1-1-2015

Evolving Heterogeneous And Subcultured Social Networks For Optimization Problem Solving In Cultural Algorithms

Yousof Gawasmeh
Wayne State University,

EVOLVING HETEROGENEOUS AND SUBCULTURED SOCIAL NETWORKS FOR OPTIMIZATION PROBLEM SOLVING IN CULTURAL ALGORITHMS

by

YOUSOF A. GAWASMEH

DISSERTATION

Submitted to the Graduate School of

Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2015

MAJOR: COMPUTER SCIENCE

Approved By:

_____

Advisor Date

_____

_____

_____

_____

DEDICATION

I dedicate this work to my parents, wife, and daughters (Yara and Reema).

ACKNOWLEDGMENTS

The outcome of this thesis couldn't be accomplished without the support and guidance of my academic adviser, Professor Robert G. Reynolds. I'm so thankful to the great effort he has done during all the years of my PhD study and research at Wayne State University. I have greatly benefited from his constructive comments and insightful suggestions to draw the path of my success; he enormously supported the ideas to get mature and great enough to be published. I would never forget his warm encouragements to me, during our research meetings, to speak my mind and to define my goals the right way.

I would also like to show my sincere gratitude to my committee, including Prof. Grosu, Prof. Hua, and Prof. Chinnam. They have been greatly tolerant and supportive and their meticulous comments were an enormous help to me to improve my work.

I would also like to express the deepest appreciation to the Department of Computer Science at Wayne State University for their assistance and trust during my PhD study and my work, as teaching and research assistant, at the department. I want also to thank my colleagues at Wayne State University for their encouragement.

Special thanks from the depth of my heart to my wife, Heba Al-Sadi, for her patience and support, and to my lovely daughter Yara, who has inspired me to make a positive difference for our future. I wish I could compensate for the time I spent away from their live in order to complete this work.

Last but not least, I would like to express my heartfelt appreciation to my parents, brothers, sisters, family, and friends for their warm encouragement and inspiration that inspired me to finish this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Introduction.

Cultural Algorithms (CA) are computational models of social evolution based upon principles of Cultural Evolution. A Cultural Algorithm is composed of a Belief Space consisting of a network of active and passive knowledge sources, and a Population Space of agents. The agents are connected via a social fabric over which information used in agent problem solving as passed. The network is termed a "social fabric" because each connection must be maintained or it will be lost. The knowledge sources in the Belief Space compete with each other to influence the decision making of agents in the Population Space. Likewise, the problem solving experiences of agents in the Population Space are sent back to the Belief Space and are used to update the knowledge sources there. It is a dual inheritance system in which both the Population and Belief spaces evolve in parallel. The Population Space can be a single culture or divided into subcultures.

Cultures are a type of complex system. A complex system is defined as "a system comprised of a (usually large) number of (usually strongly) interacting entities, processes, or agents, the understanding of which requires the development, or the use of, new scientific tools, nonlinear models, out-of equilibrium descriptions and computer simulations." [Advances in Complex Systems, 1998]. A complex system can also be defined as a system made up of a group of heterogeneous agents who interact and adapt with their environment and each other [Reynolds, Whallon, and Goodhall, 2006]. The separate behaviors of the heterogeneous agents that are aggregated together can cause higher-level behaviors to emerge from the group as a whole which work to solve problems facing the group at the group level. As a summary of complex system definitions, the complex systems are composed of different layers and in each layer there are different problems to be solved. One solution to one problem needs to be

integrated with the solutions to other problems. A complex system can be viewed as a collection of layers where each layer is associated with problems to be solved. Langton [1992] developed a model of complex systems that focus on these layers in terms of generators. Each layer is represented as a generator. As you go from a single generator, which is really one layer of a cultural system, to a system with many generators then the problem of controlling the system becomes more difficult. So, Ali and Che [Ali, 2008; Che, 2009] tested a homogeneous configuration, which have problems with high complexity. Then, we added heterogeneity to allow more flexibility and it improved performance. But still when the problem class gets closer to chaos, the system becomes less predictable and less focused. As a result, we got the idea of subcultures. Subcultures are defined in Oxford English Dictionary [Oxford Dictionary, 2005] as "a cultural group within a larger culture, often having beliefs or interests at variance with those of the larger culture".

We felt that the presence of subcultures helps to focus the problem solving process and effectively reduces the pressure on the overall system. We tested the subcultures preliminarily in a previous work [Gawasmeh and Reynolds, 2014] and we are planning to investigate the subcultures deeply in this thesis. What we are going to identify is the metrics that describe the structure of the population within the subcultures that might evolve. We will use these metrics as the mechanisms for driving the evolution of the subcultures in a system that before didn't have it.

One of the metrics is called social entropy. When the network stretches out, the social entropy value increases. Less stress gives thick connections and the flow of the information is more predictable. As the network stretches out then the connections are thinner and the movement of the information in the system becomes less predictable. We can view this stress as

an index of social entropy. When the social entropy becomes too high something should happen to reduce its value.

The social entropy characterizes the predictability of the system. In Cultural Algorithms' case, the predictability could relate to the flow of the information of the system. As the connections become more stressed they get thinner, which means that the connection is used less often and the movement of information in the system is becoming less predictable. When the connections become less stressed some vehicle should be used to pull the system back together. If the system is so complicated with many layers the whole thing cannot be pulled together. But it is possible to pull pieces together. In other words, every two or more generators could relate to each other and be pulled together to be combined in one subculture.

Three main contributions will be highlighted in this work. First, a new Heterogeneous Model will be defined using a group of homogeneous topologies that operate simultaneously in the Population Space. An additional model will be described that allows subcultures to emerge and operate between the individuals in the Population Space. This model is named Subcultures Model. Finally, evolving subcultures inside a Heterogeneous Model is big question of interest. So, we are going to describe a new model that allows evolving subcultures in heterogeneous cultures.

The next section describes briefly the Cultural Algorithm and its components. Section 3 describes the homogeneous topologies and how they were used in Cultural Algorithm. Section 4 introduces the motivation of using the Heterogeneous Model and gives a short description of how this model will be structured. Section 5will show the motivation behind introducing the Subcultures Model and give a brief description of its structure. A description of how to evolve

subcultures in a Population Space that started with the Heterogeneous Model is given in section 6. Finally, a conclusion will be given.

1.2 Cultural Algorithms.

The Cultural Algorithm is an evolutionary computation model derived from conceptual models of the Cultural Evolutionary process [Reynolds, 1978; Reynolds, 1994; Reynolds, 1999]. There are three main parts of this model: the Population Space, the Belief Space, and the interaction mechanism or protocol between them. The Population Space is a set of individuals that can support different computational models. The Belief Space is storage of the Population's Space experiences into knowledge source repositories that will reflect the changes in the individuals' behaviors. The Belief Space is divided into five main repositories of knowledge, or knowledge sources: Situational, Normative, Domain, History, and finally Topographic knowledge. The situational knowledge keeps track of the individual who did the best behavior in the population. The normative knowledge provides standards for individual behaviors to ensure that individual still within the scope of the system. The domain knowledge contains the sequence behaviors of the best individuals in the population. The history knowledge is useful in dynamic environments to store all changes in the problem landscape, such as the direction of the population behaviors either towards convergence or divergence. The topographical knowledge maintains a multi-dimensional performance grid representation of the Population Space to direct exploration into promising areas and to exploit currently productive areas. The interaction media between the Population Space and the Belief Space is called Communication Protocol. The Communication protocol transfers the individuals' experiences to the knowledge sources of the Belief Space. Then, it allows the knowledge sources to compete or work together to direct the individuals towards solving the targeted evolutionary problem. The individuals of the Population

Space were not directly interacting with each other in the early Cultural Algorithm models. But, Ali [2008] and Che [2009] added explicit homogeneous networks into Cultural Algorithms so as to allow the individuals to explicitly interact with each other.

1.3 Homogeneous Topologies.

In the early versions of Cultural Algorithms, the individual interactions were not explicit and networks didn't exist at all in the Population Space. Then, homogeneous networks used in the Population Space [Che, Ali and Reynolds, 2010; Reynolds, Che and Ali, 2010; Reynolds and Ali, 2008; Che, 2009] to transfer the knowledge between the individuals of the Population Space. Different homogeneous networks are used such as ring, square, octagon, hexagon, hexa-decagon, and global topologies [Che, Ali and Reynolds, 2010; Reynolds, Che and Ali, 2010; Reynolds and Ali, 2008]. Only one of the homogeneous topologies operates in the Population Space during the whole evolution process. The social fabric term was added into Cultural Algorithms by Ali. He used the homogeneous topologies as a vehicle of the social fabric.

Ali [2008] added the social fabric into the Cultural Algorithm framework to allow interaction among individuals, to give the individual the ability to choose which knowledge source to follow, and to investigate the influence of knowledge sources on the individuals through the social network. Ali and Che used a fixed homogeneous topology to connect the individuals with their neighbors. Ali proved that just having a social fabric to distribute the influence in the Population Space was sufficient to improve performance of the Influence function in the Cultural Algorithms. Ali tested three topologies: lbest, gbest, and square. Che added and tested three more homogeneous topologies: hexagon, octagon, and hexa-decagon. Allowing different homogeneous topologies to operate simultaneously among the Population

Space's individuals, the network becomes heterogeneous. The heterogeneous topologies are described in the next section.

1.4 Heterogeneous Topologies.

One key question of interest to those studying the emergence of social systems is the extent to which their organizational structure reflects the structures of the problems that are presented to them. Specifically, does the structure of a social or organizational network facilitate the exchange of information needed by a group to solve a problem?

In a recent study Che [2009] used Cultural Algorithms as a framework in which to empirically address this and related questions. There, a problem generator based upon Langton's model of complexity was used to produce multi-dimensional real-valued problem landscapes of varying complexities. Various homogeneous social networks were then tested against the range of problems to see whether certain homogeneous networks were better at distributing problem solving knowledge from the Belief Space to individuals in the Population Space. Langton categorized the problem complexities into fixed, periodic and chaotic classes [Langton, 1992].

The experiments suggested that different network structures worked better in the distribution of knowledge for some complexity class problems (fixed) than for others (periodic and chaotic). If this is the case, then in a situation where several different problems are presented to a group, they may wish to utilize more than one network to solve them. The heterogeneous topologies will be structured using six different homogeneous topologies used and examined in previous work [Reynolds and Gawasmeh, 2012]. Allowing different homogeneous topologies to operate together at the same time in Population Space will make the network heterogeneous. Therefore, the heterogeneous topologies name was given to this type of networks. In this thesis, the advantages of utilizing a heterogeneous network over a suite of different problem

complexities will be investigated in detail in chapters 4 and 7. It will be shown that heterogeneous approaches begin to dominate homogeneous ones as the problem complexity increases towards that of chaos. After testing the heterogeneous topologies, we could notice that when the complexity of the system increases, the individuals' connections become tenser and they start operating on scattered areas on the landscape. Therefore, we suggested a new model that divides the Population Space into subcultures. The model is described in the next section.

1.5 Subcultures Model.

Cultural Algorithms are computational models of social evolution based upon principles of Cultural Evolution. A Cultural Algorithm consists of a Belief Space consisting of a network of active and passive knowledge sources and a Population Space of agents. The agents are connected via a social fabric that is used to pass the information between these agents. The knowledge sources in the Belief Space compete with each other in order to influence the decision making of agents in the Population Space. Likewise, the problem solving experiences of agents in the Population Space are sent back to the Belief Space and used to update the knowledge sources there. It is a dual inheritance system in which both the Population and Belief spaces evolve in parallel.

The Population Space and the Belief Space are the major components of the Cultural Algorithm. Mainly, the Belief Space consists of five knowledge sources: Situational, Normative, Historical, Domain, and Topographical that will be explained in the next section of this paper. Over the years, much research has been devoted to Cultural Algorithm knowledge sources and social fabricated Population Space. However, more attention has been given to each of them as separate units and less attention has been given to the interactions among individuals as a beneficial experience. Actually, Belief Space hadn't been aware of the individuals' interactions.

Five knowledge sources listed in the Belief Space are not related to the operating topologies among the individuals and these knowledge sources actually were defined before social fabric even existed.

As previously discussed, there is a lack of knowledge in the Belief Space about the topologies among the landscape individuals and neither the homogeneous nor the non-subcultures topologies were solving the periodic problems efficiently. To overcome these problems, we introduce a new knowledge repository into the Belief Space to enable the collecting and distributing of the topologies' behaviors among the Population Space. In this work, we are adding a new knowledge repository (fabric knowledge) into the Belief Space to collect not only individuals' behaviors but also the heterogeneous topologies' behaviors. Fabric topology structure will be given and described in detail later in chapter 5. By collecting the individuals' experiences in topologies and knowledge sources, we expect to see subcultures start to evolve in the Population Space.

In this work we investigate why subcultures can emerge in the Population Space in response to the complexity of the problems presented to a Cultural System. This system is compared with other evolutionary approaches relative to a variety of benchmark problems of varying complexity. We show that the presence of subcultures can provide computational advantages in the problem landscape that are generated by multiple independent processes. These advantages can increase problem solving efficiency along with the ability to dampen the impact of increase in problem complexity. This work will also present results of an intensive testing on Cones World Problem using Cultural Algorithms.

1.6 Organization of the Dissertation.

The social fabric term was introduced into the Cultural Algorithm by Reynolds and Ali [2008] to build connections among the individuals in the Population Space. These connections formed networks that allow the knowledge to be transferred from an individual (source) to a connected individual (destination). The social fabricated Population Space can be constructed from homogeneous networks, heterogeneous networks, or both. But the real life networks are not only homogeneous and heterogeneous but also can be divided into subcultures. Those subcultures have common attributes, goals, or dependability.

Previously, Ali [2008] and Che [2009] used static and homogeneous social fabrics. Both Ali and Che [Reynolds et al., 2010] showed that some of the homogeneous topologies operated better with specific complexity class problems. In this work [Reynolds and Gawasmeh, 2012], the performance of Cultural Algorithms had been examined with a heterogeneous and Subcultures social fabric Population Space over the complete range of Langton's' optimization problem complexities [Langton, 1992]. We concluded that heterogeneous topologies work quite well at solving chaotic class problems, but performance decreases with lower complexity problems. In contrast, homogeneous topologies work quite well at solving fixed class problems, but performance decreases as the complexity of the problem increases.

The Cultural Algorithms framework is given in chapter 2. Chapter 3 describes the social fabric by giving a history of Cultural Algorithm. Two social fabric models, Homogeneous and Heterogeneous Models, are summarized in chapter 4. Then, Subcultures Model is motivated and described in chapter 5. Chapter 6 gives a complete description of the experimental environment in order to test the Heterogeneous and Subcultures Models. The analyzed results of testing Heterogeneous and Heterogeneous Models are summarized in chapters. Chapter 8 compares the

Homogeneous, Heterogeneous, and Subcultures Models in depth. Finally, the conclusions of this work and future directions will be given in chapter 10.

CHAPTER 2 CULTURAL ALGORITHM FRAMEWORK

2.1 Introduction.

The Cultural Algorithm (CA) is an evolutionary computational model derived from conceptual models of the Cultural Evolutionary process [Reynolds, 1978; Reynolds, 1994]. The Population Space, Belief Space, and Communication Protocol between the Population and the Belief Space are the three major components of this model, as shown in Figure 2-1. The Cultural Algorithms have been used to solve problems in natural languages [Stefan et. al, 2003], multi agent systems [Reynolds, Kobti, and Kohler, 2003; Reynolds, Kobti, and Kohler, 2004; Reynolds, 2003; Reynolds and Chung, 1996; Reynolds et. al, 2006; Lazar and Reynolds, 2012], and game programming [Loiacono et. al, 2008; Reynolds, Ali, and Alomari, 2006; Reynolds et. al, 2011; Vitale et. al, 2011], among others.

Cultural Algorithm is described in section 2. Section 3 introduces the main components of the Cultural Algorithm, knowledge sources of the Belief Space will be described in terms of structure, how the Belief Space will be updated, and how the knowledge sources of the Belief Space will be used to influence the Population Space in section 4.The Communication Protocol of the Cultural Algorithm components, including its three main functions: Accept (), Update (), and Influence () functions, will be described in section 5. Section 6 gives a conclusion of Cultural Algorithm framework.

2.2 Cultural Algorithm.

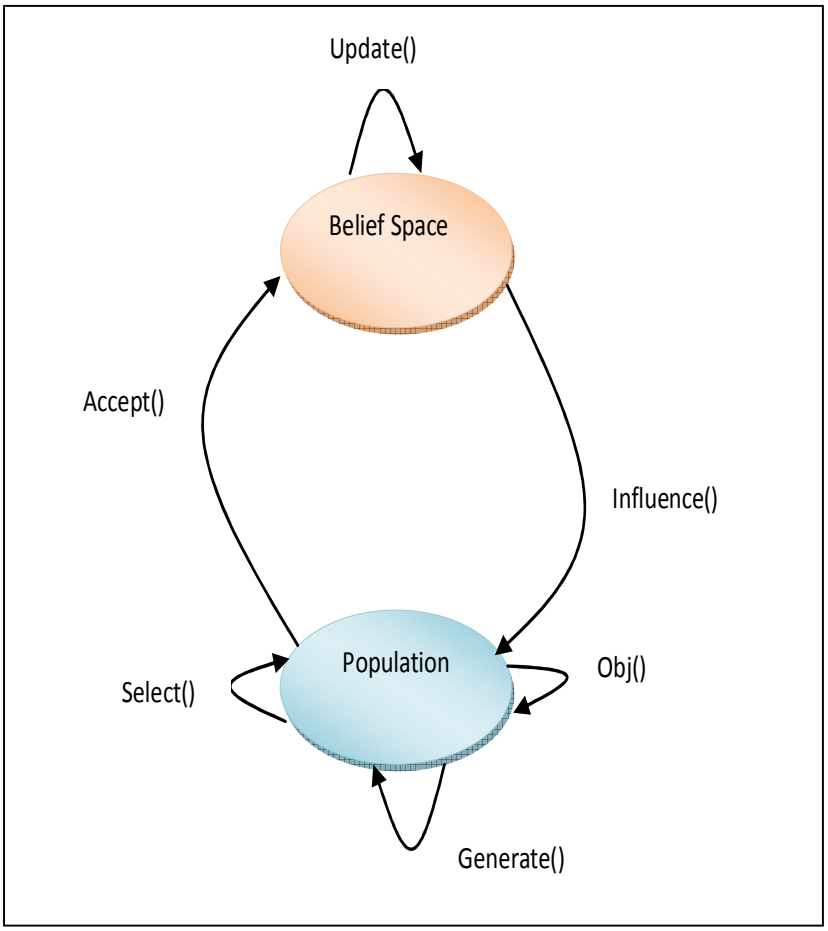


Figure 2-1: Cultural Algorithms Framework.

In the earliest Cultural Algorithm, only one knowledge source (Situational Knowledge) was used [Reynolds, 1978]. Next, four more knowledge sources were introduced into the Belief Space: Normative, Domain, History, and finally the Topographic Knowledge. The Situational Knowledge keeps track of the individual who performed the best behavior in the population. The Normative Knowledge provides standards for individual behaviors to ensure that the individual is still within the scope of the system. The Domain Knowledge contains the sequence behaviors of the best individuals in the population. The History Knowledge is useful in dynamic environment

to store all changes in the problem landscape. Topographical Knowledge gives a local view of a group of individuals within a specific region in terms of the value for the performance landscape.

The basic program of the Cultural Algorithm is shown in Figure2-2, where P (t) and B (t) represents the Population Space and the Belief Space at time (t), respectively. Obj () and Accept () functions need to be executed at the end of each iteration. Obj () is the performance function, which is used to evaluate the performance of each individual in the Population Space. Then, a group of individuals will be chosen to record their experiences to update the knowledge sources in the Belief Space using the Acceptance Function, Accept(). The Acceptance Function selects the best elements to update, but in some cases the Acceptance Function selects from different categories to explore all results of the population. Updating the Belief Space occurs via the Update () function. In the Belief Space, there are many kinds of knowledge sources. Some sources are updated by the Update Function directly and some of them indirectly by the interaction with other updated sources. Next, the Influence () function transmits the updated knowledge from the knowledge sources in the Belief Space to the individuals that reside in the Population Space to enhance their performance. Transferring the experiences between the knowledge sources of the Belief Space and the individuals of the Population Space from one side and between the knowledge sources of the Belief Space from the other side is called the Communication Protocol.

The Cultural Algorithms repeatedly produce a new generation and update the Belief Space until the termination condition is satisfied. The Cultural Algorithm is flexible since it can look for small scale and large scale solutions and run in static and dynamic environments [Ali, 2008]. The termination condition may depend on the population convergence or divergence. It may depend on constant or variable numbers of iterations.

```
Begin
        t=0;
        Initiate B(t) and P(t)
        Repeat
                Evaluate P(t) using Obj()
                Update B(t) using Accept()
                Generate P(t) using influence()
                t=t+1
                Select P(t) from P(t-1)
        Until (termination condition achieved)
End
```

Figure 2-2: Outlined Program of Cultural Algorithm.

Belief Space knowledge sources, Communication Protocol, and Population Space are the main components of the Cultural Algorithm. The Belief Space stores the extracted experience of individuals selected from the Population Space in five knowledge sources. The Belief Space's knowledge sources interact with one another, form a problem solving knowledge and influence the evolution of the population component. The Belief and the Population Spaces interact with each other via the Communication Protocol. The Communication Protocol controls the transfer of knowledge from the Population Space to the Belief Space via the Acceptance Function and directs changes in the population by the knowledge sources via the Influence Function.

2.3 The Cultural Algorithm Knowledge Sources.

The represented knowledge in the Belief Space varies depending on the nature of the problem and the expected results of the problem. The Belief Space consists of five major components: Situational Knowledge, Normative Knowledge, Domain Knowledge, History Knowledge, and Topographical Knowledge. The knowledge sources may interact with each other to ensure their compatibility, so some knowledge sources transfer the newly collected

knowledge to other knowledge sources. As seen in Figure 2-3, updating the Domain and History knowledge sources uses the newly updated Situational Knowledge source. The Influence Function can transmit the knowledge sources in two ways. First, treat the population as a complete social network and combine influences of all knowledge sources together as one influence direction. Second, separate the population into subsets of social networks and use each knowledge source to influence one of these social networks. In the next section, we are going to explain each of the knowledge sources in detail.

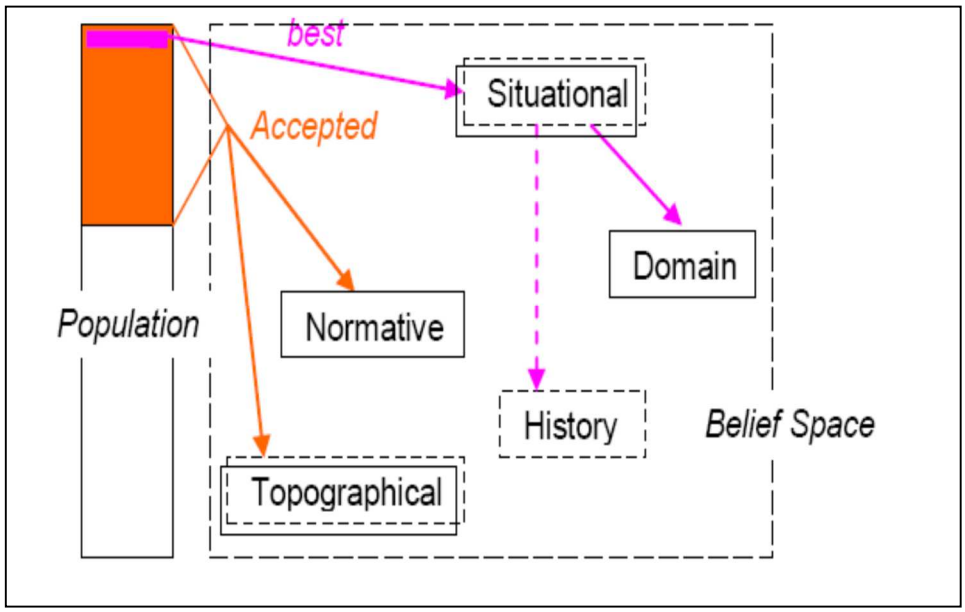


Figure 2-3: Belief Space Update Flow.

2.3.1 Situational Knowledge.

The earliest knowledge source used in applying the Cultural Algorithms was the Situational Knowledge [Reynolds, 1994; Reynolds, 1979]. It stores a set of exemplary cases that reflect the agent experiences and behaviors from the population; it usually stores the distribution of best individuals. The Situational Knowledge is also flexible since it can store exemplars or memories of successful and unsuccessful solutions. The Situational Knowledge structure is often represented as a fixed-size list of exemplars taken from the original population, as given in

Figure 2-4. The Situational Knowledge will be updated by adding new individuals' experiences that are selected by the Acceptance Function.

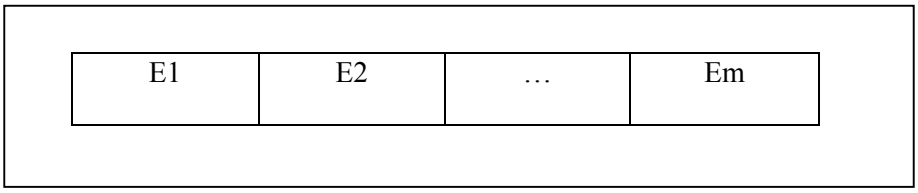


Figure 2-4: The Situational Knowledge Structure.

After each generation, the knowledge sources choose a set of individuals to influence during the next generation. When the Situational Knowledge source is selected to influence an individual, one of the elites from the knowledge base will be chosen as an example to follow. The selection of an exemplar can occur in different ways, including random selection or roulette wheel techniques.

2.3.2 Domain Knowledge.

Domain knowledge represents basic relationships between domain parameters. For example, is the domain of discourse was a configuration of cones, equations that specified relationships between cone parameters such as height, base size, and slope. If the domain was that of physics the basic relationships between the parameters could be Newton's Laws. The main benefit of this knowledge is the reasoning about the direction and the magnitude problem parameters. Figure 2-5 shows the structure of the domain knowledge.

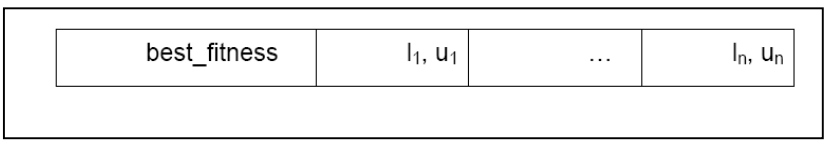


Figure 2-5: The Domain Knowledge Structure.

2.3.3 Normative Knowledge.

Chung and Reynolds added Normative Knowledge into the Cultural Algorithm [1998] in order to manage the expected performance for individual behavior. A later study [Clayton, Griffiths and Dickinson, 2001] showed the importance of norms and standards to the human social intelligence. Normative Knowledge is a set of expected ranges for the variables to provide standards for the individuals' behaviors. The Normative Knowledge is explored as rules of allowed behaviors for individuals. Normative Knowledge helps the individuals to move into good levels of behaviors especially if they are not currently in the acceptable range of experience. This change occurs by updating the agents' behaviors in order to be compatible with the standards.

The Normative Knowledge can be represented as a vector of size d (the number of dimensions) to save the ranges of the individuals' behavior and performances for each dimension. Figure 2-6 shows a representation of the Normative Knowledge where: (l_i) represents the lower bound of an individuals' behavior at dimension i, (u_i) represents the upper bound of an individuals' behavior at dimension i, (L_i) represents the lower bound of the individuals' performance that has l_i, and (U_i) represents the upper bound of the individuals' performance that has u_i. When some changes occur in the environment, Normative Knowledge will be updated by shifting the ranges (l_i and u_i) and updating the related performance values (L_i and U_i). If the offspring has gained an unexpected behavior, the Normative Knowledge would mutate the offspring randomly to adhere to the norms of the individual's behavior.

V1	V2	…	Vd
(l_1,u_1,L_1,U_1)	(l_2,u_2,L_2,U_2)		(l_d,u_d,L_d,U_d)

Figure 2-6: The Normative Knowledge Structure.

2.3.4 History Knowledge.

In order to allow backtracking and reasoning about global changes and dynamics in the environment, History Knowledge was introduced by Reynolds and Saleem [2001]. The data structure for History Knowledge is shown in Figure 2-7.

History Knowledge monitors the whole process and records the important environmental changes, individual behaviors, and events; it contains information about the direction of optimal behaviors. It provides a global view of the changes in the environment, computes the average change in parameter values within a specific region, and predicts the direction of either increasing or decreasing the parameter value. Thus, the individuals are guided by the History Knowledge to know their selection for the next movement.

The History Knowledge will be updated after every event change and the history list will also be updated to reflect changes in individuals' behavior or environment. The easiest kind of update is to find the new best value and then add it to the list. If the list size is exceeded, the oldest value will be dropped. In other words, if the best individual is different from the previous generation, its behavior will be added to the set. If the best individual didn't change for k steps, the system will be informed to backtrack and retrace the situation.

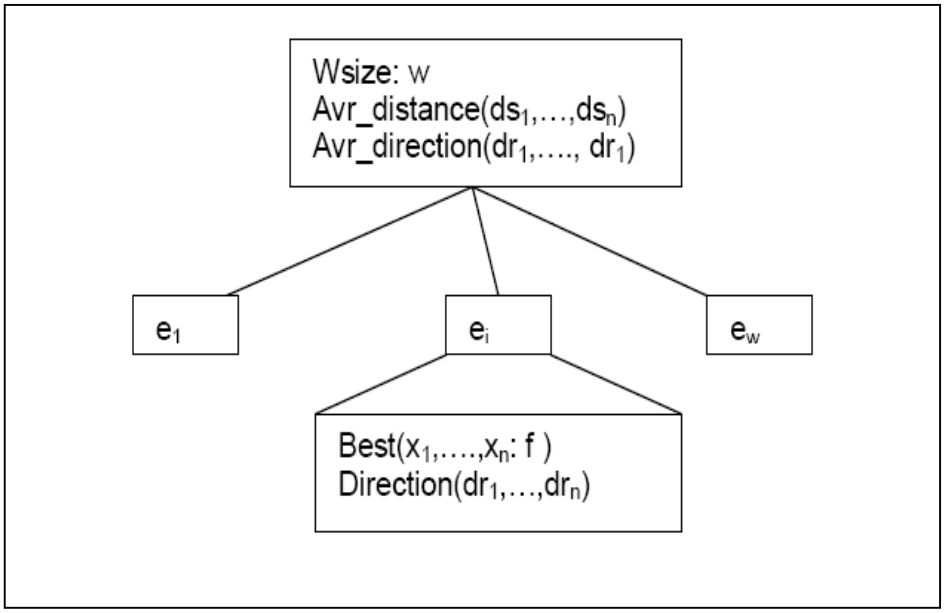


Figure 2-7: The History Knowledge Data Structure.

2.3.5 Topographical Knowledge.

Topographical Knowledge was introduced into the Cultural Algorithm by Jin and Reynolds [1999, 2000] who originally called it "regional schema". Usually, the individuals are distributed as groups within regions. Topographical Knowledge stores the optimal behaviors in these regions, and then the individuals can enhance their behavior by using this type of knowledge. Topographical Knowledge gives a local view for a group of individuals within a specific region, where they use the same kind of knowledge source or have common characteristics.

The Topological Knowledge is best described as a list of tress as seen in Figure 2-8, each tree represents a cell (Area) from the landscape. The list element starts with one node that saves the best behavior in that cell (best individual). If a better behavior showed in the cell (region), the cell will be divided into two cells and store the best behaviors in them. Topographical knowledge influences the Population Space by moving the individuals into the promising regions. If the individual was in promising cell, she would move a short distance from her location in that cell. Otherwise, the individual would move to a new promising cell.

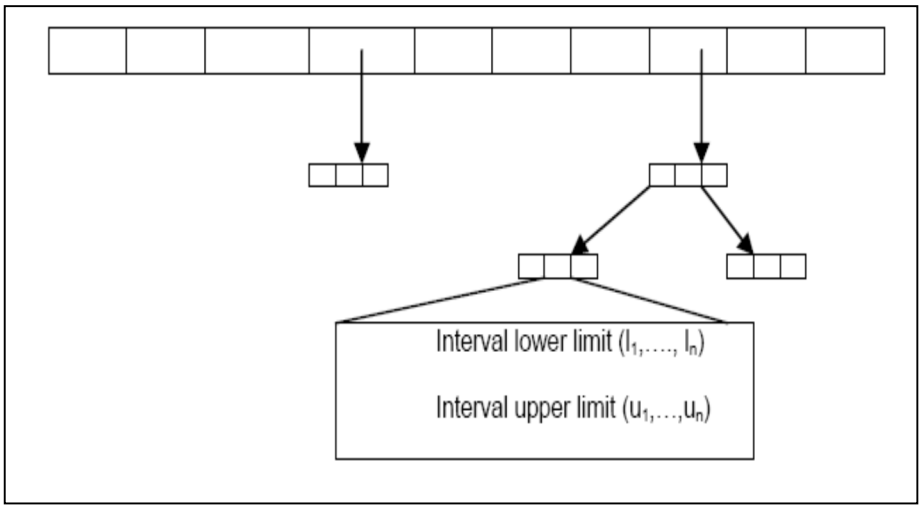


Figure 2-8: The Topographical Knowledge Structure.

2.4 The Communication Protocol.

The interaction between the Belief Space and the Population Space happens through the Communication Protocol. The Communication Protocol is divided into three main communication functions: the Acceptance Function, the Update Function, and the Influence Function. The Acceptance Function selects a subset of individuals from the Population Space to update the knowledge resources in the Belief Space. The Update Function determines how the knowledge resources in the Belief Space will be updated. After accepting some individuals using the Acceptance Function, the Belief Space will be enhanced to reflect the state of the Population

Space. The Influence Function then transfers the stored experiences in the Belief Space to the individuals that reside in the Population Space. Update and Influence Functions use the knowledge sources after each generation either to update or to use them.

