type of gene being read). The length of the array is determined by the number of spatial dimensions which the user controls using run parameters.

2.3 Run Parameters

The particulars of each run are controlled by the user with a series of run parameters. Evolutionary parameters control the evolutionary process, fitness parameters control the calculation of individual fitness, ANN parameters control the basic attributes of the networks and encoding parameters control the genome's encoding rules into the ANN phenotype.

Evolutionary parameters include number of generations, population size, crossover probability, mutation probability, type of selection and diversity maintenance parameters. Our system utilized single-point crossover variant that allows genome size to change by up to 20 % in each crossover event. Mutation is uniform (mutation probability is per gene and not per individual). When a spot in the genome is chosen for mutation, one of two actions is performed each with a probability of 0.5: Either the atom itself is randomly changed or a small genome segment beginning with the chosen atom is copied to another random location in the genome. Our system uses standard tournament selection. In the experiments described in this work we use a tournament size of 4.

The diversity maintenance measure limits the number of individuals with similar behavior profiles. An individual's behavior profile is an array made of all the output values its ANN gets for all the fitness tests (the values in the behavior profile are rounded to values in $\{-1,0,1\}$). The *distance ratio* between two behavior profiles is the number of locations where the profiles differ divided by the length of the profiles. We say that two individuals are neighbors if the distance ratio between their behavior profiles is lower than a value controlled by the user (which we typically set at 0.3). Our diversity maintenance system allows an individual to be selected only if the number of its neighbors already selected is lower than a certain threshold (typically 30). We used this method to encourage diversity not only in genome but also in behavior.

Fitness parameters include the test inputs used to calculate the fitness score and the test inputs used to calculate the benchmark score. The two calculations differ in that the fitness score calculates a much smoother function. When calculating fitness the individual is rewarded slightly for every output neuron that generates a correct output, as well as being given a bonus for the whole network giving the correct answer. The benchmark score is only affected by whether the network's answer to the test input is correct or not. In both the fitness and the benchmark score cases we normalize the scores to the [0,1000] range to make them easier to assess.

ANN network size parameters include a limit on the number of hidden layer neurons and of network links, which we set to 400 and 4000, respectively. Initial

genome size is set to 80 (limited to a maximum of 150). The number of different read codes is set to 31. The read codes determine which neuron reads which gene.

The read encoding scheme for i codes is based on a complete binary tree with i nodes that are tagged in level order starting at 1. In order to decide whether or not a given neuron reads a given gene the read codes of the neuron and the gene have to match. Two read codes match if one is an ancestor of the other in the tree.

2.4 Spatial Developmental Encoding

Our system supports multiple encoding schemes. There are several different types of actions that a gene can cause. The probability of a gene encoding a certain action is controlled by the user, who chooses how much weight to assign to each of the possible gene types. Table 2 contains the encoding weights we chose for our experiments below. We chose these values empirically and with use of common sense.

Table 2. Different action types and their weights in our experiments. an action can have a weight of 0 or higher assigned to it. An action assigned a weight of 0 is impossible to encode with any gene. The probability that an action will be encoded by a randomly generated gene is proportionate to the weight of that action.

Action name	Weight	Action description
`New Node`	10	Create a new neuron
`Move`	2	Move neuron
`Connect`	4	Connect neuron to another neuron
`Connect output`	4	Connect neuron to an output
`Connect input`	4	Connect an input to neuron
`Connect all output`	0	Connect neuron to all outputs
`Connect all input`	4	Connect all inputs to neuron
`Mutate Threshold`	2	Change neuron threshold and factor
`Split`	8	Split existing neuron, creating a new neuron next to it
`Power Split`	4	Split existing neuron, creating a new neuron with all the same connections
`Sleep`	4	Neuron sleeps, no longer performing actions
`Awaken`	4	Sleeping neuron wakes up, and resumes performing actions
`Die`	2	Neuron dies and is removed from network

Of the actions described in Table 2 `New Node` stands out as working on the entire network (by adding a neuron to it). All other actions are activated by individual neurons for which the read encoding matches.

3 The Problem Domains

In this research, we interpret our outputs in a way analogous to nature. The human brain often makes decisions when a plurality of neurons signal together rather than relying on just one neuron. In our experiments we followed this principle. The domains explored in this work (i.e., reading words or naming colors) are classification tasks where the ANN is expected to tell a number of different classes apart (e.g., distinguishing between three colors; blue, red, green). In these tasks, the output is a two dimensional 4×5 grid, and each one of the 4 rows stands for one of the possible classes (e.g., blue). Decision is made by plurality rule, with the network choosing class i if and only if row i in the output has more 1's in it than any other row.

