AN EXPLORATION ON MODULARITY EMERGENCE CONDITIONS IN EVOLUTIONARY SYSTEMS

A COMP4560: Advanced Computing Project report submitted

to the Australian National University in partial

fulfilment of the requirements for the

degree of Bachelor of Advanced Computing

in the

Research School of Computer Science

October 2017

Zhenyue Qin

Supervisors: Prof Bob McKay, Prof Tom Gedeon

Table of Contents

LIST OF FIGURES

LIST OF TABLES

DECLARATION OF AUTHORSHIP

I, ZHENYUE QIN, declare that this report titled, “An Exploration on Modularity Emergence Conditions in Evolutionary Systems” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this report has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the report is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

20/Oct/2017

ACKNOWLEDGEMENTS

I would like to thank my supervisors Prof Bob McKay and Prof Tom Gedeon for their tireless guidance, continuous support and in-depth insight. This project would not have progressed at all without their knowledgeable advice and great encouragement.

Additionally, I would also like to thank my parents Naiqin Wang and Fen Qin, who supported all my undergraduate academic expenses. Without their funding and unconditional love, my concentration on this project would have been impossible.

Furthermore, I would like to thank Siu Kei Muk for his efficient and easy-to-use genetic algorithm framework. This project would have taken much longer time without his brilliant code.

Thank you all whoever contributed to this project.

ABSTRACT

Understanding biological organisms better can assist in solving complex engineering problems by applying their desirable characteristics and structures. A long-standing biological question is how organisms can quickly adapt themselves to new environments that are constantly changing, which is called evolvability. A key aspect to understand evolvability is to know the origin of modularity. Although various theories have been proposed to explain the conditions under which modularity arises, there is no agreement among them. In computational biology, one prevalent theory argues that gene specialisation drives modularity. So far, there has been no research suggesting evidence against the plausibility of the theory. As such, I explored this theory further in order to verify its consistency with biological phenomena. I investigated evolutionary simulations, including novel crossover mechanisms, that can help modularity emerge in biological networks via gene specialisation. With these improved methods, my experiments indicated that there existed an inconsistency between the theory stating specialisation leads to modularity and observations in biology, regarding the dominant status of modular structures on evolvability. Subsequent experiments also indicated that networks with high fitness could be converted into modular structures by removing inter-module connections while their performance remained unchanged or improved. As such, I conclude that evolutionary systems may develop towards structures requiring fewer edges. In addition, allowing networks to discard unnecessary connections may resolve the inconsistency mentioned above. In the future, further investigations are needed in order to understand the average number of links in a module to guide module-based designs.

Keywords: **Computational Biology**, **Genetic Algorithms**, **Modularity, AI**.

CHAPTER 1

Introduction

1.1 Motivations

Biological organisms are complex systems that demonstrate a variety of ideal engineering characteristics, such as fault tolerance, flexibility [1]. These traits have been difficult to achieve with traditional engineering methods [2]. Thus, engineers started mimicking natural evolution approaches, such as reproduction and mutation, in order to generate and evolve entities possessing the above desirable engineering features [3]. These mimicking activities gave birth to a new field called bio-inspired computing [1], which has exhibited a wide range of successful methodologies, including artificial neural networks, genetic algorithms [2].

Despite the successes of bio-inspired computing in the past two decades, many challenging issues still remain [4]. An essential one is scalability, that is, the size of problems that current bio-inspired methodologies can solve is relatively small or moderate [4]. Another challenge is adaptability. For example, what can we do to facilitate engineered robots to evolve in order to adapt themselves into the constantly changing environments, just like biological organisms? Studies have indicated that the lack of modularity is one of the

reasons that account for the incapability of artificial biological systems for adapting into and scaling up to higher complexity [5]. For example, artificial neural networks are generally constructed to be densely connected, whereas human brains exhibit modular components taking different responsibilities, such as the hippocampus for dealing with novel situations and the amygdala for emotional controls. As such, it is essential to understand the conditions under which modularity spontaneously emerges in biology. Then, engineers may leverage these conditions to design modular systems that can solve more complex problems and are able to autonomously adapt themselves to new working environments. One good example of the module-based engineering design is the "high cohesion, low coupling" principle in software engineering [6]. This understanding in modularity led to a software engineering boom in the current and last century [6].

Specifically, modularity is defined as the divisibility of structures or functions into sub-units that perform autonomously with each other [7]. In other words, a module is a group of elements whose associations occur preferentially within the group [8]. Therefore, elements within a module will demonstrate the tendency to undertake coherent functions independently from other elements outside of the module [8] [9]. In biology, such modules exhibit ubiquity [7]. Specifically, they appear at various levels of biological organisations

[8]. For example, animals utilise different types of cells for various specialised functions, such as red blood cells for carrying oxygen, nerve cells for transmitting information, etc. [10]. Even unicellular organisms still demonstrate discrete regions to perform different activities, such as the flagellum for moving and mitotic spindle for reproducing [10].

Many biological activities and structures can be modelled in the form of networks, such as animal brains and signalling pathways [11]. A network is modular if it can be partitioned into highly connected components, and between these components, there are only sparse connections [11]. For example, In Figure 1.1, the left network is less modular than the right one. These connections can either be physical or dynamic, such as protein-protein interactions as physical connections and gene regulatory networks as dynamic connections [11]. Modularity can promote the evolvability of organisms, where evolvability is defined as the capability of rapidly adapting to novel environments [12]. Two reasons can be used to justify this statement. Firstly, a modular network may allow changes in a module without disturbing other modules [13]. Secondly, modular structures can be reutilised and combined in different ways in order to perform new functions [14].

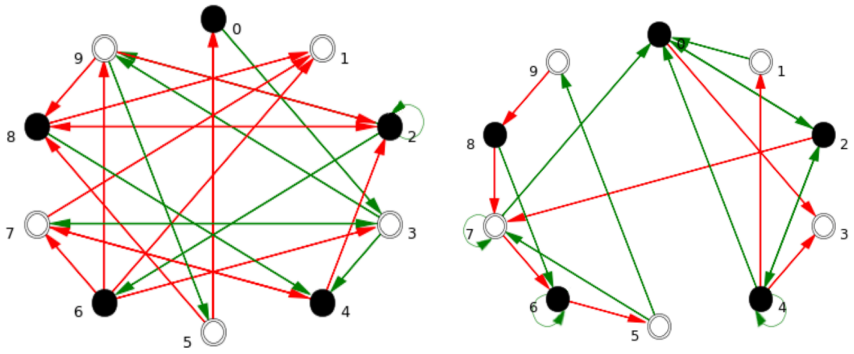


Figure 1.1 Illustration of non-modular (left) and modular (right) networks [9].

Despite the fact that modularity has gained research interest for decades [10] due to its importance in the evolution as stated previously, there is no consensus on its origin and evolutionary direction in biology [15]. Among various scenarios to explain the condition under which modularity emerges, three stand out, because their proposed conditions are commonly encountered in nature [10]. These three scenarios include a mixture of directional and stabilising selection [16], modularity-varying evolutionary goals [5], and specialisation in gene activity patterns [8].

The theory stating that directional selections favour changing some traits while stabilising selections preserve other traits may explain the origin of modularity [16]. Interactions

between changing and stabilising traits can hinder both directional and stabilising selections [8]. As such, modularity emerges in order to break the pleiotropic[1] correlations that lead to trade-offs between traits [16]. This explanation seems to be reasonable because, during the course of evolution, only a few traits are prone to change, whereas most remain unchanged [16]. Nevertheless, the extended experiments on this theory failed to produce an increasing modularity [8].

Kashtan and Alon proposed another theory, that modular changes in environments may impose an impetus in the emergence of modularity [5]. Specifically, organisms that live in environments whose sub-components are repeatedly and constantly altered demonstrate higher-level modularity than those living in stable environments. Changes in the

---

[1] **Pleiotropy** (from Greek πλείων *pleion*, "more", and τρόπος *tropos*, "way") occurs when one gene influences two or more seemingly unrelated phenotypic traits. Therefore, a mutation in a pleiotropic gene may have an effect on several traits simultaneously due to the gene coding for a product used by a myriad of cells or different targets that have the same signalling function.

environment can be fluctuations of temperatures, salinity, and so on [8]. Some experiments also confirmed that metabolic networks of bacteria that live in variable environments exhibit more modules than those in steady environments [17]. This explanation is plausible due to the ubiquity of fluctuations in the environment [8]. However, despite the fact that environments are continuously changing, it is unclear to what extent they vary modularly [8]. Therefore, the quantitative relationship between modularity-varying environments and the origin and maintenance of modularity has not been proven in biological experiments.

Espinosa-Soto and Wagner studied the conditions under which gene regulatory networks started exhibiting modular structures [8]. They concluded that modularity could arise as a by-product of gene specialisation when gene regulatory networks acquire the ability to regulate towards multiple different patterns. Specifically, the distinct sub-components in the regulatory network to regulate sharing and different gene activity patterns will hamper each other's performance. Thus, modular networks that favour fewer connections between modules of the network will break the pleiotropic effect of regulating sharing and distinct gene activity patterns. Moreover, additional gene activity patterns can further improve the modularity. Their work is persuasive since the phenomena that gene regulatory networks acquire new gene activity patterns are ubiquitous in evolution. To be more specific, the

same collection of genes exhibits different activity patterns at different phases of development or different locations in organisms [8]. Their theory can also act as an alternative explanation of why modular-varying environments result in modularity since organisms need to express different gene patterns for different environments [5] [8]. However, the experiments of Espinosa-Soto and Wagner lacked the recombination phase in their evolutionary simulations. Biologically, recombination is necessary.

Based on Espinosa-Soto and Wagner's study, Larson et al. introduced recombination hotspots and proposed their significance in evolvability [9]. Recombination hotspots, also called crossover hotspots, are chromosome regions where the average recombination rate is much higher than the average recombination rate of the whole chromosome. That is, recombination hotspots are more likely to incur crossover at their positions. Recent biological research indicates that the existence of recombination hotspots is structurally due to a mechanism of DNAs called the DNA Double-Strand Break (DBS) [18]. By computational evolutionary simulations, Larson et al. showed that functionally, recombination hotspots could preserve the modularity of DNAs when organisms undertake recombination. They utilised a multi-objective method called the Age-Fitness Pareto Optimisation in order to avoid premature convergence [19], whereas in biology, evolution

is in a single-objective manner. Therefore, it is worthwhile repeating their experiments in a standard single-objective evolutionary simulation to explore the credibility of their conclusion.

1.2 Aim

In this report, I aim to investigate novel evolutionary simulations in order to explore whether there exist methods that can expedite the evolutionary process. For example, Espinosa-Soto and Wagner did not apply recombination in their artificial evolutions [8]. Wagner also stated that recombination mechanism would not contribute to the simulated evolution in his study on epigenetic stability [20]. However, recombination, also called crossover, is assumed to be an effective method to enhance the efficacy of combining useful traits in evolutionary simulations. Therefore, it is useful to explore whether there exists a crossover mechanism that can promote modularity. Similarly, the elitism used, which is another common mechanism in artificial evolution, is also worthwhile exploring for its contribution to computational evolution in this setting.

Furthermore, I would like to explore the plausibility of the theory stating that gene specialisation drives modularity of organisms [8]. Although the experiments relating to this theory suggested a significant emergence of modular structures by gene specialisation, the

theory only reveals specialisation is the origin of modular structures. It did not explain the evolutionary direction of modular systems. Thus, I wish to discover what properties of modular structures will obtain in a long-term evolution. That is, towards what direction(s) do systems with high modularity evolve?

Moreover, experiments in [8] did not demonstrate whether structures with high modularity have gained a dominant status in survivability. In biology, there is no organism that exhibits non-modular structures. As such, one may assume non-modular creatures have become extinct. Therefore, modular individuals are expected to have far better performance than non-modular ones, especially for complicated environments. As such, within the surviving simulated organisms in the gene specialisation experiments, I will investigate the dominant status of modularity on survivability by comparing the fitness values of the eminent modular organisms to less modular ones. If the more modular ones fail to gain their expected dominance, I will also try to propose explanations and resolutions through which networks with noticeable modularity can attain their expected surviving dominance.

As an eventual goal well beyond this report, I would like to discover the nature of intelligence. One hypothesis is that intelligence can only emerge within a problem context

[21]. For humans, this problem context is our living physical world. We need to utilise our intelligence in order to engage with the surroundings. I believe that with the aid of the computational evolutionary simulations, we can eventually understand what is required to evolve intelligence. In particular, modularity may be necessary. Moreover, the engagement between the entities and the simulated world can also be assumed as related to consciousness. Overall, knowing modularity better can help us understand the origin of intelligence and consciousness.

This report is organised as follows: Chapter 2 will provide an overview of the experimental methods, including the gene regulatory network model, the calculation of fitness, the evaluation of modularity, and so on. Chapter 3 will present the results of this study with detailed experimental parameters being introduced at the beginning. Chapter 4 will contain the discussion on the experimental results. The report concludes in Chapter 5.

CHAPTER 2

METHODS

In this report, genetic algorithms are utilised as the evolutionary simulation tool. The gene regulatory network model is based on a Boolean network proposed by Wagner [20] and customised by Espinosa-Soto and Wagner [8].

