Evolving Gene Regulatory Networks for Real Time Control of Foraging Behaviours

Michał Joachimczak[1] and Borys Wróbel[1,2]
[1]Computational Biology Group, Institute of Oceanology, Polish Academy of Sciences
[2]Laboratory of Bioinformatics, Adam Mickiewicz University in Poznań, Poland
{mjoach,bwrobel}@iopan.gda.pl

Abstract

We use a genetic algorithm to obtain artificial gene regulatory networks (GRNs) controlling real time behaviour of artificial agents (animats) that gather food resources in a 2D environment. We build a system in which evolving GRNs are encoded in linear genomes. The encoding allows to determine which transcriptional factors (TFs) interact with which regulatory regions (promoters) to form a GRN. The sensory information is provided to an animat as externally driven concentration of selected TFs. Concentration of selected internally produced TFs is interpreted as signals for actuators. We first consider foraging for one food source and then scale the problem up to obtain animats that are able to switch between two types of food sources and avoid the poisonous one. We show that our system is highly evolvable, even though the genome encoding is very flexible (which results in a large search space) and though continuous product accumulation and degradation causes latencies in signal processing by the networks. We then discuss the topological properties of evolved networks and their evolutionary trajectories. Our results provide a first step toward a more ambitious goal of developing an artificial ecosystem in which multiple individuals will compete for food and mates.

Introduction

Gene regulatory networks (GRNs) are an underlying control mechanism of all living cells. Artificial gene regulatory networks are build either in order to understand how biological GRNs work or in the hope of engineering biologically-inspired systems that are, like biological systems, robust to environmental and mutational insults. Many GRN models have been proposed, and quite a few papers considered the properties of evolving GRNs (for recent examples see Kuo et al., 2006; Nicolau and Schoenauer, 2009). The model used in this work has been inspired by earlier work of Eggenberger (1997) and is similar to several models developed in recent years (e.g Andersen et al., 2009; Schramm et al., 2009). We have developed it originally for controlling development of 3-dimensional embryos with non-trivial morphologies or patterning (Joachimczak and Wróbel, 2008, 2009).

Models of multicellular development are of great interest in the field of Artificial Life, because they require considering at least two levels of biological organization: the level of molecules (genes, proteins, etc.) and the level of cells. Foraging behaviour also requires these two levels, and in this work we apply our model to control unicellular animats in an environment with a gradient of scents coming from food particles. The cells are provided with sensory information using externally driven concentrations of transcription factors. Such setup resembles chemotaxis of unicellular eukaryotic organisms which can detect gradients of substances with membrane receptors (Bagorda and Parent, 2008). However, small size of prokaryotic cells does not allow for signal to noise ratio high enough to do that, so prokaryotes evolved a different mechanism: bacterial chemotaxis is based on detecting concentration fluctuations over time and random changes in movement direction (see e.g. Alon, 2006).

What happens to the animat in our system depends not only on the GRN state but also on the laws of simulated simple Newtonian physics, and this can be exploited by evolution. The interplay between the GRN and the physical environment removes some of the computational burden from the GRN. This is analogous to the physics-GRN interplay used previously to guide developmental systems (e.g. Eggenberger, 2003; Joachimczak and Wróbel, 2009). Also, this is not the first time GRNs are used to control animat behaviour (see e.g. Bentley (2004); Taylor (2004); Quick et al. (2003) where obstacle avoidance, wall and light following were considered). Some previous papers considered the dynamical properties of GRNs in which product concentrations oscillate, decay within desired time frames or respond to noisy external signals (Kuo et al., 2004; Knabe et al., 2006).

In this paper, we will first provide a brief description of the regulatory model and the environment used in the experiments. Two experiments will be presented. In one only single food type was provided. In a more complex problem, two substances were present. One was poisonous until a certain number of particles of the other were consumed; at this point the roles were reversed. We end with a discussion of the topologies of evolved GRNs and of the evolutionary trajectories that lead to the solutions.

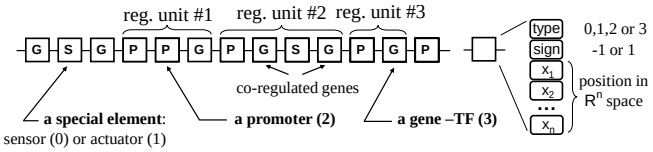

Figure 1: The genome and the structure of a single genetic element. Each element consists of a type field, a sign field, and a sequence of N real values used to determine affinity to other elements ($N = 2$ was used in this paper).