2.4.1 The Acceptance Function.

The Acceptance Function selects a subset of the Population Space's individuals to update the knowledge sources in the Belief Space. In most cases, the Acceptance Function depends on the fitness value of each individual, which determines the best and the worst individuals in the Population Space. The Acceptance Function can select the best individuals (Top 10%), the worst individuals (Bottom 10%), or a combination of both. Chung and Reynolds [1998] introduced both dynamic and static Acceptance Functions. One possible dynamic function is as follows: over time, the number of accepted individuals decreases gradually till it tends to be constant as shown below:

$$accept = top\left[p\% + \left\lfloor\left(\frac{p\%}{k}\right)\right\rfloor\right] \qquad (2.1)$$

Where p represents the percentage of the Population Space that will be accepted through the Acceptance Functions; k is the number of time steps (generations) in the current environment. When k increases, the number of accepted individuals decreases. Figure 2-9 below shows how the number of accepted individuals changes with p=40% and Population Space size = 1000,000.

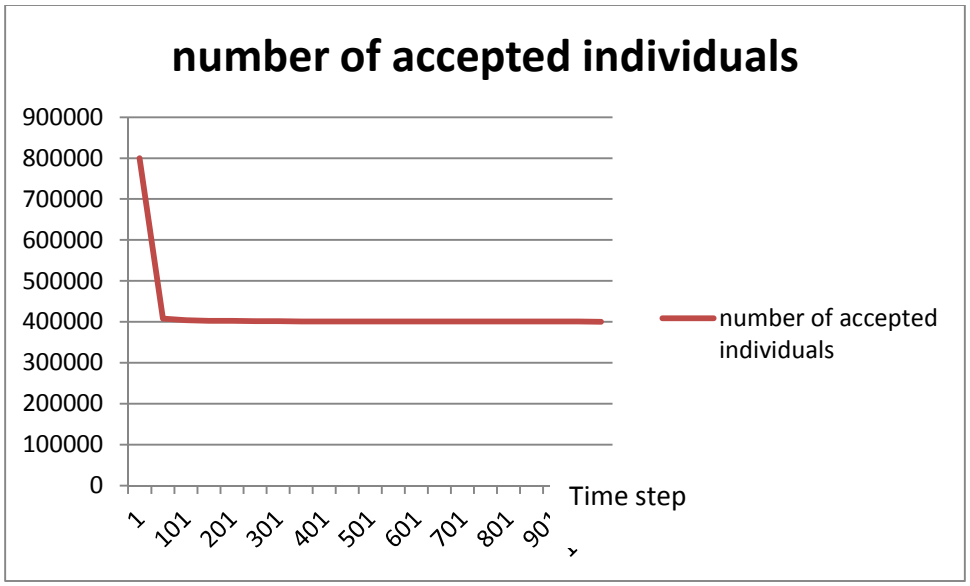


Figure 2-9: Dynamic Acceptance Function.

2.4.2 The Update Function.

As already discussed the Communication Protocol updates and integrates the five knowledge sources described above. Each knowledge source in the Belief Space has a different data structure, stores a different type of knowledge, and extracts trends from different groups of individuals. Thus, each knowledge source will be updated in a different way as reflected in Figure 2-3 that summarizes the knowledge update process in the Belief Space. Each knowledge source bases its updates on the new experiences from the current generation of offspring and its own accumulation of experiences from the old generations. However, some of the knowledge sources can be impacted by some updates in other knowledge sources. For example, History and Domain knowledge sources use Situational Knowledge as part of the update process, which is labeled with orange in Figure 2-3. Every knowledge source uses the accepted individuals from the current generation in a different way. Normative and Topographical knowledge sources are updated using the experiences from all accepted individuals, while others are updated using the

best performer within the accepted group of individuals, which is labeled with magenta in Figure 2-3.

2.4.3 The Influence Function.

The Influence Function transfers knowledge from the Belief Space to the Population Space. Thus, the five knowledge sources must be integrated together over the Population Space. The first Influence Function was introduced by Reynolds and Saleem [2001]. Reynolds and Saleem used the random integration function, in which the individual will randomly choose the knowledge source to follow. Building on that innovation, Reynolds and Peng [2005] developed the Marginal Value Theorem (MVT) to involve individuals in making decisions. In this work, the individual will transfer from area to area to follow different knowledge sources. Ali [2008] extended Peng's work with the Social Fabric by allowing individuals to communicate their potential knowledge source with neighbors' knowledge sources via a homogeneous topology. That way, the individual could decide which of the knowledge sources to follow. Che [2009] extended Ali's work by adding more homogeneous topologies to the social fabric and employing **weighted majority winning (win) schema (conflict resolution scheme)**, which is used to determine the followed knowledge source in current time step. More details will be provided in the next chapter about social fabric, homogeneous topologies, and **conflict resolution strategies**.

2.5 Conclusions.

Cultural Algorithms consist of three main components: Population Space, Belief Space, and Communication Protocol. Individuals represent the Population Space, the Belief Space stores the experiences of the Population Space, and the Communication Protocol controls the interaction between the Population Space and the Belief Space and also controls the interaction between the knowledge sources in the Belief Space. The Belief Space stores five different kinds

of knowledge sources: Situational, Domain, Normative, History, and Topographical. The population behaviors will be reflected and stored in the Belief Space. The Acceptance Function transfers the population experiences to the Belief Space. The Belief Space distributes the knowledge to the Population Space using the Influence Function. The Acceptance and Influence Functions form the Communication Protocol.

In the next chapter, the Influence Function and its social fabric evolution will be described in detail. Ali introduced three homogeneous topologies, and Che introduced three more – making six possible ways of interacting among the individuals in the Population Space [Reynolds, Che, and Ali, 2010]. The description of these topologies will be explored in the next chapter.

CHAPTER 3 SOCIAL FABRIC EVOLUTION

3.1 Introduction.

In this chapter we discuss the history of the social fabric Influence Function in Cultural Algorithms and culminate with its current implementation. The Influence Function was originally developed as a vehicle to integrate the interaction of the various knowledge sources on the population of decision making agents. In the earliest Cultural Algorithms only a single knowledge source was used at a time. However, as the problems to which Cultural Algorithms were applied grew more complex, there was more opportunity to use multiple knowledge sources, where each represented different aspects of the problem.

The integration of all knowledge sources was changed by adding new contributions to Cultural Algorithm. Three major contributions will be shown in this chapter. Section 2 shows the random selection process. Random selection process was the first process that had been used to integrate the knowledge sources. Marginal Value Theorem is another integration process and will be described in section 3. Two more integration methods, Majority Winning and Vector Voting, were added into Cultural Algorithm to handle the addition of the social fabric network. Section 4 gives a description of the social fabric and the two integration methods. A conclusion is given as a last section.

3.2 Random Selection.

Saleem was the first to explicitly integrate the different knowledge sources. He [Reynolds and Saleem, 2001] started integration of all knowledge sources into the Cultural Algorithm framework using a random selection process. At each time step. One of the available five knowledge sources was selected randomly with equal probability using a uniform distribution function. He developed the CADE, the Cultural Algorithm for Dynamic Environment, in order to track the changes in the dynamic environment and store them in the Belief Space. He added the

History and the Domain knowledge sources to the Belief Space. History Knowledge allowed reasoning about time, while the Domain Knowledge allowed reasoning about the problem domain in terms of the relationships between problem parameters. He identified the required structure for each knowledge source in the Belief Space to find and track the changes in the dynamic environment.

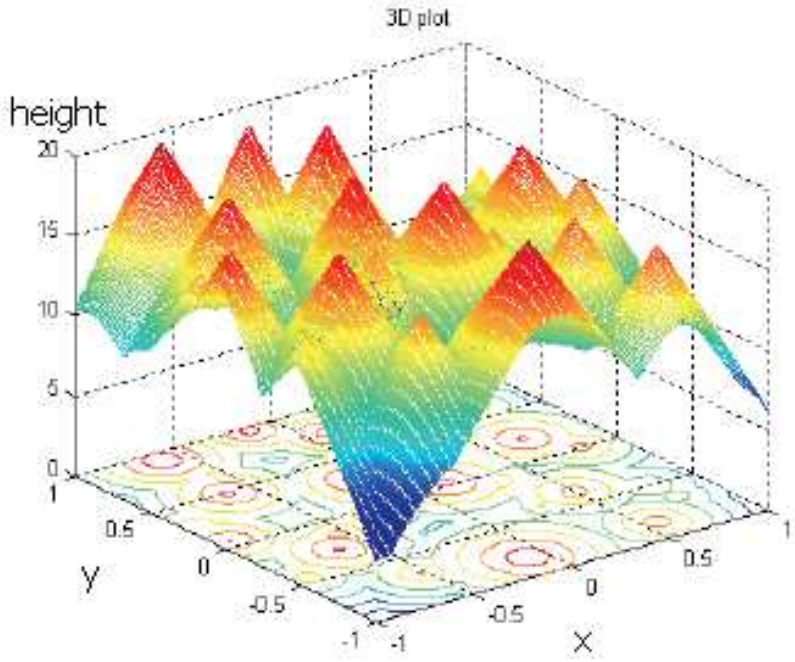


Figure 3-1: Example of Cones World Environment.

Saleem's approach was applied to the solution of problems in the Cones World environment, where the problem was to find the highest peak in a multi-dimensional landscape, where the peaks are moving over time. The Cones World environment was generated via a benchmark problem generator developed by [Morrison and De Jong, 1999] and called DF1 in their work. Figure 3-1 shows a 3D example of the Cones World environment. The random selection of the knowledge sources was the foundation of Saleem's integration function.

3.3 Marginal Value Theorem.

Next, Peng [2005] used the Marginal Value Theorem (MVT) [Charnov, 1976] to integrate the knowledge sources of the Belief Space in order to guide the problem solving process. It had been shown in foraging theory that the Marginal Value Theorem can be used to optimize the energy intake of predator/foragers within an environment. The main idea (the habitat's resources would be decreased over time if the foragers resided too long within the habitat's borders) had been taken from ecology. Thus, the forager stays within the habitat until the resources become less than the average expected value.

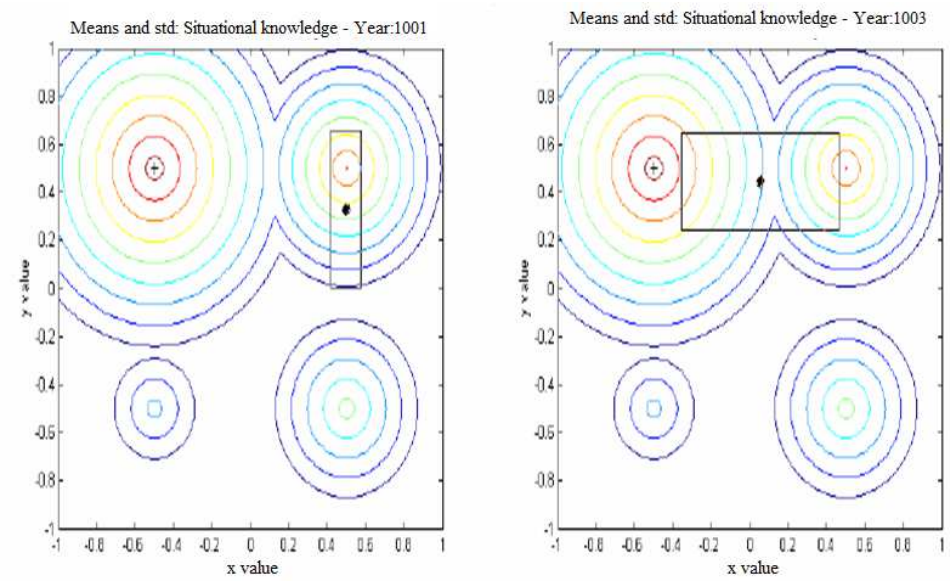


Figure 3-2: Situational Knowledge at Year 1000 and 1003.

Peng observed at each time step that the individuals generated by each knowledge source using a normal distribution could be described by a "boundary box" or a patch with a given central tendency and standard deviation. The bounding box described the area under which roughly 2/3 or 66% of the individuals influenced by the knowledge source were placed. For example, in Figure 3-2 notice the shifting of the patch for Situational Knowledge from one area on the landscape to another. In fact, the original patch orientation is rotated and then translated towards the optimal point "+" over time by the knowledge source update process.

Each of the knowledge sources was viewed by Peng as a predator. Every knowledge source (predator) controls the placement of individuals on the landscape to exploit the available resource (prey). If the average performance of those individuals controlled by knowledge source (predator) falls below the population average, the bounding box will be adjusted by moving the individuals to a different area to increase the performance. She developed the Influence Function based upon the Marginal Value Theorem, which was implemented as a basic roulette wheel. In Peng's wheel, each knowledge source is given a part that will be initialized by 20% of the wheel area and updated according to the average fitness value of the individuals who follow the knowledge source.

The roulette wheel approach was modified by Peng to emulate the action of the energy intake function. The size of a knowledge sources area under the wheel reflects its ability to exploit above population average gains. At each step, each of the individuals in the population is influenced by one of the knowledge sources based upon the spin of the wheel. Then, the individual moves into the bounding box of the selected knowledge source.

3.4 Social Fabric.

In Pengs' version, individuals acted independently and were influenced by a single knowledge source [Peng, 2005; Reynolds and Peng, 2005]. Ali [Ali, 2008; Reynolds and Ali, 2007; Reynolds and Ali, 2008] added the social fabric into the Cultural Algorithm framework to allow for the interaction among the individuals, give the individual the ability of making a decision about which knowledge source to follow, identify the minimum social structure needed to solve problems of certain complexities, and to investigate how the knowledge sources influence the individuals through a social network. In a prior work, individuals acted independently and were influenced by a single knowledge source [Reynolds and Peng, 2005]. Ali [2008] used a

fixed topology to connect the individuals with their neighbors. The topology stayed constant in shape through all the steps of the optimization process, while the individuals changed their positions in the topology each time step to assess the presence of the social fabric. Ali [2008] proved that just adding a social fabric into the Population Space improved the performance of the Influence Function in the Cultural Algorithm.

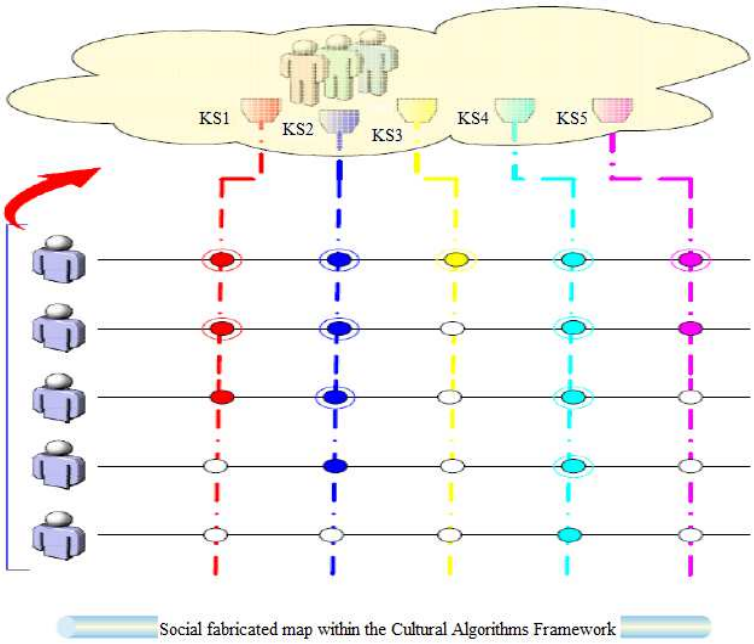


Figure 3-3: Social Fabric.

Ali viewed the social fabric as a weaving process. The concept is illustrated schematically in Figure 3-3. In the Figure, there are five different networks where each is given as a color-coded vertical line. The horizontal lines represent the Population Space's Individuals with a group of nodes, each representing a potential sharing in any of the networks. The node is blank, darkened, or a darkened-circled based upon whether the node does not participate (blank), participates sometimes (darkened), or participates frequently (darkened and circled). The interconnections among the individuals in the Population Space are woven as a social fabric.

Figure 3-4 shows how the social fabric component is embedded into the Cultural Algorithm framework. In the Figure there are 5 knowledge sources (KS1, KS2, KS3, KS4, and KS5) that correspond to the five knowledge sources in the Belief Space; Situational, Normative, Domain, History, and Topographical knowledge sources. Every knowledge source is color coded and its color is given to the individual that will be influenced at that time step.

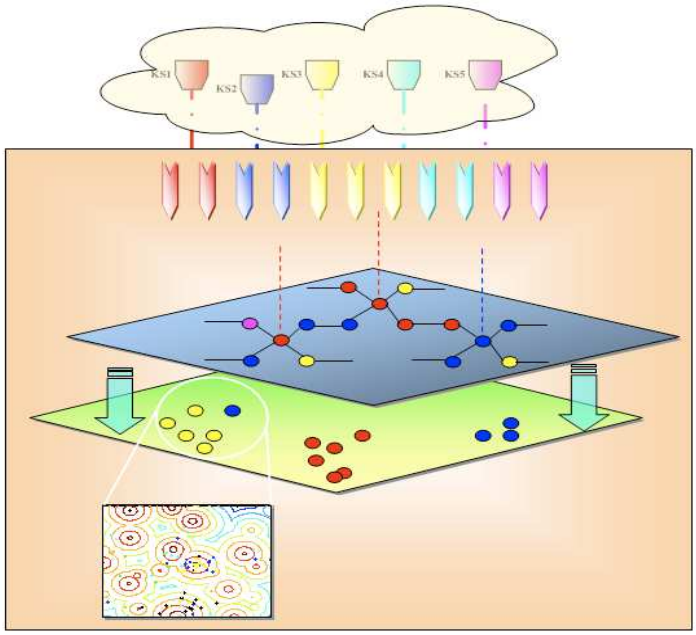


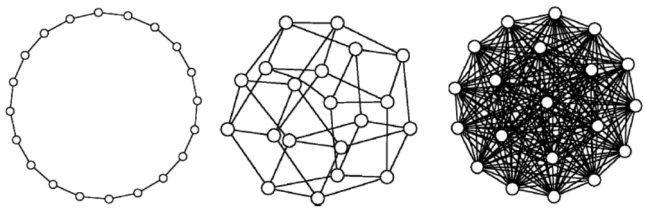
Figure 3-4: Social Fabricated Landscape.

In Figure 3-4, there are three perpendicular dotted lines that point to three individuals with a complete set of neighbors. Each individual has four neighbors to interact with, so this topology named the square topology. Ali used three other homogeneous topologies: Ring (lbest), Square, and Global (gbest) topologies [Ali, 2008]. Figure 3-5 (a), (b), and (c) shows the three topologies. The topologies differ from each other by the number of neighbors connected to each individual or what would be called the degree.

At each time step, every individual is influenced by one of the knowledge sources. Knowledge sources do not know anything about the network and the selected individuals'

position in it and vice versa. This is a double blind process. The individual then transmits the name of the influencing knowledge source to its neighbors through as many connections (hops) as specified. Next, each individual counts up the number of knowledge sources bids that it collects from its neighbors. It will have the direct influence from the knowledge source that selected it, plus the ID's of the knowledge sources transmitted to it by its immediate neighbors. The knowledge source that has been repeated the most would be the choice to follow and would direct the individual for that time step. This schema called the **Majority Win Schema** [Reynolds and Ali, 2008].

To clearly show how a topology will impact the optimization performance, Che [2009] added three more homogeneous topologies: hexagon, octagon, and hexa-decagon. In the hexagon, every individual (node) interacts with exactly six neighbors. Each individual in octagon communicates with eight neighbors. In the hexa-decagon, each individual communicates with sixteen neighbors.



(a) Ring Topology. (b) Square Topology. (c) Global Topology.

Figure 3-5: The Homogeneous Topologies Used in the Previous Models.

There are many ways to build the neighborhood topology [Langton, 1992; Jin, Girvan, and Newman, 2001; Pattison and Robins, 2002]. For example, Che gave each individual in the population an ID. The IDs range was from 1 to n, where n is the population size. Each individual has m neighbors and marked with ID k, so the neighbors IDs will be (n+k-m/2) mod n, (n+k-m/2

+1) mod n, …, k, (k+1) mod n, (k+2) mod n, …, (k+m/2) mod n. Octagon example is given in Figure 3-6.

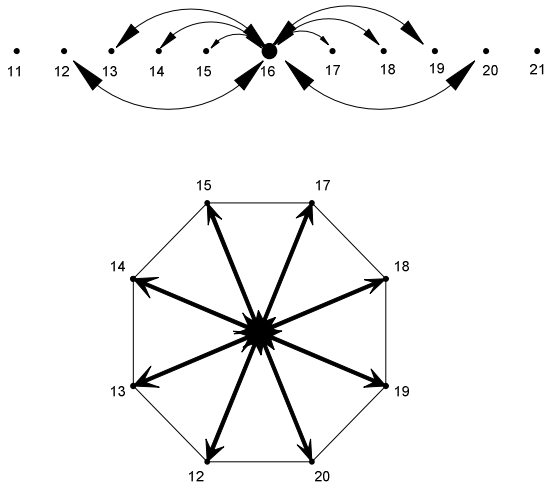


Figure 3-6: Octagon Neighborhood Topology Displayed in One Dimension and Two Dimensions.

Che [2009] employed a **Weighted Incentive Based Majority Win Scheme**. It is based on a Vector Voting model employed in the earliest version of Cultural Algorithms by Reynolds [1978]. An example of **weighted majority winning schema** with octagon topology is given in Figure 3-10. In the example, A_0, A_1… A7 and A_8 are the individuals. A_0 is the one deciding which knowledge source to follow. A_1… A_7 and A_8 are the neighbors of agent A_0. S, N, D, T, and H represent the Situational, Normative, Domain, Topographical, and History knowledge sources respectively. From Figure 3-7, Agent A has the following votes:

- Neighbors A_1 and A_6 votes for Situational Knowledge S.
- Neighbors A_2, A_4, and A_8 votes for Normative Knowledge N.
- Belief Space and neighbor A_7 votes for Domain Knowledge D.
- A_3 votes for Topographical Knowledge.
- A_5 votes for History Knowledge H.

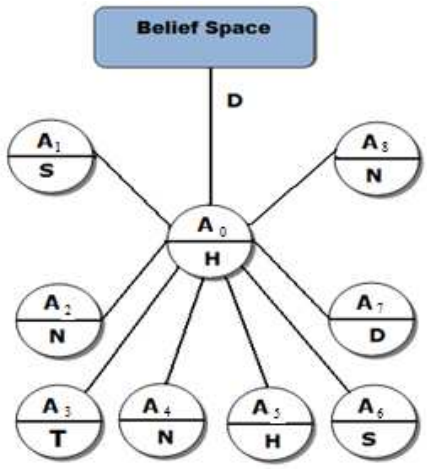


Figure 3-7: Knowledge Source Interaction at the Population Level.

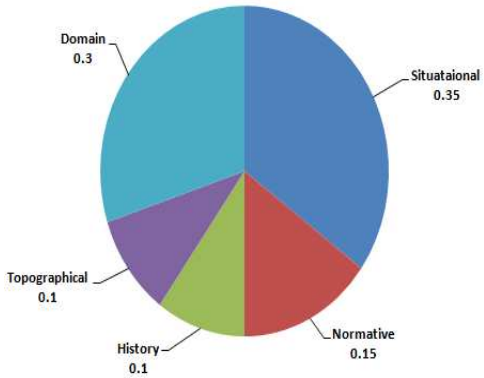


Figure 3-8: Current Average Fitness of Each Knowledge Source.

In Peng's Influence Function, A_0 will select the knowledge source directly selected by the Belief Space, so it would be the Domain Knowledge as shown in Figure 3-7. The Normative Knowledge has the most votes (3), so A_0 will follow it in Ali's Influence Function as shown in Figure 3-9. Figure 3-8 represents a roulette wheel and gives the normalized fitness of each knowledge source in the Population Space. Knowledge source fitness affects the individuals in making decision. In Che's Influence Function, A_0 would follow the Situational Knowledge even if the Normative Knowledge occurred more frequently as shown in Figure 3-10. The **weighted majority win** weights the knowledge sources in terms of the normalized fitness. If the

normalized fitness from the roulette wheel given in Figure 3-8 was used, Situational Knowledge will have the greatest weighted sum.

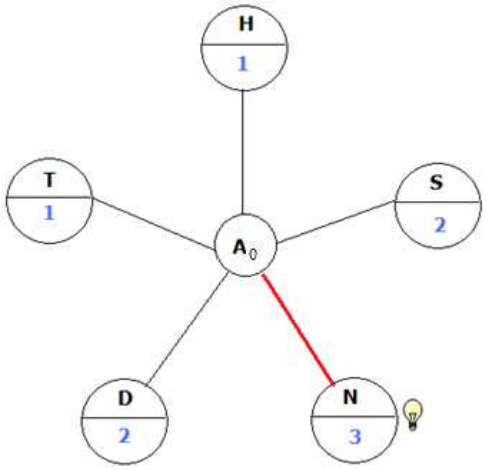


Figure 3-9: Majority Win in Belief Space.

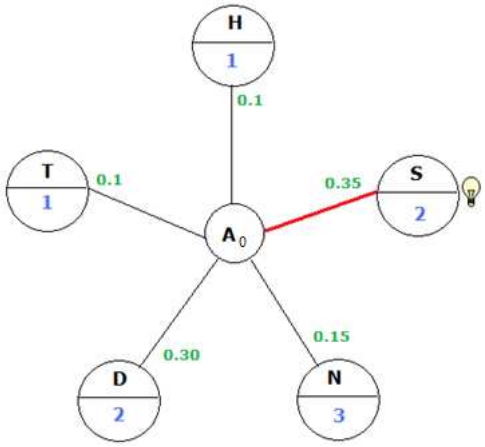


Figure 3-10: Weighted Majority Win in Belief Space.

3.5 Conclusion.

In this chapter, we described the previous contributions to Cultural Algorithm by Saleem, Ping, Ali, and Che. Saleem used random selection of knowledge sources to be followed by the agent. Ping used Marginal Value Theorem. Ali added the social fabric into the Cultural Algorithm, used the majority winning technique to select the agents' knowledge sources, and tested three homogeneous topologies to build the neighborhood networks. Ali noticed that just

adding a social fabric into the Population Space improved the performance of the Influence Function in the Cultural Algorithm. Che added three more homogeneous topologies to investigate deeply how different topologies impact the optimization performance. Also, Che employed a **weighted majority win** technique to integrate the knowledge sources in the Influence Function. A detailed description of Che's social fabricate model will be described in detail in the next chapter. New models of social fabric will be given and described in detail, as well.

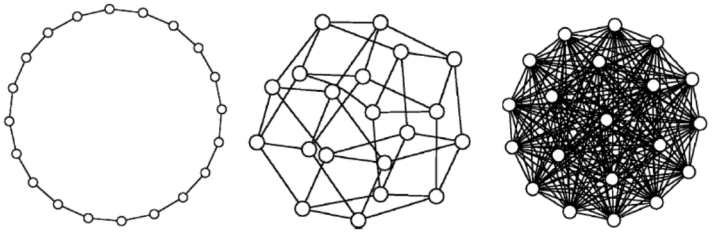
CHAPTER 4 SOCIAL FABRIC MODELS

4.1 Introduction.

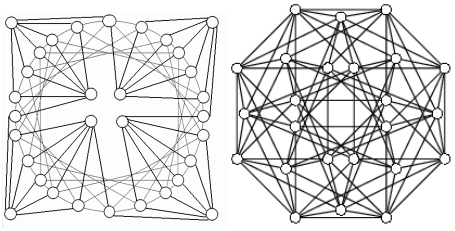
Homogeneous topologies had been used previously to allow the interaction among the Population Space's individuals. Ali [2008] proved that using the homogeneous topologies were sufficient to increase the Cultural Algorithm performance for solving evolutionary problems. The experiments suggested that different network structures worked better for some complexity classes than others. If this is the case, then in a situation where several different problems are presented to a group, they may wish to utilize more than one. So, a new Heterogeneous Model is proposed in this chapter that allows multiple homogeneous topologies to operate together in the Population Space. This chapter will be organized as follows: next section gives a description of the Ali's and Che's Homogeneous Model, the Heterogeneous Model is given in section 3, section 4 describes the Heterogeneous Model pseudo code in detail, and finally, a short conclusion is given as a last section in this chapter.

4.2 Homogeneous Topologies.

Ali [2008] introduced three homogeneous topologies and Che [Reynolds, Che, and Ali, 2010] introduced three more – making six possible homogeneous topologies: lbest (ring), gbest (global), square, octagon, hexagon, and sixteengon. Topologies differ from one another by the number of neighbors connected to each individual or what is called the degree. In a ring topology, each individual is connected only to two neighbors from the Population Space. In a square topology, each individual is connected only to four neighbors. But in the global topology, each individual is connected to all other neighbors. Hexagon, octagon, and hexa-decagon (16-gon) allows every individual (node) to interact with exactly six, eight, and 16 neighbors respectively. Figure 4-1 displays some of these homogeneous topologies.



(a) Ring topology. *(b) Square topology.* *(c) Global topology.*



(d)Hexagon Topology. *(e) Octagon Topology.*

Figure 4-1: The Homogeneous Topologies Used in the Previous Models.

The Homogeneous Model is given in Figure 4-2. In this model, a topology was selected to be used for the entire run as shown in step one. Next in each generation the knowledge source wheel is spun for each individual in order to select the knowledge source that will be the direct influence of that individual in that generation. The area under the wheel is the average performance of each knowledge source for the previous generations. The direct influences are then distributed to the agents in step three along with the direct influences of its neighbors in the topology. Next, in step four, the **weighted majority winning schema** is used to determine which knowledge source will actually influence the individual in that generation.

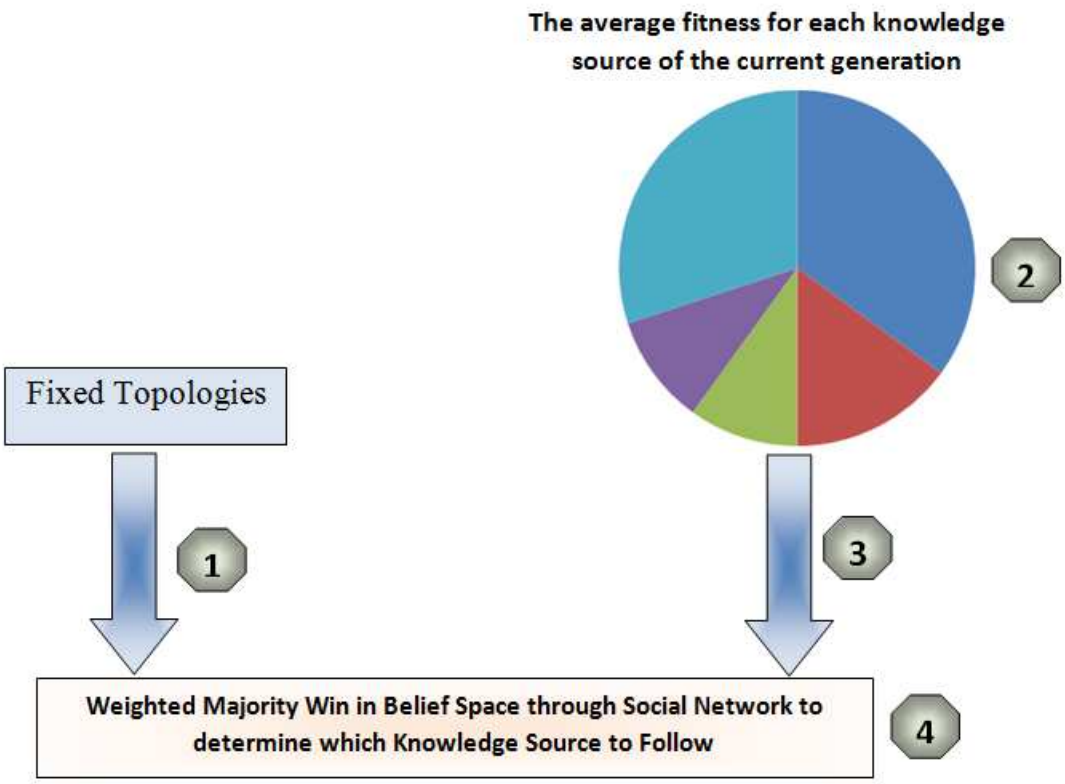


Figure 4-2: The Homogeneous Model.

Homogeneous topologies work quite well at solving fixed class problems, but performance degrades as the complexity of the problem increases [Che, 2009]. Che tested homogeneous topologies to solve the Cones World Problem using Cultural Algorithms and noticed the problem. Figure 4-3 shows the relationship between the problem class and the average number of generations to find the solution using homogeneous octagon topology. The value of (A) (more details in chapter 6) relates to the problem entropy such that 1.01, 3.35, and 3.99 respectively represent fixed, periodic and chaotic problem classes [Langton, 1992]. μ is the average number of generations used to find the solution. As we see in the figure, as the entropy increases we need more steps to find the solution.

It was clear that homogeneous topologies were not sufficient to handle the higher entropy associated with problems of periodic and chaotic class. So, multiple homogeneous topologies

operate simultaneously over the Population Space is what we suggested to evolve heterogeneous topologies. Individuals will start having different options of topologies to choose from. The new proposed topology is named a heterogeneous topology and is described in detail in the next section.

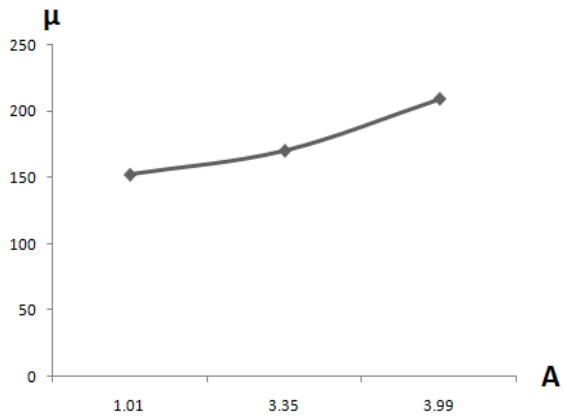


Figure 4-3: Homogeneous Octagon Topology Used in Solving Three Different Problem Classes.

