Enough is Enough: Learning to Stop in Generative Systems

Colin Roitt[0000−0002−1801−158X], Simon Hickinbotham[0000−0003−0880−4460], and Andy M. Tyrrell[0000−0002−8533−2404]

School of PET, University of York, York, United Kingdom
{colin.roitt, simon.hickinbotham, andy.tyrrell}@york.ac.uk

Abstract. Gene regulatory networks (GRNs) have been used to drive artificial generative systems. These systems must begin and then stop generation, or growth, akin to their biological counterpart. In nature, this process is controlled automatically as an organism reaches its mature form; in evolved generative systems, this is more typically controlled by hardcoded limits, which can be difficult to determine. Removing parameters from the evolutionary process and allowing stopping to occur naturally within an evolved system would allow for more natural and regulated growth. This paper illustrates that, within the appropriate context, the introduction of memory components into GRNs allows a stopping criterion to emerge. A Long Short-Term Memory style network was implemented as a GRN for an Evo-Devo generative system and was tested on one simple (single point target) and two more complex (point clouds) problems with and without structure. The novel LSTM-GRN performed well in simple tasks to optimise stopping conditions, but struggled to manage more complex environments. This early work in self-regulating growth will allow for further research in more complex systems to allow the removal of hyperparameters and allowing the evolutionary system to stop dynamically and prevent organisms overshooting the optimal.

Keywords: Generative Design · Self-regulation · EvoDevo.

1 Introduction

Evolutionary generative systems can be extremely powerful and interesting tools for generating a variety of different designs, such as 2D images [22] or 3D worlds [2]. As with many evolutionary systems, the more complex the generative system, the more parameterisation can increase, leading to parameter tuning problems. One major parameter that must be considered is when a generative system should stop generating [10].

Artificial growth in generative systems can vary greatly in implementation, but all must come to a point where they stop growing. This is typically achieved with some hardcoded limit, often even built into the mechanism that generates phenotypes, or as a limit set by a user [14, 13, 11]. However, to take full advantage of the evolutionary emergence of properties, it may prove helpful to give

more control to the evolved generative systems. Indeed, it could be considered detrimental to the principle of evolutionary systems, with hard-coding aspects of a solution reducing exploration of the algorithm. In an ideal evolutionary system, stopping would be determined dynamically based on the state of the system. The question then becomes to what extent evolved systems are capable of stopping at the right time within the context of what they design. Too many steps and compute time are wasted at best, or at worst the organism overshoots the optimal; too few, and it is unable to reach it.

Gene Regulatory Networks (GRNs) are the systems in biology that control gene expression and therefore morphogenesis (how organisms grow), making them an interface between evolutionary and developmental processes. Two biological systems, for the large part of the twentieth century, that were considered separate later became considered together as Evolutionary Development (EvoDevo) [1, 16]. Three primary characteristics of GRNs have been identified as heterochrony, spatial patterning, and interactions between genes and gene products [3].

The impact of these three characteristics is a strong indirect encoding of complex features; GRNs control the process of growth, not the final organism. In nature, a relatively small amount of DNA encodes the biology of a remarkably varied and complex organism. Indirect encoding of a genotype in a phenotype is only possible by allowing interactions between genes, thus creating a GRN, which has been shown to be powerful and key in the artificial domain [1, 8].

GRNs can be emulated within generative systems through computational structures such as feedforward neural networks and have been successfully implemented for multi-objective problems in [11]. That contribution used a fixed number of development steps and did not investigate the controlled stopping of growth. The development of neural networks into some recurrent structure with a memory may aid in the emergence of halting decisions.

The ability to stop requires context not only in the current environment but also on the historical actions made, thus requiring a construct of memory. However, it has been shown that it is problematic for evolutionary systems to develop these internal constructs, as it often requires a number of evolutionary steps that are not immediately effective in increasing the fitness score [17].

Long short-term memory networks have long been used for sequence learning as a form of recurrent neural network [12]. These systems have persistent memory that is carried over recurrent iterations, allowing them to remember and forget information within their architecture. However, the artificial evolution of LSTMs is not as well understood as the more traditional training methodologies based on backpropagation, for example. Where LSTMs have been used in evolved systems, they have not been used in the context of generative systems and often do not evolve the network directly [20, 21, 15].

Through the implementation of the recurrent Long Short-Term Memory (LSTM) networks, the work outlined in this paper will examine how evolutionary systems make use of memory during generative tasks and, crucially, reveal

if and how stopping criteria emerge; questions that become applicable to both the aesthetic- or physical-property domains of generative applications.