3.1 Color Perception

In the Color Perception task (or *CP*) the ANNs are required to identify the color of an input. In this work we define the CP task to work with a 3-dimensional input grid of size $4 \times 5 \times 5$. We see the input as made up of 4 2-dimensional grids: 3 colored "visual field" grids (*red*, *green* and *blue*) and 1 "task definition" grid that is used to differentiate between color perception and Color Reading tasks. In the CP task we expect the forth grid of the input to contain all -1's. The output is a 2-dimensional grid of size 4×5 with a correct output being one that contains a plurality of 1's in the row representing the right answer. Our convention is that the first row stands for *red*, the second stands for *green*, the third stands for *blue* and the forth is reserved for future use for inputs that do not have one dominant color. In Figs. 1 and 2 there are two examples of inputs for the CP task. In these figures, and all other figures further on that show examples of inputs, we assume that an empty square represents an input of -1 and a square containing a black circle represents an input of 1.

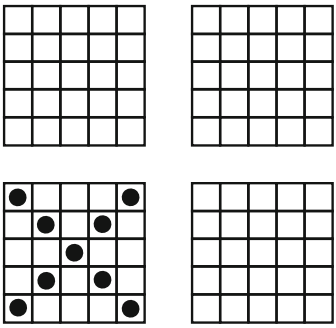

Fig. 1. Neutral CP input of $\times$ sign in blue

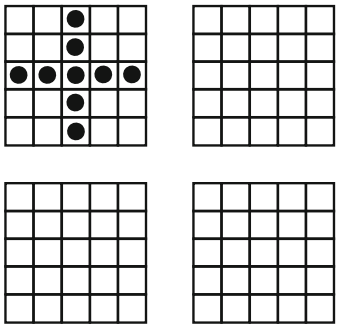

Fig. 2. Congruent CP input of $+$ sign in red

3.2 Color Reading

In the Color Reading task (or *CR*) the ANNs are required to read a colored symbol in the input. The input and output dimensions are identical to the CP task. In the CR task we expect the forth grid of the input to contain all 1's. Our convention for the output is similar to CR. We chose symbols to stand for the three base colors. The + symbol stands for *red*, the sideways `H` stands for *green* and the `H` stands for *blue*. In Figs. 2, 3 and 4 we see the symbols for red, green and blue, respectively.

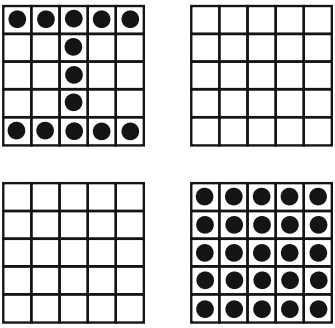

Fig. 3. Incongruent CR input of green sign (the sideways `H`) in red

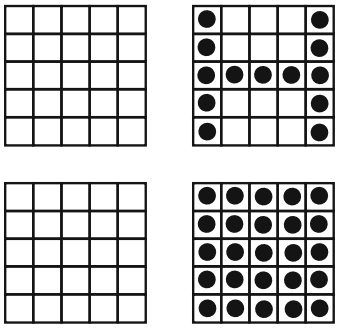

Fig. 4. Incongruent CR input of blue sign (the `H`) in green

4 Experiments

We ran several experiments in the different tasks. Each experiment was designed to check another phenomenon. In each experiment we ran the same simulation 100 times in order to get sufficient data. For some experiments we had to compare two or more types of runs. In those cases each run type got its own set of 100 simulations. Each 100 simulation experiment set took at most about a day (running on a laptop computer with Intel Core i7-4700MQ 2400 MHz 4-core processor). In all the runs below we used an elitism rate of 0.02, a mutation rate of 0.02 and a crossover rate of 0.8.

The runs generate a lot of data. For brevity we do not present all generated data here. We will present data which we think is relevant to the experiments presented.

4.1 Evolving Color Perception and Reading

First we attempted to evolve ANNs that perform each task separately. For brevity's sake we do not go into much detail as far as these runs are concerned.

To evolve ANNs for the CP task we used a population of 200 individuals, running for 120 generations. In 81 out of 100 simulations the ANNs reached a

perfect benchmark score on the CP task. To evolve ANNs for the CR task we used a population of 500 individuals, running for 900 generations. Though only 7 of the simulations resulted in individuals that had a perfect benchmark score on the CR task, they were not far off the mark. The best solution in a simulation had a mean benchmark score of 909.0457 ($\sigma = 80.3975$).