4.3 Heterogeneous Topologies.

The heterogeneous topology consists of the six homogenous topologies [Reynolds and Gawasmeh, 2012] mentioned earlier, working together in the Population Space. Individuals change the topology that they are using in each time step. Two wheels were used; one for the knowledge source and one for the topologies. The knowledge source wheel will be spun for an individual and apply **weighted majority winning scheme** to find the appropriate knowledge source to follow, and then the topology wheel will be spun at each generation separately to determine which topology individual to use in the next generation. Both wheels are updated after each generation. The knowledge source wheel calculates the average individuals' fitness under each knowledge source. The knowledge sources play an important role in this approach because each individual will have the direct influence from the knowledge source that selected it, plus the knowledge sources transmitted to it by its neighbors. Consequently, different topologies operate

together and form a heterogeneous one. The topology wheel calculates the average individuals' fitness for each topology operating among them.

Figure 4-4 demonstrates our Heterogeneous Model. Step one reflects the selection of one of the fixed topologies for use in a generation for each individual in the Population Space based upon the previous performance of each topology using a roulette wheel approach. The area under the wheel for each topology is its normalized average performance in the previous generation. The selected topology is then embedded into the Population Space for that individual in step two. In step three, the first two steps must be repeated for each individual. Then, one of the knowledge sources will be selected for use by the individual from the knowledge source wheel, which is based upon the past performance of the knowledge sources. This wheel is used in step four to generate the direct influence for each individual in the population and collect the direct influence knowledge sources for its neighbors. In step five, the **weighted majority winning** conflict resolution rules are used to determine the winning knowledge source for each individual. The individuals are then modified and evaluated. The results are used to update the selection wheels and the process starts again for the next generation.

Some changes to the Homogeneous Model are needed to allow heterogeneous topologies to evolve in the Population Space. An additional wheel was added to the Belief Space in order to save the topologies' performance. This wheel was named Topology wheel. The Influence Function behavior was also changed and this change will be explained as pseudo code in the next section.

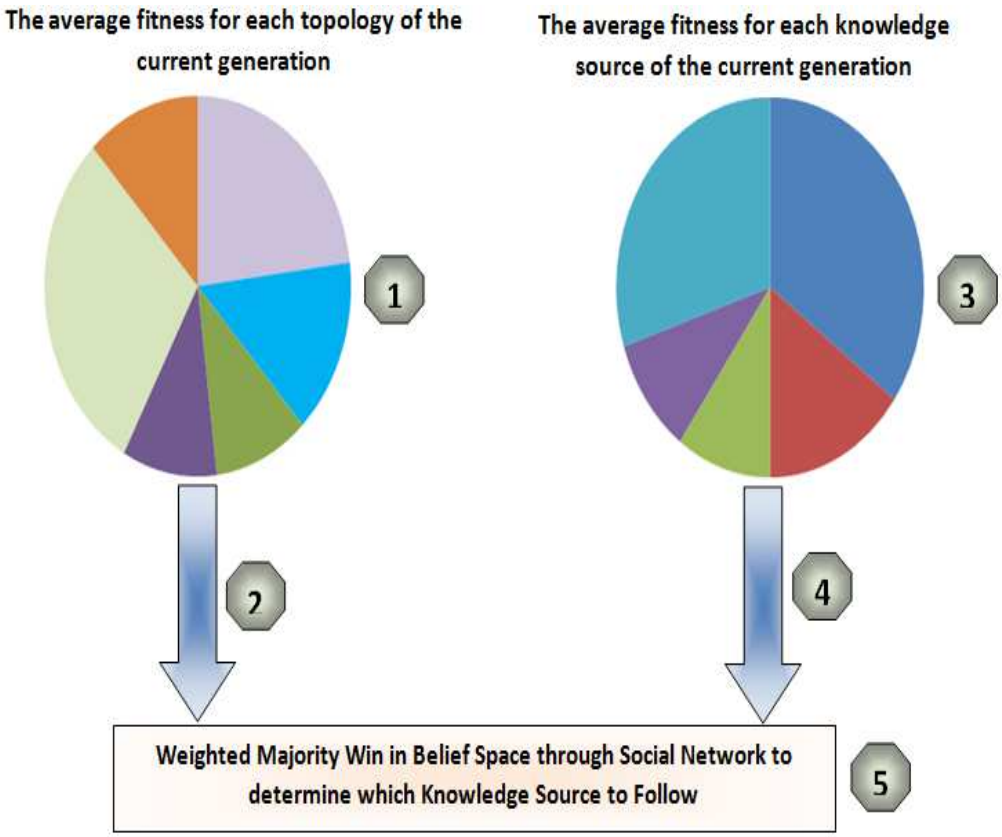


Figure 4-4: Heterogeneous Model.

4.4 Heterogeneous Model Pseudo Code.

In the Influence Function, each individual selects the topology by spinning the topology wheel. If the new chosen topology is different from the original, then the neighbors of the individual will be different. The number of neighbors of the individual may increase or decrease depending on the new topology structure used. After, the individual will spin the knowledge sources wheel to get the direct knowledge source influencer and collects his neighbors' knowledge sources to determine which of them to follow using the **weighted majority winning** approach. The pseudo code in the Influence Function in Belief Space changes the topology of the individual and selects the new followed knowledge source, as given in Figure 4-5.

Calculate the BeliefWheel (5 Knowledge Sources)

Calculate the TopologyWheel (6 Topologies)

FOR each individual A_i in population ***DO***

BEGIN

Clear the Topologies Counters (Initialize CounterTopologies to 0's)

FOR each of the neighbors R_j*DO*

BEGIN

Get the topology type of the neighbor T_j

Increment the Counter of topology T_j in CounterTopologies

END

Adjust the count of each Topology using each Topology's average fitness as weight

Pick the winning Topology based on new topology weight counts

Update the neighbors of A_i

Clear the knowledge Source Counters

Spin the BeliefWheel to get direct influence KSdirect

Increment the Counter of Knowledge Source KSdirect

FOR each of the neighbors R_j***DO***

BEGIN

Get the KS type that influenced the neighbor K_j

Increment the Counter of Knowledge Source K_j

END

Adjust the count of each KS using each KS's average fitness as weight

Pick the winning KS based on new weight counts, KSI

END

Figure 4-5: Pseudo Code of Heterogeneous Topologies Model.

Figure 4-6 gives a pseudo code for calculating the topology wheel. The topology wheel changes its values after each generation. Counting the number of individuals that use a specific

type of topology Ct and calculating the sum of fitness values of those individuals St. The average of the individual's fitness values for that topology AVt is calculated by dividing St on Ct. Each sector of the wheel is calculated by dividing AVt on the sum of all topologies' AVt.

```
FOR each individual Ai in population DO
BEGIN
        toplogyCounter[topology of Ai]++; // Ct
        sumTopologyFitness[topology of Ai] += fitnessValueOf Ai; // St
END
FOR each topology in the heterogeneous social fabricDO
BEGIN
        avgTopologyFitness[topology]=sumTopologyFitness[topology]/topologyCou
        nter[topology]; // AVt
sumAvgsFitness+=avgTopologyFitness[topology];
END
FOR each topology in the heterogeneous social fabricDO
BEGIN
        percentTopologyInWheel= avgTopologyFitness[Topology]/sumAvgsFitness;
END
```

Figure 4-6: Topology Wheel of Heterogeneous Model Pseudo Code.

4.5 Conclusion.

In this chapter, we introduced a heterogeneous topology to operate between the individuals of the Population Space. A group of homogeneous topologies were allowed to operate together simultaneously to reflect heterogeneous connections in the Population Space. We added a new wheel in the Belief Space to store the individuals' average fitness. Then, we used this wheel to allow the individuals to select the current topology at each time step. In spite of storing the topology information in the Belief Space, the knowledge sources do not have any

kind of information about the topologies. Thus, we introduced two more models to allow the dependability between the knowledge sources and topologies.

Chapter 7 shows that heterogeneous topology increases the performance of solving the problems by increasing the complexity of problem classes. Chaos landscapes are best exploited by heterogeneous networks, whereas static landscapes are best exploited by single homogeneous topologies. Heterogeneous topologies somehow produce a social organization that is better able to exploit heterogeneous landscapes. Heterogeneous topologies performed better than only a few homogeneous topologies, in terms of performance with periodic class problems, so we started looking for a third Heterogeneous Model. In the next chapter, we will propose a new model to meet our expectations.

Keeping the knowledge source and topology wheels separate from one another blinds the Influence Function of the relationship between the knowledge sources and the topologies used among the individuals. Therefore, the Subcultures Model was proposed to build the relationship between Belief Space's knowledge sources and Population Space's topologies.

CHAPTER 5 SUBCULTURES MODEL

5.1 Introduction.

Cultural Algorithms are computational models of social evolution based upon principles of Cultural Evolution. A Cultural Algorithm consists of a Belief Space consisting of a network of active and passive knowledge sources and a Population Space of agents. The agents are connected via a social fabric over which information used in agent problem solving is passed. The knowledge sources in the Belief Space compete with each other in order to influence the decision making of agents in the Population Space. Likewise, the problem solving experiences of agents in the Population Space are sent back to the Belief Space and used to update the knowledge sources there. It is a dual inheritance system in which both the Population and Belief spaces evolve in parallel. However, the knowledge about the social fabric's topologies in the Population Space is not visible to the Belief Space. By collecting the individuals' experiences in topologies and knowledge sources, we expect to see subcultures start to evolve in the Population Space.

Collecting the individuals' experiences in topologies and knowledge sources is not the only motive for using subcultures start to evolve in the Population Space. Basically, what we have is a complex system with a number of different layers of details. So, as the details of the system become more complicated, there are more layers, and then there are more opportunities to exploit those layers using subcultures. In this chapter, a new Subcultures Model is described in the next section. Subcultures Model's pseudo code is given in section 3. Section 4 gives a conclusion of the new subcultures Model.

5.2 Subcultures Heterogeneous Topologies.

Cultural Algorithms model cultural systems. Cultural systems are a particular example of complex systems. Complex system is composed of many different layers of granularity. As a

matter of fact, we don't have much of a complex system when we have a single layer. When the number of different layers increases the details of the system become more complicated. As a result, there is a potential to exploit the different layers using subcultures. Actually, each of these subcultures is dealing with the complex system layers separately rather than all at once. Evolving subcultures is related to the Cultural Algorithm components. Situational, Normative, Domain, Historical, and Topographical knowledge sources formed the Belief Space in the Cultural Algorithms, but none of these knowledge sources were related to the individuals' social connections in the Population Space level. Although social fabricated Population Spaces performed better than non-social fabricated ones, they increased the unpredictability of the resulted performance in terms of the number of generations needed to solve the problems [Reynolds and Gawasmeh, 2012]. In addition to the unpredictability problem, we still need a model that enhances the visibility between the social fabric and the Belief Space.

As the heterogeneous topology consists of six different homogeneous topologies operating together and five knowledge sources that are affecting the Population Space, we propose the topology knowledge to be in the form of a 6 by 5 matrix. For each topology, we collect information about the five knowledge sources they follow, so each cell (c[x, y]) row x and column y contains a real value which represents the average fitness of the individuals who use topology x and knowledge source y together and it can be calculated by the following equations:

$$\mathrm{m} = \sum_{i=1}^{\mathrm{n}} \sum_{T_i=\mathrm{x} \text{ and } K_i=\mathrm{y}} 1 \qquad (5.1)$$

$$\mathrm{c[x,y]} = \left(\sum_{i=1}^{\mathrm{n}} \sum_{T_i=\mathrm{x} \text{ and } K_i=\mathrm{y}} \mathrm{f_i}\right)/\mathrm{m} \qquad (5.2)$$

In the above equations, N is the number of individuals, m is the number of individuals who use topology x and knowledge source y, T_i is the topology that connects individual i to other

individuals, K_i is the knowledge source selected by individual i, and f_i is the fitness value of individual i.

The Subcultures Model allows the individual to select the topology by spinning the topology wheel. The knowledge source wheel will be divided into n wheels, where n is the number of topologies. Each wheel gives the individuals' average fitness of the knowledge sources for a specific topology. After the individual selects the topology, he will get the direct knowledge source by spinning the knowledge sources wheel related to its new topology. Figure 5-1 demonstrates how the process has been changed for the new Subcultures Model. Step one reflects the selection of one of the fixed topologies for use in a generation based upon the previous performance of each topology using a roulette wheel approach. The area under the wheel for each topology is its normalized average performance in the previous generation. The selected topology is then used to select one of the knowledge source wheels. Then, for each topology there is a separate wheel, shown in step three, which is used to select the knowledge source that is used to influence each individual based upon the past performance of the knowledge sources for the selected topology. Thus, there are six separate knowledge source wheels, one for each topology that is used for a generation. This wheel is used in step four to generate the direct influence for each individual in the population and collect the direct influence knowledge sources for its neighbors. In step five, the weighted majority win conflict resolution rules are used to determine the winning knowledge source for each individual. The individuals are then modified and evaluated. The results are used to update the selection wheels and the process starts again for the next generation.

Some changes to Cultural Algorithms are needed to evolve the subcultures. As mentioned earlier in this chapter, n Belief Space wheels are implemented, where n represents the number of

homogeneous topologies used in the Population Space. The Influence will be changed to handle the newly added Belief Space wheels. The next section will show the Influence Function changes in terms of pseudo code.

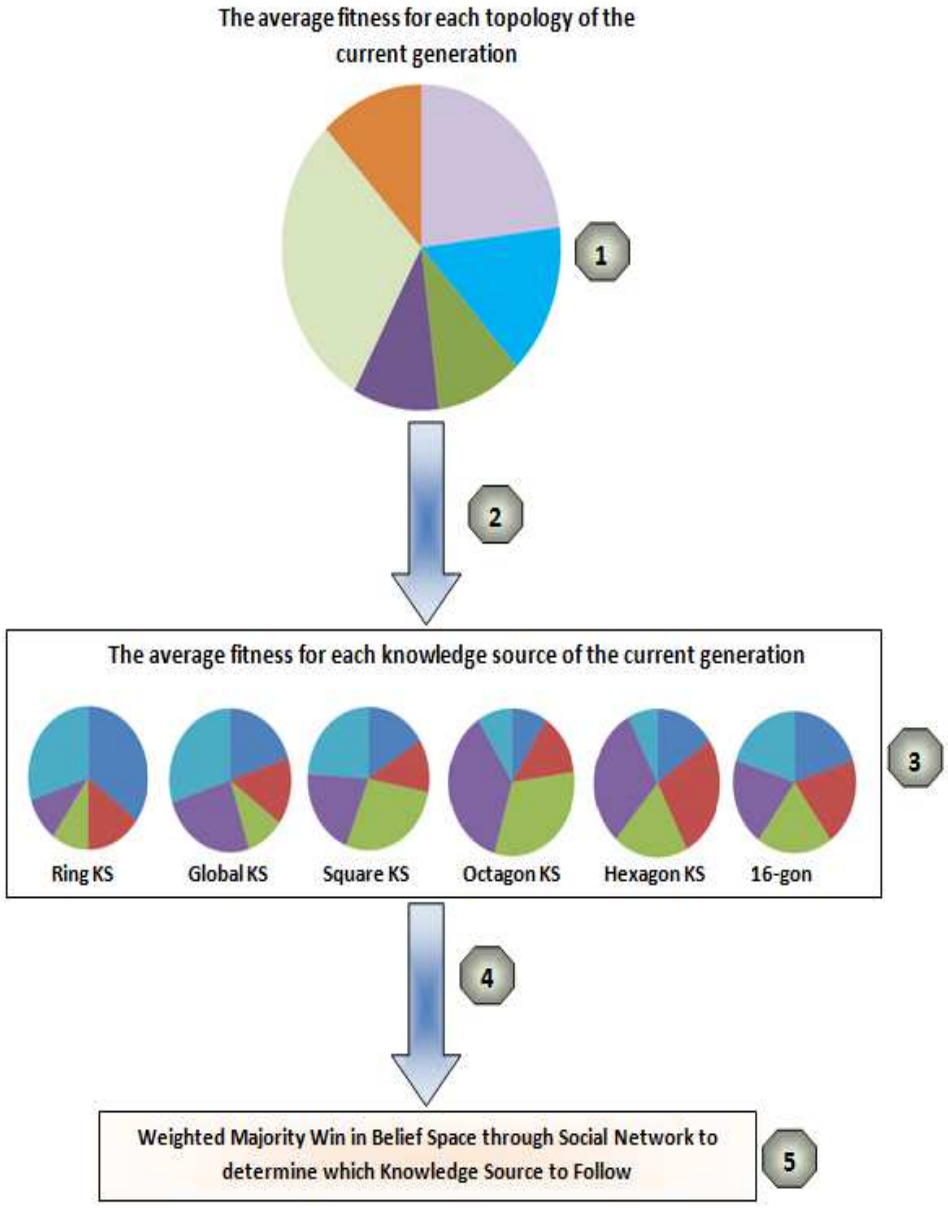


Figure 5-1: Subcultures Model.

5.3 Subcultures Pseudo Code.

In order to operate the Subcultures Model in Cultural Algorithm, two changes are needed: changes to the performance wheels and the Influnce function. Two types of wheels are used to evolve subcultures in the Population Space. First, a Topology wheel that summarizes each topology performance in the whole Population Space. Second, six Belief Wheels, one for each topology, summarize the individuals' performance for a specific topology. The change in Cultural Algorithm to calculte the topolgy and knowledge source wheels is given in Figure 5-2. Those wheels will be used in the Influence Function. The change in the Influence Function is given as a pseudo code in Figure 5-3. Each individual in the Population Space selects the topology by spinning the topology wheel and applying the weighted most winning approach. If the new chosen topology is different from the original, then the neighbors of the individual will be updated. Then, the individual will spin the knowledge sources wheel to get the direct knowledge source influencer and collects his neighbors' knowledge source to determine which of them to follow using the weighted most winning approach.

The Topology wheel is calculated the same way as in the Heterogeneous Model. Figure 5-4 gives a Pseudo Code for calculating the Belief Space wheels. The Belief Space wheels changetheir values after each generation. After each generation, the sum of fitness values of the individuals who use topology T will be calculated and stored in sumTopologyAvgsFitness [T]. In addition, the sum of fitness values of the individuals who use topology T and knowledge source KS will be calculated and saved in C [T][KS]. Then, each section of the wheels will be calculated by dividing c[T][KS] on sumTopologyAvgsFitness [T].

```
Initialize Population Space
Initialize Belief Space
j = 0
REPEAT
    Evaluate the Performance of Individuals in the Population Space
    Selected Individuals List = Apply Acceptance Function (Acceptance()).
    Apply Update Function to Update the Knowledge Sources (Update()).
    IF generation i is a multiple of 3 THEN
        Clear Topology Wheel (Each Topology has a Wheel)
        Clear KS Wheels (Each Topology has a KS Wheel)
    End If
    FOR each Individual in Pupulation Space (Individual_i) DO
    BEGIN
        Ind_T_i = Individual_i's Topology
        Ind_KS_i = Individual_i's Knowledge Source
        Update Topology Wheel of Ind_T_i with Individual_i's Performance
        Update Ind_T_i KS Wheel of Ind_KS_i with Individual_i's Perfomance
    End For
    Apply the Influence Function into the Population Space (Influence())
    j = j +1
UNTIL Termination Condition Achieved
```

Figure 5-2: Subcultures Model Pseudo Code in Cultural Algorithm.

Get the Calculated the the TopologyWheels (6 Topologies)

Get the Calculated the 6 BeliefWheels (5 Knowledge Sources) one wheel per topology

FOR each individual A_i in population ***DO***

BEGIN

Clear the Topologies Counters (Initialize CounterTopologies to 0's)

FOR each of the neighbors R_j*DO*

BEGIN

Get the topology type of the neighbor T_j

Increment the Counter of topology T_j in CounterTopologies

END

Adjust the count of each Topology using each Topology's average fitness as weight

Pick the winning Topology based on new topology weight counts T

Update the neighbors of A_i

Set the TBeliefWheel (C[T]) to the BeliefWheel of selected topology T

Clear the knowledge Source Counters

Spin the TBeliefWheel to get direct influence KSdirect

Increment the Counter of Knowledge Source KSdirect

FOR each of the neighbors R_j***DO***

BEGIN

Get the KS type that influenced the neighbor K_j

Increment the Counter of Knowledge Source K_j

END

Adjust the count of each KS using each KS's average fitness as weight using TBeliefWheel

Pick the winning KS based on new weight counts, KSI

END

Figure 5-3: Subcultures Model Pseudo Code in Influence Function.

```
FOR each individual Ai in population DO
BEGIN
        sumTopologyAvgsFitness[topology of Ai]= fitnessValueOf Ai; //St
        C[topology of Ai] [KS of Ai]+= fitnessValueOf Ai;//Stk
END

FOR each topology in the Subcultures social fabric DO
BEGIN
        FOR each topology in the Subcultures social fabric DO
BEGIN
                C[Topology][KS]=
C[Topology][KS]/sumTopologyAvgsFitness[Topology];
        END
END
```

Figure 5-4: Belief Space Wheels Mechanics for Subcultures Model Pseudo Code.

5.4 Conclusion.

This chapter introduced a new social fabric model. This model is named Subcultures Model because it divides the Population Space into subsets of related individuals. Two main changes to the Cultural Algorithm's behavior needed to allow evolving the subcultures among the Population Space's individuals. One of these changes divides the knowledge sources wheel into n wheels, where n represents the number of topologies that operate in the Population Space. The other change is to update the Influence Function to handle the additional wheels.

The next chapter describes the experimental framework that we use to test the Heterogeneous and Subcultures Models against the homogeneous ones.

CHAPTER 6 EXPERIMENTAL FRAMEWORK

6.1 Introduction.

We extended the Cultural Algorithm Toolkit 2.0 [Che, 2008] by adding heterogeneous topologies to operate among the individuals in the Population Space. The Cultural Algorithm Toolkit is embedded in the Recursive Porous Agent Simulation Toolkit (Repast). To examine the heterogeneous topologies' performance under different problems complexity classes we used the Cone's World Generator. Different social metrics will be used to describe the topologies and knowledge sources' performance. In this chapter, we will describe the Repast, Cultural Algorithm Toolkit, Cones World Generator, and the Social Metrics.

6.2 Repast.

Our extended Cultural Algorithm Toolkit 2.0 (CAT2.0) is embedded in the Recursive Porous Agent Simulation Toolkit (Repast) and it is widely used for free. In addition, it is open source, fully Object Oriented, implemented in several languages, and gives simulation results and graphing tools [North,Collier and Vos, 2006; North et al., 2007]. It was originally developed by Collier, Sallach, Howe, North and others [Collier, Howe and North, 2003] and was created at the University of Chicago. Repast is now managed and led by members from government, academic and industrial organizations. The Repast's source code can be downloaded from the website (http://repast.sourceforge.net/index.html).

Four versions of Repast were released in three different languages: (1) RepastJ (Java based), (2) RepastPy (based on Python Scripting), (3) Repast.Net (C#.Net based), and (4) RepastS (Repast Symphony, Java based). Repast can be run on all modern computing platforms like Linux, Mac OS, and Windows. Che and Ali used RepastJ that is based on Java, and they embedded CAT1.0 [Reynolds and Ali, 2007] and CAT2.0 [Reynolds, Che, and Ali, 2010] in its environment.

Repast is a very flexible environment that easily allows the Cultural Algorithm Toolkit to be embedded into it. The Repast System contains a set of agents (agent based system). These agents may or may not be homogenous and can be arranged in different ways (as a hierarchy for example). Repast allows the interactions among those agents. The set of agents' behavior (action) gets executed every tick (time step). Repast also allows for more complex dynamic scheduling because the agent's action is not the same at each tick. It may start with one action and schedules another action to execute at some future tick. In addition, this system can control simulation visualization, record data, and analyze results.

Repast functions by allowing users to update the agent's actions, add more classes into the Repast, incorporate the Repast library components into their own systems, use the standard toolbar and GUI to manipulate the parameters, and output data like histograms. In executing these actions, less time is given for GUI development and more given for modeling the real problem. However, a user may change the Repast's toolbar or GUI if needed. In our case, extensive programming is needed to model the Cultural Algorithm with all of its components: Population Space, Belief Space, problem landscape, and the Communication Protocol. This is because of the wide variety in the knowledge sources' data structures and the social fabricated Population Space.

6.3 Cultural Algorithm Toolkit (CAT 2.0).

Ali, who added the social fabric into the Cultural Algorithm, embedded the Cultural Algorithms framework in the Repast agent-based simulation toolkit. Ali implemented the social fabric Influence Function along with the Marginal Value Theorem (MVT) and random knowledge source integration Influence Functions. Ali's resultant toolkit was named Cultural Algorithms Simulation Toolkit (CAT) [Reynolds and Ali, 2008]. Figure 6-1 gives the software

architecture of how CAT is implemented in the basic Repast Integrated Development Environment [Ali, 2008].

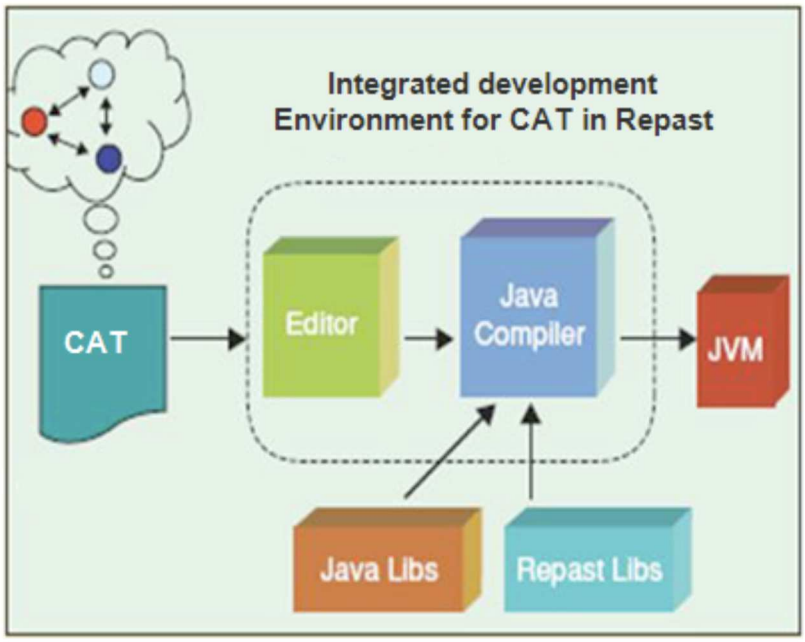


Figure 6-1: The Organization of the Implementation of CAT in the Repast Environment [Ali, 2008].

Che extended CAT1.0 to the Cultural Algorithms Toolkit (CAT 2.0) that is embedded in the Recursive Porous Agent Simulation Toolkit (Repast) as shown in Figures 6-2 and 6-3. He added an ID and a memory to each individual in the Population Space to eliminate the randomization from the old versions. In that process, each individual keeps its neighbors in the memory, and once the neighbors are assigned they stay fixed throughout the whole process. Che implemented the social fabric to cover six different homogeneous topologies, namely, ring, square, hexagon, octagon, hexa-decagon and global. They are all topologies that have been employed in CAT 2.0. This neighborhood stayed the same throughout the whole process, which is homogeneous.

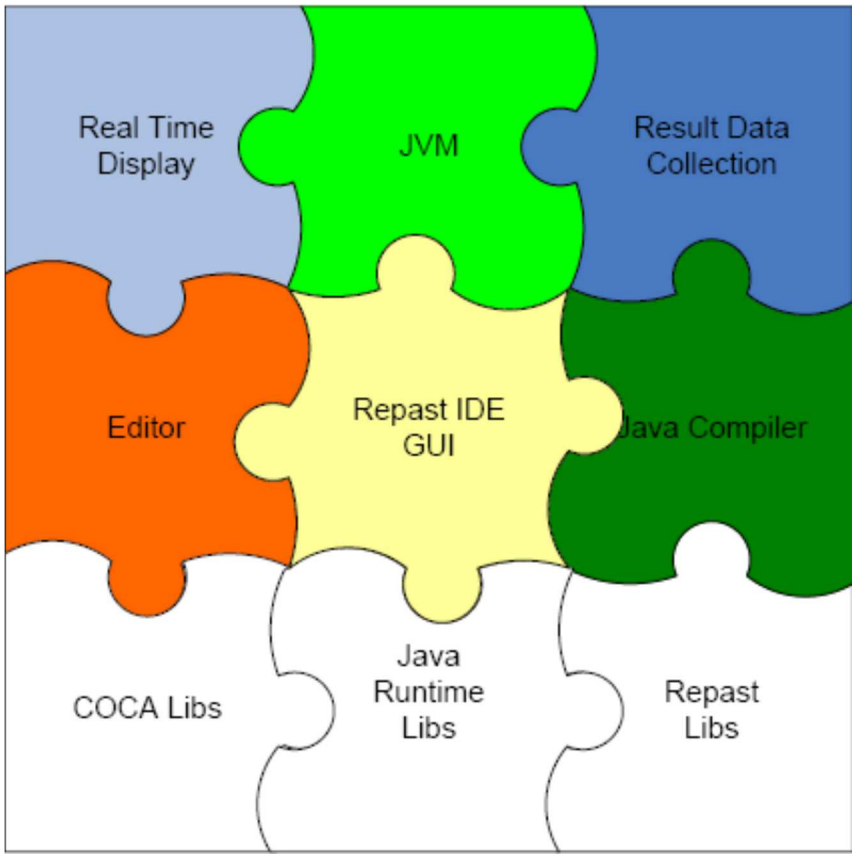


Figure 6-2: CAT2.0 System Integration Puzzle [Che, 2009].

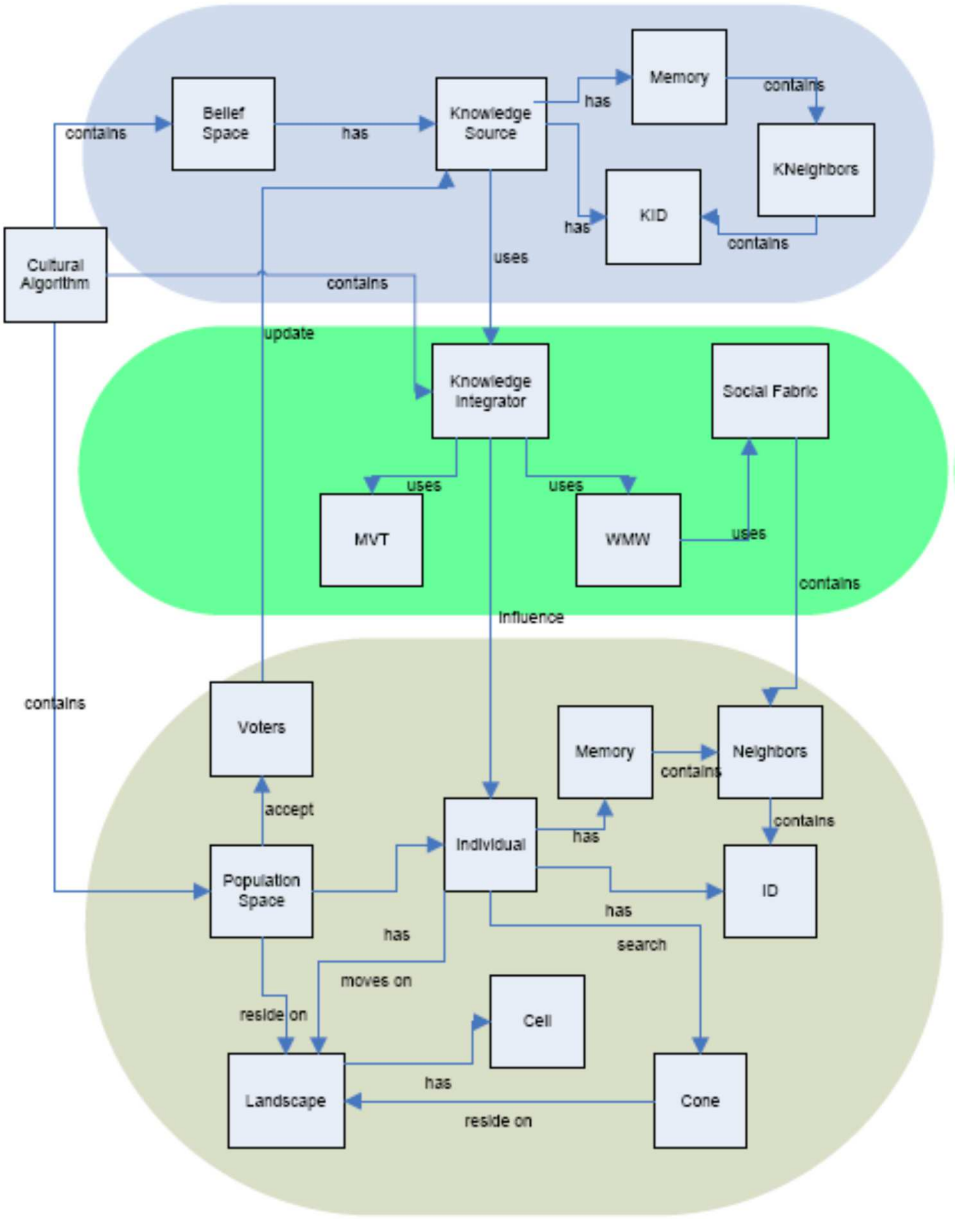


Figure 6-3: Object Oriented Diagram for CAT2.0 [Che, 2009].

