The Evolution of Evolvability in Evolutionary Robotics

David Shorten[1] and Geoff Nitschke[1]

[1]Department of Computer Science, University of Cape Town
dshorten@cs.uct.ac.za, gnitschke@cs.uct.ac.za

Abstract

Previous research has demonstrated that computational models of *Gene Regulatory Networks* (GRNs) can adapt so as to increase their *evolvability*, where evolvability is defined as a population's *responsiveness* to environmental change. In such previous work, phenotypes have been represented as bit strings formed by concatenating the activations of the GRN after simulation. This research is an extension where previous results supporting the evolvability of GRNs are replicated, however, the phenotype space is enriched with time and space dynamics with an *evolutionary robotics* task environment. It was found that a GRN encoding used in the evolution of a way-point navigation behavior in a fluctuating environment results in (robot controller) populations becoming significantly more responsive (evolvable) over time. This is as compared to a direct encoding of controllers which was unable to improve it's evolvability in the same task environment.

Introduction

An open question in artificial and natural life is whether digital and natural organisms undergoing an evolutionary process are able to become more *responsive* to changes in their environment, that is to become more *evolvable* (Wagner and Altenberg, 1996a). A prevailing hypothesis is that if the environment sufficiently varies over time, then organisms evolve the ability to be able to evolve suitable adaptations to such environmental changes faster (Wagner and Altenberg, 1996a; Draghi and Wagner, 2008). Crombach and Hogeweg (2008) as well as Draghi and Wagner (2008) have demonstrated that computational models of *Gene Regulatory Networks* (GRNs) exhibit such *evolvability*. This study's main goal is to replicate the results of this previous research (Crombach and Hogeweg, 2008; Draghi and Wagner, 2008), but in the context of *evolutionary robotics* (Nolfi and Floreano, 2000) experiments that test robot controller (behavior) evolution in environments where the goal tasks vary over time.

The representation problem in *Evolutionary Computation* (EC) (Eiben and Smith, 2003) addresses the issue of how to represent and adapt (mutate and recombine) genotypes such that a broad range of complex solutions are represented by relatively simple genotype encodings (Wagner and Altenberg, 1996b). Representation choice and associated operators has a significant impact on the evolution of viable solutions and representations which facilitate evolution have been termed *evolvable* (Wagner and Altenberg, 1996b; Rothlauf, 2006a; Simões et al., 2014). Similarly, in nature, genetic information defining the form and function of an organism is stored within its genotype, however the developmental process which translates this information into phenotypes is not well understood (Pigliucci, 2010). It has become clear that the mapping between genotype and phenotype is neither one-to-one nor linear (Gjuvsland et al., 2013). In many organisms, including the case of *Ribonucleic acid* (RNA) folding (Draper, 1992), it has been found that many genotypes can code for a single phenotype and that genetic change resulting from mutation is not proportional to phenotypic change (Pigliucci, 2010; Parter et al., 2008).

In EC, this is known as a *developmental* or *generative* (indirect encoding) genotype representation (Stanley and Miikkulainen, 2003), where effects of mutations are not only determined by representation and associated mutation operators, but also by the population's position in genotype space. This is distinct from a one-to-one mapping (direct encoding) where, for a given phenotype and its associated genotype, mutational effects are determined by the representation, mutation operators and fitness function. Thus the population's location (genotype values) in the genotype space can be viewed as an integral component of representation (Rothlauf, 2006b).

An open question in biology is whether developmental representations have occurred by chance, or if such representations have also been subject to evolution (Parter et al., 2008). A current hypothesis is that an organism's genotype representation is itself evolvable due to the evolution of *evolvability* (Wagner and Altenberg, 1996b), (Pigliucci, 2008). However, this is complicated by multiple definitions of *evolvability*[1] in both evolutionary biology(Pigliucci,

[1]For a review of evolvability in biology, the reader is referred to Pigliucci (Pigliucci, 2008).

2008), (Pigliucci, 2010; Parter et al., 2008) and EC (Tarapore and Mouret, 2014), for example, evolvability can refer to either populations or individuals (Wilder and Stanley, 2015). Similarly, within EC, numerous definitions and associated metrics have been proposed. For example, those that focus exclusively on solution fitness (Grefenstette, 1999) or variability of offspring (Lehman and Stanley, 2013). Tarapore and Mouret (Tarapore and Mouret, 2014) developed a metric which incorporated both the fitness and diversity of offspring.