The complications involved in evolving memory have been approached in several ways, such as including *helper objectives* that promote memory in a system and by pre-training an LSTM with an information maximisation objective [20]. The topological evolution of LSTM has also been seen in [21]; producing a genetic programming style tree that then builds the topology of the LSTM cell - something that has been investigated outside of evolutionary spaces [15].

The motivation is to understand how these ideas of memory affect the evolution of developmental systems, on a simple level, and into more complex systems. This background suggests that an LSTM style memory mechanism might aide the development of stopping processes in developmental growth - Section 2 details the methodology of this paper in an EvoDevo system.

LSTMs have been shown to work with sequence data well; in particular, some examples of this in an artistic generative case include generating rap lyrics [19] and typefaces[18], both examples incorporating features the network has learnt from a known corpus. With particular focus on song lyrics, an evolved system that could generate lyrics to a contextually appropriate length rather than being hardcoded could prove an interesting use case for the system outlined in this paper.

2 Methodology

To evaluate recurrency in GRNs, this paper proposes a simple LSTM-GRN that addresses the issue of halting generative developmental processes. Of the key properties of GRNs outlined above, the inclusion of recurrency is intended to facilitate the interaction of genes and their products in an artificial space, where proteins and diffusion gradients are not inherent properties of an artificial system, something that biological systems heavily leverage. The use of the system state allows for more complex interactions over time.

An LSTM was chosen above a recurrent neural network as it already has a framework for handling memory inbuilt. This new LSTM-GRN is evaluated through a range of experiments. The following sections present the GRN model and then the experimental methodology and setup of the environment in which these tests were performed.

2.1 LSTM-Gene Regulatory Network

The LSTM-GRN, as shown in Figure 1, has three major parts: the LSTM cell, the feedforward layer, and the activation function. LSTM cells are commonly placed either before or after a perceptron layer. In the current design, the fully connected feedforward perceptron layer follows the LSTM cell to process information that is captured temporally by the LSTM memory [20].

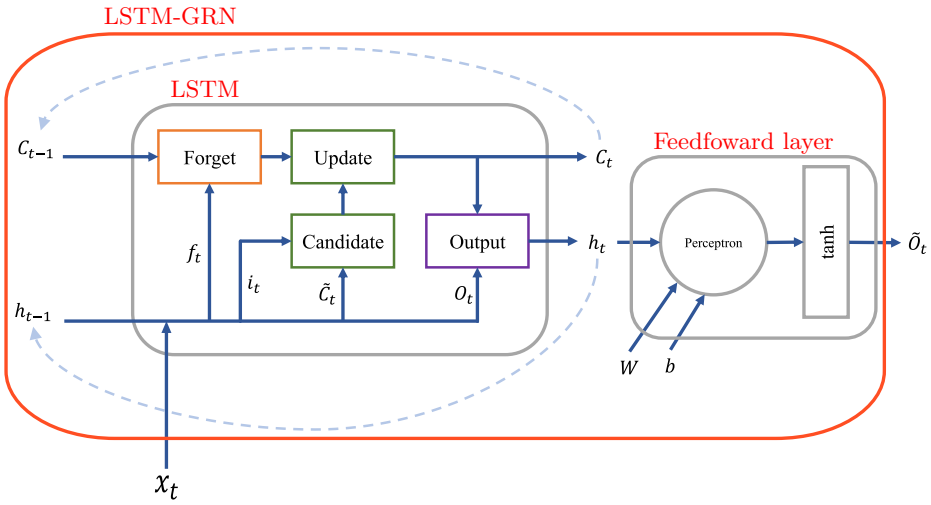


Fig. 1: Overview of the LSTM-GRN architecture shown for an N-dimensional input vector. Visible are the three components: the LSTM cell (the large box, left), the feedforward layer (the circle, middle) and the activation function (the small box, right)

LSTM Cell This part of the LSTM-GRN is a simple construction of the basic LSTM unit outlined in [12]. The unit itself contains a forget gate, an update gate, and an output with an activation function.

The LSTM cell works by having two streams of long-term and short-term memory flow through it. Short-term memory and the new data input are used, along with weights, to determine how *much* long-term memory to forget. The short-term memory and input are then calculated with additional weights to determine candidate information; this is used to update the long-term memory C_t, which is one output of the cell, fed back in at the next memory input stage, and maintained by the forget and update actions. Finally, the long- and short-term memory are used with more weights to calculate the final output, which is also fed back into the new short-term memory. The output is then passed on to the feedforward section of the LSTM-GRN constructed of a perceptron and activation function, completing one full pass through the LSTM-GRN at any given development step.