6.4 Cone's World Generator.

The Cones World Generator [Morrison and De Jong, 1999] will be used to produce evolutionary problems in terms of arbitrary complexities. Figure 6-4 shows a 3D example of the Cones' World environment. Multi-dimensional landscapes can be generated by scattering cones

in different heights, slopes, and locations over the landscape. Therefore, the Cones World generator follows two stages to generate an evolutionary landscape. First stage, in which cones are placed on a landscape. Then, cones are combined to produce a continuous functional landscape using a max function. If two cones overlap, the maximum of the two will be taken. The second stage, then iteratively goes through the landscape and dynamically adjusts the cone parameters. The base landscape is given by the equation:

$$f(< x_1, x_2, \ldots, x_n >) = max_{j=1,k}(H_j - R_j * \sqrt{\sum_{i=1}^{n}(x_i - C_{j,i})^2}) \qquad (6\text{-}1)$$

In the above equation, k is the number of cones, n is the dimensionality (2D), H_j is height of cone j, R_j is slope of cone j, and $C_{j,i}$ is coordinate of cone j in dimension i. The values for each cone (H_j, R_j, and $C_{j,i}$) are randomly generated based on user specified ranges. The second step is to specify the dynamics, which gives the ability of changing the values of Cones' parameters like the slope, height, and location. The changes of Cone's parameters will be managed by the logistic function [Morrison and De Jong, 1999]:

$$Y_i = A * Y_{i-1} * (1 - Y_{i-1}) \qquad (6\text{-}2)$$

In the logistic function, A is a constant with a range between 1.0 and 4.0, and Y_i is the value at iteration i. A bifurcation map of this function is provided in Figure 6-5. This Figure shows the relationship between the given values of A and the logistic function results Y. Value A can be a single value (a small step-size change in the Cone's parameter), a double value (two step-size changes in the Cone's parameter), and an interval of values (many step-size changes in the Cone's parameters).

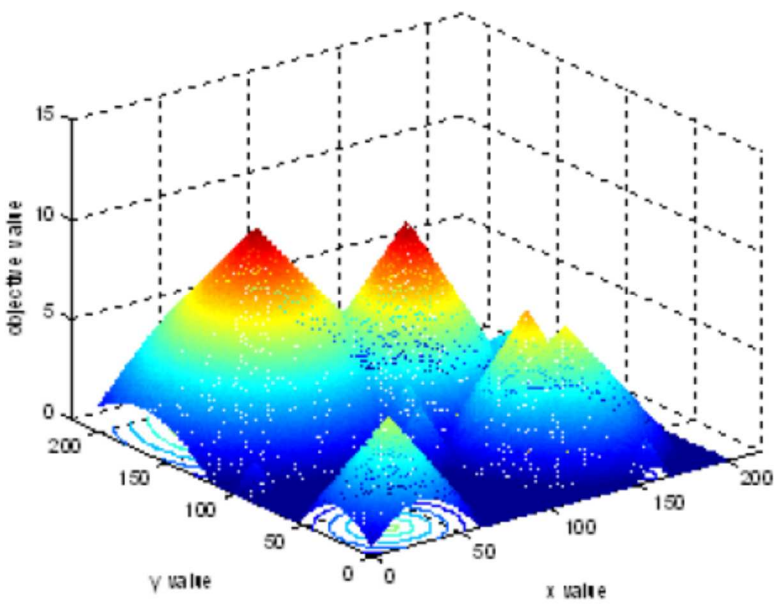


Figure 6-4: Example of cones world environment.

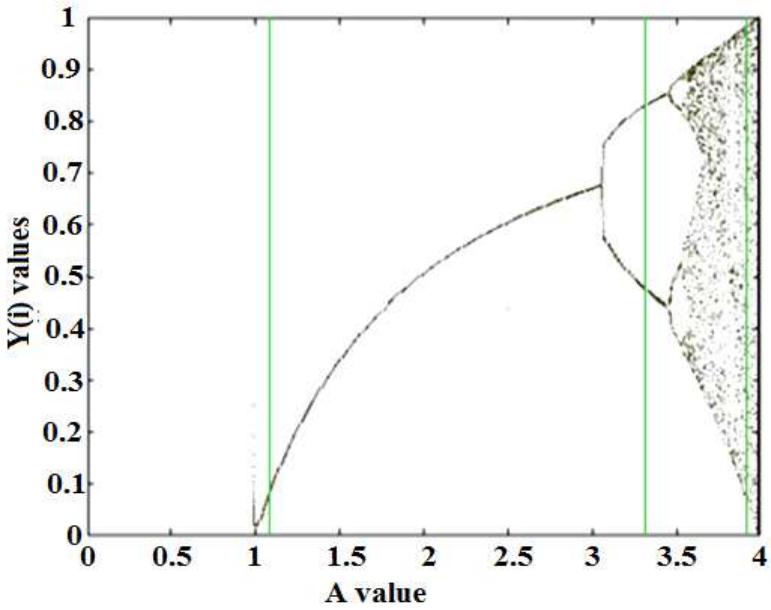


Figure 6-5: Logistic Function Y with Characteristic A Values.

Morrison and Jong attached a logistic function with a specified (A) value to each parameter in the cones world. They gave the Cone's World four parameters: height, slope, x-axis location, and y-axis location. Three values of (A) are picked in previous work [Reynolds and

Gawasmeh, 2012; Gawasmeh and Reynolds; 2014] to test the behavior of the Cultural system in three different problem classes. The green lines in Figure 6-5 represent the chosen values of (A). Picking the value 1.01 for represents the one step-change and simulates the fixed class problems. The periodic class problems can be simulated using two-step change, so a value of 3.35 has been chosen for this goal. Finally, chaotic class problems are unpredictable and need many bifurcations, so (A=3.99) is a good choice for that. Fixed, periodic, and chaotic class problems are described and proposed by Langton [1992]. In this work, three values groups of A were selected in order to test the behavior of the Cultural system in three different problem situations. A = (1.4, 1.8, 2.2 and 2.6) represent the one step-change and simulates the Fixed class problems. The periodic class problems can be simulated using a two-step generation process, so a values of 3.1, 3.2, 3.3 and 3.4 for A were chosen. Finally, chaotic class problems are produced by many generators due to the bifurcations, so A = (3.6, 3.7, 3.8 and 3.9) are good choices for that. Fixed, periodic, and chaotic class problems.

In order to test how these problem categories are reflected by changes in the corresponding social organization of the Cultural systems, an example set of landscapes from each of the three basic classes fixed (A=1.01), periodic (A=3.35), and chaotic (A=3.99) will be generated. Our task will be to measure the subcultures performance and compare it with the Homogeneous and Heterogeneous Models' performance in order to solve problems from different complexity classes.

6.5 Social Metrics.

We need social metrics [Che, 2010; Reynolds, Che, and Ali, 2010] to measure the effect and the performance of heterogeneity and subcultures social fabrics on solving evolutionary problems with different complexity classes. Three metrics (Dispersion Index, Majority and

Minority Win Scores, and Innovation Cost Index) are described in this section to monitor the Culture's vital signs in any given environment. Metrics assess the extent to which the Influence Function is able to generate diversity at each of its several stages. Therefore, the focus will be on the Influence Function. The Influence Function can be divided into two main parts:

(1) The Marginal Value Theorem (MVT). The Update Function uses the agents' experiences to update and increase the diversity of the knowledge sources. The MVT gets stored as a roulette wheel to reflect the Population Space's usage of knowledge sources and to give each agent a direct knowledge source to follow. Thus, the MVT is a co-evolutionary device that associates the knowledge sources with predators and the population of agents, as prey spread out over a functional landscape. The metric associated with this part is called the Dispersion Index. This metric measures the distance (Social Tension) on the functional landscape over which directly connected individuals are spread.

(2) Agent's Vector voting scheme and its neighbors. In the Agent's Vector Voting Scheme, the agent follows the highest weighted knowledge source and the highest weighted topology. The distribution of knowledge in the Population Space happens among the neighbors through a single or many connections (hops). To evaluate the distribution of knowledge in the Population Space, Majority Win Score, Minority Win Score, and the Innovation Cost are used as the metrics that will be used. Majority Win Score is the average value of the score when the majority wins the bidding war in a time step. Minority Win Score is the average score for the time period when a minority knowledge source wins the bidding. The Innovation Cost is the difference between the Minority Win Score and the Majority Win Score. It represents the drop in performance associated with the need to experiment with new solutions.

6.5.1 The Dispersion Index Metric (Social Tension).

The five knowledge sources influence the individuals in the Population Space, so we expect the individuals to learn from these experiences. In addition, we expect the knowledge sources to learn new experiences from the updated population. As each knowledge source will influence a group of scattered individuals from the Population Space and direct them towards the best possible solution, the distance among the network neighbors over the landscape needs to be measured to see if most of they are close to the solution. As a result, the Dispersion Index is introduced. The Dispersion Index is the sum of the Euclidean distances among the directly connected neighbors in the network. If the Dispersion Index is 0, then they will be all located at a given point. If the Dispersion Index increased, then the individuals' experiences would be different and more scattered. Social Tension is calculated using:

$$S.T = \sum_{i=1}^{N} \sum_{j=1}^{M_i} \sqrt{\sum_{k=1}^{D} \left(X_{i,k} - a_{i,j,k}\right)} \qquad (6\text{-}3)$$

In the above equation, S.T is the social Tension, N is the total number of individuals, D is the total number of dimensions of this environment, M_i is the number of neighbors directly adjacent to individual i, $X_{i,k}$ represents the coordinate on dimension k for individual i, and $a_{i,j,k}$ is the coordinate of j^{th} neighbor of individual i on dimension k.

The pseudo code for calculating Social Tension (ST) is given in Figure 6-6 where ST is the Social Tension and E_Sum is the total of Euclidean distance between an individual and its neighbor.

Initialize E_Sum and ST;

For each individual A_i in a generation

Find it neighbors $A_j[A_1,...A_K]$

E_Sum = E_Sum + E_{ij}// $E_{ij}$is the Euclidean distance between i and $_j$

ST = E_Sum/N //N is the total number of the neighbors for this topology

Figure 6-6: Pseudo Code of Social Tension.

A higher Dispersion Index means that neighbors that are adjacent in the network can get information about more diverse places on the landscape. The CAT system generates a graph of the Dispersion Index values during the running time. Figures 6-7 and 6-8 shows Dispersion Index values for two different runs using CAT 2.0.

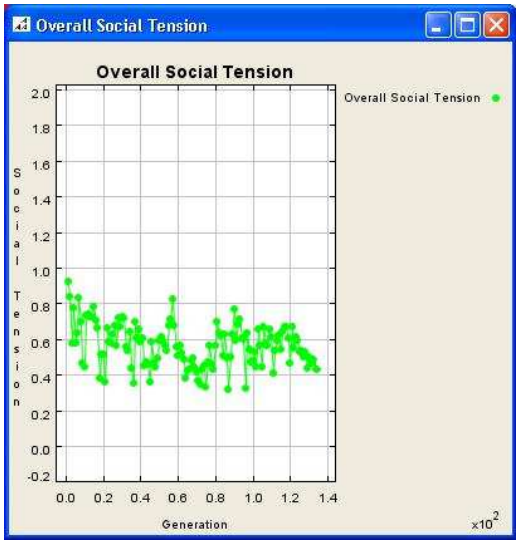


Figure 6-7: The Dispersion Index Values of Solved Cones World Problem.

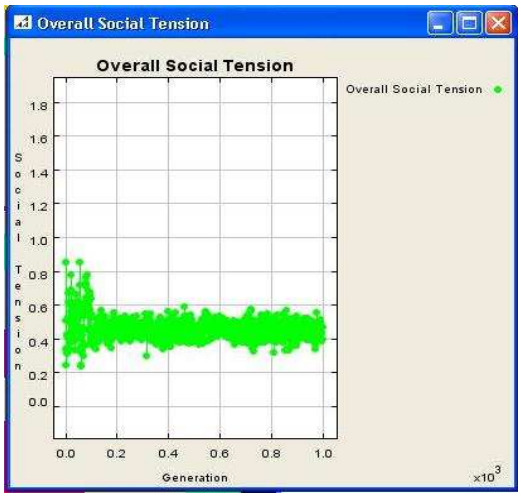


Figure 6-8: The Dispersion Index Values of Unsolved Cones World Problem.

6.5.2 Minority/Majority Win Scores and Innovation Cost Index.

The goal of the indices in this section is to compute the needed effort to allow the minority knowledge source to win instead of the majority knowledge source. This is the Innovation Cost index. The Innovation Cost index occurs when a knowledge source with few individuals finds a new promising region and as a result, has average performance for that time period that is high enough to beat the sum of the majority influences. The basic indices are defined as follows: (1) Minority Win Score: For each generation, it is the average fitness of the winning knowledge source when the minority win case occurs, (2) Majority Win Score: For each generation, the average fitness of the winning knowledge source when the majority win case occurs, (3) Innovation Cost Index: The difference between the Majority Win Score and the Minority Win Score assuming that the majority Win Score will be greater than the minority win score. For maximization problems, it is the index that reflects the cost of innovation in terms of the reduction in performance that results when the majority does not win in a given situation.

Figure 6-9 gives the pseudo code that produces the above mentioned two winning scores and the Innovation Cost Indexing the Influence function in Belief Space. As explained in the

definitions, the Innovation Cost Index depends on Majority and Minority Win Scores. Therefore, CAT system generates one graph containing the three values, as shown in Figure 6-10.

For each individual A_i in population

Spin the Topology Wheel to get a new individual network

Update individuals' neighbors

Spin the belief Wheel to get direct influence KSdirect

Find all neighbors of A_i

For each of the neighbors

Get the KS type that influenced the neighbor

Count KS that voted for Ai (including KSdirect)

Pick the KS with largest Count, KSM

Adjust the count of each KS using each KS's average fitness as weight

Pick the winning KS based on new weight counts, KSI

If KSM = KSI,

MajorityWincaseCount+1, MajorityWinScore_i = weighted count of KSI

If KSM <>KSI,

MinorityWincaseCount+1, MinorityWinScore_i = weighted count of KSI

MajorityWinScore = MajorityWinScore + MajrityWinScore_i

MinorityWinScore = MinorityWinScore + MinorityWinScore_i

MajorityWinScore = MajorityWinScore/MajorityWinCaseCount

MinorityWinScore = MinorityWinScore/MinorityWinCaseCount

InnovationCost = MajorityWinScore – MinorityWinScore

Figure 6-9: Pseudo-Code of Innovation Cost Index [Che, 2009].

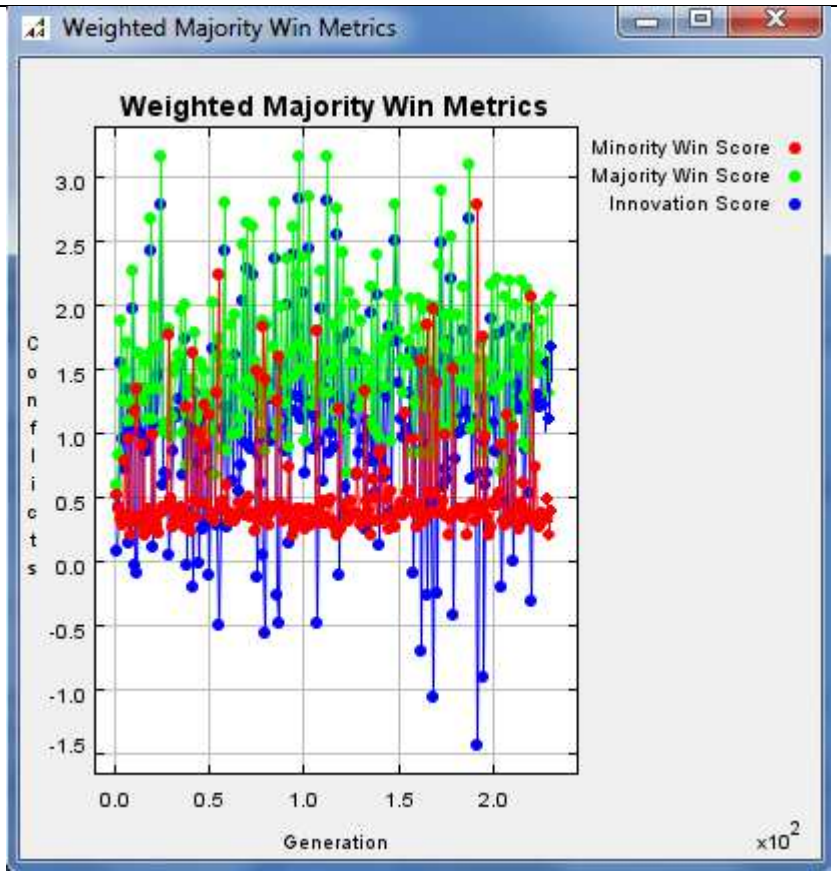


Figure 6-10: CAT System's Innovation Cost Index Results.

6.6 Conclusion.

In this chapter, we explained our experimental environment, which was close to Che's experimental environment in picking the (A) values and the number of conducted experiments. We extended CAT2.0 through adding Heterogeneous and Subcultures Models to operate in the Population Space. Then, we used Cones World problem to test our new approach, because Cones World problem gives the complete range of Langton's optimization problem complexities, from fixed to chaotic. Finally, three social metrics were explained to compare our results with a previous work. Dispersion Index and Innovation Cost Index represent the three metrics, because Innovation Cost Index is calculated using the Majority and Minority Win metrics. Those metrics are related to diversity and innovation. A higher Dispersion Index means that neighbors that are

adjacent in the network can get information about more diverse places on the functional landscape. A lower Innovation Cost Index means that innovations that are produced are more productive in exploring new promising areas over the functional landscape. The next two chapters show the results from running the CAT system with Heterogeneous and Subcultures Models.

CHAPTER 7 HETEROGENEOUS MODEL RESULTS

7.1 Introduction.

In this chapter, we compare the heterogeneous topology with one of the homogeneous topologies for the social fabric in terms of their performance and their effect on the knowledge sources within the three complexity categories (fixed, periodic, and chaotic categories). From each problem class, we selected four entropy values (A). The entropy values have been chosen from the fixed, periodic, and chaotic categories, respectively, (1.4, 1.8, 2.2, and 2.6), (3.1, 3.2, 3.3, and 3.4), and (3.6, 3.7, 3.8, and 3.9). Each entropy value (1.4, 1.8, 2.2, 2.6, 3.1, 3.2, 3.3, 3.4, 3.6, 3.7, 3.8, and 3.9) has five randomly generated example Landscapes. One of the homogeneous topologies (Lbest, Square, Hexagon, Octagon, Hexa-decagon, or Gbest) was selected to be compared with the heterogeneous topology. The Square topology was the choice over the rest of homogeneous topologies for two reasons. First, it outperformed the rest of topologies in fixed and periodic classes [Che, Ali, and Reynolds, 2010; Reynolds, Che, and Ali, 2010; Reynolds and Ali, 2008; Che, 2009]. Second, Octagon outperformed the other topologies in the chaotic class and Octagon can be structured as double Square topologies [Che, Ali, and Reynolds, 2010; Reynolds, Che, and Ali, 2010; Reynolds and Ali, 2008; Che, 2009]. A combination of ten independent runs for each landscape was tested on each of the homogeneous and heterogeneous topologies. Fifty runs will be conducted as a total for each of the landscapes. Three population sizes were selected (50, 75, and 100) to test the stability and predictability of the homogeneous and heterogeneous topologies.

The overall performance comparison among the topologies is given in the next section. We conducted t-tests to prove that significant changes in performance were introduced by the

heterogeneous topologies. The t-tests are given in section 3. Finally, section 4 gives a conclusion of the chapter.

7.2 Overall Performance Comparison.

To measure the performance of the homogeneous and heterogeneous topologies, we collected the average number of generations needed to solve the cones world problems. We also collected the standard deviation of the number of generations needed to solve the problems to help us in measuring the predictability of the homogenous and heterogeneous performance in the fixed, periodic, and chaotic problems classes.

The trend of the average number of generations needed to solve the problems with the three tested population sizes (50, 75, and 100) look similar as shown in Figure 7-1, Figure 7-2, and Figure 7-3. The only noticeable difference is the y-axis scale that represents the average number of generations needed to solve the problems for a given entropy. When the population size increases, the number of generations needed to solve the problems decreases in homogeneous and heterogeneous topologies.

Homogeneous topologies are more effective in terms of the number of generations needed to solve problems with entropies 1.4, 1.8, 2.2, 2.6, and 3.1 as shown in the blue colored rows in Tables 7-1, 7-2, and 7-33. For the entropies 3.2, 3.3, 3.4, 3.6, 3.7, 3.8, and 3.9, heterogeneous topologies outperformed the homogeneous topologies as shown in the green colored rows in Tables 7-1, 7-2, and 7-3. For the fixed category of problems (alpha = 1.4, 1.8, 2.2, and 2.6) as given in Table 7-1, 7-2, and 7-3 the homogeneous topologies approach fared better than the heterogeneous approach in terms of number of generations needed to solve the problem. Tables 7-1, 7-2, and 7-3 show the results for the periodic category of problems (alpha =3.2, 3.3, and 3.4). Heterogeneous networks were much more effective when the environment

had multiple generators (two in this case). This class (periodic) is characterized by the essential superposition of two static classes. Therefore, it makes sense that the heterogeneous approach that interweaves of subset of homogeneous topologies will be a good fit for classes of problems like this.

As more iterations (generators) are added the landscape pattern becomes more chaotic. The results of chaotic category of problems (alpha = 3.6, 3.7, 3.8, and 3.9) are shown in Table 7-1, 7-2, and 7-3. **From these Tables, we conclude that as the environment became more heterogeneous, the heterogeneous network began to outperform the homogeneous ones.** Tables 7-1, 7-2, and 7-3 show that the heterogeneous topology had a fewer number of generations used to solve the chaotic problems in all 50 runs for all of the chaotic entropies. As a summary, the heterogeneous topologies started to outperform the homogeneous topologies in terms of the number of generations needed to solve a problem in entropies higher than 3.1. So, the value 3.1 can be considered a tipping point.

The fixed problems class can be viewed is one layer of complexity and while multiple networks (heterogeneous topologies) offer an additional layer of complexity to a simple problem, a layer that just reduces the efficiency of solution in single layer problems. On the other side, the periodic and chaotic problems have multiple layers of complexity and multiple networks can increase the focus on each one of those layers separately to reduce the problem complexity and find the solution in less number of generations. Homogeneous topologies are more suitable for fixed problems class but heterogeneous topologies operate better in the periodic and chaotic problems classes.

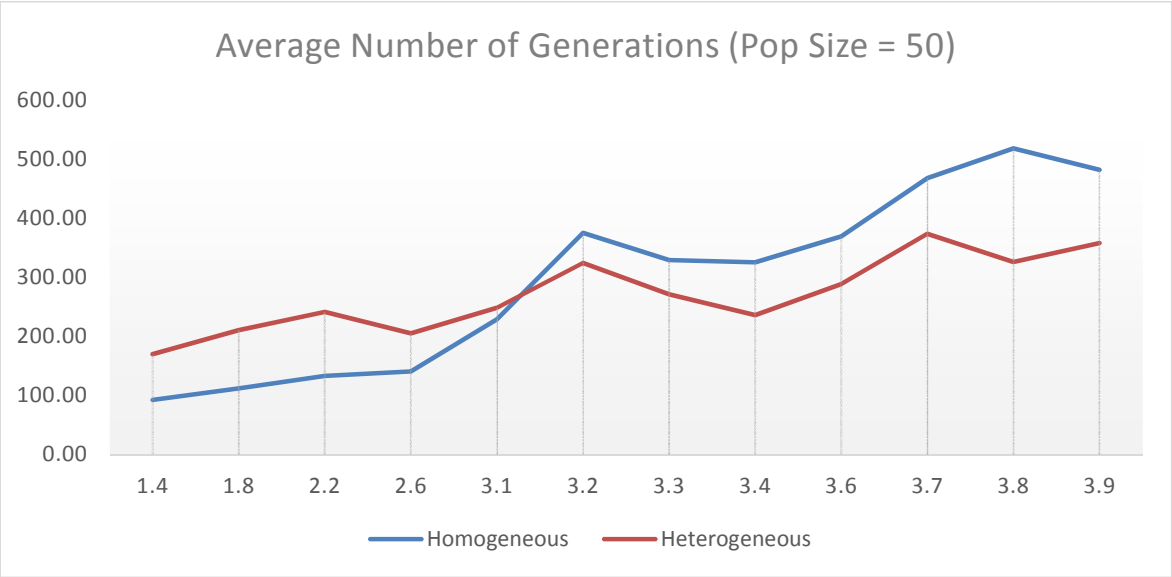


Figure 7-1: Average Number of Generations of the Homogeneous and Heterogeneous topologies for Population Size 50.

Average Number of Generations (Population Size = 50)			
	A-Value	Homogeneous	Heterogeneous
Fixed	1.4	92.38	170.20
	1.8	112.10	210.72
	2.2	133.48	241.88
	2.6	140.92	205.14
Periodic	3.1	229.08	248.90
	3.2	375.28	324.78
	3.3	329.68	271.70
	3.4	325.60	236.38
Chaotic	3.6	369.40	288.84
	3.7	468.44	373.68
	3.8	518.82	326.00
	3.9	482.06	358.46

Table 7-1: The Performance Comparison of the Homogeneous and Heterogeneous Social Fabric Topologies for Population Size 50.

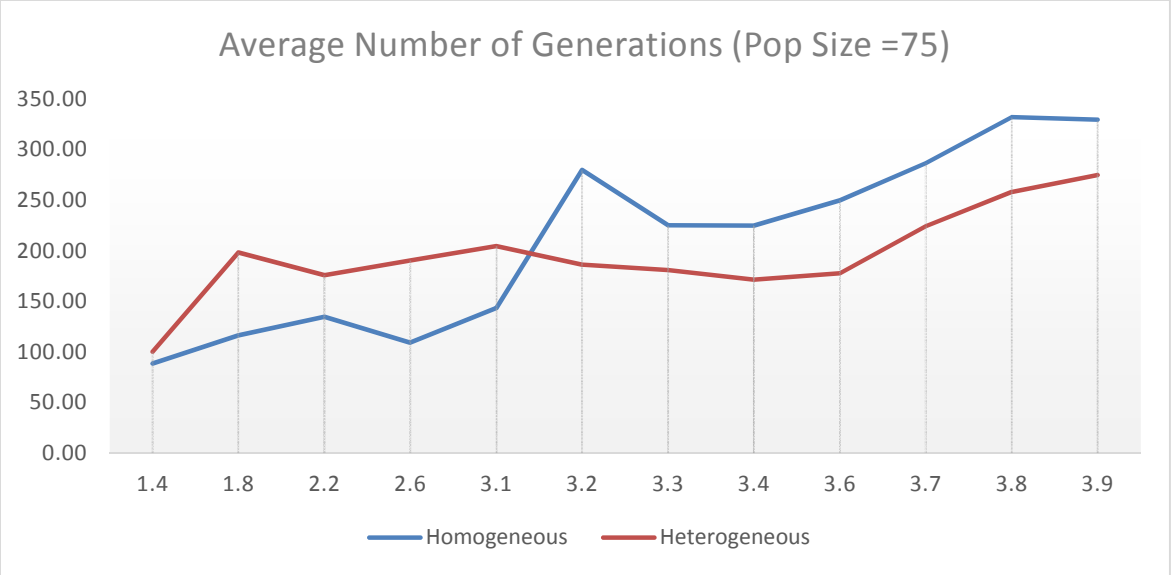


Figure 7-2: Average Number of Generations of the Homogeneous and Heterogeneous topologies for Population Size 75.

Average Number of Generations (Population Size = 75)			
	A-Value	Homogeneous	Heterogeneous
Fixed	1.4	88.64	100.34
	1.8	116.52	198.24
	2.2	134.70	175.94
	2.6	109.10	190.26
Periodic	3.1	143.24	204.50
	3.2	279.82	186.10
	3.3	225.10	181.02
	3.4	224.86	171.44
Chaotic	3.6	249.86	177.60
	3.7	286.44	224.18
	3.8	332.14	258.14
	3.9	329.46	274.62

Table 7-2: The Performance Comparison of the Homogeneous and Heterogeneous Social Fabric Topologies for Population Size 75.

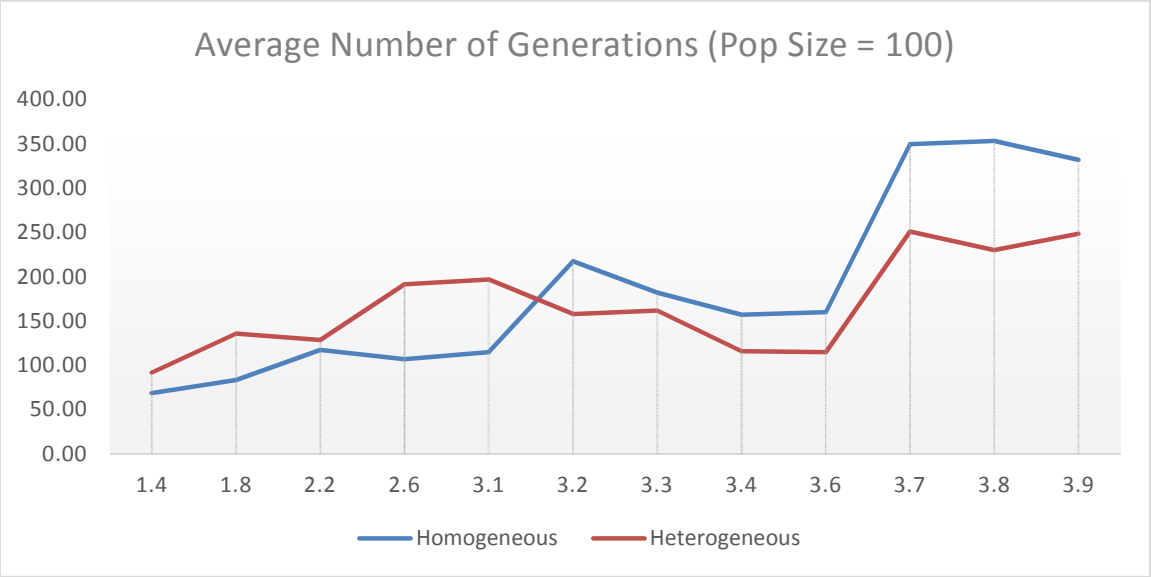


Figure 7-3: Average Number of Generations of the Homogeneous and Heterogeneous topologies for Population Size 100.

Average Number of Generations (Population Size = 100)			
	A-Value	Homogeneous	Heterogeneous
Fixed	1.4	68.50	91.86
	1.8	83.32	135.98
	2.2	117.56	128.58
	2.6	106.82	191.7
Periodic	3.1	114.82	196.82
	3.2	217.48	157.92
	3.3	182.20	162.08
	3.4	157.32	115.92
Chaotic	3.6	160.14	115.08
	3.7	349.74	251.16
	3.8	353.28	230.16
	3.9	331.94	248.64

Table 7-3: The Performance Comparison of the Homogeneous and Heterogeneous Social Fabric Topologies for Population Size 100.

Standard Deviation			
Population Size	Problem Class	Homogeneous	Heterogeneous
50	Fixed	51.91	119.25
	Periodic	247.16	169.29
	Chaotic	345.64	230.75
75	Fixed	58.82	144.28
	Periodic	170.55	107.25
	Chaotic	237.61	189.16
100	Fixed	75.19	166.12
	Periodic	128.31	89.54
	Chaotic	218.90	161.50

Table 7-4: The Standard Deviation of the Homogeneous and Heterogeneous Social Fabric Topologies for Population Sizes 50, 75, and 100.

Table 7-4 shows the standard deviation of the number of generations needed to solve each problem class for the three different population sizes. As shown in the blue rows in this table, the standard deviation of the homogeneous topologies is less than the standard deviation of the heterogeneous topologies for fixed problems classes. For the periodic and chaotic problems classes, the heterogeneous topologies were more predictable than the homogeneous topologies. The green rows of Table 7-4 show that the standard deviation of the heterogeneous topologies is less than the standard deviation of the homogeneous topologies in periodic and chaotic problems classes.

7.3 Learning Curves.

We randomly selected learning curves from homogeneous and heterogeneous topologies for each of the problem classes: fixed, periodic, and chaotic. Figures 7-4, 7-5, and 7-6 show the learning curves for fixed, periodic, and chaotic problems categories in homogeneous topologies, respectively. In heterogeneous topologies, Figures 7-7, 7-8, and 7-9 give sample learning curves for the fixed, periodic, and chaotic problems categories, respectively. The reason for showing those graphs here is to show that both approaches, homogeneous and heterogeneous, are working properly towards finding the solution in different problems classes.

These curves indicate a characteristic of learning within multi-layered complex systems, incrementally. Patterns that emerge at a lower level are combined to produce patterns at the next highest level. Thus, emergent patterns can serve as bridges from lower levels to higher levels in the emergence of a complex system. This was illustrated by Kinniard Heether in the development of a controller for a 3D car racing game [Loiacono et. al, 2008].

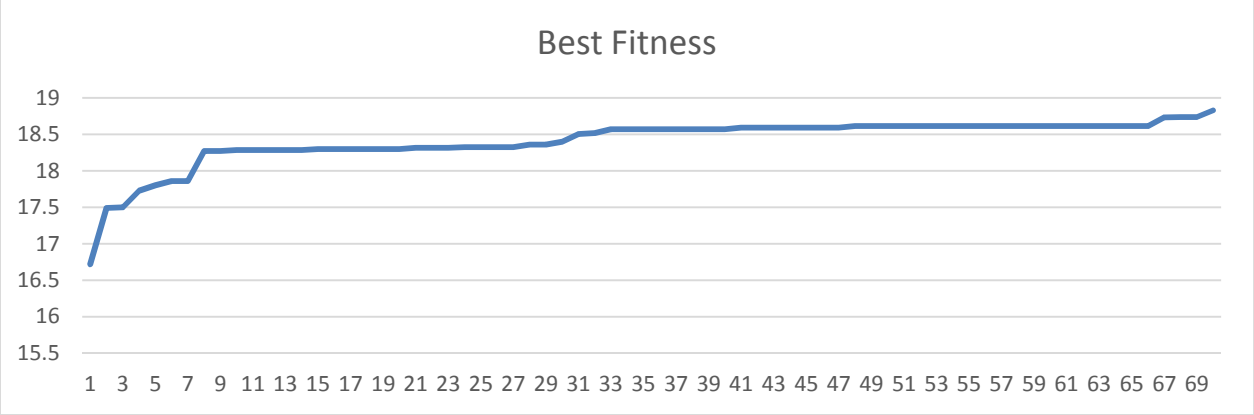


Figure 7-4: Homogeneous Learning Curve A=1.4.