The principle aim of this work is to extend the demonstration of evolvability in GRN's by Crombach and Hogeweg (2008) as well as Draghi and Wagner (2008) to an evolutionary robotics domain. In these previous studies a population's evolvability was defined as its *responsiveness*, that is, the population's ability to rapidly adapt to changes in the fitness landscape. This work maintains consistency with this definition. Hence, we define evolvability to be tantamount to a population's *adaptability* (Kirschner and Gerhart, 1998). This implies that we do not predefine sets of features that will likely propagate beneficial phenotypes (behaviors) in the evolutionary process. Rather, in line with biological literature (Pigliucci, 2008; Flatt, 2005), evolvability is an organism's (genotype's) capability to adapt and survive in its environment.

We investigate evolvability in the context of an evolutionary robotics task, where robot controller (behavior) evolution is tested using both indirect (GRN) and direct genotype encodings. The hypothesis is that an indirect encoding facilitates the improvement of responsiveness over the course of evolution in a robotics task domain with changing task environments, whereas indirect encoding does not. Here we use *responsiveness* and *evolvability* interchangeably to mean the speed with which a population adapts given task environment changes. The robotics task was way-point navigation, where responsiveness was tested via having the environment fluctuate with its own task variants. In order to facilitate this, two different way-point layouts were used.

Results indicated that evolution using the indirect (GRN) encoding facilitated the evolution of controllers that were significantly more responsive (adapted) to task environment fluctuations over evolutionary time. Comparatively, evolution using the direct (bit-string) encoding applied to this task indicated that evolved behaviors were unresponsive and unable to appropriately adapt to task environment variations over time.

Methods

Simulation and Task Environment

The evolutionary robotics simulation used a bounded two-dimensional continuous environment[2], where the environment was imposed with a 400 x 200 grid for ease of specifying task environment and simulation parameters (table 1). The task tested was *way-point navigation*. During controller evolution, the task variants were switched in order to simulate a fluctuating environment (Evolutionary Algorithm Section). Two task variants were specified, each requiring the robot to pass by (within a given distance, table 1) of a pre-specified number of way-points. The task required the robot to pass the way-points in a specific order and within its lifetime (a given number of simulation time steps, table 1). The number of way-points the robot passed by during its lifetime equalled its fitness.

Figure 1 shows the layout of the way-points for each of the two task variants, where the way-points were specially positioned to encourage the emergence of a wall-following behavior.

Robot Controller

The robot controller was a fully connected feed-forward *Artificial Neural Network* (ANN) with twelve connection weights. That is, three hidden layer neurons (Sigmoidal units), connected to two sensory input and two motor output neurons. The two sensory inputs were distance sensors, each placed $\pi/3$ radians on either side of the direction in which the robot was facing. These sensors operated similar to Infrared proximity sensors. ray casting in the sensor's field of view. If this line intersected a wall in the sensor's range, then the sensor's reading was d/r, where d was the distance to the wall and r was the sensor's range. If there was no wall in sensor range, then the sensor reading was 1.0. The controller's motor outputs determined the robot's speed and heading, where outputs were normalized in the range $[0.0, 1.0]$ and corresponded to minimum and maximum speed and heading values (table 1).

Gene Regulatory Network

The *Gene Regulatory Network* (GRN) model for robot controller encoding is based on that used in previous related work (Crombach and Hogeweg, 2008; Draghi and Wagner, 2008). Nodes in the GRN are genes, and connections between the nodes are either *excitatory* or *inhibitory*. All nodes are updated synchronously via equation 1.

$$s_i(t+1) = \begin{cases} 1: & \sum_j w_{ij} s_j(t) > \theta_i \\ 0: & \sum_j w_{ij} s_j(t) \leq \theta_i \end{cases} \quad (1)$$

Where, $s_i(t)$ is the activation of the ith node at simulation iteration t, w_{ij} is the connection weight of the directed edge from the ith to the jth node, and θ_i is the threshold of the ith node. If no such connection exists then $w_{ij} = 0$. Table 2 presents the GRN parameters. In order to facilitate the conversion of activations into bit-strings all nodes were given a unique value in the range $[0, l]$, where l is one less than the number of nodes (Gene Regulatory Network Encoding section).