The model

Our model of the genome is designed to capture some of the most essential features of evolving regulatory networks. The GRN topology is encoded in a linear genome. Genes encode transcription factors (TFs). TFs bind to promoters of genes to regulate their expression. Any network topology can be encoded: there are no limits on the size of the network, number of connections or maximum number of connections per node. This is because our primary motivation is to build a model that allows to ask questions relevant to biology (where no such limits are imposed) rather the to solve a particular optimization problem (where enforcing them might decrease the search space).

Encoding a GRN in a linear genome

The genome is a list of genetic elements that fall into three classes: elements that code for products (called genes); regulatory elements (called promoters); special elements (that code for external inputs and outputs of the regulatory network). The genome is parsed sequentially, and regulatory units are formed whenever a series of promoter elements is followed by a series of genes. Special elements are assigned to input and output nodes at a later stage. In result, each regulatory unit is composed of one or several regulatory elements and one of several genetic elements coding for TFs. Regulatory units form the nodes in the regulatory graph. When the unit is expressed (active), all TFs that belong to it are produced at the same level. Fig. 1 provides an overview of the process, together with the structure of a single genetic element.

Each genetic element encodes N coordinates ($N = 2$ was used) and thus can be assigned to a point in R^2 space. When a TF lies close enough to a promoter in this space, a connection between the respective regulatory units is formed (a cut-off distance of 5 prevents full connectivity, Fig. 2). The abstract R^2 product-promoter space should not be confused with the 2D environment in which the animat is simulated. "Sign" fields of two elements allow to determine whether the weight of a connection is positive or negative (using multiplication). Because regulatory units can have multiple promoters and multiple genes, two nodes can be connected by several edges.

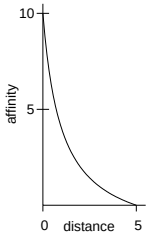

Figure 2: Translation of Euclidean distance between genetic elements into affinities (weights). Maximum weight of 10 and cut-off at the distance of 5 are used.

Genetic algorithm

Each evolutionary run was initiated with 300 genomes consisting of 5 randomly created regulatory units. Element coordinates were initiated using uniform distribution to draw a random direction and a random distance from (0,0). The population size was kept constant. Binary tournament selection (draw two individuals, keep the better one) was used.

Genetic operators in our system act on the level of genetic elements. Single element mutations can change element type, sign bit (changing all its connections from inhibitory to regulatory or vice versa) or coordinates (changing connection weights). Coordinates are changed by shifting the associated point in the abstract N-dimensional space in a random direction by a distance drawn from a Gaussian distribution. Duplications and deletions of multiple elements occur at random locations in the genome. When they occur, some points are created or removed in the abstract N-dimensional space. If $N \leq 3$, it is possible to visualize how the points move, appear and disappear. The duplication/deletion length is drawn from a geometric distribution, with equal probability of duplications and deletions. Since genetic elements cannot be created de novo and there is no recombination, all genetic elements in any individual can be traced back to the elements in one of the genomes that were present in the initial population.

GRN dynamics

During simulation of the network, regulation of a given regulatory unit (node of the graph) will result in the change in concentrations of TFs that belong to this unit. The rate of TF synthesis is a function of activation of all promoters belonging to the unit (inputs to the node). First, distances are converted to affinities using an exponential function shown in Fig. 2. Activation of each promoter is a sum of the concentration of all products binding to it weighted by their affinities. This sum (A) is used to derive product concentration (a value within $< 0, 1 >$) in the next simulation step using the equation:

$$\frac{dL}{dt} = \frac{2}{1 + e^{-(A-1)}} - L \qquad (1)$$

where the time step dt determines the simulation accuracy ($dt = 0.1$ was used), and current concentration (L) determines the intrinsic product degradation rate, so concentration increases only if the sigmoid function gives a value above L, and the degradation rate increases if the value is

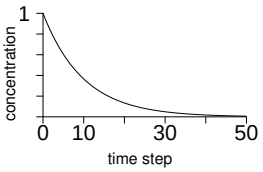

Figure 3: Time scale of exponential degradation of a transcription factor over time. All TF concentrations in the system are in the range $< 0, 1 >$.

negative. Fig. 3 illustrates the time scale of product degradation used in the system.

Animats and their environment

Animats are modelled as simple circular objects, equipped with two identical food sensors located towards the front and two actuators towards the back (Fig. 4). To evaluate the fitness of an animat, it is placed in the environment that is an open and continuous 2D space with randomly placed food particles. The animat and food particle coordinates are represented as real numbers.