All simulation code was implemented in Java 1.8.0 and Python 2.7.10. They are all publicly available at https://github.com/ZhenyueChin/Chin-GA-Project. Modularity was evaluated using the NetworkX package with the community API [22]. All the generated data can be downloaded at: https://drive.google.com/file/d/0B9dNEi7lDXnldy1sNmZuTWNXMDg/view?usp=sharing.

2.1 Genetic Algorithms

The genetic algorithm is a heuristic search algorithm used to solve optimisation problems, which is inspired by Darwin's theory of natural selection [23]. Briefly, the theory of natural selection can be summarised as organisms compete for limited resources in nature. The fittest individuals survive to reproduce offspring. Crossover and mutation occur at the

reproduction phase to increase diversity. These techniques have the following advantages compared with traditional search algorithms [24]:

- Traditional approaches begin the search with a single solution and are likely to get stuck at local optima. In contrast, genetic algorithms initialise searching from a set of candidate solutions called a population and perform a form of global search and so are more likely to find the global optimum.
- Genetic algorithms can be applied in a wide range of problems since they only consider a single indicator, namely a real-valued "fitness" score. This value is to measure "how good" a candidate solution is. Other complex actions such as derivation or integration are not required.

Candidate solutions in a genetic algorithm are represented as individuals in biology. An individual can be a haploid or a diploid, i.e. includes single or paired chromosomes, respectively. For example, bacteria are haploids, whereas humans are diploids. Each of these chromosomes contains genetic information encoded into some form of code sequences, which is called a genotype [24]. A diploid individual will have a dominance function, which applies to its two genotypes and returns another code sequence as its

phenotype, whereas the phenotype of a haploid individual is just its genotype[2] [24]. A dominance function, also called a dominance map, is an abstraction from the real, complex situation. This abstraction was originally proposed by Mendel and proved to be a useful simplification for artificial evolutions [25]. Each 'position' on the genotype is called a gene, and alternative genes at this location on the genotypes of diploid individuals are known as alleles. Furthermore, natural selection will utilise phenotypes of individuals to assess their survivability. Figure 2.1 and Figure 2.2 give detailed pseudocode and an illustration on how a dominance function works in a diploid individual. An individual contains two genotypes: 101001 and 010111 separately, and its dominance function refers to a map as 110100 to determine the phenotype of this individual, which will be 110101 here.

```
1 function dominance(genotype_1, genotype_2, dominance_map) do:
2   phenotype = list()
3   if (genotype_1.length != genotype_2.length
4           or genotype_2.length != dominance_map.length
5           or genotype_1.length != dominance_map.length):
6     throw error("lengths of genotypes and dominance map have to be
equal")
7   end if
8   for (i in range(genotype_1.length)) do:
```

[2] This is not exactly biologically correct, but is a common abstraction in genetic algorithms.

```
9     if (genotype_1[i] == dominance_map[i]
10          or genotype_2[i] == dominance_map[i]) do:
11      phenotype[i] = dominance_map[i]
12    else
13      phenotype[i] = genotype_1[i]
14    endif
15  end forloop
16 return phenotype
17 end function
```

Figure 2.1 Pseudocode of phenotype dominance function in diploid individuals.

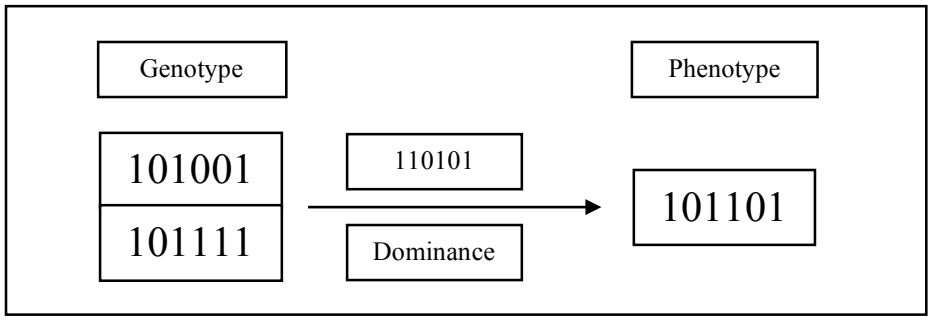


Figure 2.2 Illustration of phenotype dominance function in diploid individuals.

Every state in genetic algorithms is represented as a generation, which contains a fixed number of individuals. Genetic algorithms operate on these populations. That is, a population is a unit in genetic algorithms. The number of individuals within a population is called the population size. A suitable population size is required in order to trade-off efficiency and effectiveness. The population being too large will lengthen the computational time while being too small will lead to local optima.

Individuals in the same generation compete for reproduction opportunities. Genetic algorithms utilise a fitness function to select appropriate individuals as parents. That is, the fittest individuals are more likely to contribute their traits to subsequent generations. Two common fitness selection schemes are the proportional scheme and the tournament scheme [24]. In the proportional selection scheme, the probability of an individual being selected as a parent is proportional to its fitness over the sum of the whole fitness in the population. Formally,

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$

where i is the index of an individual, n is the population size, f_i is the fitness of individual i, p_i is the probability of individual i being selected. As for the tournament selection scheme with size $k \in N$, each parent will be the best one among the k randomly selected individuals. Of note, in situations where fitness values of individuals are very close to each other, the tournament selection scheme has stronger selective pressure towards selecting individuals with high fitness than the proportional selection scheme. This is because the former only considers the relative order of individual fitness values.

The selected parents will go through the reproduction process, including the elite reservation, crossover, and mutation. Some genetic algorithms make the fittest individuals

in the current generation automatically survive to the next generation in order to gain faster convergence. These reserved individuals are known as elites [24]. Crossover will merge chromosomes of parents into the new chromosomes of their children in some way. Mutation randomly alters genes in chromosomes of individuals. However, offspring by the reproduction operations may not fill the entire population. The proportion of children over the whole population is defined as the reproduction rate [24]. Therefore, some selection schemes may be performed in order to fill the vacancies in the population.

2.2 Model

Cells in an organism display heterogeneity in functionalities and morphologies, while they contain the same set of genes. In other words, cells interpret the same genetic material in different ways so that their behaviours and structures vary. These distinct interpretations are due to the regulation via the activation and repression of genes [20]. In brief, effects of different genes are not mutually independent. A protein that is generated by a gene may activate or repress other genes. A gene regulatory network can be a mathematical directed graph to express these relationships of genes in an organism [20]. Specifically, genes can have two different patterns, namely activation and repression. The term “gene activity

pattern" is adopted to represent the activity status of the entire set of genes. Different gene activity patterns mean the distinct cellular functions and forms [8].

I re-constructed the model that was utilised in the work done by Espinosa-Soto and Wagner, which is a network model to represent a gene regulatory network [8]. In this model, a gene regulatory network with N genes will be in the form of an adjacency matrix $A = a_{ji}$, which acts as a genotype of an individual. Each entry a_{ji} is restricted to be either 1, 0 or -1, which represents an activation, absence or repression interaction from gene j to gene i, respectively. The gene activity pattern of this network at time t can be expressed as a Boolean row vector $s_t = [s_t^0, \dots, s_t^{N-1}]$. A certain gene i can either be active ($s_t^i = 1$) or inactive ($s_t^i = -1$). The transition of state activity is modelled by the equation below

$$s_{t+\tau}^i = \sigma[\sum_{j=1}^{N} a_{ji} s_t^j]$$

where $\sigma(x)$ equals 1 if $x > 0$ and equals -1 otherwise. Figure 2.3 gives an illustration of gene activity patterns. Figure 2.4 shows two corresponding gene regulatory networks.

Despite the simplicity of this model, researchers have successfully utilised it to predict the dynamic processes of Drosophila melanogaster [26] and explore how robustness evolves in gene regulatory networks [20].

2.3 Fitness

The fitness here evaluates the likelihood that an attractor is obtained when facing perturbations [8]. In other words, Espinosa-Soto and Wagner imposed a bias of robustness on their gene regulatory network models in order to indirectly select modular networks. This is because modular networks can limit perturbations in a module so that the overall structure will not be heavily affected [27]. That is, more modular networks are more robust.

There are two or more stages in their experiments on discovering the conditions under which modularity starts emerging. In the first stage, gene regulatory networks are evolved under selective pressure towards regulating a particular gene activity pattern, while facing some perturbations. The original gene activity pattern before perturbation is called a target. In the second and further stages, networks are evolved under selective pressure to regulate new gene activity patterns, while preserving the ability to regulate the old patterns. In the

particular case where there were two gene activity patterns, the first stage lasted for 500 generations and the second took another 1500 generations.

The perturbations of targets are randomly generated in every generation when evaluating the fitness of gene regulatory networks. In Espinosa-Soto and Wagner's experiments, a network would face 500 perturbations comprising different corrupted versions of gene activity patterns. Each gene will have a probability of 0.15 to be perturbed into its opposite activity. A further study was conducted to explore a sufficient number of perturbations in order to shorten the computational time while maintaining a similar eventual improved modularity. It was concluded that 75 or 100 perturbations would lead to the noteworthy emergence of modularity [28]. Therefore, 75 perturbations are undertaken for evaluating the fitness of each gene regulatory network in order to reduce the running time.

Larson et al. applied another approach for evaluating the fitness of networks [9]. They generated a static set of perturbations at the beginning and utilised this same set of corrupted targets whenever network fitness was calculated. This method converts the original stochastic fitness evaluation into a deterministic one. However, my statistical tests indicated that this deterministic approach would impede the evolution of network

modularity (Wilcoxon signed-rank test; $p < 0.0099$). Details of this experiment are in Appendix C. Thus, I will stick to Espinosa-Soto and Wagner's perturbation methods in the rest of this work.

The fitness value of a gene regulatory network reflects its robustness in recovering from various perturbations. The error function compares an attractor of the network dynamics to the original gene activity pattern. That is, a successful network is able to regulate a corrupted pattern to its initial form. Then, the Hamming Distance G between the attractor and the original pattern was calculated. Previous experiments indicated that it normally took fewer than 20 transitions to reach the attractor [20]. Thus, non-stable attractors are assumed to be those gene regulatory networks that take more than 20 steps to attain the stability, or are cyclically stable. They are treated to have a maximum Hamming distance D_{max}. This is followed by a calculation of the contribution from each perturbation attractor to the fitness, which is defined as a developmental trajectory $\gamma = (1 - D/D_{max})^5$ [8]. Afterwards, this process is repeated to determine 75 γ_i, $1 \leq i \leq 75$. Finally, the fitness of a network is calculated as

$$f(g) = 1 - e^{-3g}$$

where g represents the arithmetic mean of the sum of all γ_i [8]. As to cases where there are more than one gene activity patterns, the arithmetic mean of $f(g)$ for all the patterns was taken. Consequently, a gene regulatory network with a high fitness is able to lead to different attractors matching different targets.

2.4 Evolutionary Simulations

Espinosa-Soto and Wagner imposed a bias towards low-density gene regulatory networks in mutation [8]. A node in the network has a probability $\mu = 0.05$ to mutate in every generation, and it either can lose or gain an interaction. The probability for a node to lose an interaction can be calculated as:

$$p(u) = \frac{4r_u}{4r_u + (N - r_u)}$$

where N is the number of gene nodes in a gene regulatory network, and r_u equals to the number of regulators of gene u. That is, the number of genes that exert effects on gene u. In contrast, the probability for a gene u to obtain an interaction is defined to be $1 - p(u)$. This mutation pattern will maintain a fixed number of edges in a gene regulatory network. That is, it can keep the sparseness of the network, which computational biology research suggests is necessary for the emergence of modularity [24]. Furthermore, an average 2 – 3

regulators for each gene has been widely observed in networks of plants, animals and bacteria [29].

Espinosa-Soto and Wagner did not apply a crossover mechanism in their simulation [8]. In the reconstructed model by Larson et al., they limited crossover to nine possible partition locations of a 10-node network, corresponding to nine possible rows for splitting the adjacency matrix of a network horizontally [9]. When two matrices A_1 and A_2 are selected for crossover at index i, matrices of their children will be produced as [9]:

$$C_1[0{:}\,i-1,:] = A_1[0{:}\,i-1,:]$$

$$C_1[i{:}\,9,:] = A_2[i{:}\,9,:]$$

$$C_2[0{:}\,i-1,:] = A_2[0{:}\,i-1,:]$$

$$C_2[i{:}\,9,:] = A_1[i{:}\,9,:]$$

I utilised Figure 2.3 as the targeted gene regulatory patterns, where white and black squares represent active and inactive gene activities [8]. The two gene regulatory networks in Figure 2.4 are illustrated as the parental networks to illustrate the process of crossover. The green edges represent gene activation, and the red edges represent gene suppression. Figure

2.5 gives a visualization of the crossover process in [9], which is referred as the horizontal crossover. In particular, when crossover happens between node 4 and 5, the horizontal crossover not only makes the parental networks exchange modular clusters, but also exchange some interactions between the two modules. This may corrupt any modules.

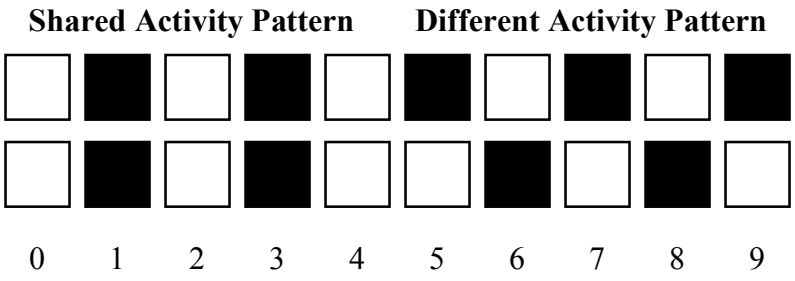


Figure 2.3 Targeted gene activity patterns.