[2] An extension of the *RoboRobo* simulator (Bredeche et al., 2013) was used for all experiments.

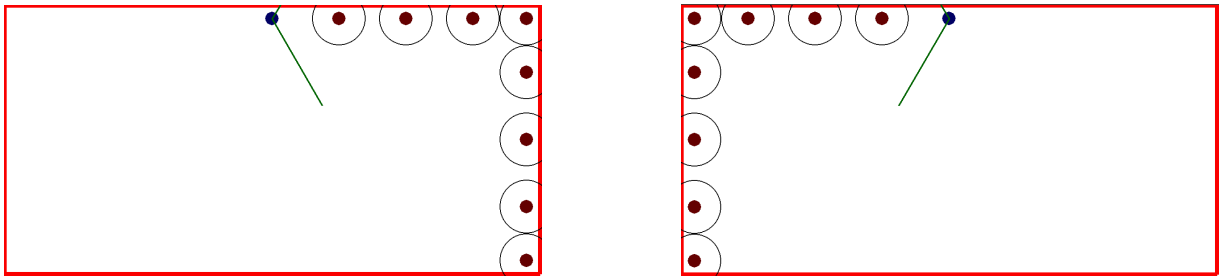

Figure 1: Visualization of the *way-point navigation task* for each of the two task variants (left and right). Way-points (brown dots) line the top-right and top-left corners of each task variant. The circles surrounding each way-point represent the distance within which the robot must pass the way-point. The robot is presented as the blue dot on either side (far left and far right) with two lines extending (representing sensor fields of view).

Mutation Operators. Table 3 specifies the mutation operators used in the *Evolutionary Algorithm* (EA) applied to evolve the GRN. The `mutate_weight`, `add_edge`, and `delete_edge` operators were applied to the GRN node connections, where for every connection, the operator was applied with a given probability (table 4). All other mutation operators acted on nodes activation and threshold functions with a given probability (table 4).

Binary (Direct) Encoding. A direct mapping function was used to map the binary genotype encoding to an ANN controller. To convert the sixty element binary string genotype to the twelve connection weight values which specify an ANN controller, the genotype string was split into twelve smaller strings of five elements each. These twelve strings were then converted into real numbers in the range $[0.0, 1.0]$ using equation 2.

$$2\left(\frac{\sum_{i\in\{0\ldots4\}} a_i 2^i}{2^5-1} - 0.5\right) \tag{2}$$

Where, a_i is the ith element of substring a.

Gene Regulatory Network (Indirect) Encoding. Using methods from previous work (Crombach and Hogeweg, 2008; Draghi and Wagner, 2008), the GRN was simulated for a given number of iterations (table 2). During this simulation time, convergence to a point attractor was tested for by determining whether node activations in the last and penultimate iterations were identical. If the GRN did not settle on a point attractor then it was marked for removal and no further evaluation took place. Preliminary testing indicated that the removal of these GRNs had a negligible effect on the evolutionary dynamics in this study's experiments. If the GRN settled on a point attractor then the node activations were converted into a bit string, where the ordering of the activations was determined by unique node identifiers (Gene Regulatory Network section). This GRN convergence test was done to maintain consistency with previous work (Crombach and Hogeweg, 2008), (Draghi and Wagner, 2008). Bit-strings were then converted into an ANN controller using the genotype to ANN direct encoding mapping method (Binary Encoding section, equation 2).

Evolutionary Algorithm (EA)

The binary and GRN encoded genotypes were evolved with an EA using deterministic tournament selection (Eiben and Smith, 2003), applied 200 times per generation. Also, the EA used mutation only (there was no recombination operator). Table 4 presents the EA parameters. The following subsections detail the EA setup for controller evolution using the binary and GRN encodings, respectively.

Direct Encoding When the EA was started, a population of bit-string genotypes were randomly generated. Each generation, each genotype was systematically selected, decoded into an ANN controller (Binary Encoding section) and tested in the way-point navigation task (Simulation and Task Environment Section) for one *robot lifetime* (table 1), after which fitness was assigned to the tested controller (genotype). One generation was when all genotypes had been tested and evaluated. Selection and mutation operators were then applied (table 4).