4.2 Mixed

We used a population of 400 individuals, running for 400 generations. We calculated the fitness score and the benchmark score using the 12 test inputs from The CP Experiment and the 21 test inputs from the CR Experiment. After the runs terminated, we checked the best individuals on congruent and incongruent inputs separately in both tasks.

Looking at congruent inputs the best solution in a simulation had a mean benchmark score of 949.999 ($\sigma = 119.6258$) in the CP task. Looking at incongruent inputs the best solution in a simulation had a mean benchmark score of 801.6637 ($\sigma = 173.5871$) in the CP task. Looking at congruent inputs the best solution in a simulation had a mean benchmark score of 893.3315 ($\sigma = 176.5132$) in the CR task. Looking at incongruent inputs the best solution in a simulation had a mean benchmark score of 446.6643 ($\sigma = 149.5968$) in the CR task.

We conducted one-way ANOVA on the 4 score types ($F(3, 396) = 208.3780$). The difference between the congruent and incongruent is significant in the CP task ($p < 0.0001$), and also in the CR task ($p < 0.0001$).

In both tasks there are no significant differences in other attributes of the individuals, which is to be expected as the individuals being compared are taken from the same population pools. These results show that interference does occur in our system and that the congruent inputs are easier for our evolved networks. However there is not clear directionality as in the Stroop case. Networks do better on congruent inputs in both the CP and CR tasks.

4.3 Weighted

We attempted to generate directionality by weighting the fitness function. In this experiment we used the same parameters as in the mixed experiment described in Sect. 4.2 except for the fitness function, which was weighted to bias evolution in favor of the CR task. Each fitness test-case from the CR test suite affected the fitness result as if it appeared 30 times in the suite.

Looking at congruent inputs the best solution in a simulation had a mean benchmark score of 749.9962 ($\sigma = 219.0449$) in the CP task. Looking at incongruent inputs the best solution in a simulation had a mean benchmark score of 493.3311 ($\sigma = 141.9752$) in the CP task. Looking at congruent inputs the best solution in a simulation had a mean benchmark score of 746.6633 ($\sigma = 246.7312$) in the CR task. Looking at incongruent inputs the best solution in a simulation had a mean benchmark score of 738.3301 ($\sigma = 191.3615$) in the CR task.

We conducted one-way ANOVA on the 4 score types ($F(3, 396) = 38.2965$). The difference between the congruent and incongruent is significant in the CP

task ($p < 0.0001$), but it is insignificant in the CR task ($p = 0.9915$). This approach successfully creates the desired asymmetry that is an attribute of the Stroop effect.

4.4 Phased and Weighted

In this experiment we added a phasing element to our simulations. We ran a population of 400 for 150 generations evaluating fitness on both CP and CR tests, using the weighted mixed test suite we used in Sect. 4.3, then we allowed the population to evolve for 400 generations on CR inputs only, and then let it evolve for 150 more generations the weighted mixed test suite.

Looking at congruent inputs the best solution in a simulation had a mean benchmark score of 716.6629 ($\sigma = 243.3185$) in the CP task. Looking at incongruent inputs the best solution in a simulation had a mean benchmark score of 458.3311 ($\sigma = 142.8719$) in the CP task. Looking at congruent inputs the best solution in a simulation had a mean benchmark score of 819.9976 ($\sigma = 234.1278$) in the CR task. Looking at incongruent inputs the best solution in a simulation had a mean benchmark score of 794.9971 ($\sigma = 186.3103$) in the CR task.

We conducted one-way ANOVA on the 4 score types ($F(3, 396) = 64.7027$). Again the difference between the congruent and incongruent is significant in the CP task ($p < 0.0001$), while it is insignificant in the CR task ($p = 0.8256$). This approach also creates an asymmetry effect similar to Stroop, and also results in a higher score on the CR task (however, this difference is not statistically significant).

4.5 Neutral CP Input

Among our 12 inputs for the CP task there are 3 `X` shaped inputs that are neither congruent nor incongruent. These are considered *Neutral* inputs (see Fig. 1 for an example of a neutral CP input).

In the experiments we ran in which a Stroop like effect appeared the results on neutral CP inputs fell somewhere in between the congruent and incongruent scores. In the weighted fitness experiment the best solution in a simulation had a mean benchmark score of 596.6617 ($\sigma = 202.648$) on neutral inputs. In the phased weighted fitness experiment the best solution in a simulation had a mean benchmark score of 569.9951 ($\sigma = 202.6477$) on neutral inputs. In light of these results we can say that our experiments are Stroop like also in the classical sense using neutral inputs. On the other hand this is also where our results differ somewhat from the Stroop effect as it appears in humans (the difference in performance between congruent and neutral tests is small to negligible).