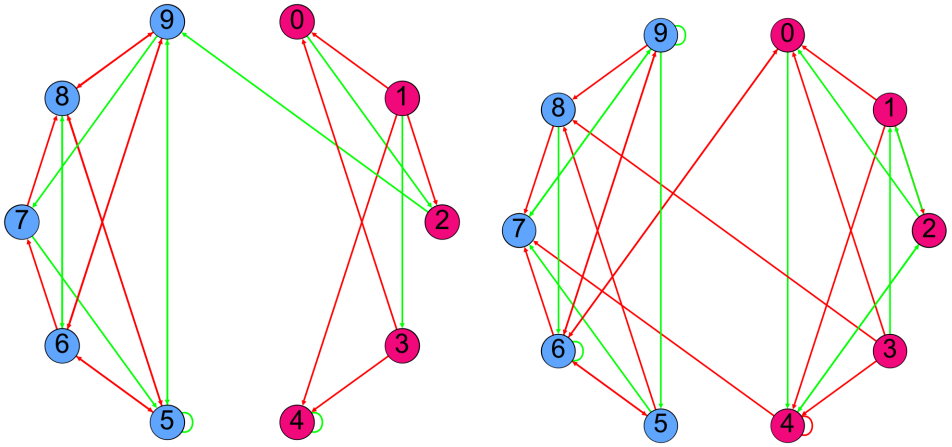


Figure 2.4 The two parent networks before crossover.

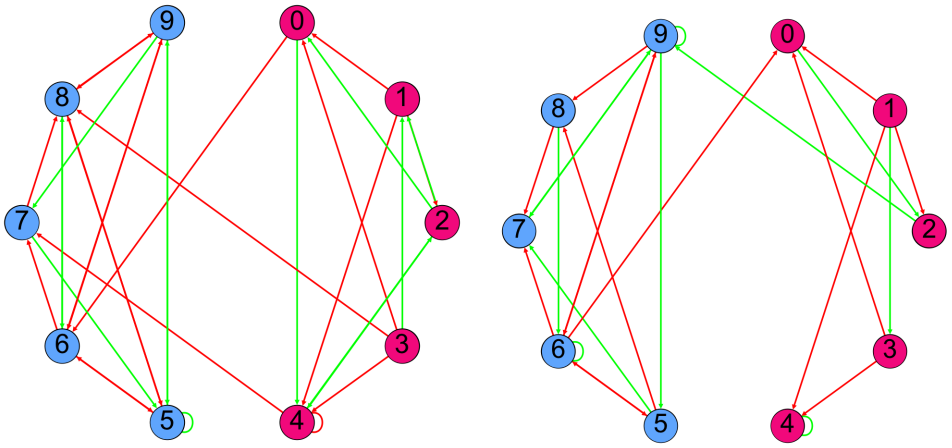


Figure 2.5 The resulting networks by the horizontal crossover mechanism in [8].

Research has shown that biological modularity exhibits high conservations between different species regarding structures and functionalities [13]. For example, cytological studies show that different morphologies of cells are due to diverse polymerization methods, whereas the basic structure of cytoskeletons remains the same in various kinds of cells. In other words, the core cell structure is highly conserved [30]. Thus, I hypothesised that in Espinosa-Soto and Wagner's model, modules that arise in different experimental simulations should also demonstrate similar conservations. That is, a module in one

artificial organism can work in conjunction with the modules in another simulated individual. This is because they correspond to the same gene regulatory patterns. As such, my crossover mechanism swaps interactions between modules in a gene regulatory network with connections between modules in another network. Compared with the crossover mechanism of Larson et al., this approach will better preserve the community structure, as Figure 2.6 illustrates. Additionally, networks that perform well after this crossover are those having conserved modules, which is an important property for modularity. In other words, this crossover mechanism imposes a bias of conservation in evolution. This crossover mechanism is referred as diagonal crossover.

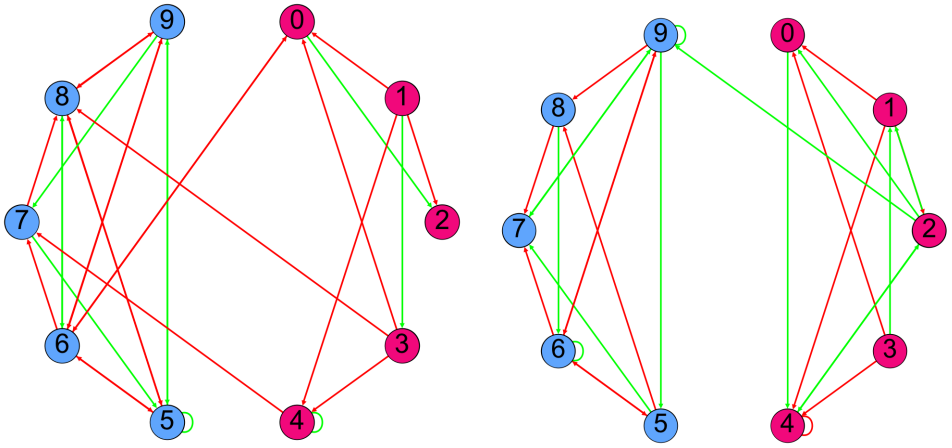


Figure 2.6 The resulting networks by the diagonal crossover mechanism.

Larson et al. also utilised an evolved probability distribution to simulate recombination hotspots. An individual gene regulatory network is defined not only by an adjacency matrix as specified previously, but also an accompanying crossover preference vector c whose length is the same as the number of nodes in the network [9]. For example, if a vector $c = < 0.01, 0.02, 0.05, 0.07, 0.3, 0.09, 0.08, 0.3, 0.08 >$, then the likelihood of crossover happening between node 3 and 4 will be 0.3. Figure 3.1 is a visualisation illustration of such a vector.

Mutations also occur to the vector of recombination hotspots. The probability of an element in the vector c being mutated is the same as the gene regulatory network mutation rate μ. For each crossover position, this mutation will randomly alter the crossover probability to be a real value sitting within $(0, 1)$. Once mutations have completed, the vector will be re-normalised.

2.5 Modularity Metric

I adopted the Q scoring system to quantify modularity in a network based on the algorithm proposed by Newman [31]. Briefly, this approach is defined as the difference between the ratio of the number of edges in the network connecting nodes within a module over the

number of all the edges, and the same quantity when assigning the nodes into the same modules yet edges are assumed to be randomly connected in the network [5]. Formally, Q is calculated as:

$$Q = \sum_{i}^{K} [\frac{l_i}{L} - \left(\frac{d_i}{2L}\right)^2]$$

where i represents one of the K potential modules within a network, L is the total number of connections in a network, l_i stands for the number of interactions in the module i, and d_i is the sum of degrees of all the nodes in module i [8]. In other words, Q considers the two ratios of both intra-module connection density and inter-module connection density [31]. A network that is considered to be good on modularity must consist of as many within-module edges and as few inter-module edges as possible. However, it will result in $Q = 0$ if all the nodes are partitioned into the same module.

The value Q will sit in the range of $[-\frac{1}{2}, 1)$. Nodes in the gene regulatory network are partitioned into different groups according to their regulating gene activity patterns. For instance, concerning the networks in Figure 2.6, nodes 0 to 4 will be assigned into one group whereas the remaining 5 nodes will be clustered into the other. This partition is of

interest in this study, since Espinosa-Soto and Wagner concluded that modules in the network will emerge corresponding to different gene activity patterns [8].

2.6 Hidden Genes

In biology, there exist genes that have no direct biological functions other than participating in the activity of gene regulatory networks. There is no implementation of these hidden genes in evolutionary simulations of [8]. In order to investigate whether they contribute to the modularity emergence of gene regulatory networks, I implemented them as additional regulatory nodes in each modular cluster. As Figure 2.7 indicates, nodes 0 to 7 are regarded as being in one cluster whereas nodes 8 to 15 are in the other cluster. The yellow and green nodes, particularly nodes 5 – 7 and nodes 13 – 15 are the sets of hidden genes for each cluster.

These hidden node genes in regulatory networks mentioned above will regulate additional gene activity patterns. Taking the network in Figure 2.7 as an example, if the original gene activity pattern is $1, -1, 1, -1, 1, -1, 1, -1, 1, -1$, then the network with hidden genes will convert the pattern to

$$1, -1, 1, -1, 1, \mathbf{1}, \mathbf{1}, \mathbf{1}, -1, 1, -1, 1, -1, \mathbf{1}, \mathbf{1}, \mathbf{1}$$

in order to facilitate the regulation process, where 1 and −1 represent activation and repression, respectively. Bold numbers represent the added hidden genes. Figure 2.8 acts as an illustration of the gene activity pattern conversion.

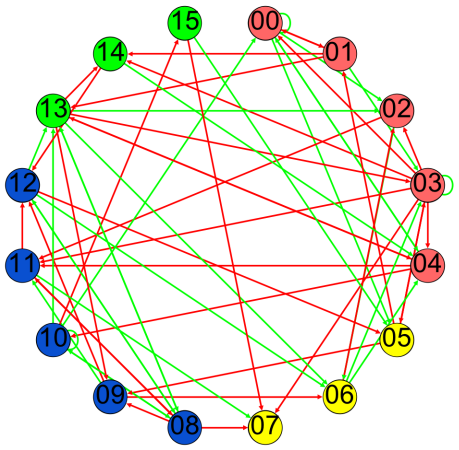


Figure 2.7 The illustration of a gene regulatory network with hidden genes. Green edges represent the activation and red edges represent the suppression.

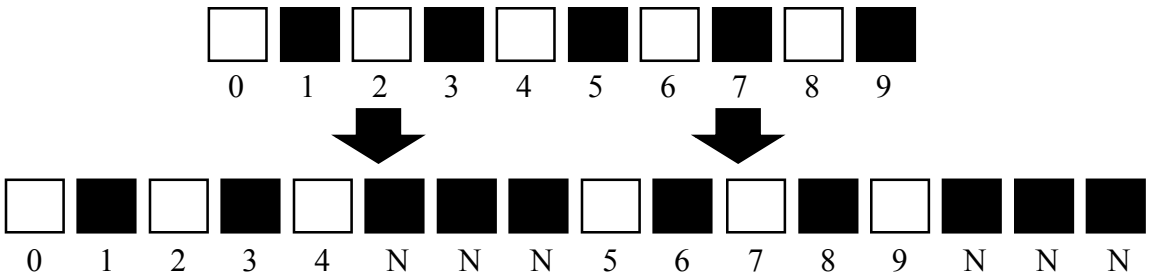


Figure 2.8 The illustration of the gene activity pattern conversion to facilitate the participation of hidden node genes. Black and white squares stand for inactive and active gene activities. "N" represents the locations reserved for hidden genes.

When evaluating the fitness for a gene regulatory network, only the locations that are not marked as "N" in the pattern will be considered to calculate the Hamming distance by comparing with the targeted activity pattern. I hypothesise this method can mimic the hidden genes.

CHAPTER 3

RESULTS

Gene activity patterns and the essential parameters of our evolutionary simulations are provided in the form of Tables 3.1 and 3.2 in order to facilitate repeatability of these experiments. The detailed explanations of these parameters are given after Table 3.2. Overall, only the elite number will be specified in each experiment, since only that may vary in different experiments. All the other parameters are specified in Table 3.1 and Table 3.2 and are consistent in the experiments, except those in the Appendix. The experiments in Appendix were conducted in an early stage of this project.

Table 3.1 Parameters of gene activity patterns.

Gene Activity Pattern	Generation to Add a New Pattern
1, -1, 1, -1, 1, -1, 1, -1, 1, -1	0
1, -1, 1, -1, 1, 1, -1, 1, -1, 1	500

Table 3.2 Parameters of the evolutionary simulation.

Edge Size	Perturbation Number	Perturbation Rate	Mutation Rate	Population Size	Tournament Size	Reproduction Rate	Maximum Generation	Elites Number
20	75	0.15	0.05	100	Proportional	0.9	2000	0 or 10

Gene Activity Patterns: the patterns that are perturbed, and towards which gene regulatory networks evolve.

Generations to add a new pattern: the generations to add new gene activity patterns towards which networks evolve.

Edge Size: the initial number of edges in the original gene regulatory networks.

Perturbation Number: the number of corrupted versions of each gene activity pattern.

Perturbation Rate: the expectation of the number of corrupted genes in a pattern.

Mutation Rate: the probability of a gene node to gain or lose an interaction in a network.

Population Size: the number of individuals in the population in every generation.

Tournament Size: the size of the tournament selection; where tournament selection is used, the size of the tournament; where proportional sections is used, it is annotated as "proportional".

Reproduction Rate: the proportion of children reproduced over the entire population. Any vacancy will be filled by the tournament scheme selecting individuals from the previous generation.

Maximum generation: the generation when the simulation will terminate after reaching it.

Elites number: number of elites in the evolutionary simulation.