In preliminary mutation operator testing it was found that using a constant mutation rate for each bit (gene) in each genotype resulted in a significantly lower task performance compared to controller evolution with GRN encoding. That is, at a high mutation rate, genotypes with high fitness were quickly found, however, convergence was sub-optimal. For relatively low mutation rates, the population converged to a set of fit genotypes, however genotypes with optimal fitness were not found. To address this, we executed 100 evolutionary runs of controller evolution with GRN encoding and recorded the Hamming distance between parent and child genotypes (Gene Regulatory Network Encoding section). The probability of each Hamming distance (number of bit-flips) occurring was then calculated and these probabilities were used to determine the number of bit-flips in a mutation of the binary direct encoding. That is, if on average, the as-

Parameter	Value
Robot speed	5 units per iteration
Robot maximum angular velocity	0.5 radians per iteration
Robot heading	$[0, 2\pi)$
Sensor range	75 units
Collision radius	20 units
Environment size	400 x 200 units
Way-point radius (navigation task)	20 units
Number of way-points	$[0, 59]$
Simulation iterations (robot lifetime)	250

Table 1: Simulation and task parameter values.

Parameter	Value Range
GRN Weights (w_{ij})	$[-2.0, 2.0]$
ANN Weights	$[-1.0, 1.0]$
ANN sensory and outputs	$[-1.0, 1.0]$
Thresholds (θ_{ij})	$[-3, 3]$
Number of nodes	60
Incoming / outgoing connections per node	$[0, 59]$
Simulation iterations (maximum t)	20

Table 2: Parameters for the *Binary* and *Gene Regulatory Network* controller encoding.

Operator	Description
`mutate_weight`	A new value for the weight of an edge is chosen from the allowed values.
`mutate_activation`	The value of the initial activation of a node is flipped.
`mutate_threshold`	A new value for the threshold of a node is chosen from the allowed values.
`add_edge`	A new incoming edge is added to the node. It connects to a random node and has a random weight.
`delete_edge`	One of the node's edges is chosen at random and removed.

Table 3: Mutation operators for the Gene Regulatory Network.

sociated bit strings of GRNs had a Hamming distance of h from their parent genotypes (with probability p), then when a binary encoded genotype was mutated, h bits would be flipped with probability p. This probability distribution was then assigned as the mutation rate for the direct encoding approach. Table 5 presents the probability of a given number of bit-flips occurring whenever the binary encoding mutation operator was applied. It was found, when these probabilities were used (table 5), that the task performance of controller evolution with direct binary encoding was comparable to the task performance to early generations of controller evolution using the GRN encoding (Results section).

Gene Regulatory Networking Encoding When the EA was started, GRNs were randomly initialized with given parameter constraints, and each GRN simulated for a given number of iterations (table 2). If the GRN settled on a point attractor, then the GRNs activations were mapped to a bit-string, which was then decoded into an ANN controller (Gene Regulatory Network Encoding section). Each decoded ANN controller was then simulated in the way-point navigation task (Simulation and Task Environment Section) for one *robot lifetime* (table 1), and fitness assigned to the tested genotype. One generation was when all genotypes had been tested and evaluated. Selection and mutation operators were then applied. The mutation operators specified in table 3 were applied to every node of the child GRNs with the probability specified in table 4. However, the `mutate_weight` operator was applied to every connection of the GRNs with a lower probability (table 4).

Experiments, Results, Discussion

Controller evolution was run for 10000 generations in the *way-point navigation* task, where robot controllers were encoded using direct binary or indirect GRN encoding. In order to investigate the conditions facilitating evolvability in this evolutionary robotics case study, task variants were switched every 200 generations. Hence, given these two controller encodings, two different evolutionary setups were run, where each setup was run 100 times to ensure viability of statistical tests on results data.