Each food particle generates a field of scent. At each location in the environment, the scent coming from a food particle is directly proportional to the distance to this scent source. Fields from each food particle sum up, forming a scent map (see right panel of Fig. 6 for an example). Animat's sensors perceive the scent at its location in a non-directional manner, so the gradient information has to be extracted using two sensors and/or movement. When a food particle is consumed its field is removed from the map.

Sensors and actuators are assigned to special elements in the genome, which come in two subtypes: input and output. The scent perceived by a sensor determines the concentration of associated input product. In addition to inputs representing sensors, special product whose concentration is always at a maximum (1) can be used to initiate gene expression. The positions of input products in the R^2 product-promoter space determine how they are connected to the rest of the GRN. However, direct connection between the input products and the output is not permitted. The output element behaves essentially as a promoter in the system, but a better way of putting it is that it is a single promoter regulating expression of a single gene, and that the concentration of the corresponding product regulates the animat's actuators. The assignment of special elements to actual inputs and outputs in the system (sensors/actuators) is based on their order in the genome, superfluous special elements are ignored.

Actuators work as thrusters and animat motion is simulated using simple Newtonian physics. The thrust force is proportional to the concentration of a product associated with the output special element. The force is not directed toward the centre of the animat, so when the activation of actuators differ, the animat is caused to spin. However, the animat cannot turn on the spot: even when only one actuator is active, the animat moves in a loop rather than rotate in place. Switching the actuators off results in continued motion because of inertia, but the animat will be eventually

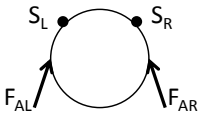

Figure 4: The placement of sensors of chemical signal (scent) and actuators on the animat.

brought to a stop due to fluid drag proportional to squared velocity. This drag also limits the maximum speed possible. To find a food particle it is thus not only necessary for the animat to properly orient itself but also to properly deal with inertia when taking turns.

Results

Designing a way to assess fitness in a chemotaxis problem

In preliminary experiments, we have assessed the fitness by measuring the energy level of an animat at the end of its lifetime divided by the maximum energy that could be obtained in a particular environment. The energy level was set to zero at the beginning of fitness evaluation. Each particle consumed by the animat increased the energy by 1.

We have noticed that if the genetic algorithm was constructed to minimize

$$f_{fitness} = 1 - \frac{energy}{energy_{max}} \tag{2}$$

the best animats would often show a suboptimal behaviour, circling towards the food (Fig. 5). The corresponding hill in the fitness landscape is very easy to find and climb, but difficult to escape from: simply circling around a map allows to find some food particles by chance and the behaviour can be further optimized by controlling the loop diameter with only a single actuator (tightening it when the scent gradient increases). To promote alternative solutions, an additional term was introduced in the fitness function. This term favours individuals that change the direction of the movement at least once during their lifetime. For such individuals $f_{fitness}$ was decreased by 10%. This helps to arrive at animats capable of controlling both actuators early during the course of evolution, even though circling behaviour remains a strong attractor for the genetic algorithm.

Using a map with fixed locations resulted in overfit individuals that simply followed trajectories optimized for a particular map. To prevent this, for each animat fitness was averaged for four maps with the same number of particles at random locations (so this average, f_{avg}, would differ slightly even for two identical genomes).

Designing sensor preprocessing for foraging behaviour

The only information about the environment made available to the animat is the state of two sensors S_L and S_R, corresponding to the concentration of the food scent in the location when the sensors would actually be at. To allow the information from the sensors to be processed by the

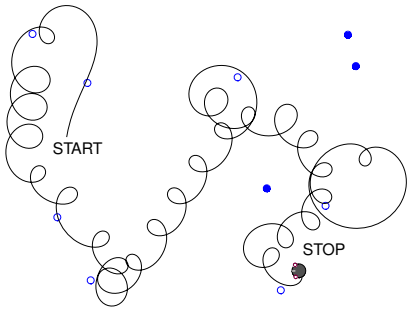

Figure 5: A common suboptimal solution in the fitness landscape: targeting the food particles by performing circular motion. Despite low average speed, it can be quite effective at targeting. Particles consumed during lifetime are drawn as empty circles.

GRN, some preprocessing of sensory information is necessary. This is because TF concentrations in the system are in the range $< 0, 1 >$ whereas the value of the scent field at a given location has no upper limit.

Our initial approach was to provide the GRN with concentrations of input products $S1$ and $S2$ that would correspond directly to the values of S_L and S_R but were restricted to $< 0, 1 >$ using sigmoid function. This, in principle, should have allowed for the emergence of simple controllers with sensors cross wired with actuators in the regulatory network, similar to Braitenberg vehicles.