At any given time in the devo loop t, the inputs are defined as the long-term and short-term relationship from the previous loop, C_{t-1} and $h_t - 1$, respectively, and the current input state x_t. The weights and biases used in each equation are given as W_f and b_f, W_i and b_i, W_c and b_c and W_o and b_o for forget, input, candidate, and output stages, respectively.

The forget gate, f_t, is defined by Equation 1 which will act on C_{t-1} to give C_t.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$

The update step requires candidate memory update values and some information from the input, and those are given as:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{3}$$

The final update step is given by:

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{4}$$

The last step is to calculate the cell output, which is also used as the short-term input for the next iteration, h_t.

$$O_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = O_t \times \tanh(C_t) \tag{6}$$

Feedforward and Activation The final two components of the LSTM-GRN are a single feedforward layer with an activation function, which here is the tanh function. This can allow the output of the LSTM cell to handle temporal data through its memory gates; in this way, the flow of information is embedded with temporal information before being handed on to the feedforward perceptron layer, which then functions as a more classical neural network unaware of any recurrency. This is given simply as:

$$\tilde{O}_t = \tanh(h_t \cdot W + b) \tag{7}$$

The Tanh layer is a construct of the LSTM-GRN output requirement such that the values can be normalised to a range between -1 and $+1$.

The network described here cannot be trained by a data set because there is no appropriate training set. Therefore, evolution holds a key solution to producing a GRN that behaves appropriately.

2.2 Evo-Devo loop

Evo-devo presents a strategy to take advantage of indirect encoding of gene regulatory networks in an evolutionary space. This process combines two straightforward approaches to both the evolution of a genome and the development of an organism through the LSTM-GRN.

The process behind Evo-devo can be considered simply as two nested loops. The evolution of the LSTM-GRN takes place in the outer loop, while the inner loop acts as a fitness function for the evolutionary selection process. The mapping

of genotype to phenotype occurs within this inner devo loop. The simple evolved genotype is interpreted as the LSTM-GRN; this controls the complex growth of the phenotype. The Devo-loop implementation is described in Algorithm 1.

Algorithm 1: Fitness function for a given individual

Data: *genome*
Result: $fitness = [distance, steps]$
$t \leftarrow TARGET$;
$curr\_loc \leftarrow (0, 0)$;
$S, D, H \leftarrow None, None, None$;
$n \leftarrow 0$;
while $n \neq MAX\_DEVO\_STEPS$ & $H < 0$ **do**
  $n \leftarrow n + 1$;
  $S, D, H \leftarrow GRN(genome, [curr\_loc_x, curr\_loc_y, H])$;
  if $H < 0$ **then**
    $step = |S| \cdot 10$;
    $direction = |D| \cdot 10$;
    $x_i = step \cdot \cos$ *(direction)*;
    $y_i = step \cdot \sin$ *(direction)*;
    $curr\_loc = (x_i, x_y)$;
  else
    END DEVO
  end
end
$distance \leftarrow |curr\_loc - t|$;
$return[distance, n]$;

For initial experiments to consider the basic functionality of this new method, the LSTM-GRN has a fixed topology and is represented in the search space as an array of values, each encoding a specific weight or bias value used in the model, W_f and b_f, W_i and b_i, W_c and b_c and W_o and b_o. This keeps the functions used for genetic operations simple and available in most libraries; in this case, the library chosen was the DEAP library for Python [9]. The evolutionary component is a standard genetic algorithm (GA); however, the inclusion of the devo steps introduces an indirect encoding that can evolve to generate complex behaviour - this is extended by the memory component in the LSTM-GRN.

Evolution is driven by a number of genetic operators that are crucial to achieving good convergence and achieving it in acceptable time. Simulated binary crossover [7] (probability: 0.5, η: 0.4) [6] was chosen for this task, as it is capable of solving a number of issues not present in a binary value crossover, but for a genome constructed of a series of real numbers. Similarly, a Gaussian mutation function (individual probability: 0.3, gene probability: 0.2, μ: 0, σ: 0.3) was used to randomly mutate individuals, allowing for frequent small mutations in the real-coded genome, but also some infrequent larger mutations. Finally,

the selection algorithm used was NSGA-II, a multiobjective algorithm that has become a common standard in evolutionary space [5].