The Wilcoxon Signed-Rank Test was used to statistically determine the validity of the experimental conclusions. Each experiment contains 40 independent trials. The evaluation metrics include both the eventual fitness values and final modularity Q scores in the last generation. Graphs of fitness and modularity Q scores are plotted by averaging the corresponding values of all the experimental trials at every generation. These graphs can help understand the trends of the entire evolutionary process. All these detailed graphs illustrating the complete evolutionary progress are attached in the Appendix.

3.1 Diagonal Crossover Mechanism Promotes Modularity

I simulated 40 independent evolutions for the development with no crossover and with each of the two crossover mechanisms, namely horizontal crossover and diagonal crossover, respectively. None of these simulations applied elitism. Overall, the diagonal crossover mechanism performed better than no crossover and the horizontal crossover, regarding both regulatory performance and modularity emergence, as Tables 3.1 and 3.2 indicate.

Table 3.1 Results for diagonal crossover driving modularity

	Diagonal Crossover	Horizontal Crossover	No Crossover
Fitness	0.9492	0.9444	0.9476
Modularity Q Score	0.3278	0.2901	0.1919

Table 3.2 Statistical significant results for diagonal crossover driving modularity.

	Fitness P	Modularity Q Score P
No Crossover < Horizontal Crossover	0.2415	9.2918e-7
Horizontal Crossover < Diagonal Crossover	0.0002	0.0372

3.2 Elitism Hampers Modularity

I simulated 40 evolutionary trials with 10 elites and without any elites. That was 80 trials in total. The experimental results indicate that elitism will hamper both the networks' regulatory capabilities and modularity emergence, as shown in Table 3.3 and Table 3.4.

Table 3.3 Results for elitism hampering the modularity.

	Without Elites	With 10 Elites
Fitness	0.9492	0.9472
Modularity Q Score	0.3278	0.2745

Table 3.4 Statistical significant results for elitism hampering the modularity.

	Fitness P	Modularity Q Score P
With 10 Elites < Without Elites	0.0019	0.0044

3.3 Recombination Hotspots Did Not Drive Modularity

The experiments in this section act as a replication of the work of [9] in order to explore the contribution of recombination hotspots to modularity. This includes 40 trials for each of the networks with and without hotspots. That is a total of 80 trials. Larson et al. utilised Age-Fitness Pareto Optimisation in [9], which is a multi-objective evolutionary algorithm to maintain the diversity of populations [19]. However, real biological evolution is single-objective. As such, I re-implemented the work in [9] with standard evolutionary algorithms to investigate whether they can still lead to the same conclusion. The results in Tables 3.5

and 3.6 indicate that recombination hotspots did not contribute to modularity in the given parameter set. In contrast, they may hinder the development of evolutionary simulations.

Table 3.5. Results for recombination hotspots not driving modularity.

	Without Hotspots	With Hotspots
Fitness	0.9492	0.9480
Modularity Q Score	0.3278	0.3153[3]

Table 3.6. Statistical significant results for recombination hotspots not driving modularity.

	Fitness P	Modularity Q Score P
With Hotspots< Without Hotspots	0.0218	0.3678

[3] Surprisingly, in Table 3.5 the fitness averages only differ in the 3rd decimal place yet it's significant, and the Q differs on the 2nd decimal point yet is not significant.

Also, as previously mentioned, recombination hotspots are represented as a normalised probability distribution. The heat map in Figure 3.1 illustrates the eventual probability distribution of recombination hotspots, where blacker blocks are more likely to incur crossover at this node. Furthermore, the heat map indicates that recombination hotspots did not appear at the boundaries of network modules, which also contradicted [9].

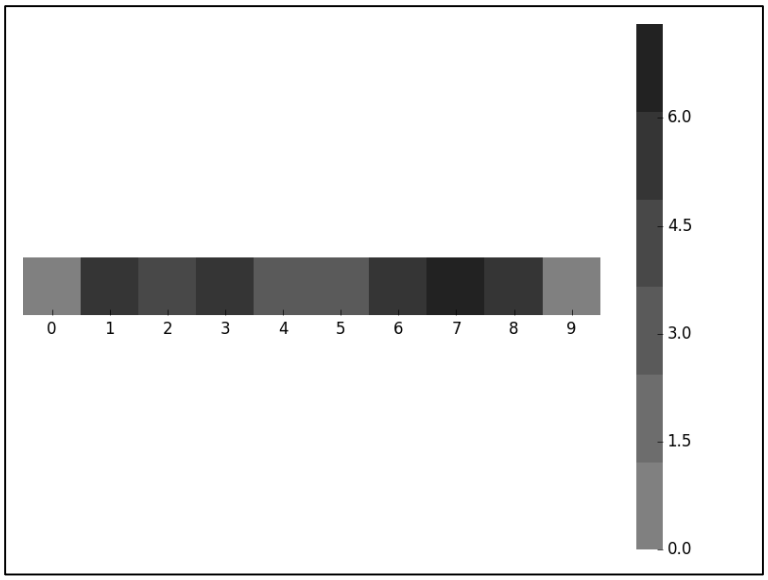


Figure 3.1 Heat map of a recombination hotspot for recombination hotspots not driving modularity [30].

3.4 Inter-Module Connections Can Hamper Network Fitness

Fitness values of gene regulatory networks were measured after removing interconnections between modules in order to understand the functionality of inter-module interactions. The results indicated that among 40 networks which had the highest fitness values and relatively low modularity Q scores in their corresponding evolutionary simulations, 24 of them demonstrated higher fitness after manually converting them into modular structures by deleting inter-module edges. Note that these modified solutions had a lower density than was expected from the evolutionary operations (sec 2.4), and thus may have been excluded from the search space. That is, there existed non-modular networks that exhibited better fitness performance after removing all the inter-module connections. For example, the right network in Figure 3.2 was the consequence of removing inter-module connections of the network in the left. The fitness value of the latter was 0.9502 after it had removed 6% connections of the former, whose fitness was 0.9472. Further statistical investigations will be conducted in the future.

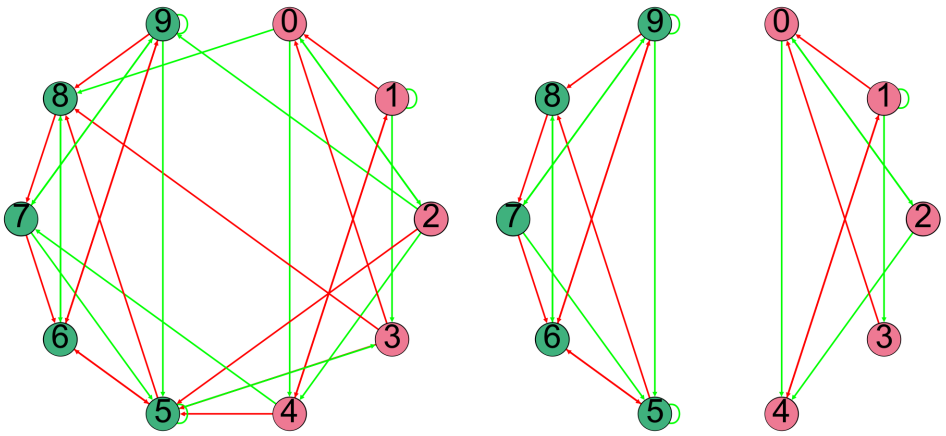


Figure 3.2 Illustration of inter-modularity connection removal. Nodes in different colours are partitioned into distinct modules. Green edges represent activation and red edges represent repression.

In summary, my experimental results indicate that diagonal crossover mechanism could better promote modularity emergence than no crossover or other mechanisms. Additionally, elitism, which is commonly utilised in genetic algorithms, would impede the emergence of modular networks. Furthermore, unlike [9], recombination hotspots did not contribute to modularity emergence in my experiments. Moreover, there existed some modular networks that demonstrated better fitness after the removal of inter-module connections.

CHAPTER 4

Discussion

As discussed in the Introduction section, there are four different aims in this work:

1. Can I refine evolutionary simulations to make them faster?
2. Do more modular networks perform better than less modular networks in my evolutionary simulations? That is, do more modular structures gain the dominant status on survivability?
3. If networks with high-level modularity do not attain surviving dominance, are there any ways to facilitate their expected dominant status?
4. Towards what direction or directions are modular structures evolving?

Overall, the experimental results suggested four discoveries: 1. crossover mechanisms can accelerate evolutionary simulations; 2. the theory of gene specialisation cannot explain the dominant status of modular structures on survivability; 3. the role of hidden genes is not related to modular structures' dominant status concerning survivability; 4. modular networks may evolve towards structures requiring fewer connections.

4.1 Crossover Mechanisms Can Accelerate Evolutionary Simulations

The Boolean model that I have utilised to simulate biological networks was originally proposed by Wagner in his study on "epigenetic stability" [20]. His work indicated that random recombination made no difference for the evolution of stability, which may be due to the freeness of random recombination on choosing locations to undertake crossover. This can corrupt the modular structures in biological networks.

Conversely, my experimental results suggested that proper recombination methods can contribute to the evolvability of organisms. The diagonal crossover proposed in this report is able to preserve underlying network modules. Although the crossover mechanism utilised by Larson et al. did not preserve community structures as well as diagonal crossover, its partitioning is still based on a network-like structure. This can be the reason why both of these two crossover mechanisms could help in obtaining modularity, with diagonal crossover better than horizontal crossover.

Meanwhile, different combinations of parental traits can increase the diversity of the population so that the evolution can be more exploratory.

4.2 Modular Systems Did Not Gain Dominance on Survivability

Early experiments in my study did not evolve high modularity, which was resolved after removing the elites in the evolutionary simulation. Details for these experiments are in Appendix G. This implies that individuals that performed optimally in the early stage might not be optimal on modularity. In other words, reserved elites in each generation did not have the most modular gene regulatory networks. Furthermore, repeats of the experiments done by Larson et al. in [9] could not reach the same conclusion that recombination hotspots promoted modularity, without the aid of a multi-objective algorithm called Age-Fitness Pareto optimization. Subsequent experiments with more complex targeted gene activity patterns and diploid individuals still could not make hotspots help in modularity, although hotspots might aid the evolutionary fitness when facing more complex targeted patterns. These experimental details are attached in Appendix H and I.

Overall, these phenomena suggest that the modularity emergence condition, namely gene specialisation promotes modular networks, may not be plausible to explain biological modularity. They indicated that modules in the simulated gene regulatory networks did not gain dominance in determining the survivability of individuals. However, biologically, modular networks are dominant and ubiquitous. In order to further investigate the

plausibility of this theory, namely specialisation driving modularity, I obtained the most optimal gene regulatory network among networks that were the most modular. Conversely, I also collected the network that was the least modular among those that had the greatest fitness value. These networks were collected from the generated results of experiments in Section 3.1.

Biologically, I expected the fitness value of the latter would be lower than the fitness of the former. Nevertheless, the situation was the converse. That is, some less modular networks were more robust than more modular ones, as Table 4.1 indicates. This is not consistent with what has been observed in biology.

Table 4.1 Modularity dominance analysis results for generated data of Section 3.1.

Generation Range	Modularity	Fitness
(500, 2000)	0.5000	0.9482
	0.1736	0.9502

There were multiple potential hypotheses that could explain this deviance from biological observations. Firstly, I hypothesise that the inconsistency was due to the targeted gene activity patterns being over-simple. That is, the number of genes in a pattern was not

sufficient or the number of patterns was not enough. A modular network may give great performance on complex tasks, but worse than non-modular ones for simple tasks. Thus, I conducted a complicated evolutionary simulation consisting 7 patterns, each of which comprised 15 gene nodes. This evolution lasted for 35,000 generations. Other detailed parameters are in Tables 4.2 and 4.3.

Table 4.2 Gene activity patterns for the 35000-generation simulation.

Gene Activity Patterns	Generation to Add a New Pattern
1, -1, 1, -1, 1, 1, -1, 1, -1, 1, 1, -1, 1, -1, 1	0
1, -1, 1, -1, 1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1	500
1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1	2000
1, -1, 1, -1, 1, -1, 1, -1, 1, -1, -1, 1, -1, 1, -1	5000
-1, 1, -1, 1, -1, 1, -1, 1, -1, 1, 1, -1, 1, -1, 1	10,000
-1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1	17,000
-1, 1, -1, 1, -1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1	26,000

Table 4.3 Parameters for the 35000-generation simulation.

Edge Size	Perturbation Number	Perturbation Rate	Mutation Rate	Population Size	Tournament Size	Reproduction Rate	Maximum Generation	Elites Number
45	75	0.15	0.05	100	Proportional	0.99	35000	0

The evolutionary progress for the best-fit individual in every generation is in Figure 4.1. I conducted the modularity dominance analysis again and the results are in Table 4.4. Overall, the complex of gene activity patterns could not resolve the issue of non-dominance of modular networks on survivability.

Table 4.4 Parameters of modularity dominance analysis results for the 35000-generation simulation.