However, such signal preprocessing resulted in very poor evolvability, for a very simple reason. The diameter of the animat is very small compared to the size of the environment, so both sensors perceive the scent at a very similar level. Unless the animat is very close to the food particle, the difference in signal levels would often be less than 1%. Although we were able to obtain some animats capable to climb the scent gradients, their overall performance was poor.

Much better results were obtained when a simple sigmoid function was used to derive the concentration of the input product $S1$:

$$S1 = \frac{1}{1 + e^{-\alpha(S_R - S_L)}} \tag{3}$$

where α controls the steepness of the function and was set so that it amplifies small differences between S_L and S_R. If S_L is equal to S_R, the $S1$ concentration is 0.5. The concentration approaches 1 or 0 depending on the difference between S_L and S_R.

Using just $S1$ was enough to evolve animats that quite efficiently search for one food source. However, we have observed that the animats turn too fast when close to the food sources and too slowly when far away. Information about the distance from sources is missing in $S1$, so to allow for better turn taking we have introduced a second input product ($S2$) which concentration depends on the perceived food scent at

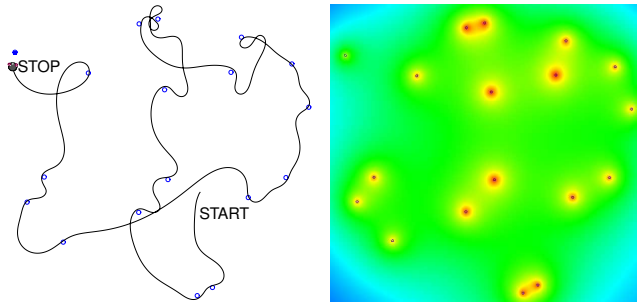

Figure 6: Left panel: best individual navigating the map with single type of food; Right panel: initial map of scent intensity that is locally perceived by animat sensors (normalized to span full colour range).

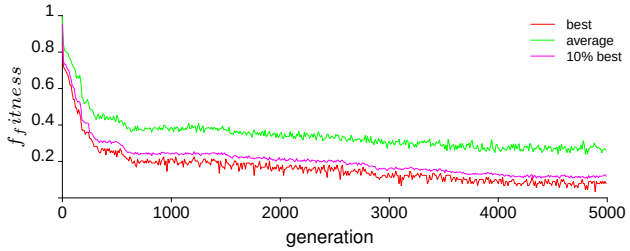

Figure 7: Fitness over generations for the problem with a single type of food source.

the animat location:

$$S2 = \frac{2}{1 + e^{-\beta(S_R + S_L)}} - 1 \tag{4}$$

where β similarly controls the steepness of the sigmoid.

Foraging for a single type of food

In the first experimental setting, maps were created by placing 20 food particles at random locations. Animat behaviour was simulated for 2000 time steps. The size of the map was such that typically about 300 time steps were required to cover the distance between the farthest food particles at maximum speed. Because about 50 steps are needed for TF degradation at the default rate (Fig. 3), latencies in information processing in the GRN quickly become an issue when there is a need to react fast.

Out of ten independent evolutionary runs of 5000 generations, seven resulted in very efficient solutions. The best animats had f_{avg} between 0.05 and 0.25, which means that around 70-90% of food particles were collected. In the remaining runs the algorithm got stuck in a solution with a circular motion and loop tightening when close to a food particle (such behaviour is shown in Fig. 5). Only about 30-40% food particles could be collected with this approach.

The behaviour of the best individual in ten runs is shown in Fig. 6 (left panel). Fairly good solutions were found quite early (Fig. 7; this could be observed also in the other runs).