Figure 2 presents task performance (average and best fitness) results for the way-point navigation task. Table 6 presents average fitness results and table 7 presents statistical test results for within and between comparisons of direct and GRN encoded populations. Table 7 presents statistical test results from pair-wise comparisons on average maximum and average fitness results in table 6. The Mann-

Parameter	Value
Population size	1000
Genotypes replaced per generation	100
Tournament size	4
Recombination	None
Generations per task variant switch	200
Number of generations	10000
Binary encoding mutation	Bit-flip
Binary encoding mutation rate	See table 5
GRN mutation	See table 3
GRN weight mutation rate	0.002
GRN node mutation rate	0.02
Genotype bit-string length	60

Table 4: Evolutionary Algorithm Parameters

Bit flips	Probability
0 / 13	0.5088 / 0.0023
1 / 14	0.2466 / 0.0017
2 / 15	0.1068 / 0.0012
3 / 16	0.0471 / 0.0009
4 / 17	0.0231 / 0.0006
5 / 18	0.0139 / 0.0004
6 / 19	0.0103 / 0.0003
7 / 20	0.0087 / 0.0002
8 / 21	0.0076 / 0.0001
9 / 22	0.0065 / 0.0001
10 / 23	0.0053 / 0.0000
11 / 24	0.0042 / 0.0000
12 / 25	0.0032 / 0.0000

Table 5: Mutation rates for direct binary encoding. Probabilities match mutation probabilities of the binary GRN encoding (for given number of bit flips). Genotypes were sixty gene bit-strings, however, mutations of more than twenty-three bit flips never occurred

Whitney U test ($p < 0.01$) with Bonferroni correction (Flannery et al., 1986) for multiple comparisons was applied to gauge statistical significance.

Table 6 presents average and maximum fitness results. These fitness results are presented for *early* and *late* stages of the evolutionary process. Early stages were at generation 25 and 425 and late stages were at generations 9225 and 9625. These generation intervals were chosen as they were an eighth into the allotted generations for task variant one, and were deemed a good measure of the population's early response to the environmental change early and late in the artificial evolution process.

One may note that we measure average and maximum fitness, rather than the rate of fitness change (relative fitness) as an indicator of a population's responsiveness. We elected to use absolute fitness, since measuring relative fitness would unfairly benefit genotypes whose fitness dropped the most after a change in the task environment. That is, given two populations, the one with the highest fitness a given interval after a task change is concluded to be the most responsive. Also, this interpretation holds in the case that the other population suffers a greater fitness decrease after the task change which might subsequently lead it to having a faster fitness increase.

In figure 2 one may observe that populations using the GRN encoding were able to significantly improve (with statistical significance, table 7) their responsiveness between early and late stages of the evolutionary process. Here we use *responsiveness* and *evolvability* interchangeably to mean the speed with which a population adapts given task environment changes (task variants). The responsiveness of populations using the direct encoding was statistically comparable during both the early and late stages of the evolutionary process (table 7).

Hence, statistical tests run between average and average maximum fitness values of populations using the GRN encoding in the early and late stages of evolution (table 7) confirm that in the late stages of evolution, populations respond more quickly to environmental change (task variants). Results indicate that populations using the GRN encoding were significantly more responsive in the late generations compared to populations using the direct encoding (table 7). Also, there was not a significant difference in the responsiveness of populations using direct encoding between the early and late stages of evolution. This is supported by previous work (Crombach and Hogeweg, 2008; Draghi and Wagner, 2008) and supports this study's hypothesis that direct genotype encoding does not facilitate responsiveness in changing environments (Introduction section). Although, in the early stages of evolution, the average fitness of populations using the direct encoding was more responsive than populations using the GRN encoding (figure 2).

In terms of the responsiveness of populations using the direct encoding, there was no significant difference between the average and average maximum fitness values of these populations between early and late stages of evolution (table 7). This indicates that direct binary encoded populations did not evolve a responsiveness to the changing task environment of this way-point navigation task, as was observed in the case of GRN encoded populations (table 7). However, in terms of the average fitness, the direct encoding was significantly more responsive in the early stages of evolution.