5 Concluding Remarks

Our system employs various measures to make the developmental process more like natural development.

We presented a new developmental spatial neuroevolution system for cognitive science research and used it to explore the Stroop effect and evolve ANNs that show some Stroop like behaviors. We successfully replicated, in our evolved networks, the phenomenon of interference due to conflict between the two tasks. We also succeeded in establishing that this conflict can be directional, by biasing the fitness function in favor of the reading task.

There is still much to be done and we want to explore these issues further and also expand our system and look into some new areas. Listed below are some avenues for future research which we believe are promising and we plan to pursue:

- We plan to examine numerical cognition, checking to see if using simple tasks as an evolutionary stepping stone improve the evolution of counting ability. We plan to follow up on work by Katz et al. [8] and Cantlon et al. [1] that suggests counting ability may have evolved from a simpler cognitive system for size perception.
- We plan to expand our inquiry into the evolutionary dynamics of evolving ANNs to perform cognitive tasks. Specifically we are interested in the effects of changing task and environment in mid run on the resulting population. Some of these effects have been demonstrated in the past in several simple domains [6,7].
- We plan to explore the Stroop effect and other similar effects such as Numerical Stroop and the Simon Effect.

Our system itself is still a work in progress, more functionality is needed in order to make it more flexible so it can cover more complex behavior. An obvious extension would be to allow for the evolution of recurrent networks that can handle domains with multiple instances that require the network to react according to new input as well as its own output (such as navigation tasks, and tasks that require networks to have memory capabilities).

Acknowledgments. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC Grant agreement number 295644.

References

1. Cantlon, J.F., Platt, M.L., Brannon, E.M.: Beyond the number domain. Trends Cogn. Sci. **13**(2), 83–91 (2009)
2. Eriksen, B.A., Eriksen, C.W.: Effects of noise letters upon the identification of a target letter in a nonsearch task. Percept. Psychophys. **16**(1), 143–149 (1974)
3. Floreano, D., Dürr, P., Mattiussi, C.: Neuroevolution: from architectures to learning. Evol. Intel. **1**(1), 47–62 (2008)
4. Gruau, F.: Automatic definition of modular neural networks. Adapt. Behav. **3**(2), 151–183 (1994)
5. Henik, A., Tzelgov, J.: Is three greater than five: the relation between physical and semantic size in comparison tasks. Mem. Cogn. **10**(4), 389–395 (1982). http://dx.doi.org/10.3758/BF03202431

6. Kashtan, N., Alon, U.: Spontaneous evolution of modularity and network motifs. Proc. Natl. Acad. Sci. USA **102**(39), 13773–13778 (2005)
7. Kashtan, N., Noor, E., Alon, U.: Varying environments can speed up evolution. Proc. Natl. Acad. Sci. **104**(34), 13711–13716 (2007)
8. Katz, G., Benbassat, A., Diesendruck, L., Sipper, M., Henik, A.: From size perception to counting: an evolutionary computation point of view. In: Proceedings of 15th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 1675–1678. ACM (2013)
9. Kitano, H.: Designing neural networks using genetic algorithms with graph generation system. Complex Syst. J. **4**, 461–476 (1990)
10. Lehman, J., Stanley, K.O.: Abandoning objectives: evolution through the search for novelty alone. Evol. Comput. **19**(2), 189–223 (2011)
11. Secretan, J., Beato, N., Ambrosio, D., Rodriguez, D.B., Campbell, A., Stanley, K.O.: Picbreeder: evolving pictures collaboratively online. In: Proceedings of SIGCHI Conference on Human Factors in Computing Systems, pp. 1759–1768. ACM (2008)
12. Simon, J.R., Rudell, A.P.: Auditory sr compatibility: the effect of an irrelevant cue on information processing. J. Appl. Psychol. **51**(3), 300 (1967)
13. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evol. Comput. **10**(2), 99–127 (2002)
14. Stroop, J.R.: Studies of interference in serial verbal reactions. J. Exp. Psychol. **18**(6), 643 (1935)
15. Yosinski, J., Clune, J., Hidalgo, D., Nguyen, S., Zagal, J., Lipson, H.: Evolving robot gaits in hardware: the hyperneat generative encoding vs. parameter optimization. In: Proceedings of 20th European Conference on Artificial Life, pp. 890–897 (2011)