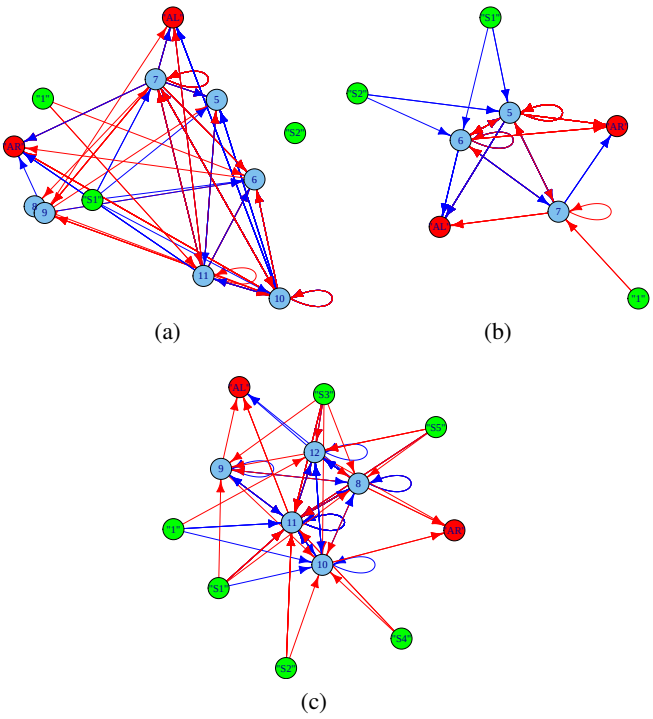

Figure 8: GRN topologies of animats foraging for one (a,b) or two (c) chemical substances; (b) shows the GRN of the best animat (generation 5000), and (a) its ancestor in generation 3000. Multiple links between nodes have been collapsed to one line.

In later generations, speed and targeting gradually improves, but even the best animats turn too widely (which later needs correction) and move only at about 60% of the maximum speed possible. However, this is an expected trade-off given the physical (inertia) and biochemical (latencies in product synthesis/degradation) constraints.

Analysis of the evolved regulatory network (Fig. 8b) shows a simple, largely symmetric topology with only three internal nodes. The best GRN uses both sensory inputs available: the directional information ($S1$) and the scent concentration at the animat location ($S2$). However, $S2$ is not critical for navigation, and in the best networks in other runs it was often disconnected. Indeed, going back from the best animat at generation 5000 to its ancestor at generation 3000 (Fig. 8a) shows that in the ancestral GRN $S2$ was not connected. Perhaps this is the primary reason why the ancestral animat is less efficient at gathering food particles.

In 2000 generations that separate these two animats, the network became less dense (see below) and the genome size roughly doubled. The number of deletions and duplications was similar, but the duplications were longer on average: 6.8 genetic elements for average duplication vs. 2.3 for deletion (despite lengths being drawn from the same distribution). It

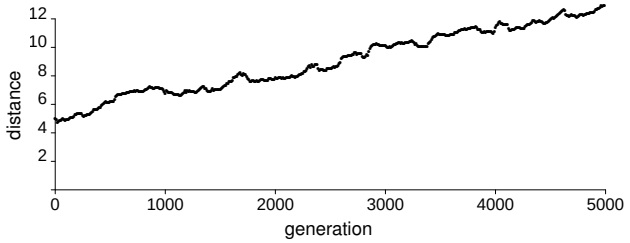

Figure 9: Measuring the spread of genetic elements over time: average distance from $(0, 0)$ for all genetic elements in each generation for the problem with single food type.

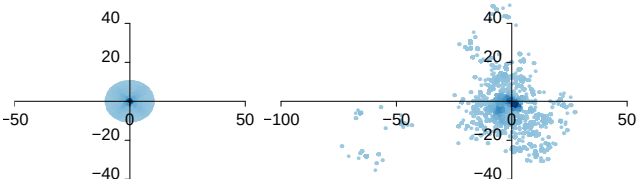

Figure 10: Distribution of genetic elements from all individuals in first generation (left) and last generation (right). Dots represent locations in R^2 space of all genetic elements in the gene pool.

is possible that this excess of duplications allows for some of the duplicated elements to take on new functions and perhaps to optimize the speed of information processing in the network. This requires changing the coordinates of points associated with the duplicated elements.

Many genetic elements in a particular genome are not important for GRN functionality and small mutations in their coordinates are neutral or almost so. This means that over time, points in product-promoter space spread away from each other, and because initial coordinates are drawn from a uniform distribution centred at 0, points spread away from the centre (Fig. 10 and Fig. 9). The unimportant points perform a random walk and slowly move beyond the interaction distance, which reduces the density of the network. This is a general property of element evolution in our system, but a similar process is at play in biological evolution: neutral mutations in duplicated genes or promoters eventually remove redundant connections in GRNs.