Thus, results demonstrate that a GRN encoding of populations of robot controllers (behaviors) evolve to become more

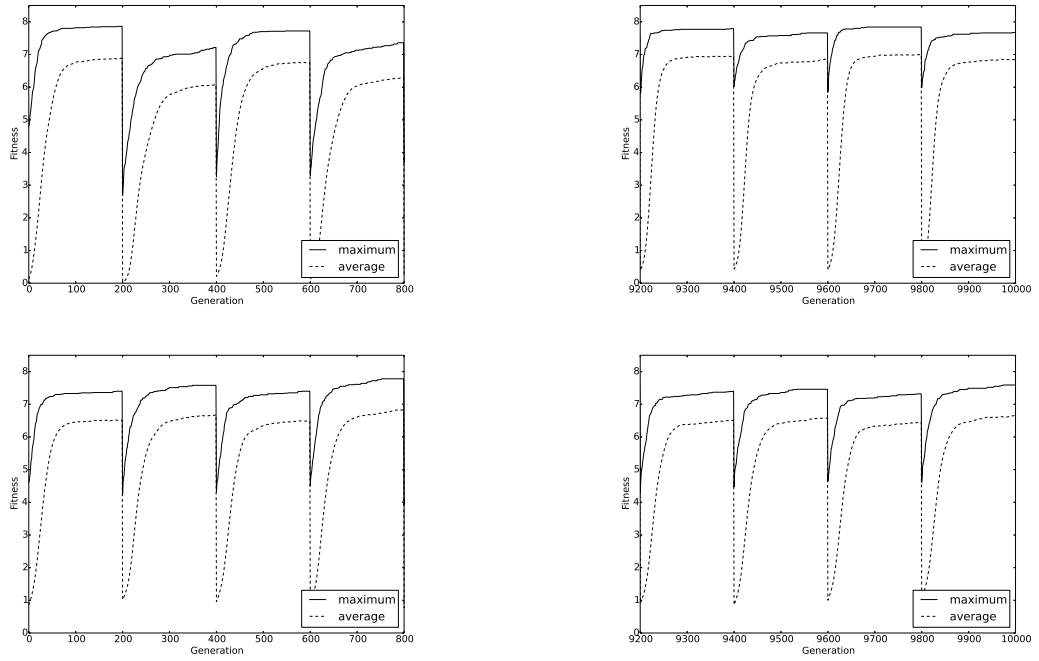

Figure 2: Average and maximum fitness for evolved GRN (top left, right) and binary encoded (bottom left, right) controllers in the *way-point navigation* task.

responsive to changes in their task environment. That is, a statistically significant difference (for *average* and *maximum* fitness values) was observed in the improvement of responsiveness of GRN versus directly binary encoded populations in the *way-point navigation* task. The contribution is that results support previous work on the efficacy of GRN encodings for conferring evolvability in changing task environments (Crombach and Hogeweg, 2008), as well as extending previous work (Crombach and Hogeweg, 2008; Draghi and Wagner, 2008) into an evolutionary robotics task environment with both time and space dynamics.

A goal of this study was to investigate the environmental and evolutionary conditions that facilitate the evolution of evolvability. Previous researchers have demonstrated that many-to-one genotype-to-phenotype mapping (redundant mappings) result in evolvability in EAs (Shipman, 1999; Ebner et al., 2001a) as well as increased EA task performance. That is, a highly redundant mapping enables some mutations to have negligible impact on the fittest phenotypes, meaning the EA is better able explore the search space via neutral networks (Ebner et al., 2001b).

Redundancy, and the closely related notion of robustness (Wagner, 2005), is theorized to have played a key role in the increased responsiveness of evolved GRN encoded behaviors, given that the GRN encodings are more redundant than the direct encodings. During their evaluation, GRN's are decoded into bit-strings before these bit-strings are decoded into ANN's, which are then evaluated by the EA. Given that the decoding from GRN's into bit-strings is a many-to-one mapping, the redundancy in the GRN encoded search space is considerably higher than that in the directly encoded search space.

That is, the bit-strings are sixty characters long, which implies that there are $2^{60} \approx 10^{18}$ genotypes in this space. In the GRNs, each node has fifty-nine possible connections and each connection can either connect to one of the sixty nodes or not connect. This implies that each node has 59^{61} possible configurations. Note that although many of these configurations are equivalent, where the only difference is the ordering of the connections, each one forms a distinct encoding and thus represents part of a distinct genotype. There are sixty nodes, so the number of genotypes in this space is $\left(59^{61}\right)^{60} \approx 10^{6000}$.

Also, consider that in this study's way-point navigation task, there were only nine fitness values, where fitness was equated with how many of eight way-points a robot passed in its lifetime. The ninth fitness value was to account for the robot not passing any way-points (Simulation and Task Environment Section). Thus, given the GRN encoding, nine possible phenotypes (way-point navigating behaviors) were represented by a high dimension and highly redundant genotype space. That is, in this task, there were many possible genotype to phenotype (controller) mappings, where controller behavior was equated with one of nine fitness values.