The output of this evolvable generative network is then used to drive the morphological development of two structures outlined in Section 3. Source code made available at https://github.com/ColinRoitt/LSTM-GRN

3 Experiments

In order to show that the LSTM can work within this context, this paper first presents a simple "single point" problem where the organism must reach a point in an artificial environment. Then to extend that to allow for a more visible devo process a "point cloud" based problem is presented. Finally the idea is extended to that of structural growth to see if the organism is able to evolve the appropriate strategies to reach a task-specific point arrangement.

The approaches presented here are inspired by biological forms, specifically the natural and efficient growth exhibited in an experiment with slime mould *Physarum* polycephalum [23]. Wasted energy, while an abstract concept within these experiments, is represented by optimised growth strategies. Learnt behaviours that react efficiently to an environment.

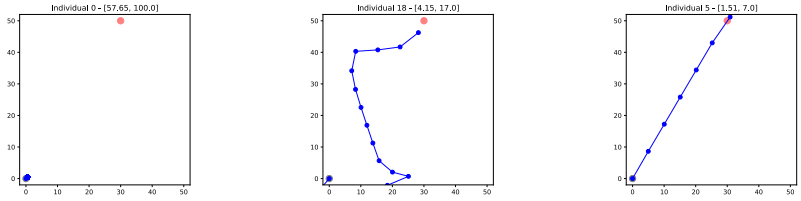
In the first experiment, a single point is used as a target for a single structure to grow from its starting point to the target. The second experiment is a more complex problem: An organism is placed in the middle of a field of targets randomly placed in a band at a specified radius. The organism can then, over each devo step, take a number of actions: step forward, change the angle, or branch. Thus, it is possible for an LSTM-GRN to develop a structure to cover the target area, efficiently collecting many targets.

4 Results

The aim here is to first validate the basic principle that a stopping criteria can be met by a recurrent system and then go on to evaluate these established abilities in a more complex problem.

4.1 Single-Point Target

The organism begins as a point in the bottom left of the arena shown in Figure 2. Through each Devo step, the LSTM-GRN outputs a learnt step size and direction in which to grow. Fitness, measured in the final Devo step, is given by the distance between the final point reached and the *food* and the number of steps taken. In this series of experiments, the position of the target is fixed at the top of the arena, and the organism is blind to its location; the only feedback that reaches the organism of its quality is after the devo process is complete. Throughout the evolution of the organism, it must evolve the correct series of steps to efficiently reach the target and, crucially, when to stop growing so as not to expend unnecessary energy.



(a) Gen: 1, steps - 100, distance - 57.65 (b) Gen: 40, steps - 17, distance - 4.15 (c) Gen: 70, steps - 7, distance - 1.51

Fig. 2: An example of some of the visual selected best individuals evolved over 3 of the 100 generations. The organism can be seen to go through a number of stages of evolution; first finding the target and then optimising its path to the target. This experiment was carried out 7 times and achieved similar results in each run.

The evolutionary process was given 100 generations and a maximum of 100 development steps. The final and best individual terminated its development cycle after 7 steps and achieved a final distance from the target of 0.29 units. Individuals from three different generations produced through an evolutionary run are shown in Figure 2. Starting from a single point with little growth to a long indirect path to the target, before finally finding the shortest path and refining the step size to hit the centre of the target around generation 70. This illustrates that the LSTM-GRN presented here is capable of balancing both the pathfinding and termination criteria to encode a solution.

These early results suggest that it is possible to evolve a GRN that can perform a stopping operation in an appropriate manner. However, it is important to note that this is a relatively simple and crucially static test; The target is in a fixed location, so location information can be encoded directly into the genome and *saved* via evolution not development. To provide a more challenging problem, the LSTM-GRN should be able to interpret spatial input based on its environment, which will be addressed with more complex issues in the next section.

4.2 Point Cloud

In this series of experiments, more targets are placed in the environment and the organism must be able to sense its surroundings and follow routes efficiently to collect the *food*.

Although the organism's ability remains the same as in the previous experiment, it now has the ability to push and pop its current location to and from a stack in memory, allowing for branching growth. Another ability it is given is a look-ahead vector pointing to its nearest food source. This is a critical piece of information as the points are no longer fixed between evaluations and are distributed randomly within a specified radius.