Generation Range	Modularity	Fitness
(26000, 35000)	0.5506	0.9100
	0.4150	0.9419

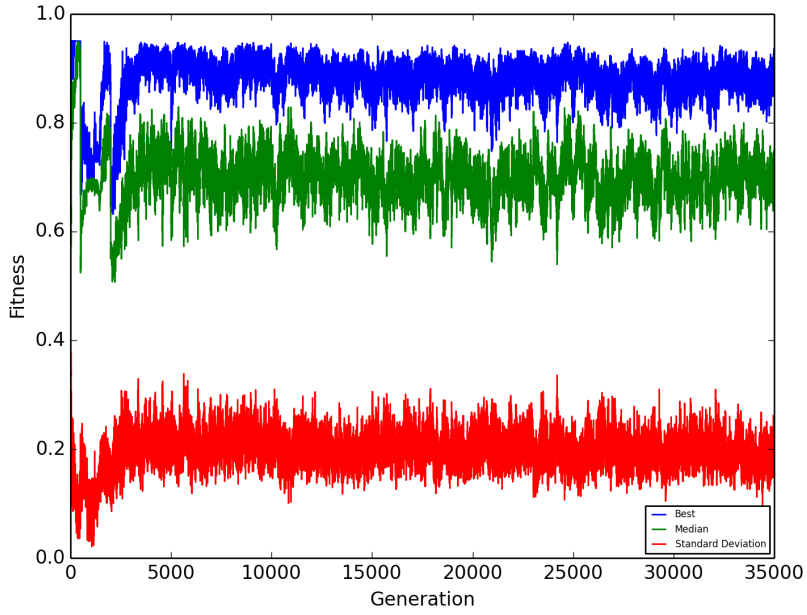


Figure 4.1 The 35000-generation evolutionary fitness process.

4.3 Hidden Genes Cannot Facilitate the Dominance of Modular Networks

Another hypothesis is that it was the interaction connecting different modules that caused the non-dominance of modular networks. The existence of these interactions might have put overmuch pressure on some nodes that were regulating gene activity patterns as well as communicating with other modules. Biologically, there exist hidden regulatory genes. These genes in the network do not directly participate in the regulatory process. Therefore,

I hypothesised these genes may take the responsibility of assisting interacting between modules so that other genes can focus on their regulatory functions. Thus, I conducted experiments for networks containing hidden genes. The experimental parameters were the same as those in Section 3.1, except they included hidden gene nodes. However, the modular networks still did not gain dominance in evolution, and regulatory genes apart from the hidden ones still communicate with genes in other modules. This is shown in Table 4.3 and Figure 4.2.

Table 4.3 Modularity dominance analysis results for the evolution with hidden genes.

Generation Range	Modularity	Fitness
(500, 2000)	0.500	0.9482
	0.0898	0.9502

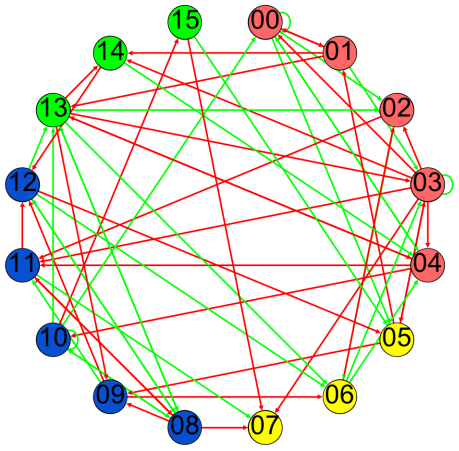


Figure 4.2 The gene regulatory network with hidden genes at the final generation. Green edges represent activation and red edges represent suppression.

4.4 Modular Networks May Evolve Towards Requiring Fewer Connections

As results in Section 3.4 suggested, interactions between modules sometimes do not contribute to and even hamper the regulation activity of networks. That is, a network can gain a better performance by removing those inter-module connections, which indicates that modular networks require fewer connections in total. However, the evolutionary simulation utilised in this report adopted a mutation pattern that maintains a consistent number of edges in the entire network. Thus, modular solutions cannot discard the extraneous edges. These unwanted edges can account for the non-dominant status of

modular networks since those networks with high modularity Q scores may not be the optimum solutions concerning modularity. That is, there are other more desirable networks with potential modular structures and could have demonstrated highest fitness values. Nonetheless, they cannot demonstrate high Q scores due to their additional edges. These extra edges may also impede the evolutionary process of recombination hotspots on ascertaining the module boundaries. Therefore, our results showed that recombination hotspots do not aid the evolution of modularity.

Clune et al. stated that the evolutionary origin of modularity is due to the cost associated with every connection in the network [11]. They demonstrated this by their experiments indicating that there was a significant emergence of modular networks after imposing a penalty on the number of edges in the network [11]. That is, modularity arose in order to minimise connection costs. Specifically, they made simulated organisms evolve towards two objectives, namely to maximise performance and to minimise edge costs. However, in reality, biological organisms evolve in a single-objective fashion. That is, they are only selected under the pressure of fitting the (changing) environment. Therefore, the theory stating that modularity comes from minimising connection costs may not be sufficiently plausible.

My results revealed a converse causality of Clune et al.'s explanation on modularity. To be specific, that connecting costs of modular networks are lower may be because modular networks need fewer edges to support their activities than non-modular ones. It may be also due to this, Clune et al. can recognise and select more modular systems by choosing structures in which there are fewer connections. Nevertheless, my results suggest that containing fewer edges is a property of more modular networks, not their evolutionary origin.

4.5 Future Work

In the future, I will relax the mutation constraint on the number of total edges so that a network can potentially discard its extraneous connections. I hypothesise that this will lead to some individuals with both high fitness and modularity, so that the phenomenon that the most modular networks not gaining the dominance on survivability will disappear. Additionally, my experiments suggested that modular solutions with good performance might come from initial non-modular solutions. In other words, directly searching for modular structures in evolutionary simulations may return worse solutions than manufacturing modular solutions with good performance. Additionally, insights into the ideal average number of connections in modular structures may suggest inspiration in

other engineering applications such as pruning densely connected artificial neural networks, minimising the number of logic gates in evolutionary simulations, and so on.

CHAPTER 5

CONCLUSION

In sum, I found that the diagonal crossover mechanism can promote the emergence of modularity. In contrast, elitism hampers the rise of modular networks, which indicates that early optimal individuals did not demonstrate high-level modularity. Further experiments also indicated that the theory on the origin of modularity resulting from specialisation has limitations on explaining the surviving dominance of modular systems in biology. This might be the reason why artificial organisms with recombination hotspots did not show advantages in obtaining modularity, without the highly specific choice of Age-Fitness Pareto optimisation that Larson et al. applied in their study [9]. I initially hypothesised the reason was due to the inter-module connections. One possible remedy of introducing hidden nodes in gene regulatory networks failed to show improvements. In contrast, networks that have high fitness values could demonstrate better performance after converting them into modular structures by removing their inter-module edges. This suggests that modular networks initialised by gene specialisation may evolve towards structures requiring fewer number of connections in total. In the future, a well-understood advance in the modularity evolutionary direction and the average number of edges in every

module can assist in designing engineered systems that are capable of solving more complex problems and autonomously adapting to new working environments.

CHAPTER 6

BIBLIOGRAPHY

[1] C. Pintea, "Bio-inspired computing," in *Advances in bio-inspired computing for combinatorial optimization problems*, Springer Berlin Heidelberg, 2014, pp. 3-19.

[2] D. Mange and M. E. Tomassini, Bio-inspired computing machines: Towards novel computational architectures, PPUR presses polytechniques, 1998.

[3] C. Teuscher, D. Mange, A. Stauffer and G. Tempesti, "Biosystems," in *Bio-inspired computing tissues: towards machines that evolve, grow, and learn*, 2003, pp. 235-244.

[4] X. Yang, Z. Cui, R. Xiao, A. Gandomi and M. Karamanoglu, Swarm intelligence and bio-inspired computation: theory and applications, Newnes, 2013.

[5] N. Kashtan and U. Alon, "Spontaneous evolution of modularity and network motifs," *Proceedings of the National Academy of Sciences of the United States of America,* vol. 102, no. 39, pp. 13773-13778, 2005.

[6] M. Hitz and B. Montazeri, “Measuring Coupling and Cohesion In Object-Oriented Systems,” 1995.

[7] G. Schlosser and G. Wagner, Modularity in development and evolution, University of Chicago Press, 2004.

[8] C. Espinosa-Soto and A. Wagner, “Specialization can drive the evolution of modularity,” *PLoS computational biology,* vol. 6, no. 3, p. e1000719, 2010.

[9] A. Larson, A. Bernatskiy, C. Cappelle, K. Livingston, N. Livingston, J. Long, J. Schwarz, M. Smith and J. Bongard, “Recombination Hotspots Promote the Evolvability of Modular Systems,” in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, 2016.

[10] G. Wagner, M. Pavlicev and J. Cheverud, “The road to modularity,” *Nature reviews. Genetics,* vol. 8, no. 12, pp. 921-931, 2007.

[11] J. Clune, J. Mouret and H. Lipson, “The evolutionary origins of modularity,” *Proc. R. Soc. B,* vol. 280, no. 1755, p. 20122863, 2013.

[12] M. Pigliucci, “Is evolvability evolvable?,” *Nature reviews. Genetics,* vol. 9, no. 1, p. 75, 2008.

[13] M. Kirschner and J. Gerhart, “Evolvability,” *Proceedings of the National Academy of Sciences,* vol. 95, no. 15, pp. 8420-8427, 1998.

[14] G. Wagner and L. Altenberg, “Perspective: complex adaptations and the evolution of evolvability,” *Evolution,* vol. 50, no. 3, pp. 967-976, 1996.

[15] G. Wagner and J. Mezey, “The role of genetic architecture constraints in the origin of variational modularity,” in *Modularity in development and evolution*, 2004, pp. 338-358.

[16] G. Wagner, “Homologues, natural kinds and the evolution of modularity,” in *American Zoologist*, vol. 36, 1996, pp. 36-43.

[17] M. Parter, N. Kashtan and U. Alon, “Environmental variability and modularity of bacterial metabolic networks,” *BMC evolutionary biology,* vol. 7, no. 1, p. 169, 2007.

[18] H. Dooner, “Extensive interallelic polymorphisms drive meiotic recombination into a crossover pathway,” *The Plant cell,* vol. 14, no. 5, pp. 1173-1183, 2002.

[19] M. Schmidt and H. Lipson, “Age-fitness pareto optimization,” in *Genetic Programming Theory and Practice VIII*, Springer New York, 2011, pp. 129-146.

[20] A. Wagner, “Does evolutionary plasticity evolve?,” *Evolution,* vol. 50, no. 3, pp. 1008-1023, 1996.

[21] B. R. P. Josh, How the Body Shapes the Way We Think: A New View of Intelligence, MIT Press, 2004.

[22] NetworkX, “NetworkX,” NetworkX, 2017. [Online]. Available: https://networkx.github.io/. [Accessed 25 09 2017].

[23] L. Tsoukalas and R. Uhrig, Fuzzy and Neural Approaches in Engineering, Wiley, 1997.

[24] R. Poli, W. Langdon and N. McPhee, A field guide to genetic programming, http://lulu. com, 2008.

[25] W. a. M. G. Bateson, Mendel's principles of heredity, Courier Corporation, 2013.

[26] R. Albert and H. G. Othmer, "The Topology of the Regulatory Interactions Predicts the Expression Pattern of the Segment Polarity Genes in Drosophila Melanogaster.," *Journal of Theoretical Biology,* vol. 223, no. 1, pp. 1-18, 2003.

[27] A. Aderem, "Systems biology: its practice and challenges," *Cell,* vol. 121, no. 4, pp. 511-513, 2005.

[28] M. Totten, "Exploring the Evolution of Modularity in Gene Regulatory Networks," 2015.

[29] V. Blondel, J. Guillaume, R. Lambiotte and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment,* no. 10, p. P10008, 2008.

[30] A. Gavin, P. Aloy, P. Grandi, R. Krause, M. Boesche, M. Marzioch, C. Rau, L. Jensen, S. Bastuck, B. Dümpelfeld and A. Edelmann, "Proteome survey reveals modularity of the yeast cell machinery," *Nature,* vol. 440, no. 7084, p. 631, 2006.

[31] M. Newman and M. Girvan, “Finding and evaluating community structure in networks,” *Physical review E,* vol. 69, no. 2, p. 026113, 2004.

[32] S. Muk, “Flexible Code Foundation Development For Further Study In Gender-based GA with Recombination Hotspots,” 2016.

[33] W. Callebaut and D. Rasskin-Gutman, Modularity: understanding the development and evolution of natural complex systems, MIT press, 2005.

CHAPTER 7

APPENDIX

A. Project Description & Independent Study Contract

Australian National University

INDEPENDENT STUDY CONTRACT

Note: Enrolment is subject to approval by the projects co-ordinator

SECTION A (Students and Supervisors)

UniID: u5505995

SURNAME: Qin FIRST NAMES: Zhenyue

PROJECT SUPERVISOR (*may be external*): Bob McKay

COURSE SUPERVISOR (*a RSCS academic*): Tom Gedeon

COURSE CODE, TITLE AND UNIT: COMP4560 Advanced Computing Project 12 units

SEMESTER ☒ S1 YEAR: _2017_ ☒ S2 YEAR: _2017_

PROJECT TITLE:

~~Gender-based GA with Recombination Hotspots~~
An Exploration on Modularity Emergence Conditions in Evolutionary Systems

LEARNING OBJECTIVES:

Experience with co-evolutionary meta-strategies
Experience with testing of evolutionary algorithms
Experience with data analysis

PROJECT DESCRIPTION:

• Brief literature survey
• Background: As important difference between GAs and eukaryotic genetics is recombination hotspots: in GAs, recombination locations are almost always randomly selected, whereas in (diploid) eukaryotes, the locations are very far from random. There is strong evidence that the locations are under evolutionary selection. The interest is to determine whether recombination hotspots are algorithmically important. Previous work has suggested that recombination hotspots in a typical monoploid GA representation applied to static optimisation problems do not yield any advantage. However that work did not look at dynamic, and particularly co-evolutionary, problems (there is some evidence that hotspots are selected for in immune-related chromosome regions of the human genome).
• Extend existing code base to perform experimental comparisons
• Design and run an experimental evaluation of hotspots in genetic algorithms
• Statistical analysis of results.
• Optional task: extend code to include further features from evolution in biological systems
• Write report

Australian National University

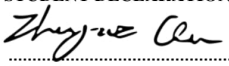
ASSESSMENT (as per course's project rules web page, with the differences noted below):

Assessed project components:	% of mark	Due date	Evaluated by:
Report: name style: _____ (e.g. research report, software description...)	45		Sumudu Mendis
Artefact: name kind: _____ (e.g. software, user interface, robot...)	45		Tom Gedeon
Presentation:	10		

MEETING DATES (IF KNOWN):

Weekly

STUDENT DECLARATION: I agree to fulfil the above defined contract:

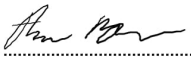


..... 20 February 2017...