Foraging for two types of food

The chemotaxis problem can be made more difficult by introducing more types of food. Evolving animats that search for two types of food may be seen as a first step towards evolving even more complex behaviours, such as the ability to avoid obstacles or to search for mates, perhaps with separate modules in the network controlling different behaviours. The task was formulated so that consuming an appropriate food particle increases the energy by 1, wrong particle results in a decrease by 1. Poison changes to food and vice versa when energy reaches a certain value (5). When energy

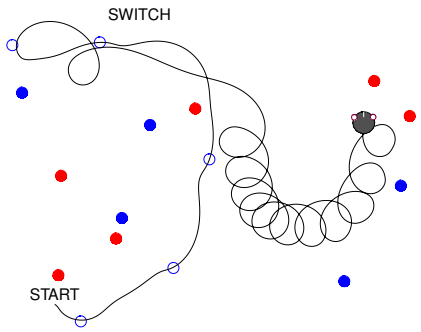

Figure 11: The path of the best individual from generation 2600 for the problem with two food sources. After seeking blue particles, the animat switches to circular motion strategy, similar to that observed in the previous experiment (Fig. 5). This behaviour is replaced later in evolution with direct targeting. Consumed particles are drawn as empty circles.

drops below zero the animat becomes immobile. 30 food particles of type one (blue) and 30 of type two (red) were placed in the environment, and this rather high density of particles was required so that poison avoidance could evolve (otherwise accidental consumption would be too rare to affect fitness).

To allow perception of two substances in the same fashion as for one, four special genetic elements were used as GRN input ($S1$ and $S2$ for the first type, and $S3$ and $S4$ for the second). To increase evolvability, one more element ($S5$) had to be introduced. The concentration of its product would be 0 until the energy reaches 5 for the first time, and 1 from then on, signalling that a behaviour switch is necessary. The best animats evolved before this mechanism was introduced would move slowly enough to collect only about 5 particles during their lifetime.

In this experimental setup ten independent evolutionary runs were performed, but with individual lifespan increased to 7000 time steps so that more particles could be collected. In three runs f_{avg} for the best individuals was between 0.19 and 0.26, which means that the animats extracted around 70% of energy available to them in the environment. The animats showed the desired behaviour: they first searched for blue particles and switched to search for red as soon as signal $S5$ was set to 1. In four runs the best animats would gather around 50% of energy by efficiently collecting blue particles, but then collected red using the circular motion approach (a manifestation of same attractor in the fitness landscape as seen on Fig. 5). The best animats in the remaining runs would gather only blue particles and then stop.

Fig. 12 shows the behaviour of the best animat in ten runs, its GRN has been presented in Fig. 8c. Information from all externally provided signals ($S1 - S5$) is used. This animat actively avoids wrong (red) food particles when searching for blue. However, after the behaviour switch, when it

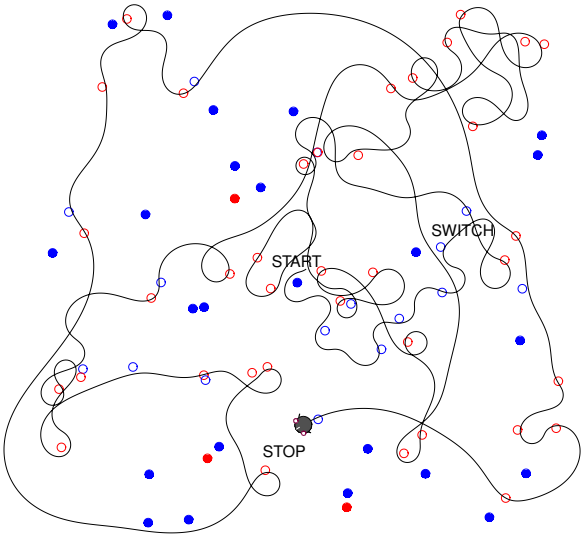

Figure 12: The path of the best individual from the final generation (5000) for the problem with two food sources. The switch in behaviour occurs after 5 blue particles are consumed. Particles consumed are marked as empty circles.[1]

actively seeks red particles, it will consume any blue particles that accidentally come its way. The difference in the avoidance behaviours likely stems from the fact that the evolutionary pressure to avoid red particles at the beginning is stronger: consuming them when low on energy will be lethal.

Fig. 13 shows that evolution of foraging for two types for food was less gradual than for one type (Fig. 7), though in some runs the plateaus were less pronounced; their lengths varied. The best individual from the first plateau (generation 2600) actively and efficiently searches for blue particles, and avoids the red, but uses the circular motion strategy after the food/poison switch (Fig. 11). This behaviour allows to gain energy because at this stage there is more red particles than blue. The best individual from generation 3100 (the second plateau) already seeks the red particles actively, but moves rather slowly. The third plateau in fitness is reached by improving the speed.

A large fitness improvement between generation 2900 and 3900 corresponds to an increase of genome size (Fig. 14). The duplications that lead to this increase tend to create new connections between existing nodes in the GRN rather than create new nodes. This is not surprising: duplication of genetic elements results more readily in a new product-promoter pair than in a new regulatory unit. However, it was rare for the duplications to occur before the onset of the episodes of fitness improvement. Rather, they tended to occur at the very end of these episodes or during the plateaus.