Importantly, the organism retains the ability to terminate its growth, reducing the number of steps it has to take. An upper bound of 1,000 devo steps was imposed to make the experiments tractable, this is compared to the previous 100 devo steps.

The results presented in Figure 3 show solutions taken from the Pareto front of the final population. The phenotype does present reasonable coverage of the environment and notably skips around the gap in the middle of the space. However, the key feature that must be investigated is the stopping of the growth process at an appropriate point.

It is apparent that although an acceptable path has been discovered through the environment, the organism has not terminated its growth. However, there are a number of potential reasons why this may be the case. There are many more targets in the arena that it may try to collect; this increase in the *score* fitness may be winning out in the selection process over any organisms that halt before this is done. This is a feature that can be seen in a number of results.

The final Pareto front (Figure 4) from another run of this experiment yielded a typical distinct split across the solution space, and although it is clear there are changes across generations (Figure 5), it is not yet clear how much of the solution space is reachable with the current configuration of the LSTM-GRN.

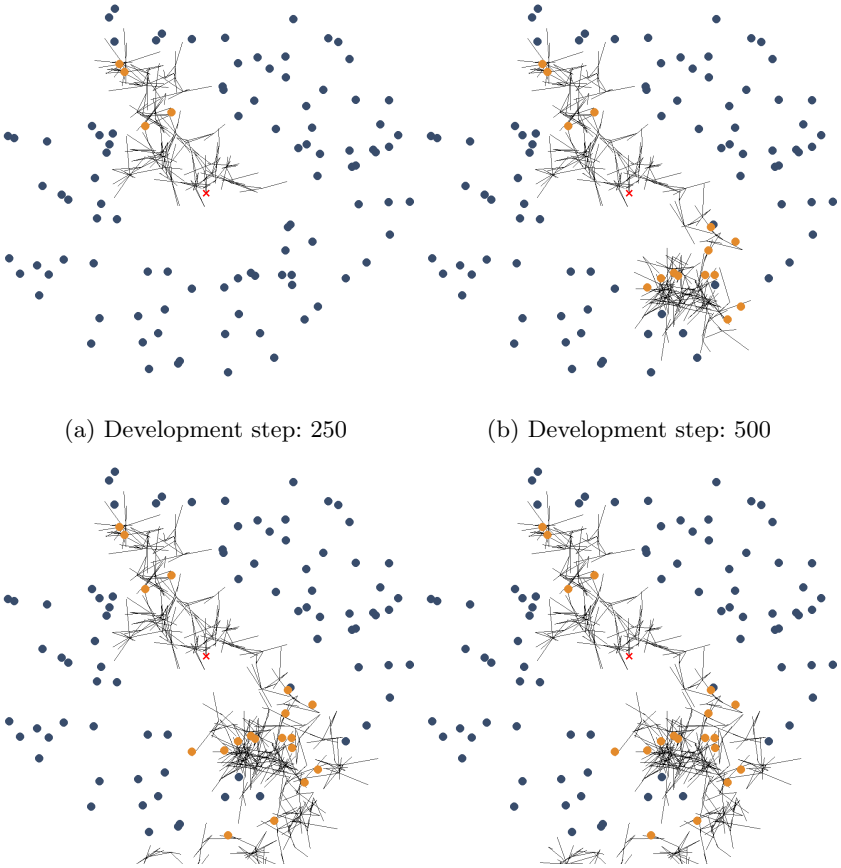
A range of phenotypes emerge in the final population. A consistent trait through the populations of this task is the large circles that are present in Figure 6a. These structures are large and repetitive, but they cover a large area and collect a good number of targets, as highlighted in orange. These structures emerge frequently when no termination action is activated in the majority of the population.

Figure 6b is similar to the experiment run for Figure 3, however, in this iteration of the experiment, the LSTM-GRN has a limited distance in which it can see the direction of its nearest target. This results in large swooping arcs compared with the unconstrained look-ahead of quite sharp turns. This is likely due to it often lacking an understanding of its environment, and as such the best strategies to maximise score involve finding ways to cover the space as much as possible.

4.3 Structured Cloud

The final experiment replaces the random field of targets with deliberate structures. In this, the targets are of the form of capital letters Z and a straight line. There are several reasons why these layouts were chosen. Letters of the alphabet have been used in the past as a reference point for evolvable generative systems, such as in the Biomorphs example [4]. However, mainly it gives some complexity to the task of finding the shortest path, similar to solving mazes. Within the straight line, a very clear optimal route is developed, with the letter Z that also provides the challenge of tight turns for exploration.