Signature Date

SECTION B (Supervisor):

I am willing to supervise and support this project. I have checked the student's academic record and believe this student can complete the project.



..... 20 February 2017...

Signature Date

REQUIRED DEPARTMENT RESOURCES:

SECTION C (Course coordinator approval)

…….. ……..

Signature Date

SECTION D (Projects coordinator approval)

…….. ……..

Signature Date

B. Description of Artefacts

Figure 7.1 presents a list of all submitted program code files and clear descriptions on their authorship. All the files that were implemented by Zhenyue Qin (the author of this report) will be in the colour of black and all the other programs that were implemented by Siu Kei Muk will be in the colour blue. However, Muk's programs only served as a basic gene algorithm framework for my work [32]. All the experiments and research-related work were designed and conducted by me. Additionally, I added a lot of extensions to the original framework in order to conduct those further research. Please review the code for more detailed descriptions.

```
├── experiments
│   ├── exp1_soto_larson_perturbation_comparison
│   │   ├── HaploidGRN2Target10MatrixChinMain.java
│   │   └── HaploidGRN2Target10MatrixSotoMain.java
│   ├── exp2_crossover_comparison
│   │   ├── HaploidGRN2Target15MatrixChinMain.java
│   │   └── HaploidGRN2Target15MatrixLarsonMain.java
│   ├── exp3_larson_work_reproduction
│   │   ├── HaploidGRN2Target10MatrixLarsonSPXMain.java
│   │   ├── HaploidGRN2Target10MatrixNoXMain.java
│   │   ├── HaploidGRNMatrixMain.java
│   │   └── HotspotHaploidGRNMatrixMain.java
│   ├── exp4_hotspot_aid_matrix_x_validation
│   │   ├── DiploidGRNFastMatrixSPXMain.java
│   │   ├── DiploidGRNMatrixFixedSPXMain.java
│   │   └── HotspotDiploidGRNFastMatrixSPXMain.java
│   ├── exp5_crossover_aid_validation
│   │   └── DiploidGRNFastMatrixNoXMain.java
│   ├── exp6_haploid_hotspot_aid_matrix_x_validation
│   │   ├── HaploidGRNFixedPointXMatrixMain.java
│   │   ├── HaploidGRNMatrixMain.java
│   │   ├── HaploidGRNMatrixMainWithHiddenGenes.java
│   │   ├── HaploidGRNMatrixMainWithHiddenLayer.java
│   │   └── HotspotHaploidGRNMatrixMain.java
│   ├── exp7_modularity_analyzer
│   │   └── ModularityAnalyzer.java
```

```
│   ├── experiment8
│   │   ├── GenderDiploidGRNMain.java
│   │   └── GenderHotspotDiploidGRNMain.java
│   ├── experiment9
│   │   ├── GenderDiploidGRN2TargetMain.java
│   │   └── GenderHotspotDiploidGRN2TargetMain.java
│   └── experiment10
│       ├── DiploidGRN3Target10FastMatrixMain.java
│       ├── HaploidGRN2Target10MatrixChinMain.java
│       ├── HaploidGRNMatrixMain.java
│       ├── HotspotDiploidGRNFastMatrixSPXMain.java
│       ├── HotspotHaploidGRNMatrixMain.java
│       ├── TestField1.java
│       ├── TestField10.java
│       ├── TestField11.java
│       ├── TestField12.java
│       ├── TestField2.java
│       ├── TestField3.java
│       ├── TestField4.java
│       ├── TestField5.java
│       ├── TestField6.java
│       ├── TestField7.java
│       ├── TestField8.java
│       └── TestField9.java
├── ga
│   ├── collections
│   │   ├── BaseStatistics.java
│   │   ├── DetailedGenderStatistics.java
│   │   ├── DetailedStatistics.java
│   │   ├── GenderPopulation.java
│   │   ├── Individual.java
│   │   ├── Population.java
│   │   ├── PopulationMode.java
│   │   ├── SimpleElitesStatistics.java
│   │   └── Statistics.java
│   ├── components
│   │   ├── GRNs
│   │   │   └── DirectedEdge.java
│   │   ├── chromosomes
│   │   │   ├── Chromosome.java
│   │   │   ├── Coupleable.java
│   │   │   ├── CoupleableWithHotspot.java
│   │   │   ├── Diploid.java
│   │   │   ├── GenderDiploid.java
│   │   │   ├── GenderHotspotDiploid.java
│   │   │   ├── Haploid.java
│   │   │   ├── SimpleDiploid.java
│   │   │   ├── SimpleHaploid.java
│   │   │   ├── SimpleHotspotDiploid.java
│   │   │   ├── SimpleHotspotHaploid.java
│   │   │   └── WithHotspot.java
```

```
│   │   ├── genes
│   │   │   ├── DataGene.java
│   │   │   ├── EdgeGene.java
│   │   │   ├── Gene.java
│   │   ├── hotspots
│   │   │   ├── Hotspot.java
│   │   │   └── MatrixHotspot.java
│   │   └── materials
│   │       ├── EdgeMaterial.java
│   │       ├── GRN.java
│   │       ├── GRNFactory.java
│   │       ├── GRNFactoryNoHiddenTarget.java
│   │       ├── GRNFactoryWithHiddenTargets.java
│   │       ├── GRNWithHiddenTargets.java
│   │       ├── Material.java
│   │       └── SimpleMaterial.java
│   ├── frame
│   │   ├── frames
│   │   │   ├── Frame.java
│   │   │   ├── SimpleDiploidFrame.java
│   │   │   ├── SimpleDiploidMultipleTargetFrame.java
│   │   │   ├── SimpleGenderHotspotDiploidFrame.java
│   │   │   ├── SimpleGenderHotspotMultipleTargetFrame.java
│   │   │   ├── SimpleHaploidFrame.java
│   │   │   ├── SimpleHotspotDiploidFrame.java
│   │   │   └── SimpleHotspotDiploidMultipleTargetFrame.java
│   │   └── states
│   │       ├── DiploidState.java
│   │       ├── HotspotState.java
│   │       ├── SimpleDiploidState.java
│   │       ├── SimpleGenderDiploidState.java
│   │       ├── SimpleGenderHotspotState.java
│   │       ├── SimpleHaploidState.java
│   │       ├── SimpleHotspotDiploidState.java
│   │       ├── SimpleHotspotHaploidState.java
│   │       └── State.java
│   ├── operations
│   │   ├── dominanceMapMutators
│   │   │   ├── DiploidDominanceMapMutator.java
│   │   │   └── ExpressionMapMutator.java
│   │   ├── dynamicHandlers
│   │   │   └── DynamicHandler.java
│   │   ├── expressionMaps
│   │   │   ├── DiploidEvolvedMap.java
│   │   │   ├── ExpressionMap.java
│   │   │   └── ProjectionMap.java
│   │   ├── fitnessFunctions
│   │   │   ├── FitnessFunction.java
│   │   │   ├── FitnessFunctionMultipleTargets.java
│   │   │   ├── GRNFitnessFunction.java
│   │   │   ├── GRNFitnessFunctionMultipleTargets.java
```

```
│   │   │   ├── GRNFitnessFunctionMultipleTargetsFast.java
│   │   │   ├── GRNFitnessFunctionMultipleTargetsFastHidden.java
│   │   │   ├── GRNFitnessFunctionMultipleTargetsHidden.java
│   │   │   ├── GRNFitnessFunctionMultipleTargetsSeparateHidden.java
│   │   │   └── GRNFitnessFunctionSingleTarget.java
│   │   ├── hotspotMutators
│   │   │   ├── HotspotMutator.java
│   │   │   └── RandomHotspotMutator.java
│   │   ├── initializers
│   │   │   ├── DiploidGRNHiddenTargetInitializer.java
│   │   │   ├── DiploidGRNInitializer.java
│   │   │   ├── GenderDiploidGRNInitializer.java
│   │   │   ├── GenderHotspotDiploidGRNInitializer.java
│   │   │   ├── HaploidGRNHiddenTargetInitializer.java
│   │   │   ├── HaploidGRNInitializer.java
│   │   │   ├── HotspotDiploidGRNHiddenTargetInitializer.java
│   │   │   ├── HotspotDiploidGRNInitializer.java
│   │   │   ├── HotspotHaploidGRNInitializer.java
│   │   │   └── Initializer.java
│   │   ├── mutators
│   │   │   ├── BinaryGeneMutator.java
│   │   │   ├── GRNEdgeMutator.java
│   │   │   ├── GRNModularisedEdgeMutator.java
│   │   │   └── Mutator.java
│   │   ├── postOperators
│   │   │   ├── PostOperator.java
│   │   │   ├── SimpleFillingOperator.java
│   │   │   └── SimpleFillingOperatorForNormalizable.java
│   │   ├── priorOperators
│   │   │   ├── PriorOperator.java
│   │   │   ├── SimpleElitismOperator.java
│   │   │   └── SimpleGenderElitismOperator.java
│   │   ├── reproducers
│   │   │   ├── BaseCoupleReproducer.java
│   │   │   ├── BaseReproducer.java
│   │   │   ├── CoupleReproducer.java
│   │   │   ├── DiploidMatrixReproducer.java
│   │   │   ├── DiploidReproducer.java
│   │   │   ├── GRNDiploidFixedMatrixReproducer.java
│   │   │   ├── GRNDiploidMatrixReproducer.java
│   │   │   ├── GRNHaploidMatrixDiagonalReproducer.java
│   │   │   ├── GRNHaploidMatrixHorizontalReproducer.java
│   │   │   ├── GRNHaploidMatrixReproducer.java
│   │   │   ├── GRNHotspotDiploidEvolvedSPXMatrixReproducer.java
│   │   │   ├── GRNHotspotHaploidMatrixReproducer.java
│   │   │   ├── GenderHotspotReproducer.java
│   │   │   ├── GenderReproducer.java
│   │   │   ├── HaploidReproducer.java
│   │   │   ├── HotspotDiploidMatrixReproducer.java
│   │   │   ├── HotspotDiploidReproducer.java
│   │   │   ├── Reproducer.java
```

Figure 7.1 Submitted program code files.

Table 7.1 gives the details of the code for testing correctness. All the tests and experiments were implemented with Java 1.8.0 and Python 2.7.10.

Table 7.1 Test descriptions.

Test Type	Test File	Test Purpose	Test Results
Unit Test	GRNFitnessFunctionTest.java	Test the correctness of fitness evaluation	Pass
	TournamentSelectionTest.java	Test the correctness of tournament selection	Pass
	CrossoverTest.java	Test the correctness of crossover mechanism	Pass
Integration Test	IntegrationTest.java	Test the correctness of the genetic algorithm flow, including the correctness of mutation and reproduction	Pass

ZhenyueChin / **Chin-GA-Project**

Branch: master ▾ **Chin-GA-Project** / **README.md** Find file | Copy path

Zhenyue Qin changed some namings `2675f90` 11 minutes ago

0 contributors

44 lines (30 sloc) 1.76 KB

Modularity Exploration in Evolutionary Systems

build passing downloads 29.32 M

Zhenyue Qin, u5505995@anu.edu.au,
Supervisors: Prof Tom Gedeon and Prof Bob Mckay. Much appreciate to them.

This readme file serves as a basic guidance on the usage of evolutionary simulations.

Instructions

System Requirements

JDK

```
1.7 or above (this is to execute Maven - it still allows you to build against 1.3
and prior JDK's).
```

Memory:

```
No minimum requirement.
```

Disk:

```
Approximately 10MB is required for the Maven installation itself. In addition to
that, additional disk space will be used for your local Maven repository. The size
of your local repository will vary depending on usage but expect at least 500MB.
```

Operating System:

```
Windows:
  Windows 2000 or above.
Unix based systems (Linux, Solaris and Mac OS X) and others:
  No minimum requirement.
```

Installation

```
1. Deploy the Maven project
2. Install networkx, numpy, pandas and community API for pythons.
3. Run evolutionary simulations in the evolution folder.
P.S. Intellij is highly recommended.
```

License

Figure 7.2 The README file.

Table 7.2 gives descriptions of the generated experimental results. To reiterate, these generated data can be downloaded from: https://drive.google.com/file/d/0B9dNEi7lDXnldy1sNmZuTWNXMDg/view?usp=sharing.