Measure	Fitness
Early maximum GRN encoding	0.88 (0.18)
Late maximum GRN encoding	0.96 (0.11)
Early maximum direct encoding	0.86 (0.17)
Late maximum direct encoding	0.85 (0.17)

Measure	Fitness
Early average GRN encoding	0.33 (0.11)
Late average GRN encoding	0.45 (0.16)
Early average direct encoding	0.39 (0.15)
Late average direct encoding	0.39 (0.16)

Table 6: Maximum (**left**) and average (**right**) fitness and standard deviations (in parentheses) for *way-point navigation* for *early* and *late* stages of evolution. Results have been normalized, where given values are portions of the minimum and maximum possible task performance: 0 and 1.0, respectively. Early stages were at generation 25 and 425 and late stages were at generations 9225 and 9625.

	EM GRN	LM GRN	EM DE	LM DE	EA GRN	LA GRN	EA DE	LA DE
EM GRN	•	✓	✗	•	•	•	•	•
LM GRN	✓	•	•	✓	•	•	•	•
EM DE	✗	•	•	✗	•	•	•	•
LM DE	•	✓	✗	•	•	•	•	•
EA GRN	•	•	•	•	•	✓	✓	•
LA GRN	•	•	•	•	✓	•	•	✓
EA DE	•	•	•	•	✓	•	•	✗
LA DE	•	•	•	•	•	✓	✗	•

Table 7: Statistical test results from pair-wise comparisons on average fitness results in table 6. ✓ signifies a statistically significant difference between two data-sets ($p < 0.01$) using the Mann-Whitney U test and Bonferroni correction. ✗ signifies that the difference between two data-sets is not significant and • signifies that a test was not done. EM is an abbreviation for *Early Maximum* (average maximum fitness in early evolution), LM is *Late Maximum*, EA is *Early Average* (average fitness in early evolution), LA is *Late Average*, and DE is *Direct Encoding*.

It is theorized that the larger size of the space of the GRN encoding is the cause of it exhibiting lower evolvability in the early stages of evolution. That is, given that phenotypes may not be uniformly distributed over the space (Pigliucci, 2010; Parter et al., 2008), finding target phenotypes after initialization may be more challenging. During the course of evolution, however, the population can move to areas of the space biased towards the targets. Moreover, other work has shown that direct encodings do exhibit a baseline of evolvability that is comparable to certain generative encodings (Tarapore and Mouret, 2015).

This study's results are also supported by related work (Ciliberti et al., 2007), that similarly modeled GRNs, where GRN instances were individual genotypes decoded into expression patterns (phenotypes). Ciliberti et al. (2007) discovered that such a GRN encoding was robust (and redundant) as a large number of genotypic changes had no phenotypic impact.

To demonstrate this for our experimental results, current research is investigating the relationship between robustness, redundancy and evolvability for the GRN versus directly encoded search spaces. This is being done for way-point navigation and more complex evolutionary robotics tasks.

Conclusion

This research presented an evolutionary robotics study that replicated and extended previous work testing the *evolvability* of populations of *Gene Regulatory Networks* (GRNs). Evolvability was defined as a population's speed of adaptation to changing task environments. Direct binary encodings of robot controllers were compared to indirect GRN encodings in controller evolution to accomplish a way-point navigation task. Task variants were alternated during controller evolution to confirm previous results that GRNs facilitate the emergence of evolvability in environments with alternating tasks. Results indicated that, for the GRN encoding approach, populations became significantly more adapted to task variation over time, and thus evolvable. This was compared to a direct encoding of controllers which was unable to achieve a high level of evolvability in the same task environment. This work thus demonstrates that the previous results are valid in a substantially more complicated domain and suggests approaches for aiding robots in dealing with dynamic environments. The findings were theorized to be a result of increased redundancy and robustness of the indirect GRN encoding of the search space. However, definitively demonstrating increased robustness and redundancy resulting in increased evolvability for the GRN encoding of this and other more complex evolutionary robotics tasks remains the subject of ongoing research.