Figure 7a is the first example of a response to a deliberate structure in the environment. As before, the organism starts in the middle of the arena, and the expected structure would be for the branching to occur towards the top right



(a) Development step: 250 (b) Development step: 500

(c) Development step: 750 (d) Development step: 1000

Fig. 3: An example of branches developing when directional distance information is given to the network about the nearest point (unconstrained distance). Growth starts in the centre of the arena.

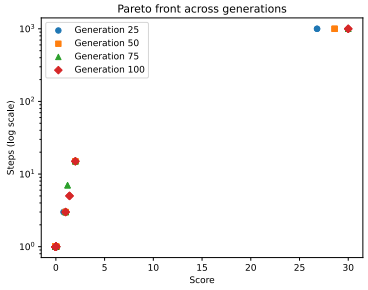


Fig. 4: The solutions generated in this run occupy tight clusters suggesting difficult local minima to escape. The final generation is marked with the red diamond.

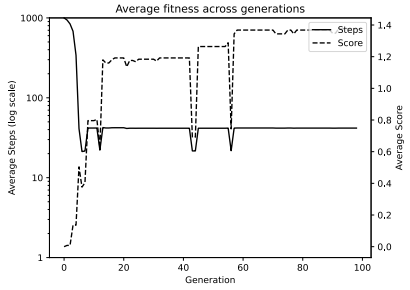
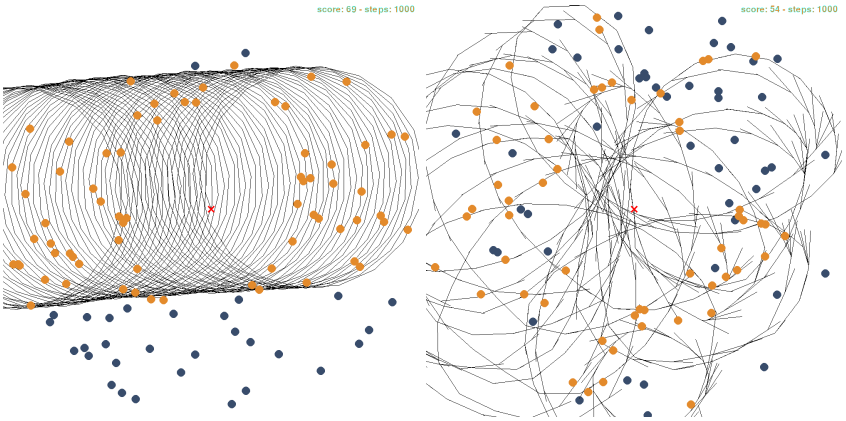


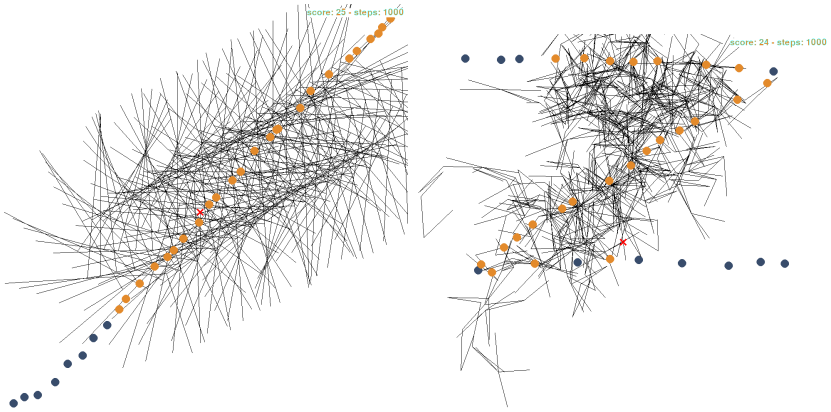
Fig. 5: Average of both fitness measures across 100 generations - steps (minimised) and score (maximised). The average step stabilises around generation 20 as the populations become fixed at the maximum or minimum number of steps. The score shows good improvement over the generations.



(a) Large circles commonly evolved a strategy to maximise score in a landscape few networks perform well minimising the number of steps.

(b) A branching organism developed when directional distance information given to the GRN is constrained to within 300 units.

Fig. 6: Two common results from the point cloud test



(a) The organism explores a straight line from a centre point out in both directions - it explores more effectively towards the top right, neglecting the bottom left.

(b) In order to cover the area of the letter Z, the organism behaves similarly to the examples seen when presented with a point cloud.