Table 7.2 Descriptions of generated experimental results.

The Name of Generated Results	Descriptions
complex-gene-activity-patterns-do-not-solve-inconsistency	Data related to Table 4.2 and 4.3.
elitism-reduce-modularity	Data related to Section 3.2.
hotspot-diploid-3-target-15-gene	Data related to Appendix I.
hotspot-haploid-3-target-15-gene	Data related to Appendix H.
different-crossover-mechanism-comparisons	Data related to Section 3.1.
larson-experiments-repeats	Data related to Section 3.3.
larson-modularity-worse-to-soto	Data related to Appendix C.

C. Comparison Between Stochastic and Deterministic Perturbations

Table 7.3 Parameters of gene activity patterns for comparison between perturbations.

Gene Activity Pattern	Generation to Add a New Pattern
1, -1, 1, -1, 1, -1, 1, -1, 1, -1	0
1, -1, 1, -1, 1, 1, -1, 1, -1, 1	300

Table 7.4 Parameters of the evolutionary simulation for comparison between stochastic and deterministic perturbations.

Edge Size	Perturbation Number	Perturbation Rate	Mutation Rate	Population Size	Tournament Size	Reproduction Rate	Maximum Generation	Elites Number
20	300	0.15	0.05	100	Tournament with Size 3	0.9	1050	10

Table 7.5 Results for comparison between stochastic and deterministic perturbations.

	Deterministic	Stochastic
Fitness	0.8813	0.9108
Modularity Q Score	0.1012	0.1840

Table 7.6 Statistical significant results for comparison between stochastic and deterministic perturbations.

	Fitness P	Modularity Q Score P
Stochastic < Deterministic	0.0281	0.0099

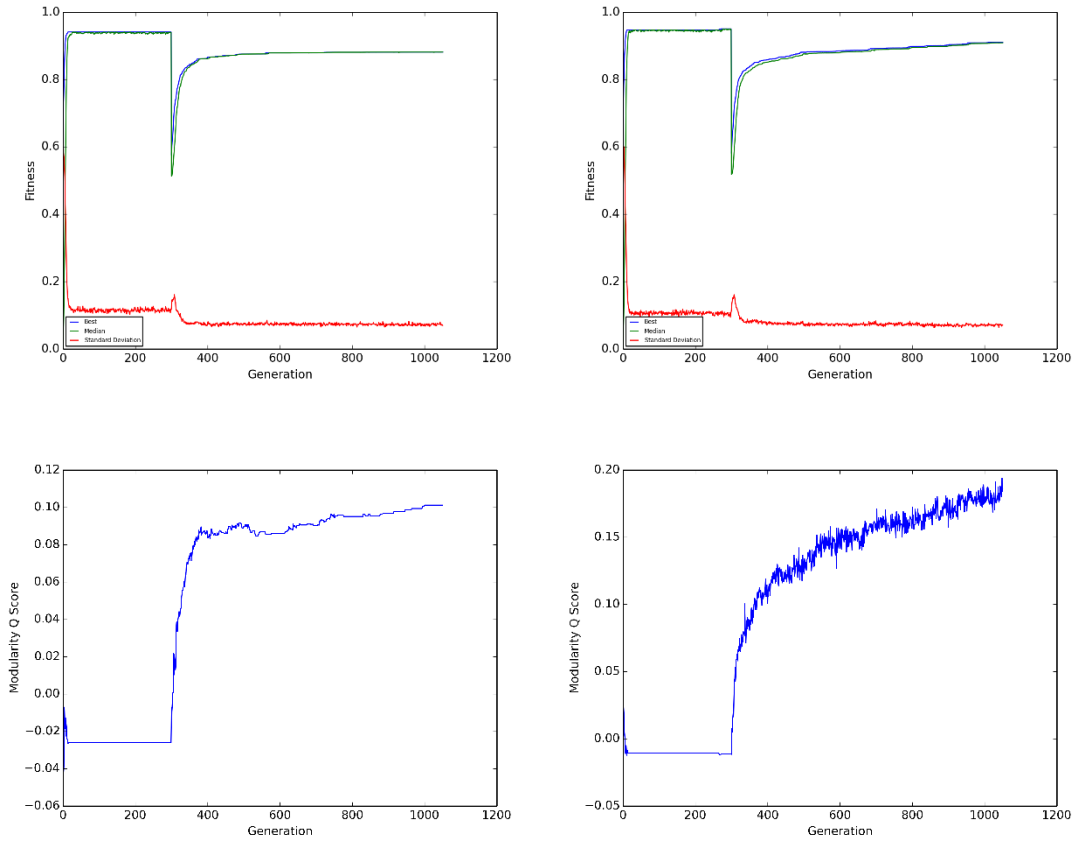
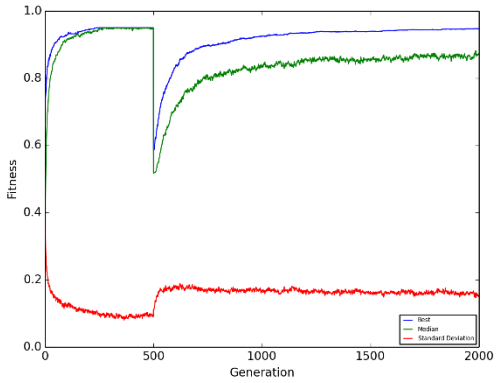
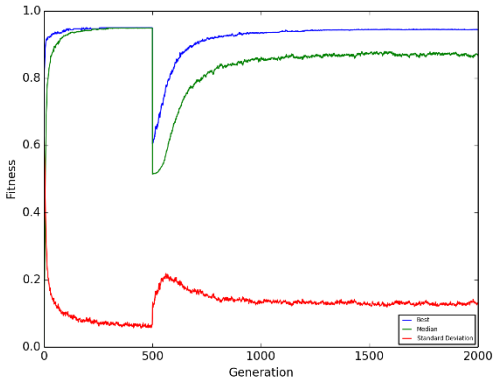


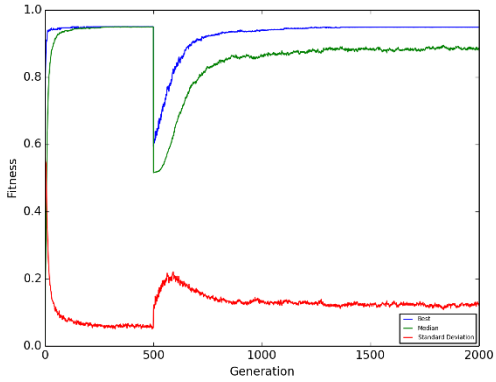
Figure 7.3 The evolutionary progresses for comparison between perturbations. (Top-Left) Fitness with perturbations of Larson et al. (Top-Right) Fitness with perturbations of Espinosa-Soto and Wagner. (Bottom-Left) Modularity with

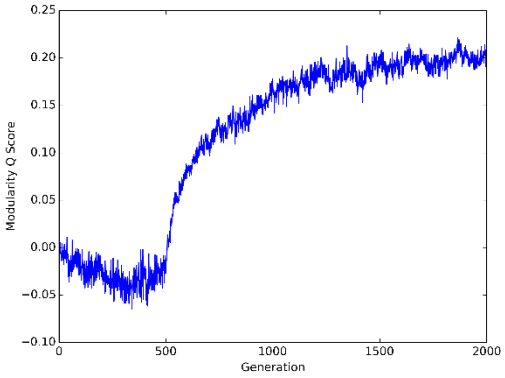
perturbations of Larson et al. (Bottom-Right) Modularity with perturbations of Espinosa-Soto and Wagner.

D. Graphs of Result 3.1









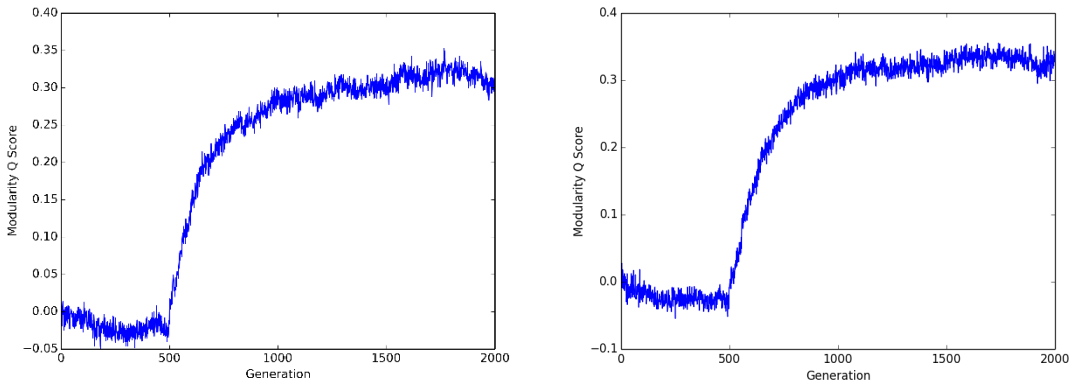


Figure 7.4 The evolutionary progresses of Result 3.1. (Top-Left) Fitness with no crossover mechanism. (Top-Right) Fitness with the horizontal crossover mechanism. (Centre-Left) Fitness with the diagonal crossover mechanism. (Centre-Right) Modularity with no crossover mechanism. (Bottom-Left) Modularity with the horizontal crossover mechanism. (Bottom-Right) Modularity with the diagonal crossover mechanism.

E. Graphs of Result 3.2

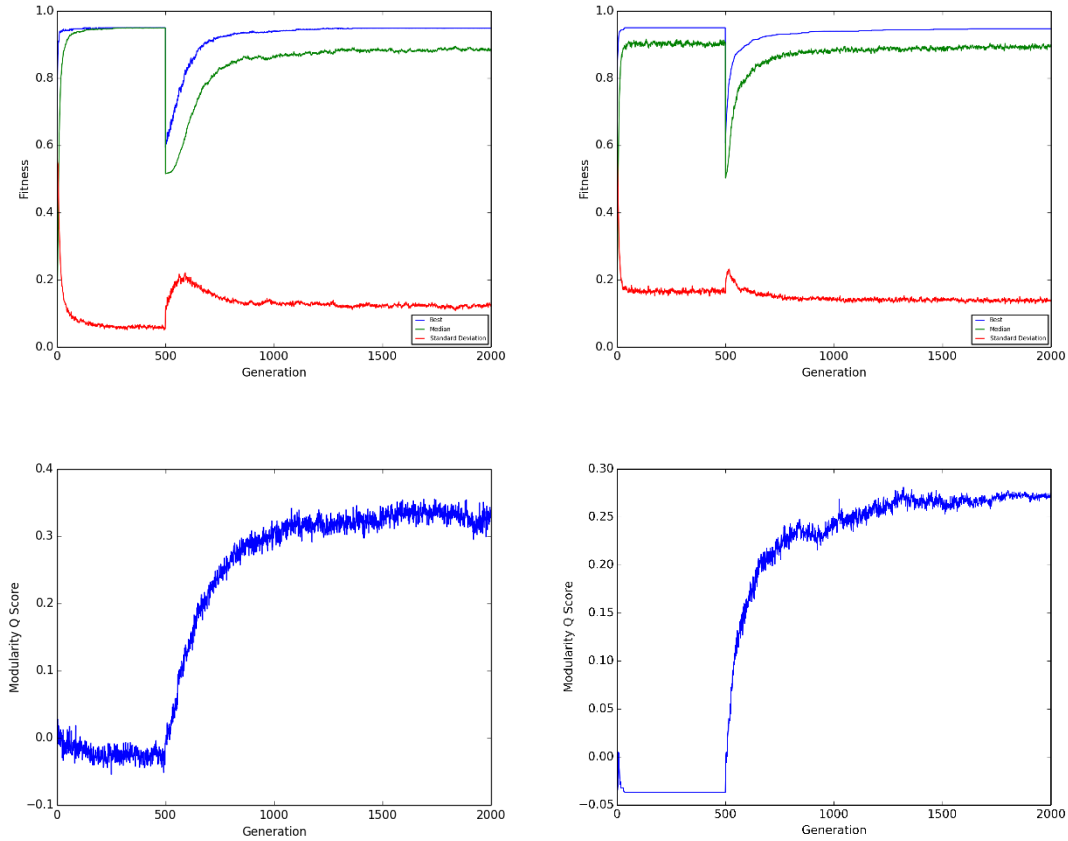


Figure 7.5 The evolutionary progresses of Result 3.2. (Top-Left) Fitness for without elitism. (Top-Right) Fitness for with elitism. (Bottom-Left) Modularity for without elitism. (Bottom-Right) Modularity for with elitism.