[1] Videos of animat behaviours are available at:
`http://www.evosys.org/alife12chemotaxis`

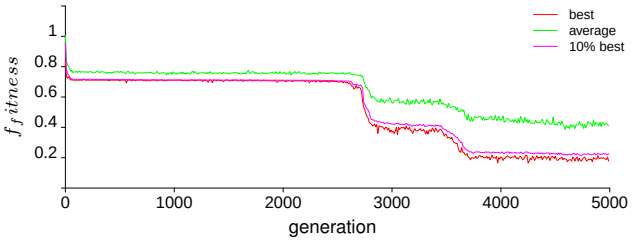

Figure 13: The fitness for the problem with two food types. Three stages corresponding to improved behaviour are seen.

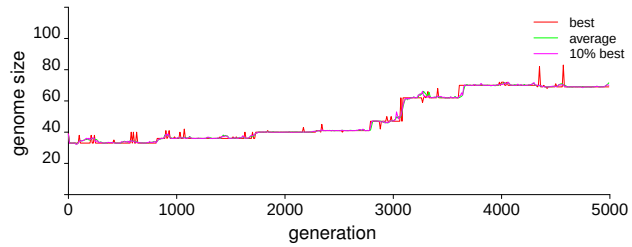

Figure 14: The genome size (the number of genetic elements) for the problem with two food types.

This suggests that even though the duplications may prepare the stage for the improvements, the episodes themselves are actually initiated when the elements acquire new functions, and the points in the promoter-product space need to move some distance before that can happen.

Discussion

The genetic algorithm used in this work did not include elitism nor recombination. Together with small population size and the fact that the fitness was evaluated using random scent maps would mean that the best genomes, subject to the Muller's ratchet, would not necessarily be maintained in the population. Even so, good solutions were obtained. Random genomes grew through duplications, with better and better fitness thanks to the divergence of duplicated elements. The evolvability was good enough to scale the system to a more complex foraging problem, in which several navigating behaviours are required. The best animat displayed 3 behaviours, activating them in a proper fashion: first seeking blue particles and avoiding red, and then seeking red particles after food/poison switch. Although preprocessing of sensory information was necessary to obtain good evolvability in the foraging tasks, all the information available to the animats came from the scent concentrations perceived at the locations of two sensors (Fig. 4).

Before this research platform could be used to address biologically relevant questions pertaining to the properties of evolving networks, a few issues need to be addressed. First of all, evolved networks are fairly small, even for the more complex problem. Secondly, to observe any emerging trend in properties with confidence, networks from multiple evolutionary histories will have to be analysed. This is because

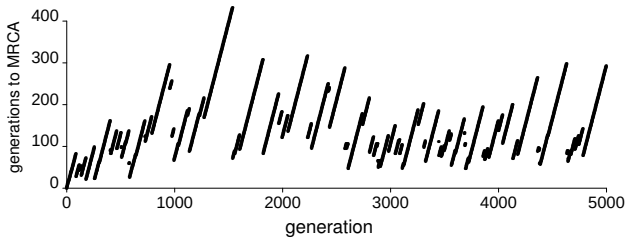

Figure 15: The number of generations to the most recent common ancestor (MRCA) for the entire population in each generation of the experiment with one food source. Average: 148.7; the value for the experiment with two sources was similar.

individuals in a single evolving population are not very divergent. For a given generation, all individuals have a common ancestor about 150 generations earlier (Fig. 15), so they represent a single successful lineage rather than multiple lineages evolving independently. To analyse general trends in properties, evolutionary runs will have to be repeated many times. Alternatively, such analysis will require constructing a system in which multiple lineages can co-exist.

Artificial GRNs have computational properties equivalent to recurrent neural networks. However, when compared with typical perceptron-based neural networks, GRNs have richer dynamics coming from product accumulation and degradation. This results in lower response time, but can allow e.g. to integrate noise or produce signals that change gradually. We provide a more in-depth discussion of evolvability of regulatory networks together with comparison to perceptron-like GRNs in a parallel paper (Joachimczak and Wróbel, 2010).