Fig. 7: Two examples of structure being latent within the solution space.

and bottom left. The actual phenotype that evolves is not significantly different from the expected result. However, it does take a lot of steps exploring either side of the line rather than taking advantage of the proximity of the nearby targets.

The quality of the exploration strategy of the organism is more difficult to evaluate qualitatively here due to the tight structure of the targets. The organism clearly does not adhere rigidly to the intended structure. However, the area below the bottom line of the letter Z (Figure 7b) was briefly explored; then abandoned and wasteful areas of exploration, such as the acute top angle of the structure, are quite close to each other, so it would not be unreasonable to search in space for more densely packed targets. Also note that the LSTM-GRN was unable to stop earlier than the termination criteria allowed.

5 Discussion

The LSTM-GRN presented here can certainly evolve to stop, as shown in the first experiment; however, as the complexity grows, it becomes harder to interrogate small changes in efficacy. It may simply be a case of expanding the evolutionary search strategy in selection or mutation or perhaps allowing some change to the topology of the GRN. It is not unreasonable to expect this incongruity, as it has been pointed out that the search space for evolving memory is *"deceptively large"* [20]. It is also likely that it is simply challenging to escape the local minima of the fitness space.

An important distinction to make is the difference in complexity between the two experiments outlined here. The success of the more simple experiment in halting the devo process does suggest that this problem space becomes quite challenging to search over, with many local minima seen throughout the solution space.

There are a number of explanations that resolve the discrepancy presented between these experiments. One such explanation is that the phenotypically controlled organism never found all the points in the environment, and, given the current experimental set-up, this may be a requirement to begin minimising the step fitness. That is to say, there must first be a path to a maximum score fitness before the evolutionary strategy is able to optimise the route thus minimising the steps fitness - as seen in the first experiment.

Another limitation of this fitness strategy is that the collection of targets is a binary measure. Using some proximity such that almost every mutation in development process has some immediate feedback on quality may help guide the search.

There is also an argument to suggest that the more complex problem is simply poorly suited to this type of solution. If this were to be used, for example, in the context of designing supports for a bridge, there may be some requirement to have targets to grow to, but also some fixed boundary definitions. And fitness may need to be based on other characteristics required by that of a bridge, such as physical properties.

The question also arises of when the LSTM-GRN is learning to stop or optimising to stop similar to questions on generalisability. Using the steps taken as a fitness measure may lean into this idea that it's simply an optimisation process, rather than the LSTM-GRN learning to dynamically grow to its environment. There is a great deal of nuance between these two ideas, and while they are not the same thing, they are quite closely coupled. The inclusion of proxy fitness measures may aid this, for example, including some amount of energy an organism can use up and replenish within the environment.

6 Future Work

Many more avenues for exploration are possible and this paper only scratches the surface of deep integration of recurrent systems into Evo-Devo environments and generative designs stopping their growth.

Within the current framework, it would be valuable to adjust the fitness function in order to guide the evolution process in a more granular way. For example, including an amount of reward for proximity to a target would resolve the issue with the binary scoring system. This makes small changes to the organism's growth more likely to receive either positive or negative feedback, allowing these small mutations time to be taken advantage of through generations.

This problem of learning to stop or optimising to stop can be a challenge to extract; the simplest way would be to remove any idea of duration from the fitness function. In this way, the system is no longer optimising for stopping, it

would have to learn when a stopping condition was met. One premise is the idea of giving the organism some amount of energy. It can move around and collect targets and gain more energy, or it can terminate with its current energy level. To encourage termination, a strong penalty can be included for an organism running out of energy or '*dying*'. Removing direct references to stopping early in the fitness function now removes one-half of the challenge of preventing a GRN simply optimising for stopping. Then allowing the GRN to decide when to stop based on energy levels and the environment. Upon a search of the literature there seems to be little discussion of these artificial ideas of energy within the context of EvoDevo algorithms.

7 Conclusion

While it is clear that including reccurency does allow termination criteria to be handled internally by a GRN in more simple environments, it remains to be seen whether this is true for much more complex generative systems. There is clear evidence for the efficacy of the LSTM-GRN in handling generative tasks, but it will become increasingly important to set a benchmark for how stopping is handled across a number of GRN constructs.

It becomes clear through consideration of the results here that optimising to stop and learning when to stop are similar but slightly different tasks. It remains to be seen what the best way to ensure a system is learning when to stop, but it is apparent a development in this area will significantly aide the understanding of stopping in generative systems as a whole.