F. Graphs of Result 3.3

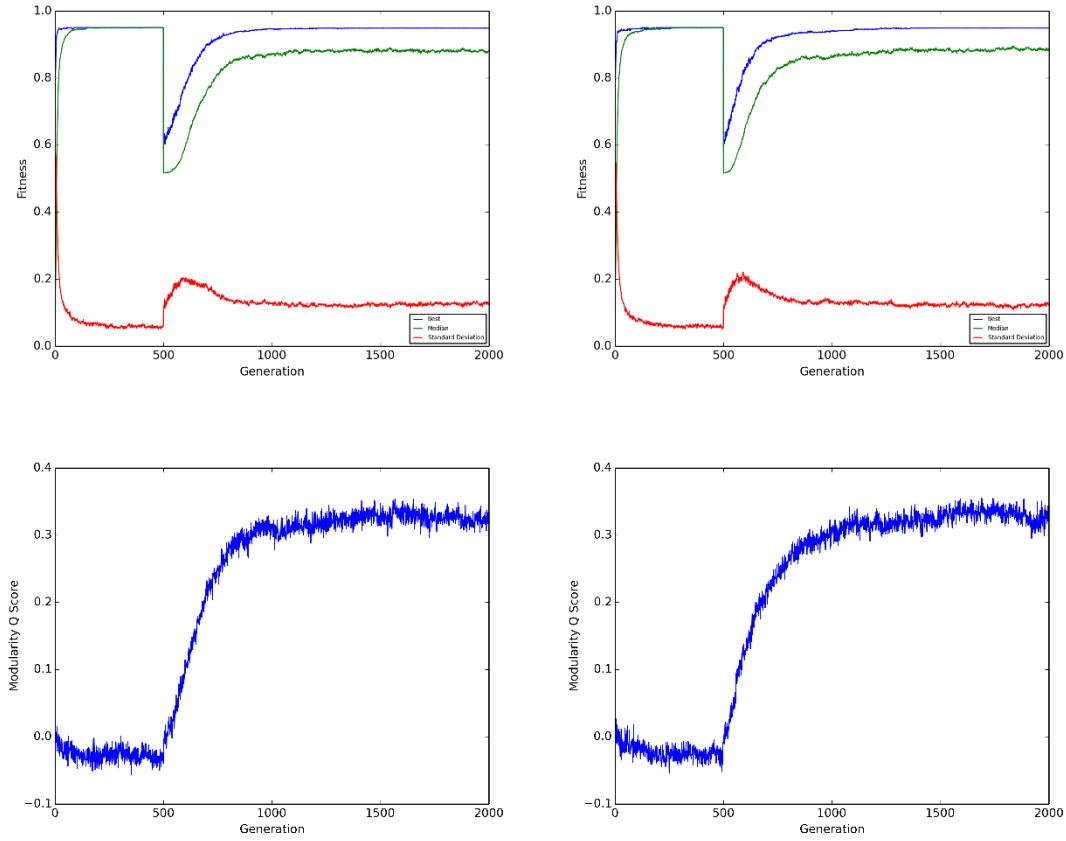


Figure 7.6 The evolutionary progresses of Result 3.3. (Top-Left) Fitness for without hotspots. (Top-Right) Fitness for with hotspots. (Bottom-Left) Modularity for without hotspots. (Bottom-Right) Modularity for with hotspots.

G. Early Simulations that did not Evolve High Modularity

Table 7.7 Parameters of gene activity patterns for early simulations that did not evolve high modularity.

Gene Activity Pattern	Generation to Add a New Pattern
1, -1, 1, -1, 1, -1, 1, -1, 1, -1	0
1, -1, 1, -1, 1, 1, -1, 1, -1, 1	300

Table 7.8 Parameters of the evolutionary simulation for early simulations that did not evolve high modularity.

Edge Size	Perturbation Number	Perturbation Rate	Mutation Rate	Population Size	Tournament Size	Reproduction Rate	Maximum Generation	Elites Number
20	300	0.15	0.05	100	Tournament with Size 3	0.9	1050	10

Table 7.9 Results for early simulations that did not evolve high modularity.

	Generation 300	Generation 1050
Fitness	0.8680	0.9442
Modularity Q Score	0.0201	0.0689

Table 7.10 Statistical significant results for early simulations that did not evolve high modularity.

Fitness P (Generation 1050 < Generation 300)	Modularity Q Score P (Generation 300 < Generation 1050)
3.2612e-7	0.0078

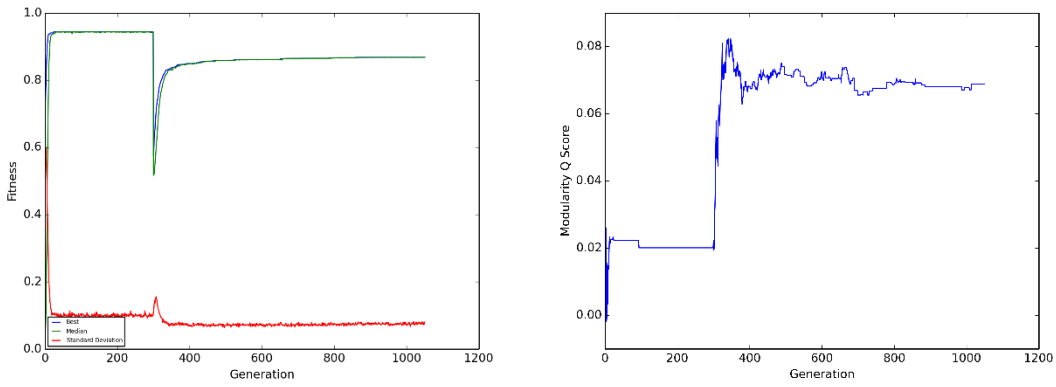


Figure 7.7 The evolutionary progresses for early simulations that did not evolve high modularity. (Left) Fitness. (Right) Modularity.

H. Recombination Hotspots with Three Gene Activity Patterns

Table 7.11 Gene activity patterns of recombination hotspots with three activity patterns.

Gene Activity Patterns	Generation to Add a New Pattern
1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1	0
1, -1, 1, -1, 1, -1, 1, -1, 1, -1, -1, 1, -1, 1, -1	500
1, -1, 1, -1, 1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1	2000

Table 7.12 Parameters of the evolutionary simulation for recombination hotspots with three activity patterns.

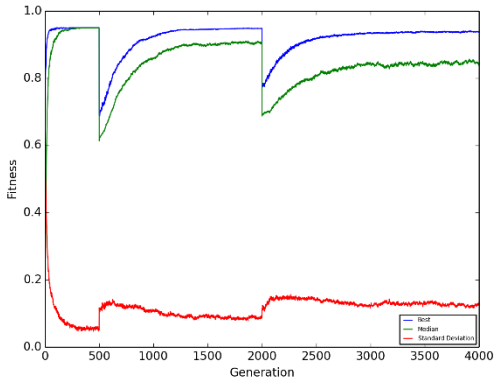
Edge Size	Perturbation Number	Perturbation Rate	Mutation Rate	Population Size	Tournament Size	Reproduction Rate	Maximum Generation	Elites Number
20	75	0.15	0.05	100	Proportional	0.9	4000	0

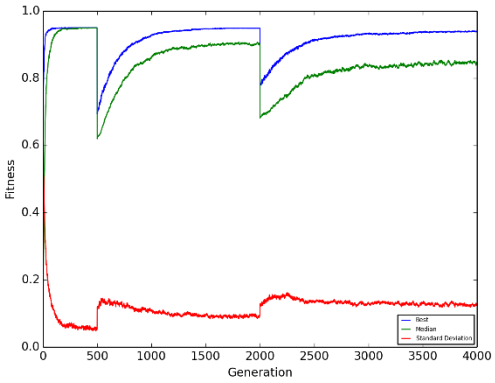
Table 7.13 Results for recombination hotspots with three activity patterns.

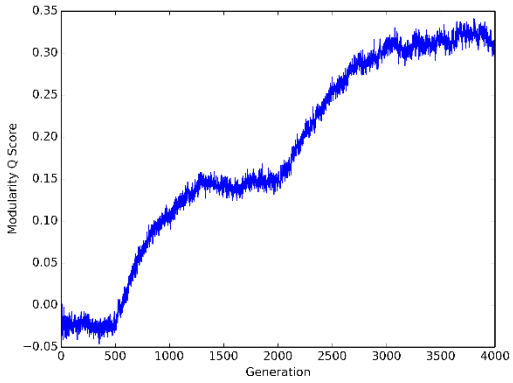
	With Hotspot	Without Hotspot
Fitness	0.9403	0.9340
Modularity Q Score	0.3146	0.3087

Table 7.14 Statistical significant results for recombination hotspots with three activity patterns.

	Fitness P (Wilcoxon Signed-Rank)	Modularity Q Score P (Wilcoxon Signed-Rank)
Without Hotspot < With Hotspot	0.0266	0.3468







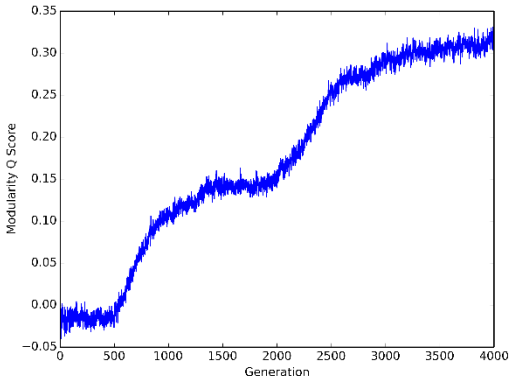


Figure 7.8 The evolutionary progresses for recombination hotspots with three activity patterns. (Top-Left) Fitness for without hotspots. (Top-Right) Fitness for with hotspots. (Bottom-Left) Modularity for without hotspots. (Bottom-Right) Modularity for with hotspots.

I. Recombination Hotspots with Diploids

Table 7.15 Gene activity patterns of recombination hotspots with diploids.

Gene Activity Patterns	Generation to Add a New Pattern
1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1	0
1, -1, 1, -1, 1, -1, 1, -1, 1, -1, -1, 1, -1, 1, -1	500
1, -1, 1, -1, 1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1	2000

Table 7.16 Parameters of the evolutionary simulation for recombination hotspots with diploids.

Edge Size	Perturbation Number	Perturbation Rate	Mutation Rate	Population Size	Tournament Size	Reproduction Rate	Maximum Generation	Elites Number
20	75	0.15	0.05	100	Proportional	0.9	4000	0

Table 7.17 Results for recombination hotspots with diploids.

	With Hotspot	Without Hotspot
Fitness	0.9454	0.9456
Modularity Q Score	0.1775	0.1797

Table 7.18 Statistical significant results for recombination hotspots with diploids.

	Fitness P (Wilcoxon Signed-Rank)	Modularity Q Score P (Wilcoxon Signed-Rank)
With Hotspot < Without Hotspot	0.8937	0.7881

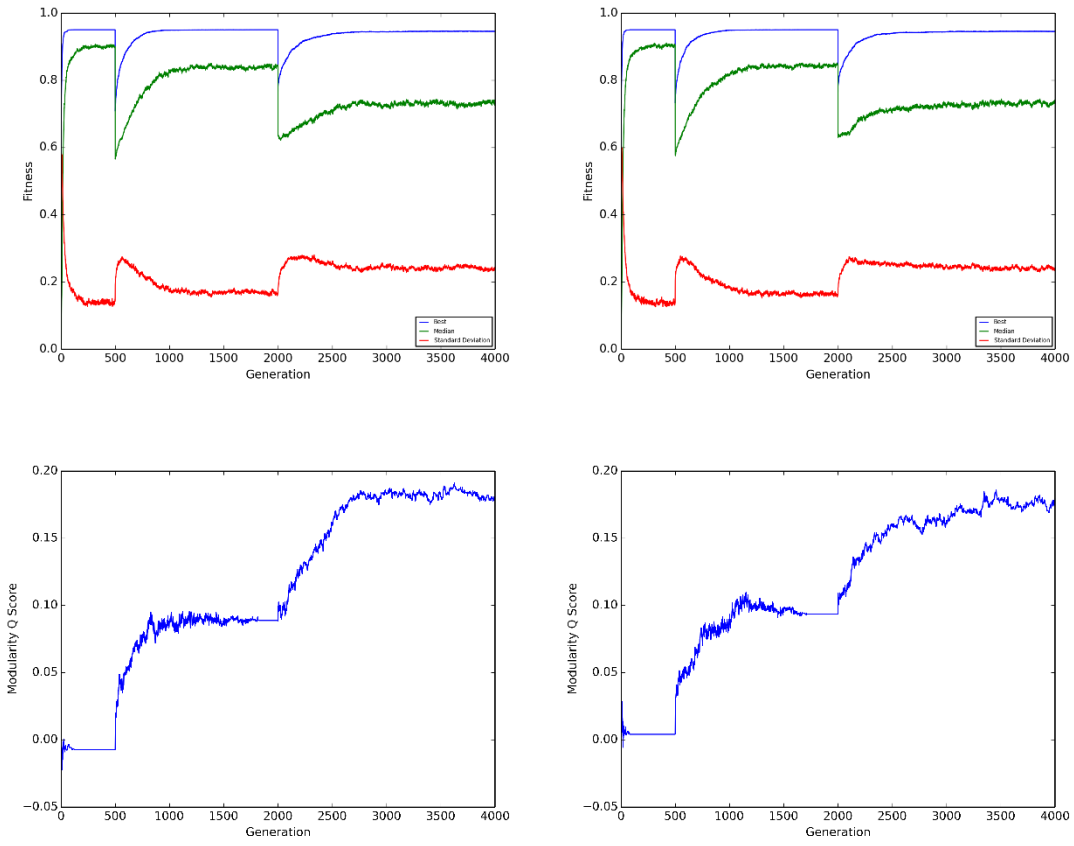


Figure 7.9 The evolutionary progresses for recombination hotspots with diploids. (Top-Left) Fitness for without hotspots. (Top-Right) Fitness for with hotspots.

(Bottom-Left) Modularity for without hotspots. (Bottom-Right) Modularity for with hotspots.