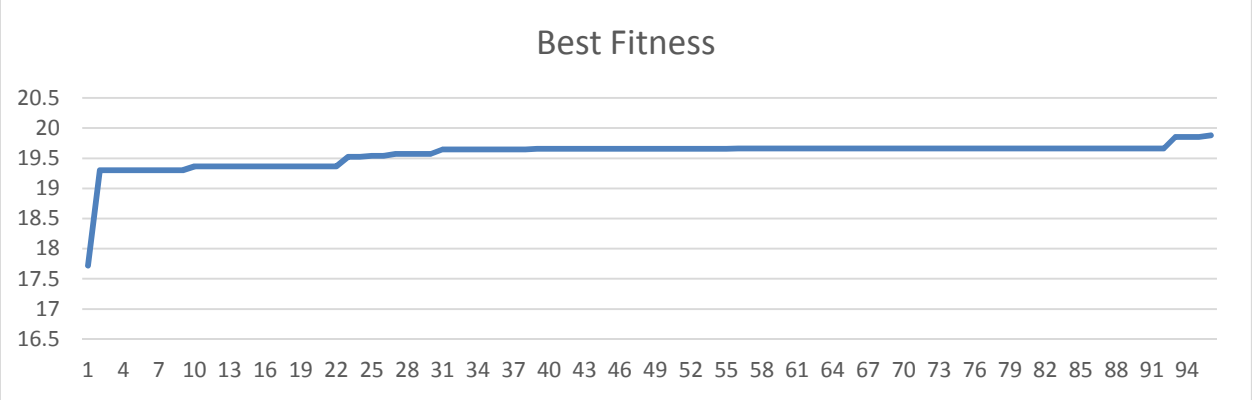


Figure 7-5: Homogeneous Learning Curve A=3.2.

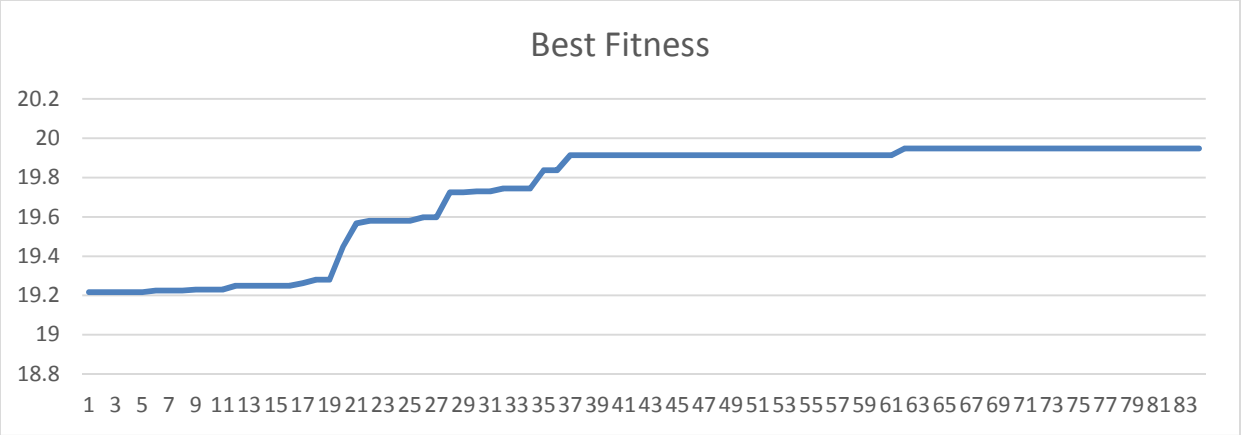


Figure 7-6: Homogeneous Learning Curve A=3.8.

Figure 7-7: Heterogeneous Learning Curve A=1.4.

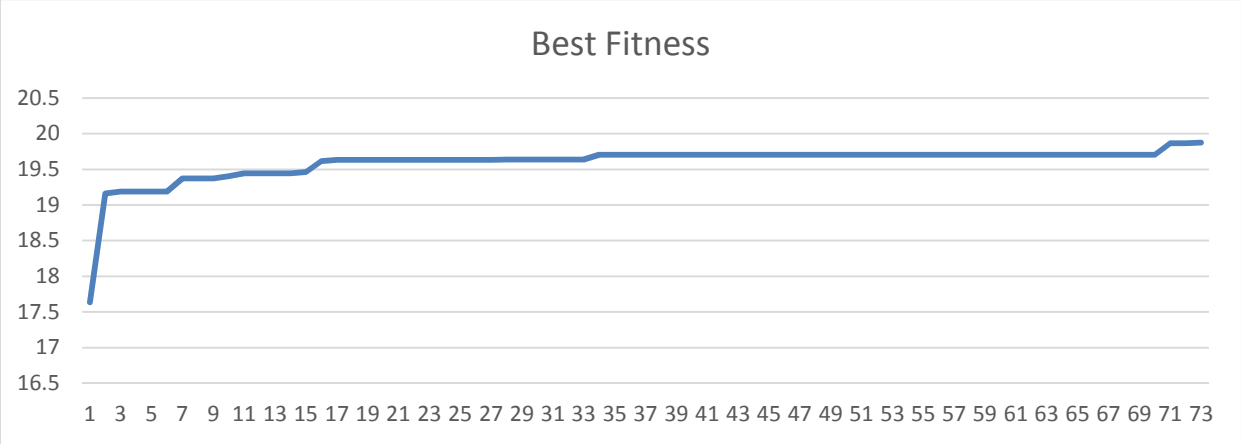


Figure 7-8: Heterogeneous Learning Curve A=3.2.

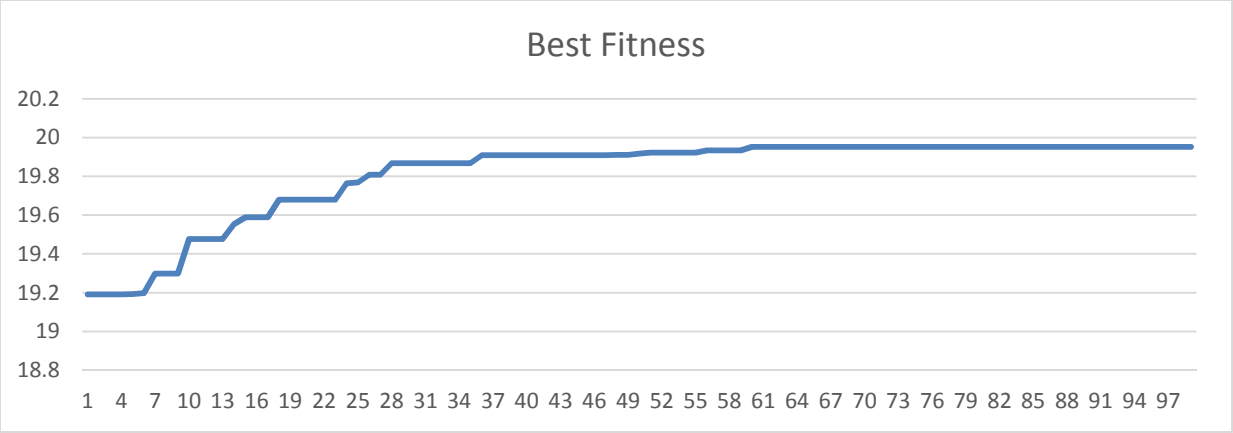


Figure 7-9: Heterogeneous Learning Curve A=3.8.

7.4 T-tests.

We conducted a t-test to test the influence of the heterogeneous topologies on the cultural algorithm performance and their effect on the knowledge sources within the three complexity categories (fixed, periodic, and chaotic categories). We suggested the following null hypothesis:

"Heterogeneous Topology doesn't affect the performance of the cultural algorithm over the homogeneous topologies". Tables 7-5, 7-6, and 7-7 give the t-test results of the homogeneous and heterogeneous topologies for the population sizes 50, 75, and 100, respectively.

	A-Value	Homogeneous (σ)	Heterogeneous (σ)	Calculated t-value	Two tailed table t-value	
Fixed	1.4	37.24	60.86	7.712312	1.98	Reject
	1.8	29.41	102.96	6.512269	1.98	Reject
	2.2	21.92	102.42	7.318343	1.98	Reject
	2.6	90.90	198.78	2.077496	1.98	Reject
Periodic	3.1	170.41	154.21	0.609803	1.98	Accept
	3.2	379.88	257.11	0.778464	1.98	Accept
	3.3	239.43	160.49	1.422361	1.98	Accept
	3.4	212.87	121.49	2.573982	1.98	Reject
Chaotic	3.6	423.24	314.08	1.080825	1.98	Accept
	3.7	359.44	290.78	1.449289	1.98	Accept
	3.8	306.92	93.35	4.250026	1.98	Reject
	3.9	386.87	198.20	2.010612	1.98	Reject

Table 7-5: T-tests of Homogeneous and Heterogeneous Topologies of Population Size 50.

	A-Value	Homogeneous (σ)	Heterogeneous (σ)	Calculated t-value	Two tailed table t-value	
Fixed	1.4	54.50	41.59	1.206878	1.98	Accept
	1.8	70.92	158.44	3.328864	1.98	Reject
	2.2	44.58	84.79	3.044207	1.98	Reject
	2.6	71.32	239.41	2.297323	1.98	Reject
Periodic	3.1	108.76	147.15	2.367365	1.98	Reject
	3.2	283.60	128.20	2.129276	1.98	Reject
	3.3	126.25	93.61	1.983203	1.98	Reject

	3.4	134.49	84.02	2.382002	1.98	Reject
Chaotic	3.6	266.87	165.32	1.627612	1.98	Accept
	3.7	240.85	195.38	1.419515	1.98	Accept
	3.8	190.71	222.91	1.783672	1.98	Accept
	3.9	311.18	218.82	1.019343	1.98	Accept

Table 7-6: T-tests of Homogeneous and Heterogeneous Topologies of Population Size 75.

	A-Value	Homogeneous (σ)	Heterogeneous (σ)	Calculated t-value	Two tailed table t-value	
Fixed	1.4	45.40	68.78	2.004245	1.98	Reject
	1.8	45.98	218.49	1.667722	1.98	Accept
	2.2	64.80	44.78	0.989258	1.98	Accept
	2.6	128.94	265.00	2.036616	1.98	Reject
Periodic	3.1	88.75	106.93	4.172487	1.98	Reject
	3.2	212.11	120.56	1.726211	1.98	Accept
	3.3	105.84	59.03	1.173946	1.98	Accept
	3.4	84.55	67.17	2.711094	1.98	Reject
Chaotic	3.6	151.08	105.21	1.730662	1.98	Accept
	3.7	199.34	213.43	2.386892	1.98	Reject
	3.8	250.50	128.75	3.091001	1.98	Reject
	3.9	264.44	186.93	1.81886	1.98	Accept

Table 7-7: T-tests of Homogeneous and Heterogeneous Topologies of Population Size 100.

From Table 7-5, the t-test rejects the null hypothesis for the fixed problem class entropies (1.4, 1.8, 2.2, and 2.6) and there is a significant statistical difference between the heterogeneous and homogeneous topologies. Actually, the homogeneous topologies were clearly more effective in the fixed class problem in population size 50. In the periodic class, there was a significance difference between the homogeneous and heterogeneous topologies for the highest entropy in this class (alpha = 3.4). The last two rows of the Table 7-5 indicate that there is a significance difference in the performance of the homogeneous and heterogeneous topologies for the highest two entropies of the chaotic problem classes (3.8 and 3.9). Those results were published by Reynolds and Gawasmeh in 2012 [Reynolds and Gawasmeh, 2012].

When the population size increases as shown in Tables 7-6 and 7-7, the pattern of reject and accept of the null hypothesis becomes less clear. The issue resides in the Belief Space, it is completely blind to the interactions between the individuals on the landscape. So, the Belief Space provided Influence and Acceptance Functions that don't distinguish between the individuals according to their networks and that makes the individuals look like a single clan as shown in Figure 7-10. This issue demonstrated the need of subcultures and that includes the need to update the Belief Space, Influence Function, and Acceptance Function.

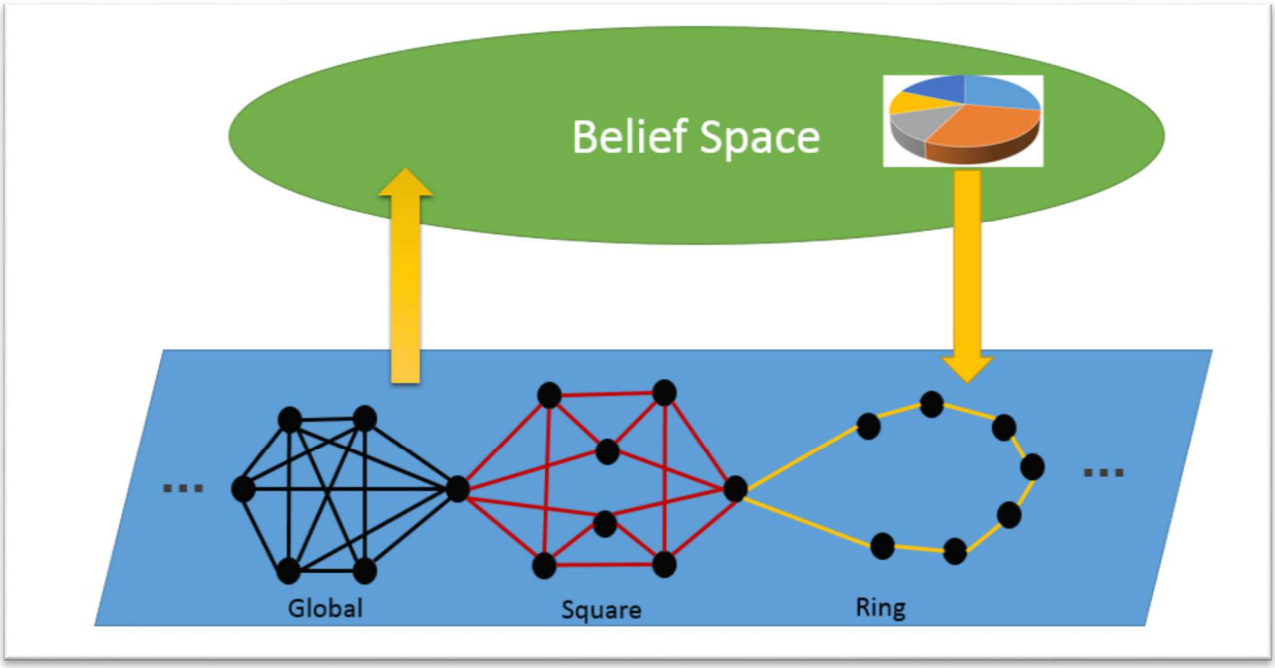


Figure 7-10: Heterogeneous Model Big Picture.

7.5 Social Metrics.

In this section we examine the difference in performance in terms of their effects on the movement of information through the social fabric. We will use two basic metrics:

1). Social tension measures the distance between the network neighbors over the landscape. If the Social Tension increases, then the individuals' experiences will be more variable. In this section, we are going to compare the average Social Tension of homogeneous

and heterogeneous topologies with population sizes 50, 75, and 100 as shown in figures 7-11, 7-12, and 7-13 respectively.

A higher Social Tension means that neighbors that are adjacent in the network can get information about more diverse places on the landscape and more dissemination of information. Heterogeneous topologies could achieve higher Social Tension in periodic and chaotic problems than the Homogeneous topologies for population sizes 50, as shown in Figure 7-11. Figures 7-12 and 7-13 shows that Social Tension values were close for homogeneous and heterogeneous topologies in population sizes 75 and 100. In other words as population size increases the differences in experiences were reduced because the density of individuals has decreased over the landscape and there are less differences between individuals for the same landscape.

2). The Innovation Cost Index's goal is to compute the needed effort to allow the minority knowledge source to win over the majority knowledge source. This will happen when a Knowledge Source with few individuals finds a new promising region and as a result has an average performance for that time period that is high enough to overcome the sum of the majority influences.

A lower Innovation Cost Index means that innovations that are produced are more productive than with a high Innovation Cost Index which means that there is not much difference between them. Heterogeneous topologies achieved a lower Innovation Cost Index in periodic and chaotic classes for all experimented population sizes as shown in figures 7-14, 7-15, and 7-16. On the other side, Homogeneous topologies could collect a lower Innovation Cost Index for the fixed class problems. The Innovation Cost Index and Social Tension social metrics support the measured performance results in the previous sections.

As population size increases the Innovation Cost for the same landscapes increases because of the fact that there are more individuals associated with the majority than the minority. In some sense, the minority will need to produce a more convincing demonstration of a new area in order for the innovation to proceed.

Taken together, these indexes suggest that increased population size is not necessarily a good thing in a non-sedentary population. In other words, too many people weigh down the exploration activities of the population over time. On the one hand variability in experience is reduced, and momentum for change as expressed in terms of Innovation Cost is dampened. This fits in with observation by Engelbrecht [2014].

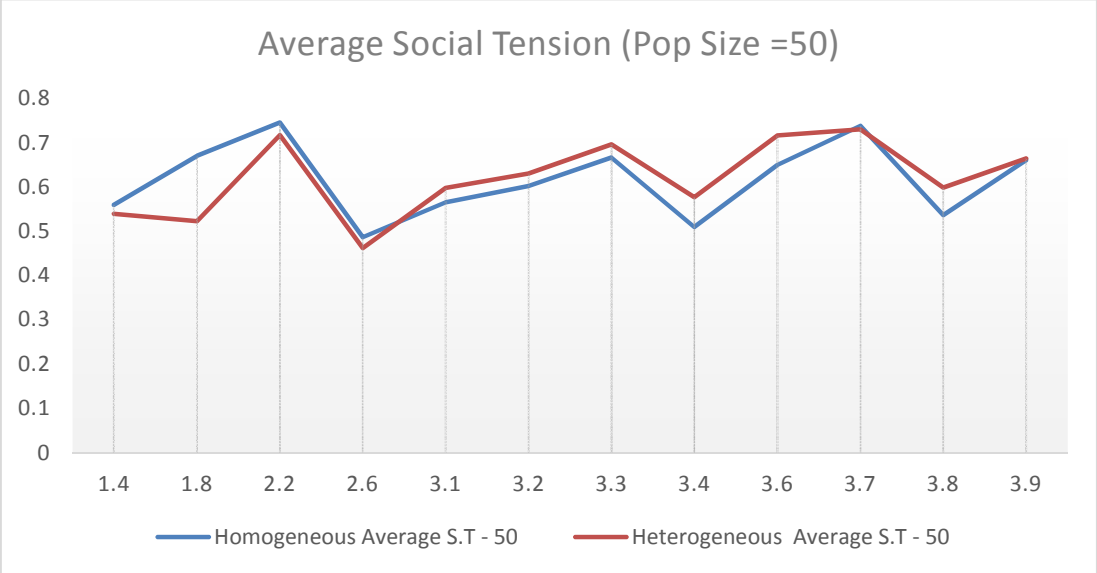


Figure 7-11: Average Social Tension in Population Size 50.

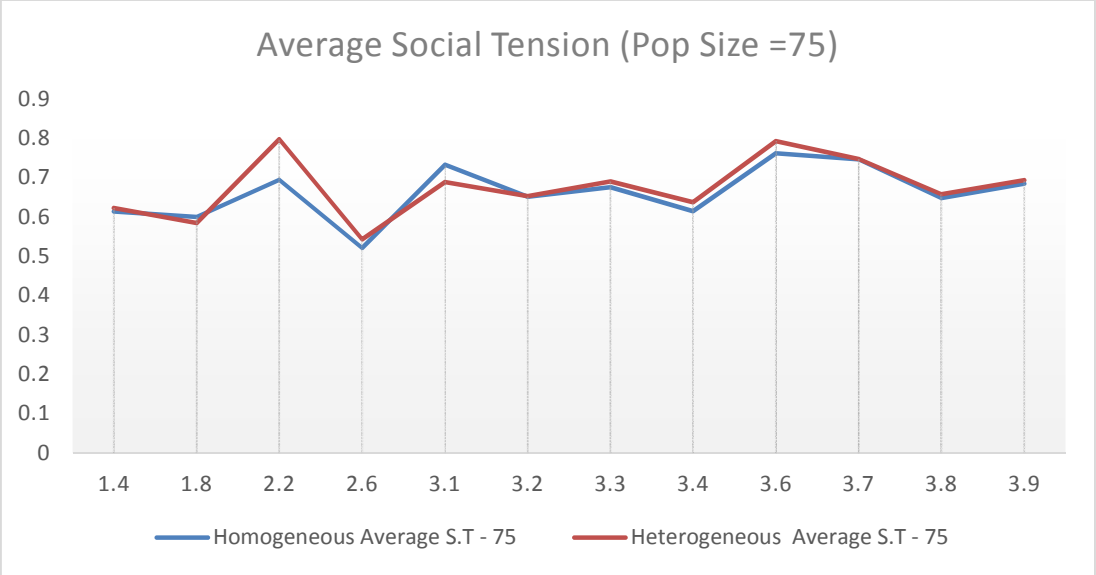


Figure 7-12: Average Social Tension in Population Size 75.

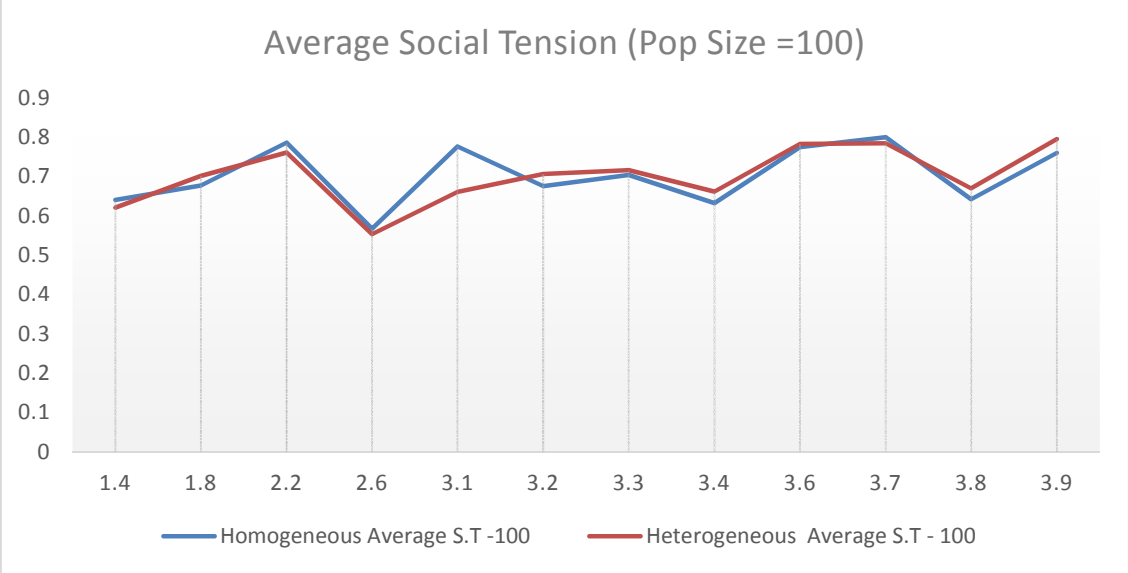


Figure 7-13: Average Social Tension in Population Size 100.

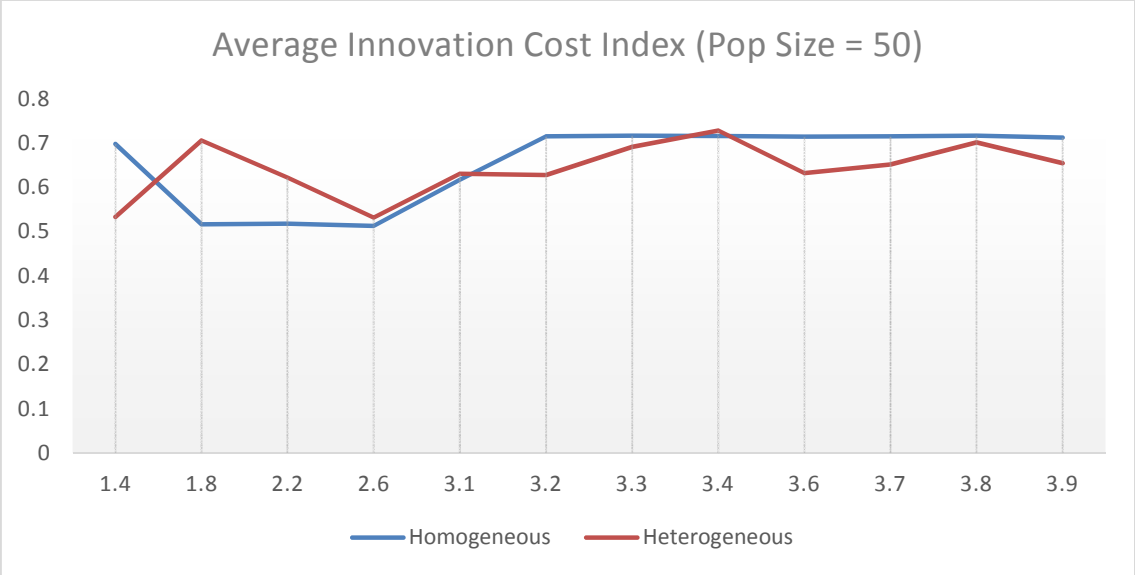


Figure 7-14: Average Innovation Cost Index in Population Size 50.

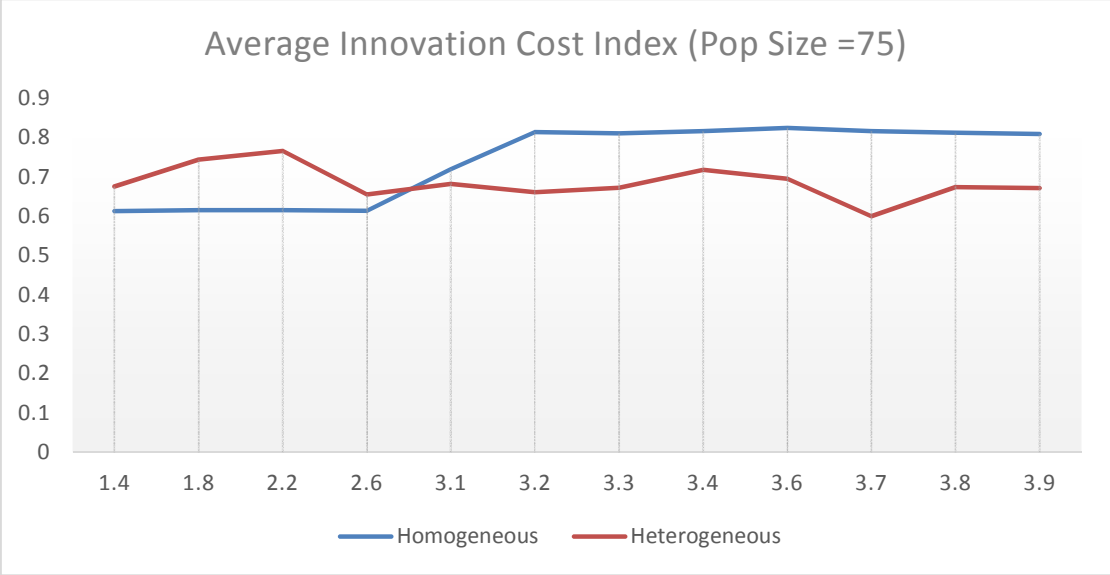


Figure 7-15: Average Innovation Cost Index in Population Size 75.

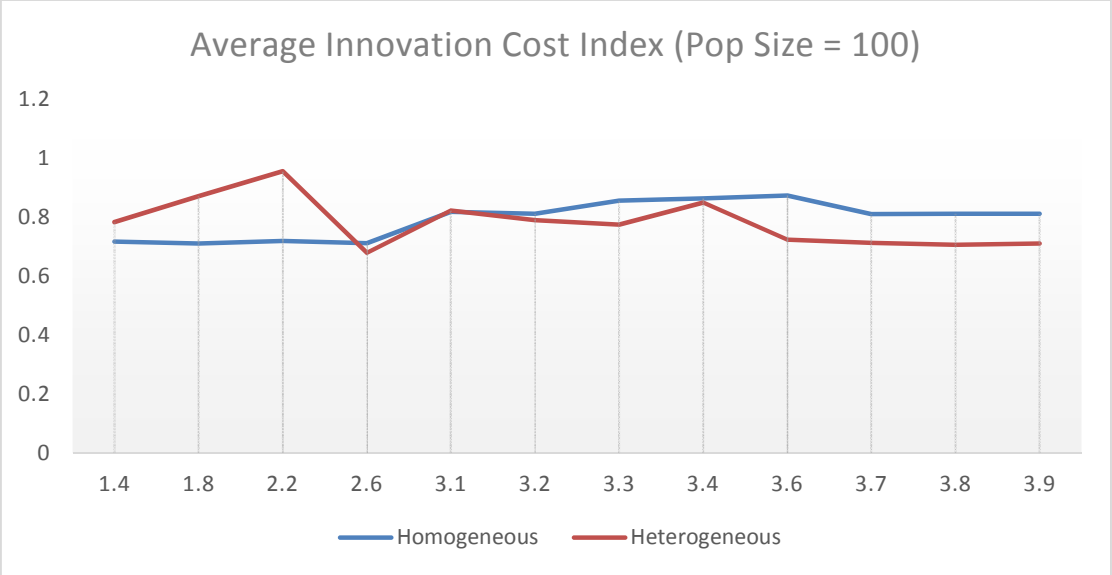


Figure 7-16: Average Innovation Cost Index in Population Size 100.

7.6 Conclusions.

In this chapter, the heterogeneous and the homogeneous topologies were compared. For static problems with predictable patterns of distribution, the homogeneous networks were the most effective. As more and more static problems were blended together to increase complexity, the heterogeneous approach became the dominant strategy in terms of the number of problems solved in the given time frame. This was because there was work for more than one topology to perform. The approach effectively allowed a team of networks to work on the problem space, each exploiting those patterns most suited for it.

So, for static problems with fixed cone distribution parameters the homogeneous topologies were the most effective in solving them. On the other hand, the heterogeneous topologies were more effective in solving problems of periodic and chaotic classes.

In addition, while increased population sizes would appear to improve performance, here they tended to dampen performance since they reduced the variability in performance and increased the cost for innovation.

The question is then, how can we increase the size of our population while at the same time reducing the impact of increased population size on performance? When we conducted the t-test on different population sizes, we noticed the need of dividing the Population Space into subcultures and updating the Belief Space, the Influence function, and the Acceptance Function. In other words, subcultures might be the answer to our problem. If the population did increase, but was broken into sub-populations of smaller size, then the metrics in those sub-populations might better support exploration of the population. The next chapter introduces the subcultures and gives more comparable results to the heterogeneous and homogeneous topologies.

CHAPTER 8 SUBCULTURES MODEL RESULTS

8.1 Introduction.

One of the manifestations of cultural systems, particularly in recent times, has been the emergence of subcultures. Each subculture is characterized by a topology and a set of concepts of information that are passed through the topology. In this chapter, we employ a single Population Space with multiple topologies. Each topology is viewed as a subculture. The Belief Space keeps information about the performance of each subculture as a combination of a topology and a knowledge source. The topologies compete with each other to find patterns in a problem landscape. The more generators that combined to produce a landscape, the more underlying patterns there are for each topology to undercover. This subcultures social fabric is compared with the performance of systems that use the homogeneous and heterogeneous social fabrics.

One of the questions that we wish to answer here is whether subcultures are indeed able to maintain the exploratory tendencies of a population, as the size of that population grows. One of the things that we observed in the previous chapter was that for a single population, the variability of individual experience is reduced and the Innovation Cost is increased as population size increases. If we are to use subcultures to subdivide the population into networked subgroups, will that help to ameliorate the negative impact of population growth on innovation?

In this chapter, the use of subcultures topologies with the homogeneous and heterogeneous topologies for the social fabric will be compared in terms of their relative performance. Their effect on the knowledge sources within the three complexity categories will also be tested (fixed, periodic, and chaotic categories). Four entropy points were selected from each problem class. The entropy values had been chosen from the fixed, periodic, and chaotic categories, respectively, (1.4, 1.8, 2.2, and 2.6), (3.1, 3.2, 3.3, and 3.4), and (3.6, 3.7, 3.8, and 3.9). Again, the best performing homogeneous configuration is used in the comparison here.

The chapter includes: overall performance comparison among the topologies, the subcultures predictability, social metrics comparison to show effect of the heterogeneous topology on the Social Tension and Innovation Cost Index measures, and, we conducted t-tests to prove the significant changes introduced by the subcultures topologies.

8.2 Overall Performance Comparison.

Tables 8-1, 8-2, and 8-3 summarize the experiments results. Each row represents the entropy of the problem. The homogeneous, heterogeneous, and subcultures columns represent the individuals' interaction approach that was used to weave the social fabric. The Topology column specifies the topology that has been used in the runs. The values in Columns 2, 3, and 4 give the average number of generations needed to solve a problem in 50 runs.

For Static problems the heterogeneous and subcultures versions were in general outperformed by the homogeneous topologies, as shown in the blue rows of Tables 8-1, 8-2, and 8-3. The presence of just a single generator function did not require the additional complexity of either a heterogeneous topology or the emergence of subcultures. Their presence in an environment produced by just one generator process does not contribute to the effectiveness of the optimization process here.

For the periodic class, problems with (A= 3.2, 3.3, and 3.4) as shown in Tables 8-1, 8-2, and 8-3, the subcultures topologies outperformed all of the other topologies in terms of the average number of steps needed to solve the problem. For periodic class problems with entropy 3.1, subcultures outperformed the heterogeneous topologies as shown in Figures 8-2, 8-4, and 8-6. And, subcultures topology was very close to the homogeneous topologies' average number of generations. In fact, the subcultures were more effective than the homogeneous topologies in solving problems with entropy 3.1 for population size 75 as shown in Figure 8-3 but not for

population sizes 50 and 100 as shown in Figures 8-1 and 8-5. Overall, subcultures were the most effective topology in solving problems from the periodic class. This suggests that as the number of generators for the problem increase, so can the utility of subcultures to track them.