References

Bredeche, N., Montanier, J.-M., Weel, B., and Haasdijk, E. (2013). Roborobo: A fast robot simulator for swarm and collective robotics. *arXiv preprint*.

Ciliberti, S., Martin, O., and Wagner, A. (2007). Innovation and robustness in complex regulatory gene networks. *Proceedings of the National Academy of Sciences*, 104(34):13591–13596.

Crombach, A. and Hogeweg, P. (2008). Evolution of evolvability in gene regulatory networks. *PLoS Comput Biol*, 4(7):e1000112.

Draghi, J. and Wagner, G. P. (2008). Evolution of evolvability in a developmental model. *Evolution*, 62(2):301–315.

Draper, D. (1992). The rna-folding problem. *Acc. Chem. Res.*, 25(4):201–207.

Ebner, M., Langguth, P., Albert, J., Shackleton, M., and Shipman, R. (2001a). On neutral networks and evolvability. In *Proceedings of the 2001 Congress on Evolutionary Computation*, pages 1–8. IEEE.

Ebner, M., Shackleton, M., and Shipman, R. (2001b). How neutral networks influence evolvability. *Complexity*, 7(2):19–33.

Eiben, A. and Smith, J. (2003). *Introduction to Evolutionary Computing*. Springer, Berlin, Germany.

Flannery, B., Teukolsky, S., and Vetterling, W. (1986). *Numerical Recipes*. Cambridge University Press, Cambridge, UK.

Flatt, T. (2005). The evolutionary genetics of canalization. *The quarterly review of biology*, 80(3):287–316.

Gjuvsland, A., Vik, J., Beard, D., Hunter, P., and Omholt, S. (2013). Bridging the genotype-phenotype gap: what does it take? *J Physiol.*, 591(8)::2055–2066.

Grefenstette, J. (1999). Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3. IEEE.

Kirschner, M. and Gerhart, J. (1998). Evolvability. *Proceedings of the National Academy of Sciences*, 95(15):8420–8427.

Lehman, J. and Stanley, K. (2013). Evolvability is inevitable: Increasing evolvability without the pressure to adapt. *PloS one*, 8(4):e62186.

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines.* MIT Press, Cambridge, USA.

Parter, M., Kashtan, N., and Alon, U. (2008). Facilitated variation: how evolution learns from past environments to generalize to new environments. *PLoS Comput Biol*, 4(11):e1000206.

Pigliucci, M. (2008). Is evolvability evolvable? *Nature Reviews Genetics*, 9(1):75–82.

Pigliucci, M. (2010). Genotype phenotype mapping and the end of the genes as blueprint metaphor. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1540):557–566.

Rothlauf, F. (2006a). Introduction. In *Representations for Genetic and Evolutionary Algorithms*, pages 1–7. Springer Berlin Heidelberg.

Rothlauf, F. (2006b). *Representations for Genetic and Evolutionary Algorithms*. Springer-Verlag, Berlin, Germany.

Shipman, R. (1999). Genetic redundancy: Desirable or problematic for evolutionary adaptation. In *Proceedings of the 4th International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 337–344. Springer.

Simões, L. F., Izzo, D., Haasdijk, E., and Eiben, A. E. (2014). Self-adaptive genotype-phenotype maps: neural networks as a meta-representation. In *Parallel Problem Solving from Nature–PPSN XIII*, pages 110–119. Springer.

Stanley, K. and Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130.

Tarapore, D. and Mouret, J.-B. (2014). Comparing the evolvability of generative encoding schemes. In *Proceedings of ALife*, pages 1–8. MIT Press.

Tarapore, D. and Mouret, J.-B. (2015). Evolvability signatures of generative encodings: beyond standard performance benchmarks. *Information Sciences*, 313:43–61.

Wagner, A. (2005). Distributed robustness versus redundancy as causes of mutational robustness. *Bioessays*, 27(2):176–188.

Wagner, G. and Altenberg, L. (1996a). Complex adaptations and the evolution of evolvability. *Evolution*, 50(1):967–976.

Wagner, G. and Altenberg, L. (1996b). Perspective: Complex adaptations and the evolution of evolvability. *Evolution*, pages 967–976.

Wilder, B. and Stanley, K. (2015). Reconciling explanations for the evolution of evolvability. *Adaptive Behavior*.