We have observed that animats in the final generation have usually low maximum TF concentrations, rarely above 0.3. This may stem from the evolutionary pressure to reduce the response time of the networks. In a system in which concentrations represent some continuous variables (such as the activity of a sensor or actuator), it is relative changes of concentrations that are important. Intrinsic TF degradation is exponential, so resulting relative changes do not depend on the concentration itself (Fig. 3). However, relative changes caused by regulation do depend on current concentration: a low concentration allows for a larger relative change, so keeping TF expression low permits to react faster to changing environmental signals. In biological systems lower concentrations would result in a decreased signal-to-noise ratio, but in our system GRNs there is no noise. The only thing that prevents using extremely low expression levels is the limit of maximum connection weight. It will be interesting to investigate if adding noise to gene expression will affect the properties of evolved networks and the way information is encoded in changing concentrations of TFs.

Our results demonstrate that a slightly simplified model previously employed for artificial embryogenesis (Joachim-

czak and Wróbel, 2009) can be used to obtain GRNs controlling real-time foraging behaviours of unicellular artificial organisms. In our future work, we plan to bring two problems together with the goal to build a system in which multicellular animats will develop from single cells and co-evolve competing for resources.

Acknowledgements

This work was supported by the Polish Ministry of Science and Education (project N519 384236). The computational resources used in this work were obtained thanks also to the support of the project N303 291234, the Tri-city Academic Computer Centre (TASK) and the Interdisciplinary Centre for Molecular and Mathematical Modelling (ICM, University of Warsaw; project G33-8).

References

Alon, U. (2006). *An Introduction to Systems Biology: Design Principles of Biological Circuits (Chapman & Hall/CRC Mathematical & Computational Biology)*. Chapman & Hall, 1 edition.

Andersen, T., Newman, R., and Otter, T. (2009). Shape homeostasis in virtual embryos. *Artif. Life*, 15(2):161–183.

Bagorda, A. and Parent, C. A. (2008). Eukaryotic chemotaxis at a glance. *J Cell Sci*, 121(16):2621–2624.

Bentley, P. J. (2004). Adaptive fractal gene regulatory networks for robot control. In *Workshop on Regeneration and Learning in Developmental Systems in the Genetic and Evolutionary Computation Conference (GECCO 2004)*.

Eggenberger, P. (1997). Evolving morphologies of simulated 3D organisms based on differential gene expression. In *Proceedings of the Fourth European Conference on Artificial Life*, pages 205–213, Cambridge, MA. MIT Press.

Eggenberger, P. (2003). Genome-physics interaction as a new concept to reduce the number of genetic parameters in artificial evolution. In *Congress on Evolutionary Computation, CEC '03*, volume 1, pages 191–198.

Joachimczak, M. and Wróbel, B. (2008). Evo-devo *in silico*: a model of a gene network regulating multicellular development in 3D space with artificial physics. In *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 297–304. MIT Press, Cambridge, MA.

Joachimczak, M. and Wróbel, B. (2009). Evolution of the morphology and patterning of artificial embryos: scaling the tricolour problem to the third dimension. In *Proceedings of 10th European Conference on Artificial Life (ECAL 2009)*. Springer.

Joachimczak, M. and Wróbel, B. (2010). Processing signals with evolving artificial gene regulatory networks. In *Artificial Life XII: Proceedings of the Twelfth International Conference on the Simulation and Synthesis of Living Systems*. MIT Press, Cambridge, MA.

Knabe, J. F., Nehaniv, C. L., Schilstra, M. J., and Quick, T. (2006). Evolving biological clocks using genetic regulatory networks. In *Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems*, pages 15–21. MIT Press/Bradford Books.

Kuo, D., P., Banzhaf, W., and Leier, A. (2006). Network topology and the evolution of dynamics in an artificial genetic regulatory network model created by whole genome duplication and divergence. *BioSystems*, 85(3):177–200.

Kuo, D. P., Leier, A., and Banzhaf, W. (2004). Evolving dynamics in an artificial regulatory network model. In *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 571–580. Springer Berlin / Heidelberg.

Nicolau, M. and Schoenauer, M. (2009). On the evolution of scale-free topologies with a gene regulatory network model. *BioSystems*, 98(3):137–148.

Quick, T., Nehaniv, C. L., Dautenhahn, K., and Roberts, G. (2003). Evolving embodied genetic regulatory network-driven control systems. In *Advances in Artificial Life*, pages 266–277.

Schramm, L., Jin, Y., and Sendhoff, B. (2009). Emerged coupling of motor control and morphological development in evolution of multi-cellular animats. In Kampis, G. and Szathmáry, E., editors, *Proceedings of 10th European Conference on Artificial Life (ECAL 2009)*. Springer.

Taylor, T. (2004). A genetic regulatory network-inspired real-time controller for a group of underwater robots. In *Proceedings of the Eighth Conference on Intelligent Autonomous Systems (IAS-8)*, pages 403–412.