In chaotic class problems, when the number of generators becomes large, the utility of subcultures is increased. The presence of subcultures in this chaotic environment impacts the number of generations needed to solve a problem successfully. The last four rows of Tables 8-1, 8-2, and 8-3 show that subcultures were the most effective topology in solving chaotic class problems in terms of the number of generations. It may be that the presence of subcultures allows the system to focus on a solution once it is in the ballpark due to its ability to support individual variability and reduce Innovation Costs.

Figures 8-1, 8-3, and 8-5 show the average number of generations needed to solve problems for entropy values 1.4, 1.8, 2.2, 2.6 , 3.1, 3.2, 3.3, 3.4, 3.6, 3.7, 3.8, and 3.9 using homogeneous and subcultures topologies. From the figures, we can see that entropy 3.1 is a critical point where subcultures start to outperform the homogeneous topologies in solving the given problems. When the number of layers of complexity increase, multiple networks are needed to regain the focus on parts of the landscape instead of exploring the landscape as a single unit. Homogoeneous topologies explore the landscape as a coherent unit but subcultures topology explores the landscape with small groups of individuals the form subcultures over parts of the landscape. Therefore, subcultures topology can solve problems efficiently when more layers of complexity added to the landscape.

The average number of generations needed to solve problems with twelve entropies from fixed class problems (1.4, 1.8, 2.2, and 2.6), periodic class problems (3.1, 3.2, 3.3, and 3.4), and chaotic class problems (3.6, 3.7, 3.8, and 3.9) for homogeneous, heterogeneous, and subcultures

topologies is given in Figures 8-2, 8-4, and 8-6. The figures suggest a few points: (1) entropy 3.1 is still a critical (or tipping) point as observed earlier because heterogeneous and subcultures topologies outperform the homogeneous topologies after that point forward for population sizes 50, 75 and 100; (2) subcultures outperform the heterogeneous topologies almost for every given entropy; (3) subcultures and heterogeneous topologies outperform the homogeneous topology in chaotic class problems; (4) and the trend of the average number of generations needed to solve the problems with the three tested population sizes (50, 75, and 100) look similar with one noticeable difference in the y-axis scale that represents the average number of generations needed to solve the problems of a given entropy, when the population size increases, the number of generations needed to solve the problems decreases in homogeneous and heterogeneous topologies.

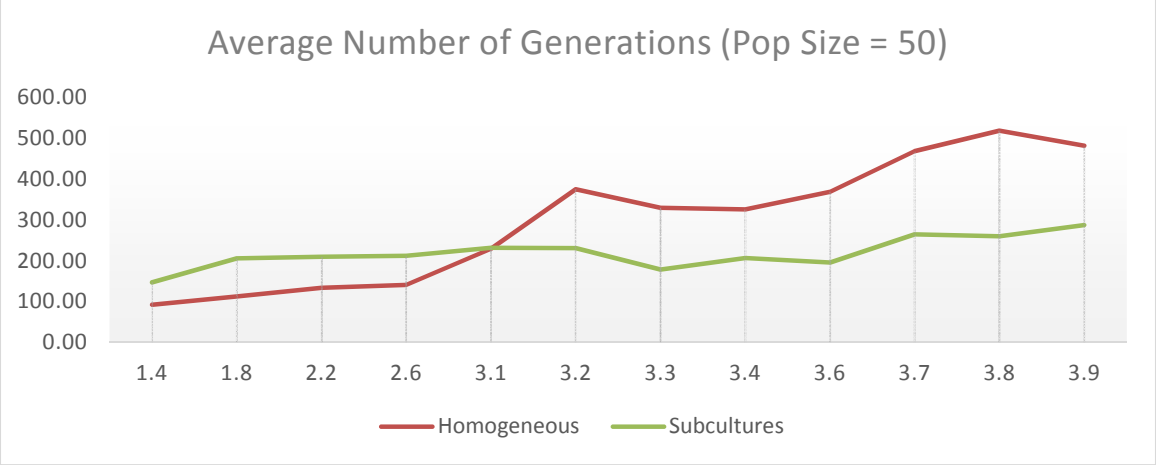


Figure 8-1: The Average Number of Generations Comparison of Homogeneous, and Subcultures Topologies for Population Size 50.

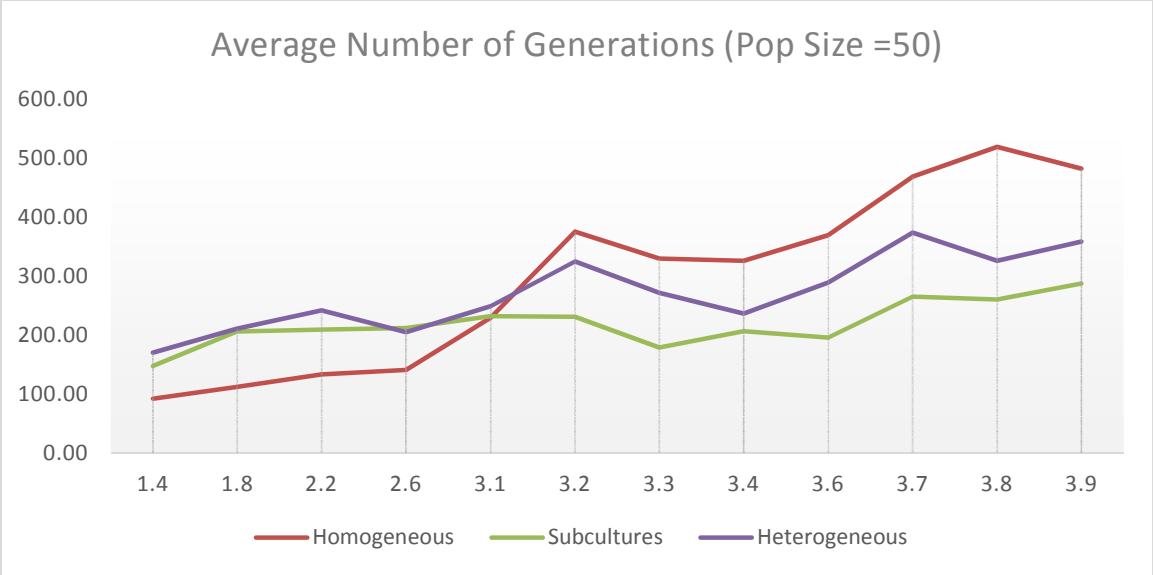


Figure 8-2: The Average Number of Generations Comparison of Homogeneous, Heterogeneous, and Subcultures Topologies for Population Size 50.

		Average Number of Generations (Population Size = 50)		
	A-Value	Homogeneous	Heterogeneous	Subcultures
Fixed	1.4	92.38	170.20	147.36
	1.8	112.10	210.72	206.16
	2.2	133.48	241.88	209.48
	2.6	140.92	205.14	211.90
Periodic	3.1	229.08	248.90	231.76
	3.2	375.28	324.78	230.72
	3.3	329.68	271.70	178.72
	3.4	325.60	236.38	206.58
Chaotic	3.6	369.40	288.84	195.90
	3.7	468.44	373.68	264.88
	3.8	518.82	326.00	260.16
	3.9	482.06	358.46	287.32

Table 8-1: The Performance Comparison of the Homogeneous and Subcultures Social Fabric Topologies for Population Size 50.

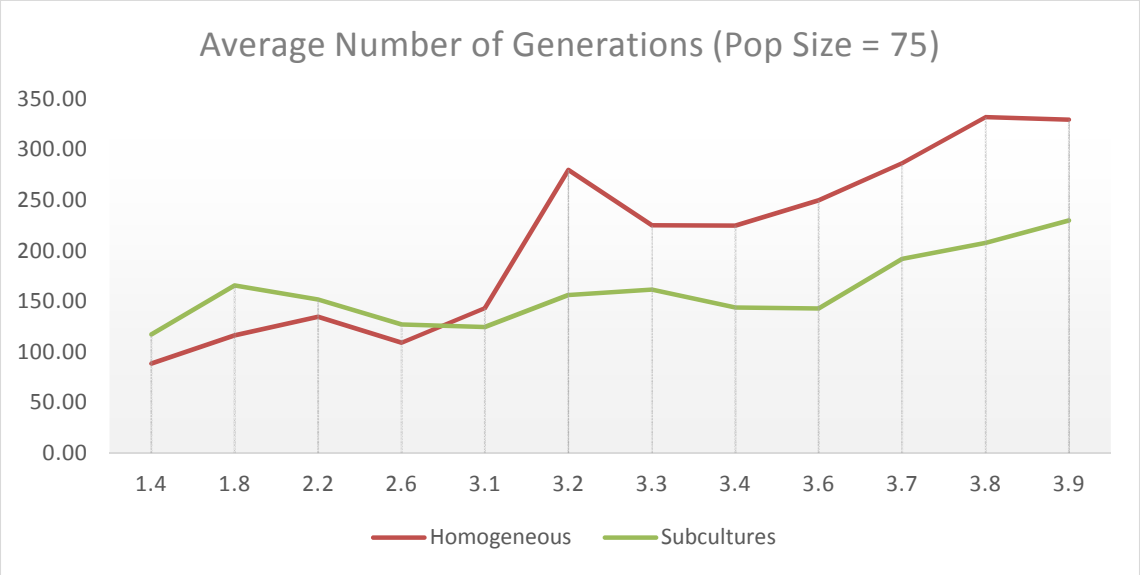


Figure 8-3: The Average Number of Generations Comparison of Homogeneous and Subcultures Topologies for Population Size 75.

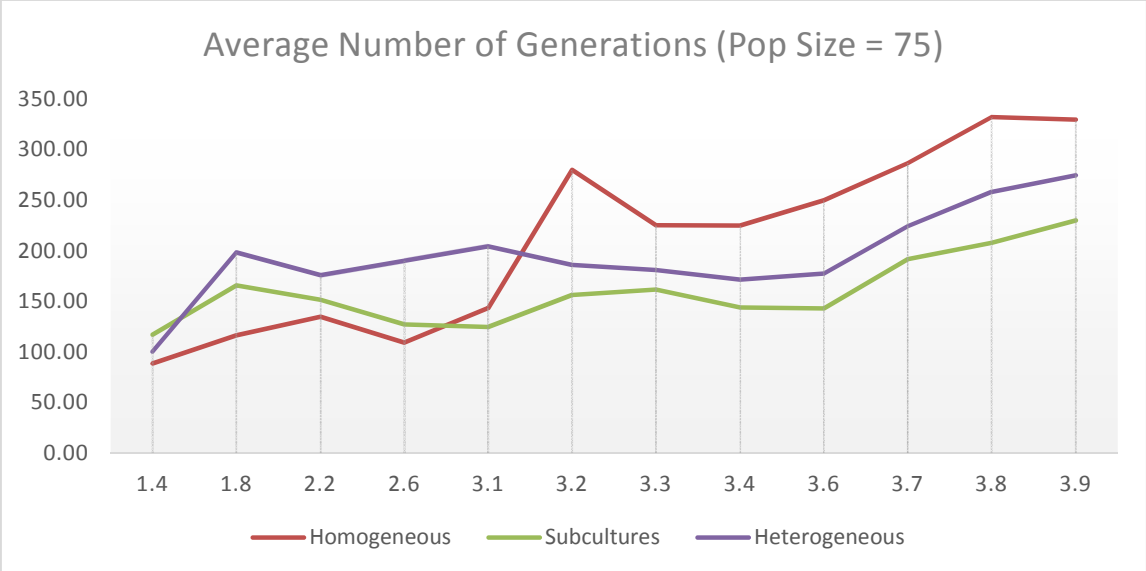


Figure 8-4: The Average Number of Generations Comparison of Homogeneous, Heterogeneous, and Subcultures Topologies for Population Size 75.

		Average Number of Generations (Population Size = 75)		
	A-Value	Homogeneous	Heterogeneous	Subcultures
Fixed	1.4	88.64	100.34	117.18
	1.8	116.52	198.24	165.74
	2.2	134.70	175.94	151.64
	2.6	109.10	190.26	127.12
Periodic	3.1	143.24	204.50	124.66
	3.2	279.82	186.10	156.14
	3.3	225.10	181.02	161.66
	3.4	224.86	171.44	143.98
Chaotic	3.6	249.86	177.60	142.86
	3.7	286.44	224.18	191.84
	3.8	332.14	258.14	207.86
	3.9	329.46	274.62	230.00

Table 8-2: The Performance Comparison of the Homogeneous and Subcultures Social Fabric Topologies for Population Size 75.

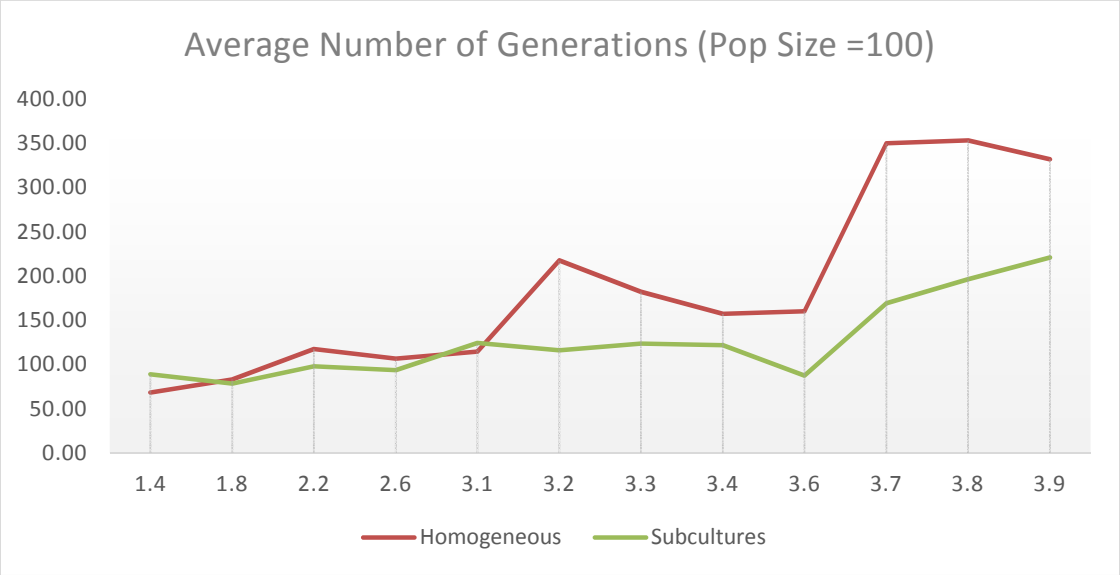


Figure 8-5: The Average Number of Generations Comparison of Homogeneous and Subcultures Topologies for Population Size 100.

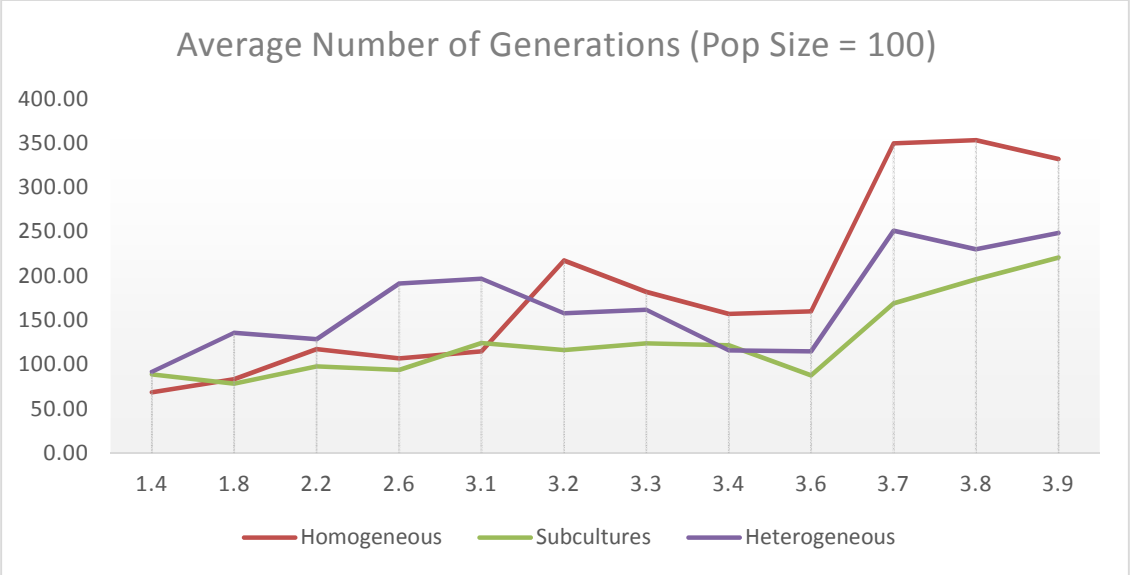


Figure 8-6: The Average Number of Generations Comparison of Homogeneous and Subcultures Topologies for Population Size 100.

		Average Number of Generations (Population Size = 100)		
	A-Value	Homogeneous	Heterogeneous	Subcultures
Fixed	1.4	68.50	91.86	88.90
	1.8	83.32	135.98	78.36
	2.2	117.56	128.58	97.96
	2.6	106.82	191.70	93.82
Periodic	3.1	114.82	196.82	124.34
	3.2	217.48	157.92	116.22
	3.3	182.20	162.08	123.84
	3.4	157.32	115.92	121.76
Chaotic	3.6	160.14	115.08	87.72
	3.7	349.74	251.16	169.3
	3.8	353.28	230.16	196.18
	3.9	331.94	248.64	220.76

Table 8-3: The Performance Comparison of the Homogeneous and Subcultures Social Fabric Topologies for Population Size 100.

In terms of time on a Core i5 desktop with 8 Giga Byte Memory, Homogenous topologies were the fastest by running 16.5 iteration per second, Heterogeneous topologies were the slowest by running 14.25 iteration per second, and Subcultures topologies were in the middle by running 15.92 iterations per second. Calculating the topology wheels and updating the individual neighbors were the main cause of the overhead in Heterogeneous and Subcultures Models.

8.3 Subcultures Predictability.

One of our hypotheses is that subcultures can favorable impact the variability of individuals in the population. We had said earlier that as population size increases then variability expressed in terms of our dispersion metric was reduced. In this section we examine that hypothesis in more detail.

Predictability can be measured by standard deviations. In this section, we collected the number of generations' standard deviations for each of the topologies (homogeneous, heterogeneous, and subcultures) on different levels; (1) entropy values level as shown in Tables 8-4, 8-5, and 8-6; (2) problems classes (fixed, periodic, and chaotic) level that is given in Table 8-7; and (3) topologies (homogeneous, heterogeneous, and subcultures) level is collected in Table 8-8.

Standard Deviation (Population Size = 50)			
A-Value	Homogeneous	Subcultures	heterogeneous
1.4	37.24	76.61	60.86
1.8	29.41	132.10	102.96
2.2	21.92	64.86	102.42
2.6	90.90	157.36	198.78
3.1	170.41	125.01	154.21
3.2	379.88	108.98	257.11
3.3	239.43	57.55	160.49
3.4	212.87	101.39	121.49
3.6	423.24	202.33	314.08

3.7	359.44	133.98	290.78
3.8	306.92	98.29	93.35
3.9	386.87	213.89	198.20

Table 8-4: The Standard Deviation Comparison of the Homogeneous, Heterogeneous, and Subcultures Social Fabric Topologies for Population Size 50.

Standard Deviation (Population Space =75)			
A-Value	Homogeneous	Subcultures	heterogeneous
1.4	54.50	65.70	41.59
1.8	70.92	92.24	158.44
2.2	44.58	70.71	84.79
2.6	71.32	92.21	239.41
3.1	108.76	76.70	147.15
3.2	283.60	89.76	128.20
3.3	126.25	63.29	93.61
3.4	134.49	77.58	84.02
3.6	266.87	148.46	165.32
3.7	240.85	138.56	195.38
3.8	190.71	67.01	222.91
3.9	311.18	150.70	218.82

Table 8-5: The Standard Deviation Comparison of the Homogeneous, Heterogeneous, and Subcultures Social Fabric Topologies for Population Size 75.

Standard Deviation (Population Size = 100)			
A-Value	Homogeneous	Subcultures	heterogeneous
1.4	45.40	90.11	68.78
1.8	45.98	25.07	218.49
2.2	64.80	43.31	44.78
2.6	128.94	118.70	265.00
3.1	88.75	101.88	106.93
3.2	212.11	70.53	120.56
3.3	105.84	80.19	59.03
3.4	84.55	67.18	67.17
3.6	151.08	62.09	105.21
3.7	199.34	144.08	213.43
3.8	250.50	165.80	128.75
3.9	264.44	161.91	186.93

Table 8-6: The Standard Deviation Comparison of the Homogeneous, Heterogeneous, and Subcultures Social Fabric Topologies for Population Size 100.

For entropies 1.4, 1.8, 2.2, and 2.6 of population sizes 50 and 75 that are represented as blue colored rows in Tables 8-4 and 8-5, homogeneous topologies were the most predictable by showing the lowest standard deviation in terms of the number of generations needed to solve fixed class problems. In the periodic and chaotic classes' problems of population sizes 50 and 75, subcultures topologies were the most predictable with the lowest standard deviation for entropies 3.1, 3.2, 3.3, 3.4, 3.6, 3.7, 3.8, and 3.9 that are shown as orange colored rows in Tables 8-4 and 8-5. In the runs with population size 100, homogeneous topologies were the most predictable for entropies 1.4 and 3.1, but the subcultures topologies were the most predictable for the rest of entropies. Heterogeneous topologies were more predictable than homogeneous topologies for entropies 3.2, 3.3, 3.4, 3.6, 3.7, 3.8, and 3.9 as shown in Tables 8-4, 8-5, and 8-6.

The value 3.1 is a critical entropy point in terms of the predictability for the following summarized facts that are noticed in Tables 8-4, 8-5, and 8-6; (1) homogeneous topologies were more predictable than heterogeneous topologies for entropies less than or equal to 3.1, (2) subcultures topologies were the most predictable for entropies greater or equal to 3.1 of population sizes 50 and 75. For population size 100, subcultures topologies were the most predictable for entropies higher than the critical entropy point 3.1, and (3) homogeneous topologies were the most predictable for entropies that were either less than 3.1 of population sizes 50 and 75, or less than or equal 3.1 of population size 100.

Standard Deviation				
Population Size	Problem Class	Homogeneous	Heterogeneous	Subcultures
50	Fixed	51.91	119.25	108.48
	Periodic	247.16	169.29	95.63
	Chaotic	345.64	230.75	159.00
75	Fixed	58.82	144.28	77.03
	Periodic	170.55	107.25	72.51
	Chaotic	237.61	189.16	124.45
100	Fixed	75.19	166.12	72.52
	Periodic	128.31	89.54	74.48
	Chaotic	218.90	161.50	138.30

Table 8-7: The Standard Deviation Comparison of the Homogeneous, Heterogeneous, and Subcultures Social Fabric Topologies for Fixed, Periodic, and Chaotic Problems Classes.

The needed number of generations' standard deviations for each of the problem classes (fixed, periodic, and chaotic) was collected in Table 8-7. Each entry in the table were collected over four entropy values for 200 runs. For fixed class problems, homogeneous topologies were the most predictable with population sizes 50 and 75. In case of population size 100, subcultures topologies were more predictable but the subcultures topologies' standard deviation values were very close to homogeneous topologies' standard deviation. For periodic and chaotic class problems, subcultures topologies were the most predictable by showing the lowest number of generations' standard deviations over the ones of homogeneous and heterogeneous topologies. Heterogeneous topologies were more predictable than homogenous topologies for the periodic and chaotic class problems.

Standard Deviation (All Runs)			
	Homogeneous	heterogeneous	Subcultures
Population size = 50	280.62	183.87	124.42
Population size = 75	186.12	150.82	95.68
Population size = 100	172.74	144.43	103.45

Table 8-8: The Standard Deviation Comparison of the Homogeneous, Heterogeneous, and Subcultures Social Fabric Topologies for All Runs.

Table 8-8 shows the standard deviations of all runs for homogeneous, heterogeneous, and subculture topologies using the population sizes 50, 75, and 100. As shown in the table, heterogeneous topologies were more predictable than the homogenous topologies for all runs of all entropies that we used in our experiments. The most predictable of all was the subcultures as the subcultures standard deviations of the number of generations were the lowest among all topologies.

From table 8-8 we see that as population size is increased in both homogeneous and heterogeneous topologies, standard deviation is reduced accordingly as we go from 50 to 100. Not so with subcultures for all complexity classes. Notice that variability in subcultures actually increases with population size as we go from 75 to 100 for all runs. This suggest that the power of subcultures is amplified as the size of the population gets large enough. In other words, when there is a small population, the inherent variability is sufficient to drive discovery, but as the population grows there is more opportunity to retain variability by breaking populations into sub networks that retain advantages that previously accrued to smaller populations.

8.4 Subcultures and Homogeneous T-tests.

In Tables 8-9, 8-10, and 8-11 we investigate the statistical significance of the performances between the subcultures configuration and the best performing homogeneous topology in each category over a wide range of entropy values in terms of the number of generations used for successfully solved problems. We conducted a two tailed t-test with a significance level (α) of 0.05 for the null hypothesis:

"Subcultures Topologies don't affect the performance of the cultural algorithm over the homogeneous topologies".

As can be seen, for the fixed class, the square homogeneous topology uses less generations and standard deviations per solved problem than the subcultures approach. We have four entropy values of the fixed class problems 1.4, 1.8, 2.2, and 2.6. For entropy values 1.4 and 1.8, the given t-test tables show a rejection of the null hypothesis for population sizes 50 and 75 but not 100. It seems that subcultures can utilize the bigger population sizes more than homogeneous topologies and this suggestion will be investigated further in the next section. For entropy values 2.2 and 2.6, the null hypothesis is only rejected (homogeneous topologies were more effective in solving problems) for population size 50 but not 75 and 100.

For the periodic class, subcultures use significantly fewer generations and lower standard deviations than either the heterogeneous or homogeneous topologies as shown in the previous section. There is a statistically significant difference in the number of generations used by the latter two approaches homogeneous and subcultures. It is clear that the presence of the subcultural structure as a social fabric is the key to its improved performance there. Four entropy values are used to test the periodic class problems. The entropy value 3.1 is a critical point because there is no statistically significant difference between the homogenous and subcultures topologies performance of population sizes 50, 75, and 100. For entropies, 3.2, 3.3, and 3.4, there is a statistically significant difference because the hypothesis has been rejected (subcultures outperformed the homogeneous topologies) for those entropies for population sizes 50, 75, and 100 as seen in Tables 8-9, 8-10, and 8-11.

In the chaotic situation with many generators heterogeneity in the topologies becomes more important since both heterogeneous categories outperformed the homogeneous one. There was an observed statistical difference in performance between the homogeneous and subcultures configurations which suggests that subcultures do have a significant impact on the number of

generations used to solve a problem successfully. The null hypothesis was rejected for the four entropy values that in chaotic class problems 3.6, 3.7, 3.8, and 3.9 for the population sizes 50, 75, and 100.

	A-Value	Homogeneous (σ)	Subcultures (σ)	Calculated t-value	Two tailed table t-value	
Fixed	1.4	37.24	76.61	4.564097	1.98	Reject
	1.8	29.41	132.10	4.914624	1.98	Reject
	2.2	21.92	64.86	7.848958	1.98	Reject
	2.6	90.90	157.36	2.761772	1.98	Reject
Periodic	3.1	170.41	125.01	0.089663	1.98	Accept
	3.2	379.88	108.98	2.586513	1.98	Reject
	3.3	239.43	57.55	4.334895	1.98	Reject
	3.4	212.87	101.39	3.569412	1.98	Reject
Chaotic	3.6	423.24	202.33	2.615218	1.98	Reject
	3.7	359.44	133.98	3.752294	1.98	Reject
	3.8	306.92	98.29	5.675213	1.98	Reject
	3.9	386.87	213.89	3.115	1.98	Reject

Table 8-9: T-tests of Homogeneous and Subcultures Topologies of Population Size 50.

	A-Value	Homogeneous (σ)	Subcultures (σ)	Calculated t-value	Two tailed table t-value	
Fixed	1.4	54.50	65.70	2.364202	1.98	Reject
	1.8	70.92	92.24	2.991334	1.98	Reject
	2.2	44.58	70.71	1.433017	1.98	Accept
	2.6	71.32	92.21	1.093008	1.98	Accept
Periodic	3.1	108.76	76.70	0.987187	1.98	Accept
	3.2	283.60	89.76	2.93997	1.98	Reject
	3.3	126.25	63.29	3.176412	1.98	Reject
	3.4	134.49	77.58	3.683554	1.98	Reject
Chaotic	3.6	266.87	148.46	2.477518	1.98	Reject
	3.7	240.85	138.56	2.40734	1.98	Reject
	3.8	190.71	67.01	4.347462	1.98	Reject
	3.9	311.18	150.70	2.034086	1.98	Reject

Table 8-10: T-tests of Homogeneous and Subcultures Topologies of Population Size 75.

	A-Value	Homogeneous (σ)	Subcultures (σ)	Calculated t-value	Two tailed table t-value	
Fixed	1.4	45.40	90.11	1.429613	1.98	Accept
	1.8	45.98	25.07	0.669761	1.98	Accept
	2.2	64.80	43.31	1.778144	1.98	Accept
	2.6	128.94	118.70	0.524517	1.98	Accept
Periodic	3.1	88.75	101.88	0.498202	1.98	Accept
	3.2	212.11	70.53	3.203265	1.98	Reject
	3.3	105.84	80.19	3.107656	1.98	Reject
	3.4	84.55	67.18	2.328396	1.98	Reject
Chaotic	3.6	151.08	62.09	3.135064	1.98	Reject
	3.7	199.34	144.08	5.187544	1.98	Reject
	3.8	250.50	165.80	3.697962	1.98	Reject
	3.9	264.44	161.91	2.535386	1.98	Reject

Table 8-11: T-tests of Homogeneous and Subcultures Topologies of Population Size 100.

8.5 Subcultures and Heterogeneous T-tests.

We showed that subcultures outperformed the homogeneous and heterogeneous topologies for periodic and chaotic classes of problems in terms of standard deviation and average number of generations. However, the significance difference between the subcultures and heterogeneous wasn't clear from the t-tests we conducted as shown in Tables 8-12, 8-13, and 8-14.

	A-Value	Heterogeneous (σ)	Subcultures (σ)	Calculated t-value	Two tailed table t-value	
Fixed	1.4	60.86	76.61	1.650748	1.98	Accept
	1.8	102.96	132.10	0.192519	1.98	Accept
	2.2	102.42	64.86	1.889822	1.98	Accept
	2.6	198.78	157.36	0.188538	1.98	Accept
Periodic	3.1	154.21	125.01	0.610528	1.98	Accept
	3.2	257.11	108.98	2.381707	1.98	Reject
	3.3	160.49	57.55	3.856168	1.98	Reject
	3.4	121.49	101.39	1.331657	1.98	Accept
Chaotic	3.6	314.08	202.33	1.759021	1.98	Accept

	3.7	290.78	133.98	2.40295	1.98	Reject
	3.8	93.35	98.29	3.434379	1.98	Reject
	3.9	198.20	213.89	1.725067	1.98	Accept

Table 8-12: T-tests of Heterogeneous and Subcultures Topologies of Population Size 50.

	A-Value	Heterogeneous (σ)	Subcultures (σ)	Calculated t-value	Two tailed table t-value	
Fixed	1.4	41.59	76.61	1.531429	1.98	Accept
	1.8	158.44	132.10	1.25351	1.98	Accept
	2.2	84.79	64.86	1.55636	1.98	Accept
	2.6	239.41	157.36	1.740245	1.98	Accept
Periodic	3.1	147.15	125.01	3.402177	1.98	Reject
	3.2	128.20	108.98	1.353637	1.98	Accept
	3.3	93.61	57.55	1.211491	1.98	Accept
	3.4	84.02	101.39	1.697896	1.98	Accept
Chaotic	3.6	165.32	202.33	1.105545	1.98	Accept
	3.7	195.38	133.98	0.954703	1.98	Accept
	3.8	222.91	98.29	1.527419	1.98	Accept
	3.9	218.82	213.89	1.187502	1.98	Accept

Table 8-13: T-tests of Heterogeneous and Subcultures Topologies of Population Size 75.

	A-Value	Heterogeneous (σ)	Subcultures (σ)	Calculated t-value	Two tailed table t-value	
Fixed	1.4	68.78	76.61	1.531429	0.184638	Accept
	1.8	218.49	132.10	1.25351	1.852615	Accept
	2.2	44.78	64.86	1.55636	3.475647	Reject
	2.6	265.00	157.36	1.740245	2.383605	Reject
Periodic	3.1	106.93	125.01	3.402177	3.469984	Reject
	3.2	120.56	108.98	1.353637	2.111037	Reject
	3.3	59.03	57.55	1.211491	2.715519	Reject
	3.4	67.17	101.39	1.697896	0.434686	Accept
Chaotic	3.6	105.21	202.33	1.105545	1.583585	Accept
	3.7	213.43	133.98	0.954703	2.247843	Reject
	3.8	128.75	98.29	1.527419	1.144623	Accept
	3.9	186.93	213.89	1.187502	0.797171	Accept

Table 8-14: T-tests of Heterogeneous and Subcultures Topologies of Population Size 100.

8.6 Deeper look to the population size.

By comparing the blue and red colored curves in Figures 8-7, 8-8, and 8-9, we noticed that when the population size changed from 50 to 75, all social fabricated topologies could utilize the additional individuals to enhance the performance in solving the problems over all categories; fixed, periodic, and chaotic. After changing the population size to 100, subcultures could utilize the additional individuals for all investigated entropies and that can be seen as a gray colored curve in Figure 8-9. However, homogeneous and heterogeneous topologies couldn't utilize the added individuals to improve the performance for all of the investigated entropies as the gray curve could come above the red curve as shown in Figures 8-7 and 8-8. After this investigation, the question of concern is which of the topologies can utilize the added individuals to the Population Space to improve the problem solvability performance.