8 Acknowledgements

The authors acknowledge the support of a School-funded PhD studentship and the support of the EPSRC project RIED EP/V007335/1

References

1. Banzhaf, W.: On the dynamics of an artificial regulatory network. In: Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P., Kim, J.T. (eds.) Advances in Artificial Life. pp. 217–227. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
2. Broughton, T., Tan, A., Coates, P.S.: The use of genetic programming in exploring 3D design worlds. In: CAAD futures 1997. pp. 885–915. Springer Netherlands (1997)
3. Davidson, E.H.: Genomic Regulatory Systems: In Development and Evolution. Elsevier (Jan 2001)
4. Dawkins, R.: The evolution of evolvability. On growth, form and computers pp. 239–255 (2003)
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (Apr 2002)

6. Deb, K., Agrawal, R.B., Others: Simulated binary crossover for continuous search space. Complex Systems (1994)
7. Deb, K., Sindhya, K., Okabe, T.: Self-adaptive simulated binary crossover for real-parameter optimization. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. pp. 1187–1194. GECCO '07, Association for Computing Machinery, New York, NY, USA (Jul 2007)
8. Eggenberger, P.: Evolving morphologies of simulated 3d organisms based on deferential gene expression. In: Harvey, I., Husbands, P. (eds.) Proceedings of the 4th European Conference on Artificial Life. pp. 205–213. Springer (1997)
9. Fortin, F.A., De Rainville, F.M., Gardner, M.A., Parizeau, M., Gagné, C.: DEAP: Evolutionary algorithms made easy. J. Mach. Learn. Res. (2012)
10. Frazer, J.: Chapter 9 - creative design and the generative evolutionary paradigm. In: Bentley, P.J., Corne, D.W. (eds.) Creative Evolutionary Systems, pp. 253–274. Morgan Kaufmann, San Francisco (Jan 2002)
11. Hickinbotham, S., Dubey, R., Friel, I., Colligan, A., Price, M., Tyrrell, A.: Evolving design modifiers. In: 2022 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1052–1058 (Dec 2022)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (Nov 1997)
13. Hornby, G.S., Lipson, H., Pollack, J.B.: Evolution of generative design systems for modular physical robots. In: Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164). vol. 4, pp. 4146–4151 vol.4. IEEE (2001)
14. Hornby, G.S., Pollack, J.B.: Body-brain co-evolution using l-systems as a generative encoding. In: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation. pp. 868–875. GECCO'01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (Jul 2001)
15. Jozefowicz, R., Zaremba, W., Sutskever, I.: An empirical exploration of recurrent network architectures. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. pp. 2342–2350. ICML'15, JMLR.org (Jul 2015)
16. Lohmann, I.: The birth of evo-devo. Nat. Rev. Mol. Cell Biol. **24**(5), 311 (May 2023)
17. Ollion, C., Pinville, T., Doncieux, S.: With a little help from selection pressures: evolution of memory in robot controllers. In: Artificial Life 13. MIT Press (Jul 2012)
18. Phon-Amnuaisuk, S., Salleh, N.D.H.M., Woo, S.L.: Pixel-Based LSTM generative model. In: Computational Intelligence in Information Systems. pp. 203–212. Springer International Publishing (2019)
19. Potash, P., Romanov, A., Rumshisky, A.: GhostWriter: Using an LSTM for automatic rap lyric generation. In: Màrquez, L., Callison-Burch, C., Su, J. (eds.) Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 1919–1924. Association for Computational Linguistics, Lisbon, Portugal (Sep 2015)
20. Rawal, A., Miikkulainen, R.: Evolving deep LSTM-based memory networks using an information maximization objective. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016. ACM, New York, NY, USA (Jul 2016)
21. Rawal, A., Miikkulainen, R.: From nodes to networks: Evolving recurrent neural networks (Mar 2018)

22. Secretan, J., Beato, N., D'Ambrosio, D.B., Rodriguez, A., Campbell, A., Folsom-Kovarik, J.T., Stanley, K.O.: Picbreeder: a case study in collaborative evolutionary exploration of design space. Evol. Comput. **19**(3), 373–403 (May 2011)
23. Tero, A., Takagi, S., Saigusa, T., Ito, K., Bebber, D.P., Fricker, M.D., Yumiki, K., Kobayashi, R., Nakagaki, T.: Rules for biologically inspired adaptive network design. Science **327**(5964), 439–442 (Jan 2010)