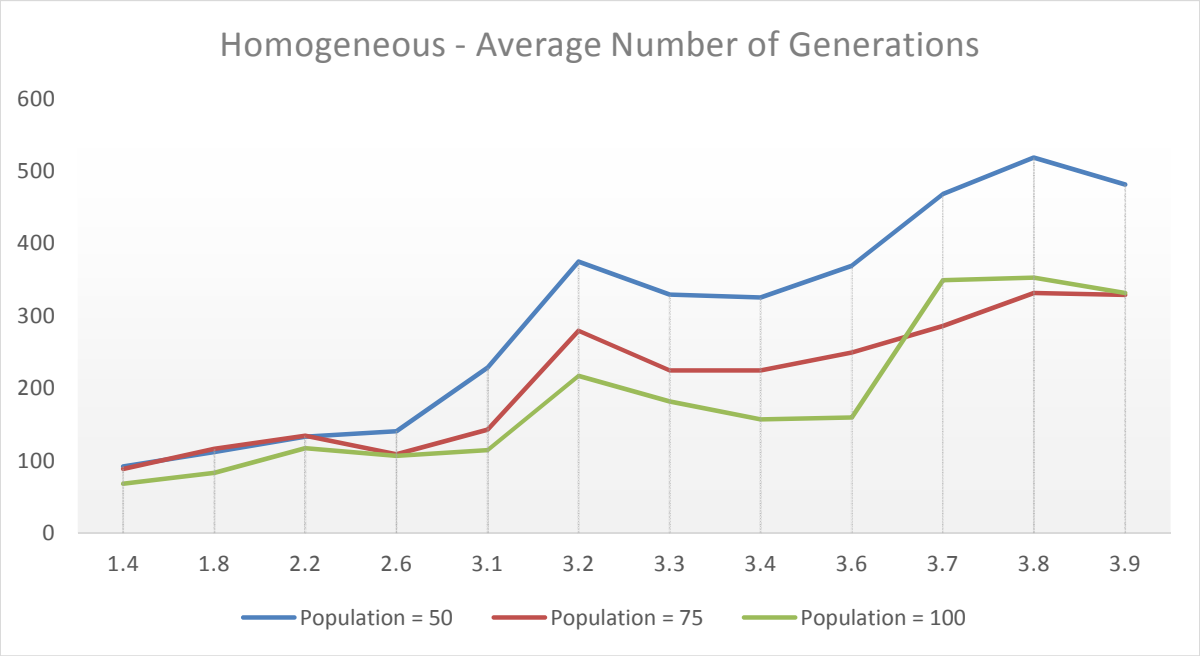


Figure 8-7: The Average Number of Generations Comparison of Homogeneous Topologies for Population Size 50, 75, and 100.

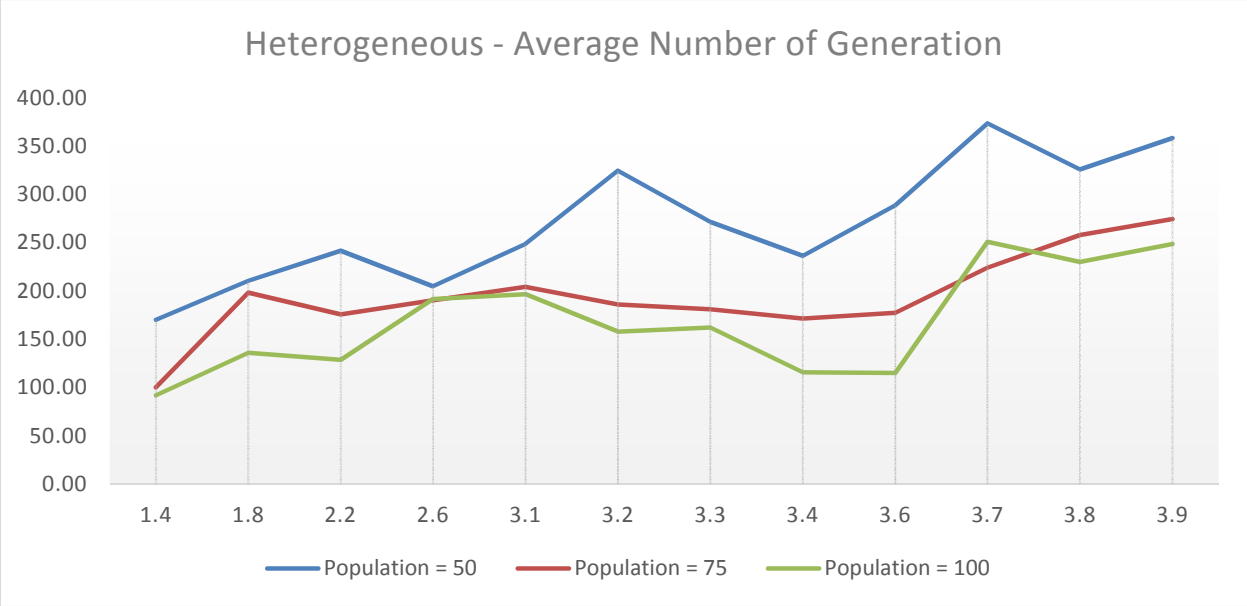


Figure 8-8: The Average Number of Generations Comparison of Heterogeneous Topologies for Population Size 50, 75, and 100.

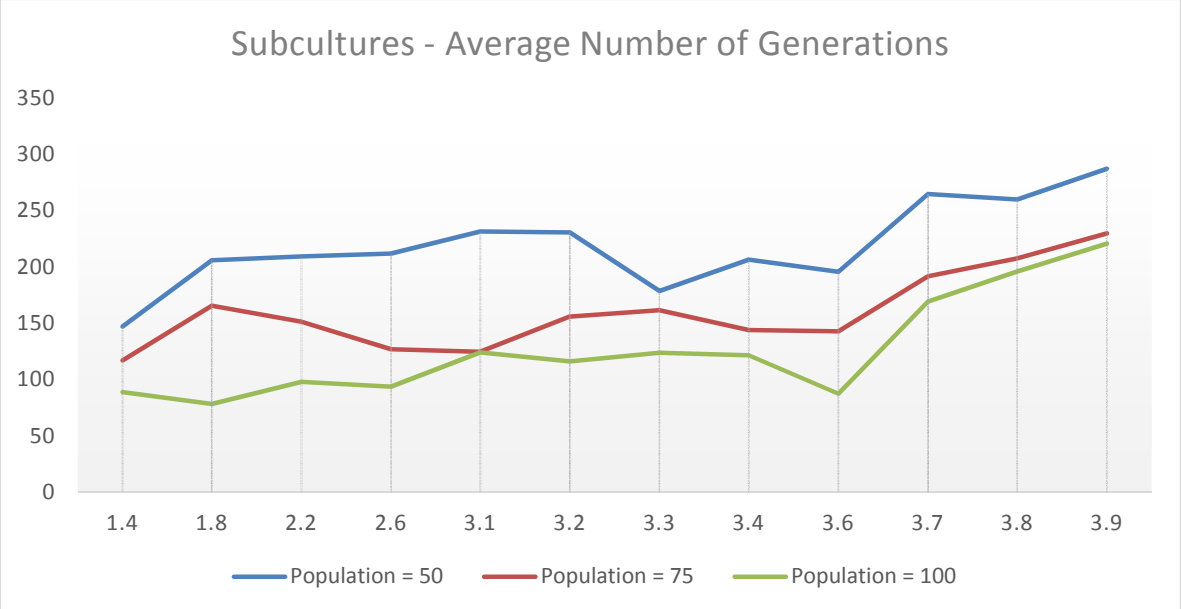


Figure 8-9: The Average Number of Generations Comparison of Subcultures Topologies for Population Size 50, 75, and 100.

	Changing Population Size from 50 to 75	Changing Population Size from 75 to 100
Homogeneous	21%	1%
Heterogeneous	26%	11%
Subcultures	40%	33%

Table 8-15: Performance Improvement by Changing the Population Sizes from 50 to 75, and 100.

As shown in Table 8-15, when the population size changed from 50 to 75, heterogeneous topologies (26%) could utilize the added individuals more that the homogeneous topologies (21%). However, the subcultures had the highest utilization percentage (40%) when the population changed to 75. When the population size changed from 75 to 100, the homogeneous had the lowest utilization percentage (1%) and subcultures could have the highest utilization (33%) among the investigated social fabricated topologies.

8.7 Social Metrics.

In the previous section it was clear that overall variability declined as a population increased. But we observed over all runs that variability actually increased with increased population size when subcultures were used. So, the question is how this variability is used within the different classes of complexity by subcultures. That is the focus of this section.

Figures 8-10, 8-12, and 8-14 give a comparison between the Social Tension averages of Homogeneous and Subcultures for population sizes 50, 75, and 100, consecutively. In population sizes 50 and 75, the average Social Tension of subcultures topology is higher than the one of the Homogeneous topologies for all entropies in periodic and chaotic problems classes (A= 3.1, 3.2,3.3, 3.4, 3.6, 3.7, 3.8, and 3.9). The average Social Tension of subcultures topology in population size 100 tended to be higher than the one for Homogeneous topologies for entropy values above 3.1 as shown in Figure 8-14. As we discussed in earlier chapters, the higher Social

Tension, the more variant information will flow between the Population Space's individuals to solve the problem faster. Therefore, the Social Tension is one of the indicators that explain why Subcultures topology outperforms Homogeneous topologies in periodic and chaotic problem classes.

The average Social Tension of Subcultures is higher than the ones of homogeneous and heterogeneous topologies for entropies greater or equal 3.1 in population sizes 50 and 75 as can be seen in Figures 8-11 and 8-13. Figure 8-15 shows that Subcultures' average Social Tension tended to be higher than the ones for homogeneous and heterogeneous topologies for entropies greater than 3.1 in population size 100.

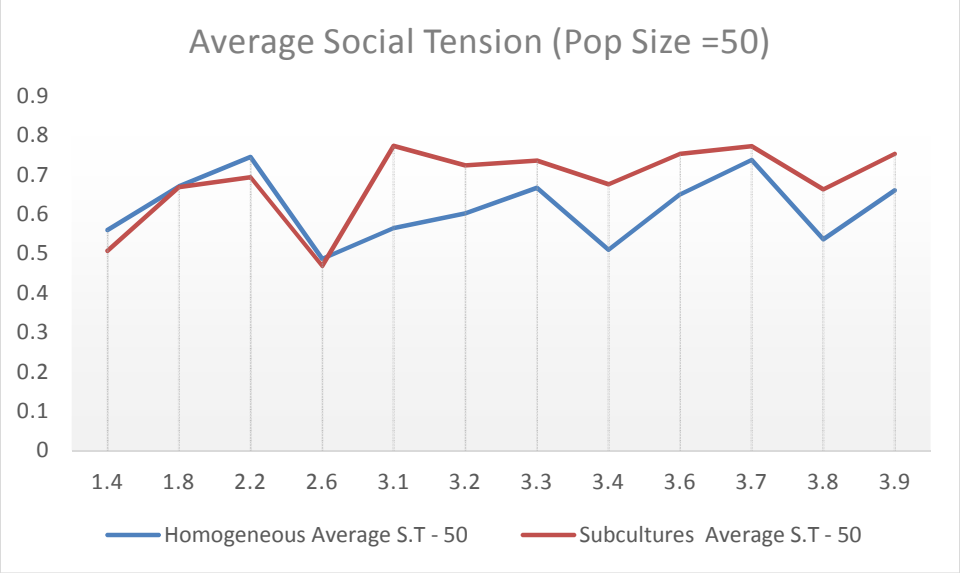


Figure 8-10: Average Social Tension Comparison of Homogeneous and Subcultures (Pop Size =50).

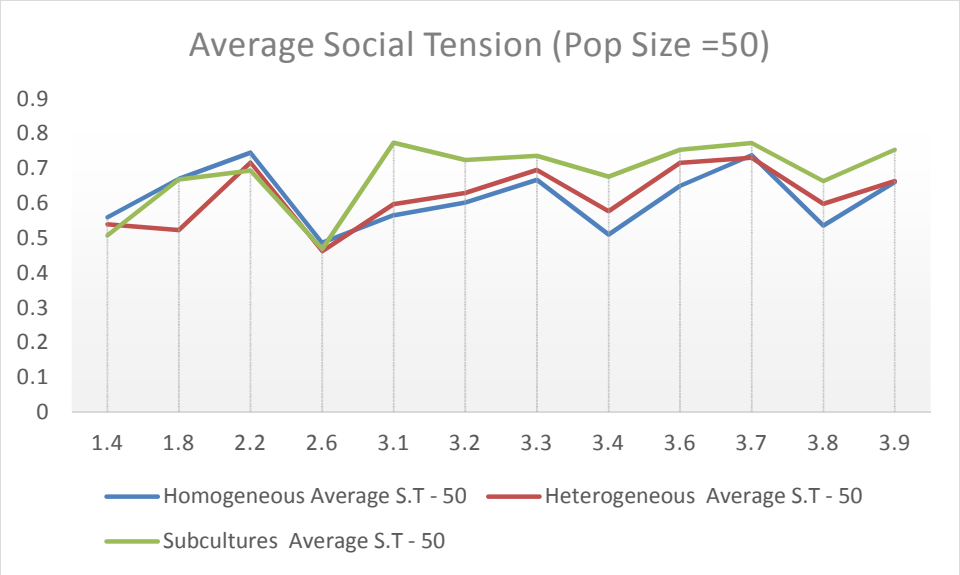


Figure 8-11: Average Social Tension Comparison of Homogeneous, Heterogeneous, and Subcultures (Pop Size =50).

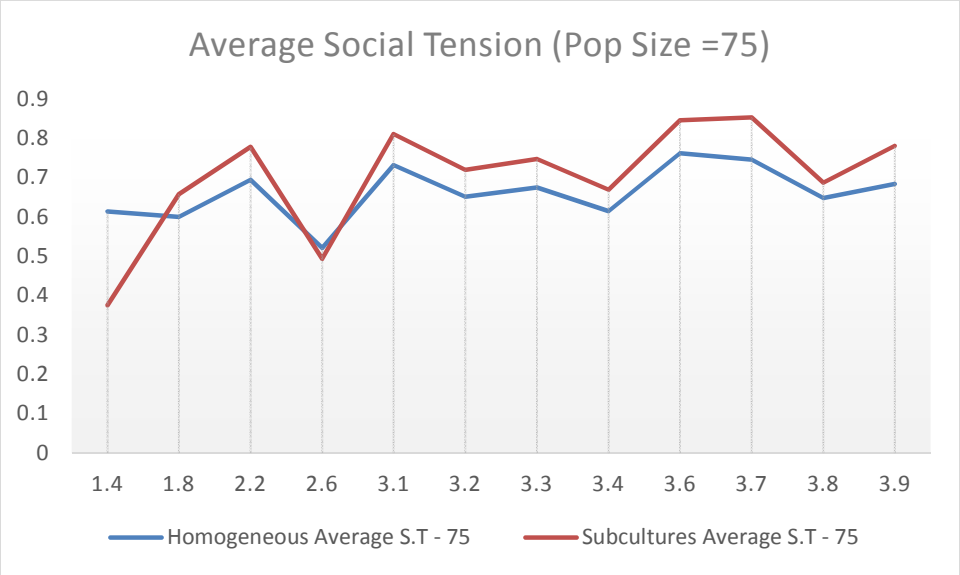


Figure 8-12: Average Social Tension Comparison of Homogeneous and Subcultures (Pop Size =75).

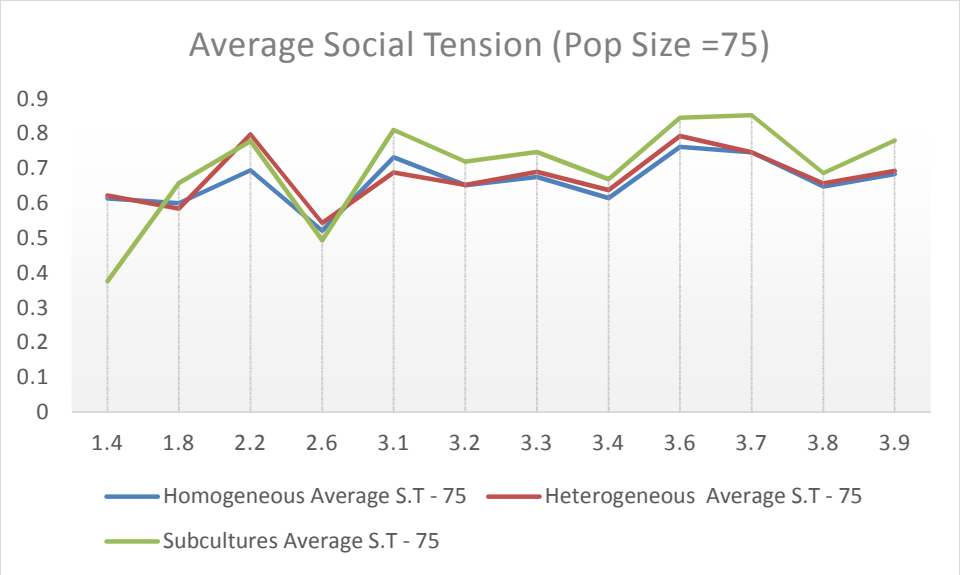


Figure 8-13: Average Social Tension Comparison of Homogeneous, Heterogeneous, and Subcultures (Pop Size =75).

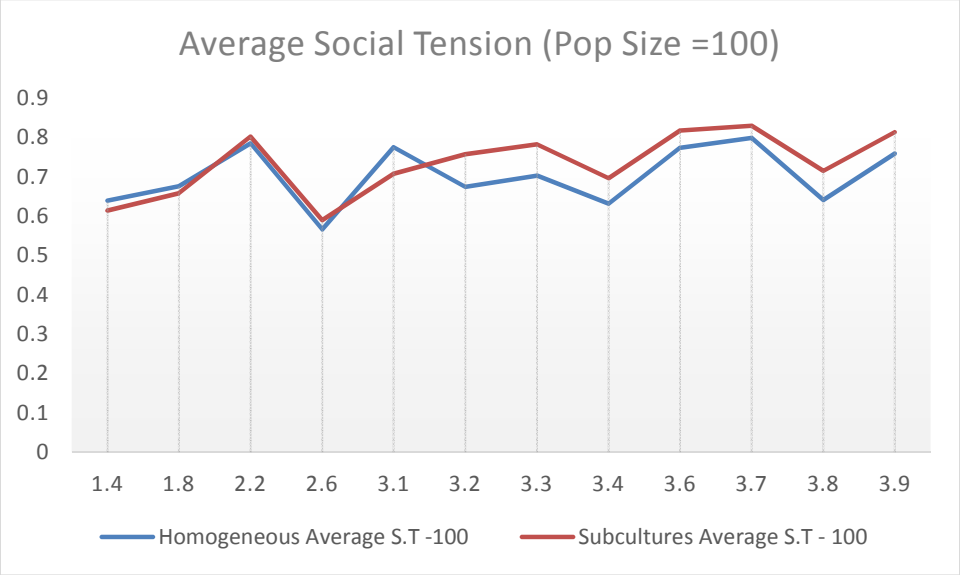


Figure 8-14: Average Social Tension Comparison of Homogeneous and Subcultures (Pop Size =100).

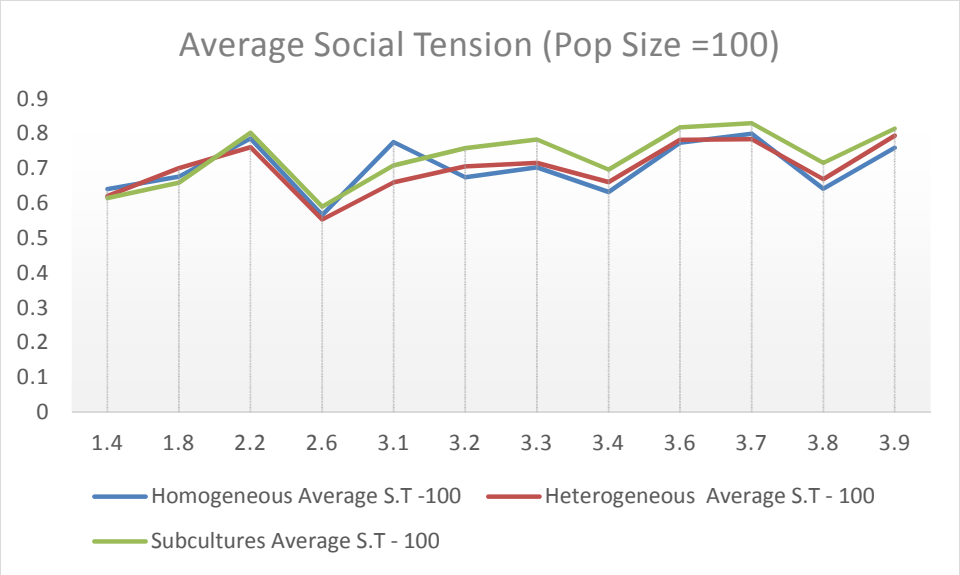


Figure 8-15: Average Social Tension Comparison of Homogeneous, Heterogeneous, and Subcultures (Pop Size =100).

As the population size increases, the Social Tension increases because more individuals in the Population Space bring variant experiences. This trend can be seen in Figures 8-16, 8-17, and 8-18. Figures 8-16, 8-17, and 8-18 give the average Social Tension of all tested population sizes for homogeneous, heterogeneous, and subcultures topologies, respectively.

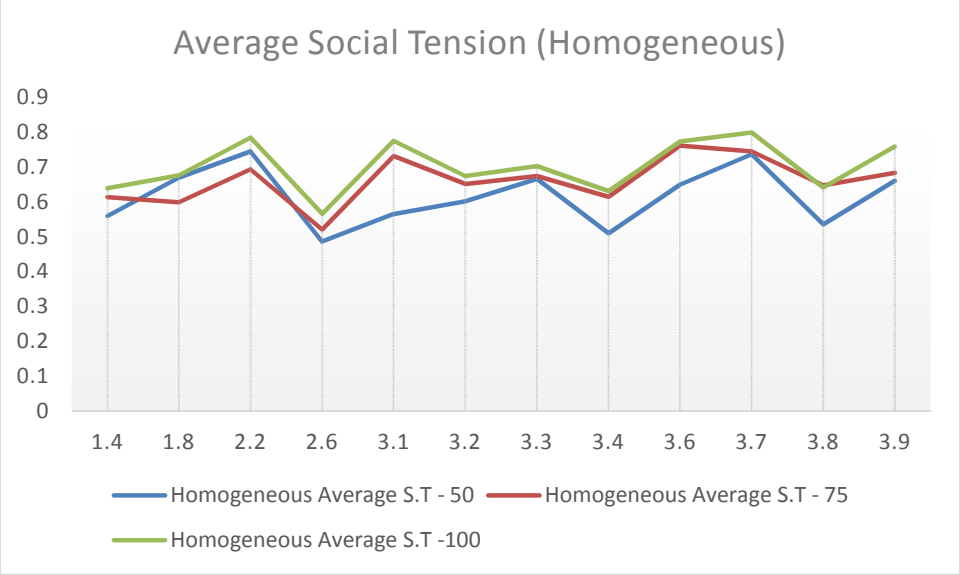


Figure 8-16: Average Social Tension of Homogeneous Topologies in Population Sizes 50, 75, and 100.

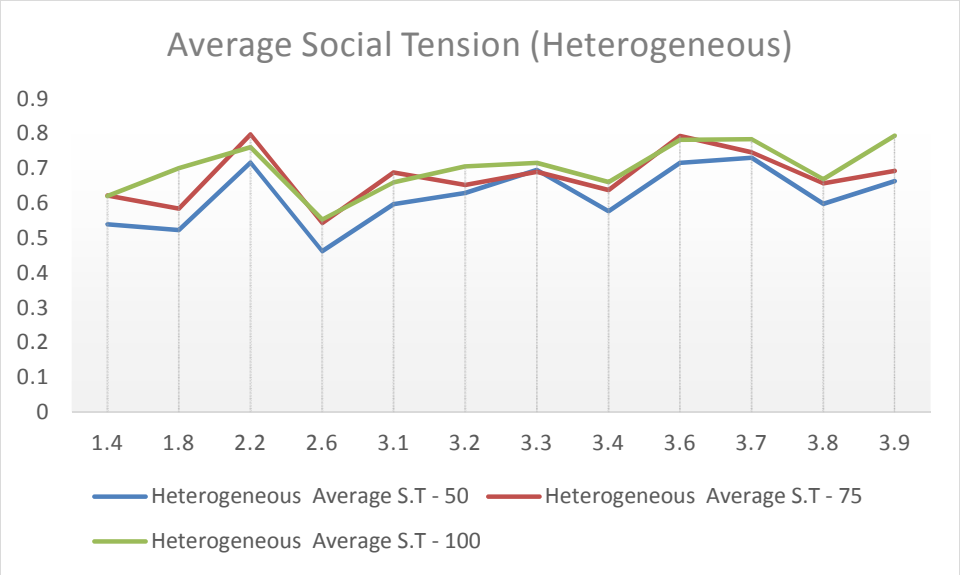


Figure 8-17: Average Social Tension of Heterogeneous Topologies in Population Sizes 50, 75, and 100.

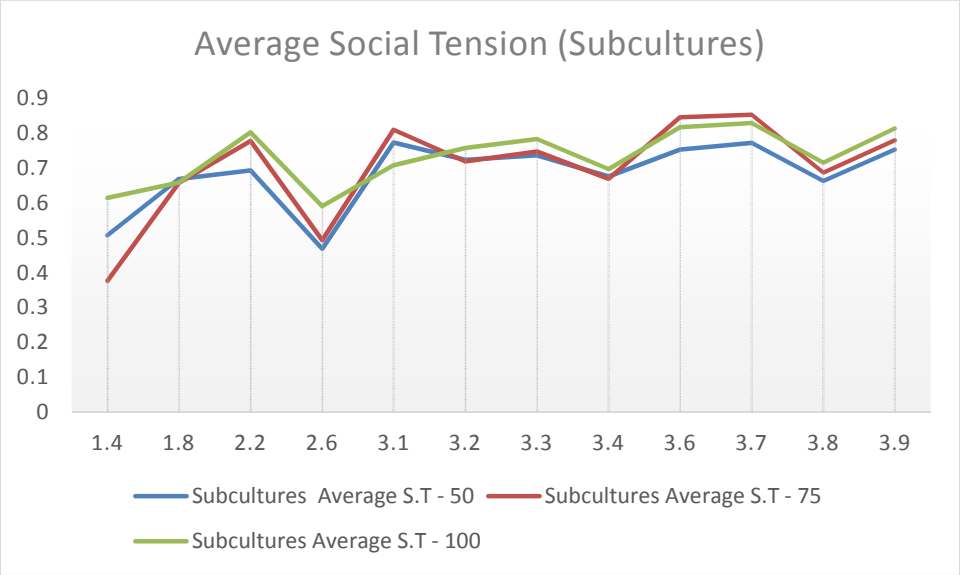


Figure 8-18: Average Social Tension of Subcultures Topologies in Population Sizes 50, 75, and 100.

Figures 8-19, 8-21, and 8-23 compare the average of Innovation Cost Index of Homogeneous and Subcultures topologies in populations' sizes 50, 75, and 100, consecutively. As a conclusion from the given figures, the Subcultures topology achieved a smaller Innovation Cost Index than the Homogeneous topologies in the periodic and chaotic problems classes. The

question of concern is which trend of Innovation Cost Index is better; the lower or the higher. The Innovation Cost Index value represents the needed effort to allow the minority knowledge source to win over the majority knowledge source to allow the minority knowledge source to explore new promising areas over the landscape of the problem. As a result, the lower the Innovation Cost Index of Subcultures topology in periodic and chaotic problems classes, the higher the performance will be achieved in solving the problems.

The Subcultures topology's Innovation Cost Index is lower than the ones for homogeneous and heterogeneous topologies for periodic and chaotic problems classes in population sizes 50, 75, and 100 as shown in Figures 8-20, 8-22, and 8-24, consecutively.

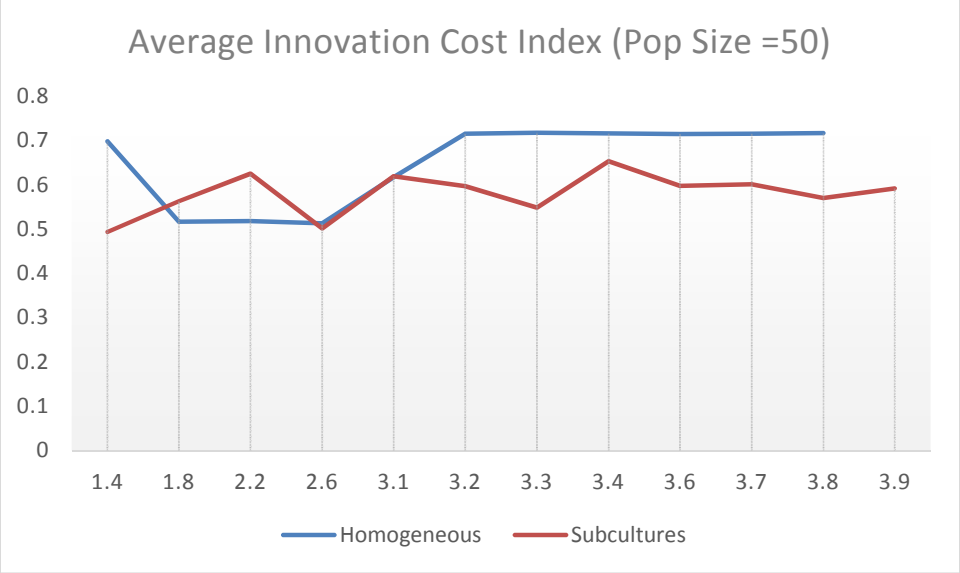


Figure 8-19: Average Social Innovation Cost Index Comparison of Homogeneous and Subcultures (Pop Size = 50).

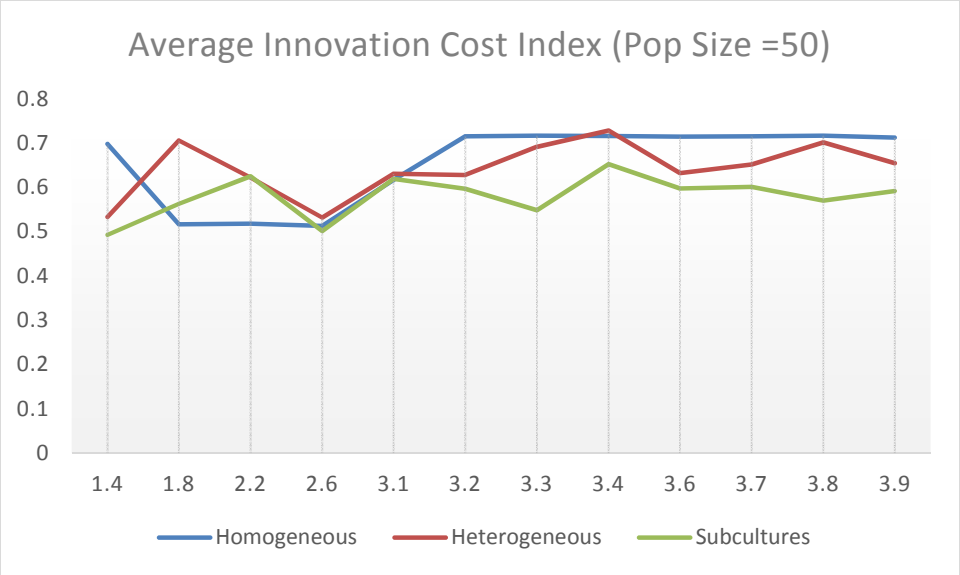


Figure 8-20: Average Social Innovation Cost Index Comparison of Homogeneous, Heterogeneous, and Subcultures (Pop Size = 50).

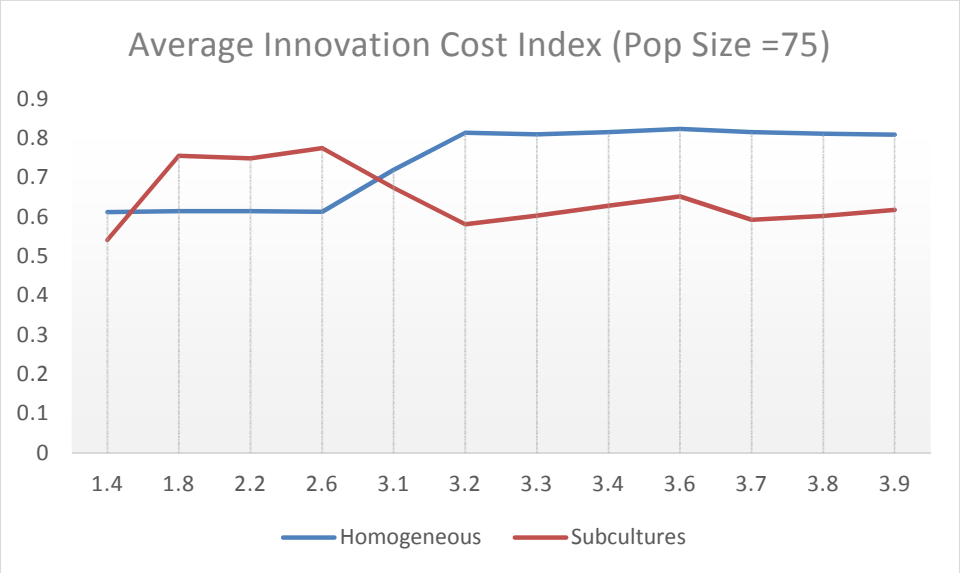


Figure 8-21: Average Social Innovation Cost Index Comparison of Homogeneous and Subcultures (Pop Size = 75).

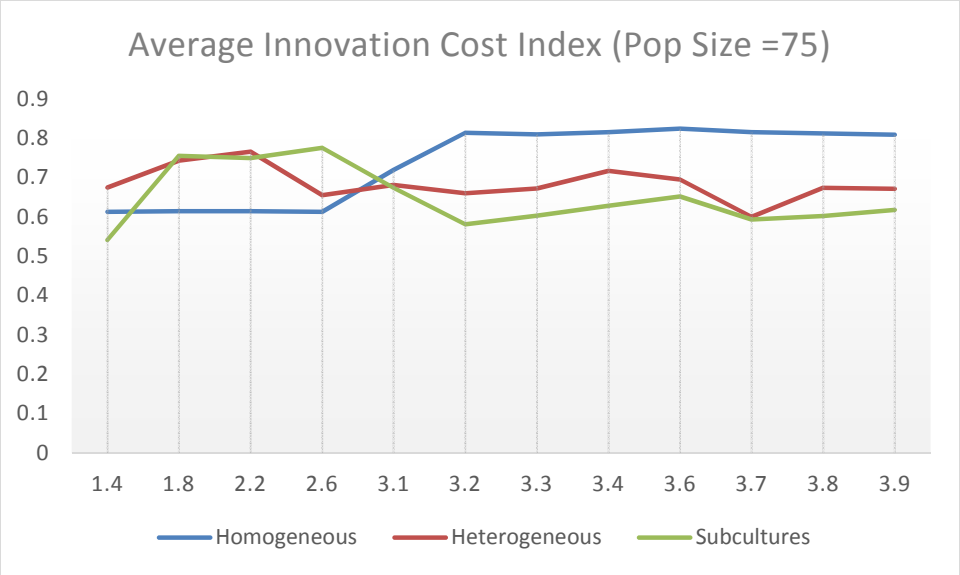


Figure 8-22: Average Social Innovation Cost Index Comparison of Homogeneous, Heterogeneous, and Subcultures (Pop Size = 75).

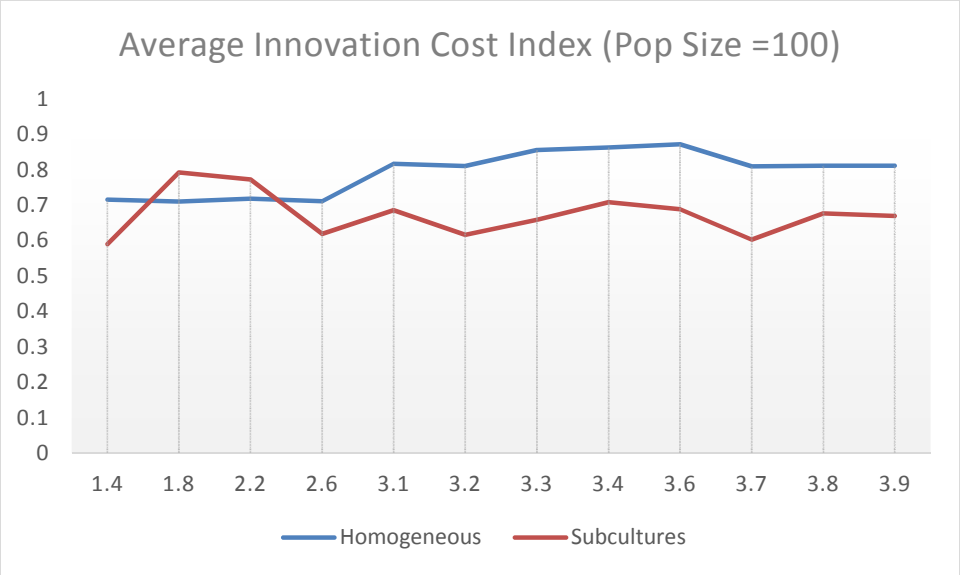


Figure 8-23: Average Social Innovation Cost Index Comparison of Homogeneous and Subcultures (Pop Size = 100).

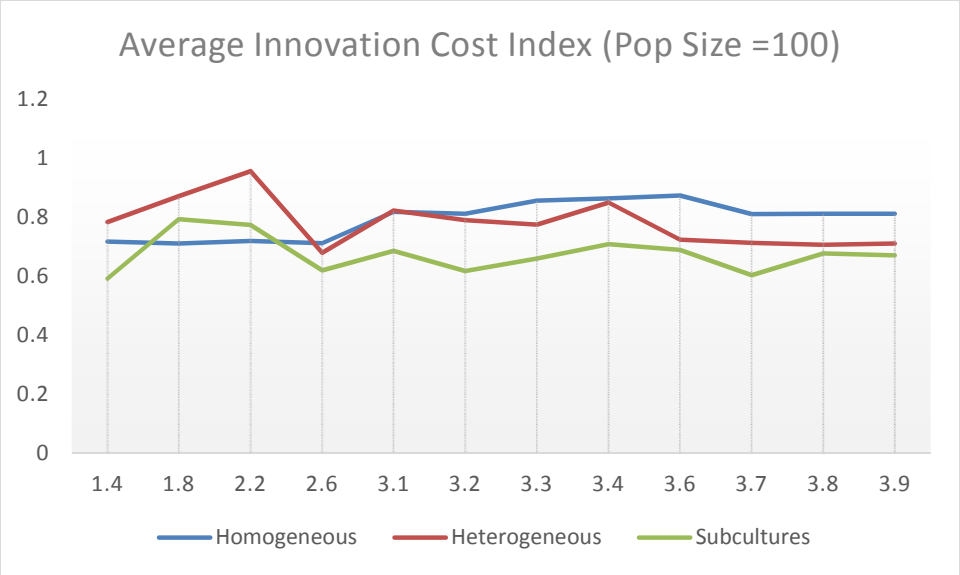


Figure 8-24: Average Social Innovation Cost Index Comparison of Homogeneous, Heterogeneous, and Subcultures (Pop Size = 50).

When the population size increases, more noise will be added to the problem on the landscape. Therefore, the minority knowledge sources need more effort to win over the majority knowledge sources as seen in Figures 8-25, 8-26, and 8-27.

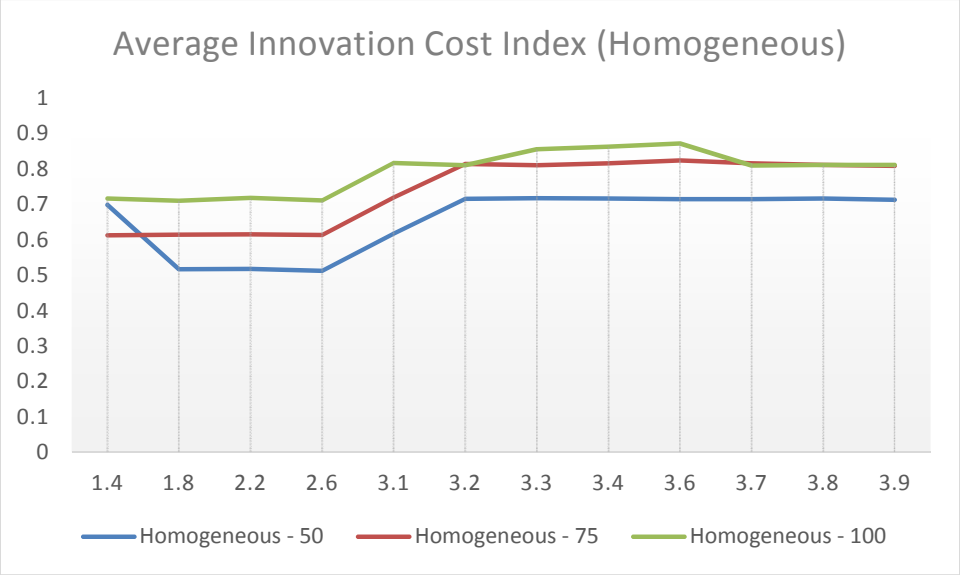


Figure 8-25: The Average Innovation Cost Index Cost of Homogeneous Topologies in Population Sizes 50, 75, and 100.

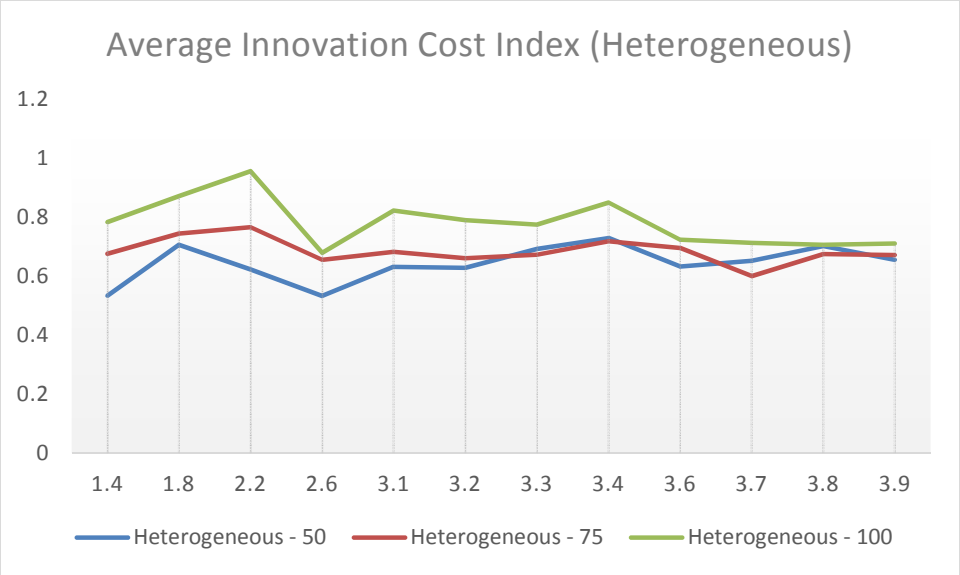


Figure 8-26: The Average Innovation Cost Index of Heterogeneous Topologies in Population Sizes 50, 75, and 100.

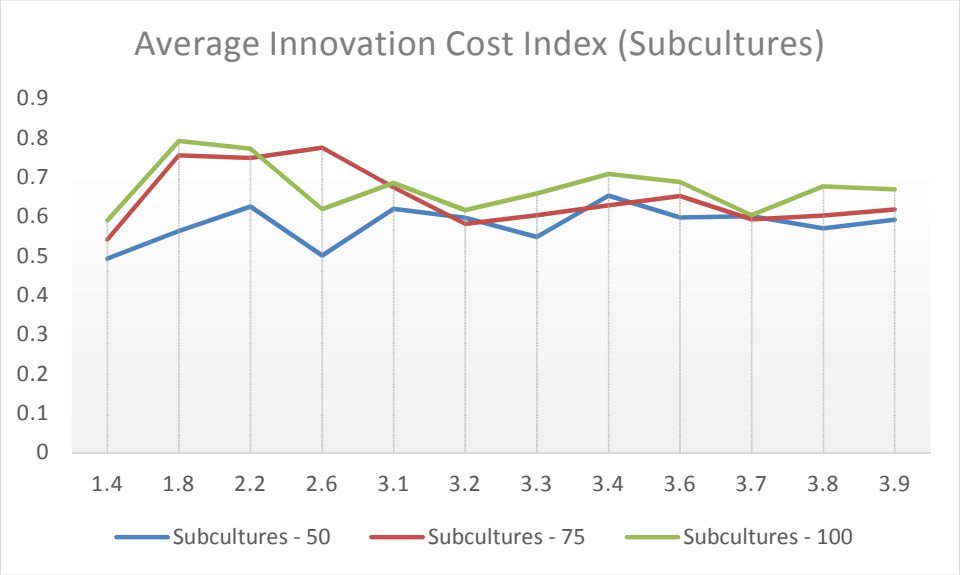


Figure 8-27: The Average Innovation Cost Index of Subcultures Topologies in Population Sizes 50, 75, and 100.

8.8 Conclusion.

In this chapter, a subcultures topology was compared with other homogeneous and heterogeneous topologies over twelve entropy values (1.4, 1.8, 2.2, 2.6, 3.1, 3.2, 3.3, 3.4, 3.6, 3.7, 3.8, and 3.9). The results of the statistical tests conducted suggest in computational terms two basic ways why subcultures can emerge successfully in certain social situations. First, with multiple pattern generators in the search space the subcultures can be useful in sorting out the different patterns produced by each generator, assuming of course that there are not too many. Second, the presence of subcultures tends to reduce the overall variability in the problem solving process along with dampening down the impact of increased problem complexity.

Homogeneous topologies performance were more efficient and predictable in fixed problems class with the tested entropies 1.4, 1.8, 2.2, and 2.6. Entropy value 3.1 was a critical point (or singularity) for the topologies to start defining their trends in solving different problem categories. Subcultures showed the highest performance predictability and efficiency in periodic and chaotic problems classes as was shown previously for entropies 3.2, 3.3, 3.4, 3.6, 3.7, 3.8, and 3.9.

For the periodic class, subcultures use significantly fewer generations than either the heterogeneous or homogeneous topologies. There is no statistically significant difference in the number of generations used by the latter two approaches. It is clear that the presence of the subcultural structure in the heterogeneous topology is the key to its improved performance there.

In the chaotic situation with many generators heterogeneity in the topologies becomes more important since both heterogeneous categories outperformed the homogeneous one. There was no observed statistical difference in performance between the two heterogeneous classes which suggests that subcultures do not have a significant impact on the number of generations

used to solve a problem successfully. However, the presence of subcultures tends to reduce the standard deviation of the number of generations used to get a solution, even though it was not tested for significance here.

For a given population size, the reason for the performance of subcultures as complexity can be seen at the network level in terms of the social metrics. First, subcultures are able to maintain a reduced index over the other two approaches as complexity increases. In addition, it is able to exploit increases in population size in order to enhance individual variability and reduce Innovation Cost as population sizes increase. This suggests that as complexity of problems increase along with population size, subcultures will become more and more important in the problem solving process.

CHAPTER 9 CONCLUSIONS AND FUTURE WORK

In this work, we introduced two new social fabricated models to operate between the individuals of the Population Space; Heterogeneous and Subcultures Models. In both models, we are using the homogeneous topologies that were introduced in previous work [Ali, 2008; Che, 2009; Reynolds, Che, and Ali, 2010]. The idea is forming heterogeneous topologies by allowing multiple homogenous topologies to operate together between the individuals of the Population Space.

In the Heterogeneous Model, a group of homogeneous topologies were allowed to operate together simultaneously in the Population Space. We added a new wheel in the Belief Space to store the individuals' average fitness. Then, we used this wheel to allow the individuals to select the current topology at each time step. In spite of storing the topology information in the Belief Space, the knowledge sources do not have any kind of information about the topologies.

The results were promising, chapter 7 shows that heterogeneous topology increases the performance of solving the problems by increasing the complexity of problem classes. Chaos landscapes are best exploited by heterogeneous networks, whereas static landscapes are best exploited by single homogeneous topologies. Heterogeneous topologies somehow produce a social organization that is better able to exploit heterogeneous landscapes. Heterogeneous topology performed better than a few homogeneous topologies, in terms of performance with periodic class problems.

Keeping the knowledge sources and topology wheels separate from one another blinds the Influence Function of the relationship between the knowledge sources and the topologies used among the individuals. Therefore, the Subcultures Model was proposed to build the relationship between Belief Space's knowledge sources and Population Space's topologies.

The Subcultures Model divides the Population Space into subsets of related individuals. The introduction of subcultures into Cultural Algorithms required more changes into Cultural Algorithms Model's components. Two main changes to the Cultural Algorithm's behavior needed to allow evolving the subcultures among the Population Space's individuals. One of these changes is dividing the knowledge sources wheel into n wheels, where n represents the number of topologies that operate in the Population Space. The other change is to update the Influence Function to handle the additional wheels.

We used Cones World problem as a landscape to test our new approaches, because Cones World problem gives the complete range of Langton's optimization problem complexities, from fixed to chaotic. In order to test how these problem categories are reflected by changes in the corresponding social organization of the Cultural systems needed to solve them, an example set of four landscapes from each of the three basic classes fixed (A=1.4, 1.8, 2.2 and 2.6), periodic (A=3.1, 3.2, 3.3 and 3.4) and chaotic (A=3.6, 3.7, 3.8 and 3.9) will be generated. The task will be to measure the Subcultures heterogeneous Performance and compare it with the homogeneous and heterogeneous topologies' performance in solving problems from the different complexity classes. The Square topology was used to represent the set of homogeneous social fabrics because it outperformed the other homogeneous topologies. Each complexity (1.4, 1.8, 2.2, 2.6, 3.1, 3.2, 3.3, 3.4, 3.6, 3.7, 3.8 and 3.9) has five randomly generated example Landscapes. A combination of ten independent runs for each landscape (from the generated example Landscapes) was tested on each of the proposed homogeneous, heterogeneous, and Subcultures topologies. Fifty runs will be conducted as a total for each of the topologies on each of the complexities. Three population sizes were used to add more variability to our experiments. The population are 50, 75 and 100. To investigate the results of our experiments, we used social

metrics Dispersion Index and Innovation Cost Index. Those metrics are related to diversity and innovation. A higher Dispersion Index means that neighbors that are adjacent in the network can get information about more diverse places on the functional landscape. A lower Innovation Cost Index means that innovations that are produced are more productive in exploring new promising areas over the functional landscape. The results of our experiments is summarized in the next few paragraphs.

In chapter 7, the heterogeneous topology and the homogeneous topologies were compared. For static problems with predictable patterns of distribution, the homogeneous networks were the most effective. As more and more static problems were blended together to increase complexity, the heterogeneous approach became the dominant strategy in terms of the number of problems solved in the given time frame. This was because there was work for more than one topology to perform. The approach effectively allowed a team of networks to work on the problem space, each exploiting those patterns most suited for it.

So, for static problems with fixed cone distribution parameters the homogeneous topologies were the most effective in solving them. On the other side, the heterogeneous topologies were more effective in solving problems of periodic and chaotic classes. When we conducted the t-test on different population sizes, we noticed the need of dividing the Population Space into subcultures and updating the Belief Space, the Influence Function, and the Acceptance Function. After comparing the heterogeneous with the homogeneous topologies in chapter 7, we added the subcultures to give more comparable results to the heterogeneous and homogeneous topologies in chapter 8.

A subcultures topology was compared with other homogeneous and heterogeneous topologies, in chapter 8, over twelve entropy values (1.4, 1.8, 2.2, 2.6, 3.1, 3.2, 3.3, 3.4, 3.6, 3.7,

3.8, and 3.9) that represent fixed, periodic, and chaotic problems classes. The results of the statistical tests conducted suggest in computational terms two basic ways why subcultures can emerge successfully in certain social situations. First, with multiple pattern generators in the search space the subcultures can be useful in sorting out the different patterns produced by each generator, assuming of course that there are not too many. Second, the presence of subcultures tends to reduce the overall variability in the problem solving process along with dampening down the impact of increased problem complexity.

Homogeneous topologies were more efficient and predictable in fixed problems class with the tested entropies 1.4, 1.8, 2.2, and 2.6. Entropy value 3.1 was a critical point (or singularity) for the topologies to start defining their trends in solving different problem categories. Subcultures showed the highest predictability and efficiency in periodic and chaotic problems classes as was shown previously for entropies 3.2, 3.3, 3.4, 3.6, 3.7, 3.8, and 3.9.

For the periodic class, subcultures use significantly fewer generations than either the heterogeneous or homogeneous topologies. There is no statistically significant difference in the number of generations used by the latter two approaches. It is clear that the presence of the subcultural structure in the heterogeneous topology is the key to its improved performance there.

In the chaotic situation with many generators heterogeneity in the topologies becomes more important since both heterogeneous categories outperformed the homogeneous one. There was no observed statistical difference in performance between the two heterogeneous classes which suggests that subcultures do not have a significant impact on the number of generations used to solve a problem successfully. However, the presence of subcultures tends to reduce the standard deviation of the number of generations used to get a solution, even though it was not tested for *significance* here.

In future work the goal will be to map real world problems into these complexity classes to that one can predict what social network will be most effective. The subcultures social fabricated Population Space can be used in solving Multi-Objective problems to be compared with other homogeneous and heterogeneous social fabricated Population Spaces in [Reynolds and Liu, 2011].

REFERENCES

[Advances in Complex Systems, 2010] Advances in Complex Systems, Journal. 2010. URL http://www.worldscinet.com/acs/mkt/aims_scope.shtml.

[Ali, 2008] Ali, M.Z. 2008. *Using Cultural Algorithms to Solve Optimization Problems with a Social Fabric Approach*, Dissertation, Wayne State University.

[Bratton and Kennedy, 2007] Bratton, D. and Kennedy, J. 2007. *Defining a Standard for Particle Swarm Optimization*, in Proceedings of the 2007 IEEE Swarm Intelligence Symposium, Honolulu, HI, pp: 120-127.

[Che, 2009] Che, X. 2009. *Weaving the Social Fabric: Optimization Problem Solving in Cultural Algorithms using the Cultural Engine*, Dissertation, Wayne State University.

[Che, Ali, and Reynolds, 2010] Che, X., Ali, M.Z., and Reynolds, R.G. 2010. *Robust evolution optimization at the edge of chaos: Commercialization of culture algorithms*. IEEE Congress on Evolutionary Computation, pp: 1-8.

[Charnov, 1976] Charnov, E. 1976. *Optimal foraging, the marginal value theorem*, Theoretical population biology 9(2): 129.

[Chung and Reynolds, 1998] Chung, C. and Reynolds, R.G. 1998. *CAEP: an evolution-based tool for real-valued function optimization using Cultural Algorithms*, International Journal on Artificial Intelligence Tools, vol. 7, o. 3, pp: 239-291.

[Clayton, Griffiths, and Dickinson, 2001] Clayton, N., Griffiths, D., and Dickinson A. 2001. *Declarative and Episodic-like Memory in Animals: Personal Musings of a Scrub Ja*. The Evolution of Cognition. C. Heyes and L. Huber. The MIT Press, Cambridge, Massachusetts, pp: 273–288.

[Collier, Howe, and North, 2003] Collier, N., Howe, T., and North, M. 2003. *Onward and Upward: The Transition to Repast 2.0*, in Proceedings of the First Annual North American Association for Computational Social and Organizational Science Conference, Electronic Proceedings, Pittsburgh, PA USA.

[Engelbrecht, 2014] Engelbrecht, A.P. 2014. *Fitness function evaluations: A fair stopping condition?*. In Proceedings of the 2014 Symposium Series on Computational Intelligence, pp: 181:188.

[Gawasmeh and Reynolds, 2014] Gawasmeh, Y.A., and Reynolds, R.G. 2014, *A Computational Basis for the Presence of Sub Cultures in Cultural Algorithms*. In Proceedings of the 2014 IEEE Symposium Series on Computational Intelligence, Orlando, FL, pp: 328-335.

[Jin, Girvan, and Newman, 2001] Jin, E., Girvan, M., and Newman, M. 2001. *Structure of Growing Social Networks,* PHYSICAL REVIEW-SERIES E-4; PART 1, pp: 46132-46132.

[Jin and Reynolds, 1999] Jin, X. and Reynolds, R.G. 1999. *Using Knowledge-Based Evolutionary Computation to Solve Nonlinear Constraint Optimization Problems: a Cultural Algorithm Approach*, in Proceedings of the 1999 Congress on Evolutionary Computation (CEC99), Washington, DC., pp: 1672-1678.

[Jin and Reynolds, 1999] Jin, X. and Reynolds, R.G. 1999. *Using Knowledge-Based System with Hierarchical Architecture to Guide the Search of Evolutionary Computation*, in Proceeding of the Eleventh IEEE International Conference on Tools with Artificial Intelligence (ICTAI'99), Chicago IL, Nov., pp: 29-36.

[Langton, 1992] Langton, C. 1992. *Life at the Edge of Chaos*, Artificial life II, pp: 41–91.

[Lazar and Reynolds, 2002] Lazar, A. and Reynolds, R.G. 2002. *Evolution-based Learning of Ontological Knowledge for a Large-scale Multi-agent Simulation*, in Proceedings of the 6th Joint Conference on Information Science. USA, pp: 659-662.

[Loiaciano et. al, 2008] Loiacono, D., Togelius, J., Lanzi, P.L., Kinnaird-Heether, L., Lucas, S.M., Simmerson, M., Perez, D., Reynolds, R.G. and Saez, Y. 2008. *The wcci 2008 simulated car racing competition*. In Proceedings of the IEEE Symposium on Computational Intelligence and Games.

[Morrison and De Jong, 1999] Morrison, R. and De Jong, K. 1999. *A Test Problem Generator for Non-Stationary Environments*, in Proceedings of the Congress on Evolutionary Computing, pp: 25-31.

[North, Collier, and Vos, 2006] North, M.J., Collier, N.T., and Vos, G.R. 2006. *Experiences creating three implementations of the repast agent modelling toolkit*, ACM Transactions on Modelling and Computer Simulation, Vol. 16 No.1, pp: 1-25.

[North et. al, 2007] North, M.J., Tatara, E., Collier, N.T., and Ozik, J. 2007. *Visual Agent-based Model Development with Repast Simphony*, in Proceedings of the Agent 2007 Conference on Complex Interaction and Social Emergence, Argonne National Laboratory, Argonne, IL USA.

[Oxford Dictionary, 2005] Oxford Dictionary of English. 2nd revised ed. Oxford: Oxford University Press. 2005.

[Pattison and Robins, 2002] Pattison, P. and Robins, G. 2002. *Neighborhood-based models for social networks* Sociological Methodology, pp: 301-337.

[Peng, 2005] Peng, B. 2005. *Knowledge Swarms in Cultural Algorithms for Dynamic Environments*, Ph.D. Thesis, at Wayne State University.

[Reynolds, 1978] Reynolds, R.G. 1978. *On Modeling the Evolution of Hunter-Gatherer decision-Making Systems,* Geographical Analysis 10(1), pp: 31-46.

[Reynolds, 1994] Reynolds, R.G. 1994. *An introduction to cultural algorithms*, in Proceedings of the 3rd Annual Conference on Evolutionary Programming, World Scientific Publishing, London, pp: 131-139.

[Reynolds, 1999] Reynolds, R.G. 1999. *An overview of Cultural Algorithms, Advances in Evolutionary Computation*, D. Corne, M. Dorigo and F. Glover, Eds. New Yourk: McGraw-Hill, pp: 367-378.

[Reynolds, 2003] Reynolds, R.G. 2003. *Predicting the Past Using Multi-Agent Models of Cultural Evolution*, Midwest Sociological Society, Chicago, IL., April 16-19.

[Reynolds and Ali, 2007] Reynolds, R.G. and Ali, M.Z. 2007. *Exploring knowledge and population swarms via an agent-based Cultural Algorithms Simulation Toolkit (CAT),* Evolutionary Computation, 2007. CEC 2007. IEEE Congress on , vol., no., pp: 2711-2718, 25-28.

[Reynolds and Ali, 2008] Reynolds, R.G. and Ali, M.Z. 2008. *Computing With the Social Fabric: The Evolution of Social Intelligence within a Cultural Framework,* IEEE Computational Intelligence Magazine, pp: 18-30.

[Reynolds, Ali, and Alomari, 2006] Reynolds, R.G., Ali, M.Z., and Alomari, R.S. 2006. *Optimization Problem Solving using Predator/Prey Games and Cultural Algorithms*. CIG 2006, pp: 119-125.

[Reynolds, Che, and Ali, 2010] Reynolds, R.G., Che, X., and Ali, M.Z. 2010. *Weaving the social fabric: The past, present and future of optimization problem solving with cultural algorithms*, International Journal of Intelligent Computing and Cybernetics, Vol. 3 Iss: 4, pp: 561–592.

[Reynolds and Chung, 1997] Reynolds, R.G. and Chung, C. 1997. *A Cultural Algorithm to Evolve Multi-Agent Cooperation Using Cultural Algorithms*, in Evolutionary Programming VI, P. J. Angeline, pp: 323-334.

[Reynolds and Gawasmeh, 2012] Reynolds, R.G. and Gawasmeh, Y.A. 2012. *Evolution of heterogeneous Social Networks for Optimization and Problem Solving in Cultural Algorithms*. In proceedings of 2012 IEEE World congress on Computational Intelligence, Brisbane, Australia, pp: 1-8.

[Reynolds and Liu, 2011] Reynolds, R.G. and Liu, D. 2011. Multi-objective cultural algorithms. In proceedings of 2011 IEEE Congress on Evolutionary Computation, pp: 1233-1241.

[Reynolds, Kobti, and Kohler, 2003] Reynolds, R.G., Kobti, Z., and Kohler, T. 2003. *A Multi-Agent Simulation Using Cultural Algorithms: The Effect of Culture on the Resilience of Social Systems*, Proceedings 2003 IEEE Proceedings of Congress on Evolutionary Computation, Canberra, Australia, December 8-12, pp: 1743-1750.

[Reynolds, Kobti, and Kohler, 2004] Reynolds, R.G., Kobti, Z., and Kohler, T. 2004. *The Effect of Culture on the Resilience of Social Systems in the Village Multi-Agent Simulation*, in Proceedings of IEEE International Congress on Evolutionary Computation, Portland, OR, June 19, 24, pp: 1743-1750.

[Reynolds et. al, 2011] Reynolds, R.G., O'Shea, J., Che, X., Gawasmeh, Y.A., Meadows, G., and Fotouhi, F. 2011. *Agile Design of Reality Games Online*. Book Chapter, Source Title: Multi-Agent Applications with Evolutionary Computation and Biologically Inspired Technologies: Intelligent Techniques for Ubiquity and Optimization.

[Reynolds and Peng, 2005] Reynolds, R.G. and Peng, B. 2005. *Knowledge Learning and Social Swarms in Cultural Algorithms. Journal of Mathematical Sociology*. London, Routledge. Vol 29, pp: 1-18.

[Reynolds and Saleem, 2001] Reynolds, R.G., and Saleem, S. 2001. *The impact of environmental dynamics on cultural emergence,* Perspectives on Adaptation in Natural and Artificial Systems, pp: 253–280.

[Reynolds, Whallon, and Goodhall, 2002] Reynolds, R.G., Whallon, R., and Goodhall, S. 2002. *The Impact of Resource Access on Learning by Emulation in Hunter-Gatherer Foraging Systems: A Multi-Agent Model*, In Proceedings of World Congress on Computational Intelligence, May 12-19, Honolulu, Hawaii.

[Stefan et. al, 2003] Stefan, J., Reynolds, R.G., Fotouhi, F., Aristar, A., Lu, S., and Dong, M. 2003. *Evolution Based Approaches to the Preservation of Endangered Natural Languages*, in Proceedings of the IEEE International Congress on Evolutionary Computation, Australia, pp: 1980-1987.

[Vitale et. al, 2011] Vitale, K., Reynolds, R.G., O'Shea, J., and Che, X. 2011. *Learning group behavior in games: Using Cultural Algorithms: The land bridge game engine example*, Swarm Intelligence IEEE Symposium, pp: 1-9.

ABSTRACT

EVOLVING HETEROGENEOUS AND SUBCULTURED SOCIAL NETWORKS FOR OPTIMIZATION PROBLEM SOLVING IN CULTURAL ALGORITHMS

by

YOUSOF GAWASMEH

August 2015

Advisor: Professor Robert G. Reynolds

Major: Computer Science

Degree: Doctor of Philosophy

Cultural Algorithms are computational models of social evolution based upon principle of Cultural Evolution. A Cultural Algorithm are composed of a Belief Space consisting of a network of active and passive knowledge sources and a Population Space of agents. The agents are connected via a social fabric over which information used in agent problem solving is passed. The knowledge sources in the Belief Space compete with each other in order to influence the decision making of agents in the Population Space. Likewise, the problem solving experiences of agents in the Population Space are sent back to the Belief Space and used to update the knowledge sources there. It is a dual inheritance system in which both the Population and Belief spaces evolve in parallel over generations.

A question of interest to those studying the emergence of social systems is the extent to which their organizational structure reflects the structures of the problems that are presented to them. In a recent study [Reynolds, Che, and Ali, 2010] used Cultural Algorithms as a framework in which to empirically address this and related questions. There, a problem generator based upon Langton's model of complexity was used to produce multi-dimensional real-valued problem landscapes of varying complexities. Various homogeneous social networks were then

tested against the range of problems to see whether certain homogeneous networks were better at distributing problem solving knowledge from the Belief Space to individuals in the population. The experiments suggested that different network structures worked better in the distribution of knowledge for some optimization problems than others. If this is the case, then in a situation where several different problems are presented to a group, they may wish to utilize more than one network to solve them. In this thesis, we first investigate the advantages of utilizing a heterogeneous network over a suite of different problems. We show that heterogeneous approaches begin to dominate homogeneous ones as the problem complexity increases. A second heterogeneous approach, Subcultures Model, will be introduced by dividing the social fabric into smaller networks.

The three different social fabrics (homogeneous, heterogeneous and Subcultures) were then compared relative to a variety of benchmark landscapes of varying entropy, from static to chaotic. We show that as the number of independent processes that are involved in the production of a landscape increases, the more advantageous subcultures are in directing the population to a solution. We will support our results with t-test statistics and social fabric metrics performance analysis.

AUTOBIOGRAPHICAL STATEMENT

YOUSOF GAWASMEH

Mr. Yousof Gawasmeh is currently a PhD Candidate in Department of Computer Science, Wayne State University. He joined Wayne State in 2007. He received his Bachelor's degree in Computer Science from the Yarmouk University in Jordan. He subsequently received a Master of Science degree from the New York Institute of Technology. He is the currently a member of "Artificial Intelligence Lab" studying under the direction of Prof. Robert Reynolds. Yousof's research concerns "Evolving Heterogeneous and Subcultures Networks for Optimization and Problem Solving Using Cultural Algorithms" and has recently produced several articles in that area. He is also interested in Game Programming, Big Data, Data Mining, and Multi-Objective Cultural Algorithms.

In conjunction with his research activities, Yousof had been a Graduate Teaching Assistant in the Computer Science Department at Wayne State University and is interested in teaching both Computer Hardware and Software classes. As a Graduate Teaching Assistant for five consecutive years (2007 to 2012), he had taught courses in both Computer Hardware and Software including CSC 2000, CSC 2200, and CSC 4420. He had received the best Graduate Teaching Assistantship award in 2011 as a recognition for his excellence in teaching.

Prior to joining Wayne State University he had served as an instructor in Software Engineering Department at Philadelphia University – Jordan for two years (2005-2007). In the last two years, Yousof worked as a software developer for Unisys and General Motors – OnStar. He has recently joined Amazon.com.