1-1-2015

The Impact Of Increased Optimization Problem Dimensionality On Cultural Algorithm Performance

Yang Yang
Wayne State University,

THE IMPACT OF INCREASED OPTIMIZATION PROBLEM DIMENSIONALITY ON CULTURAL ALGORITHM PERFORMANCE

by

YANG YANG

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2015

MAJOR: COMPUTER SCIENCE

Approved By:

_____

Advisor Date

ACKNOWLEDGMENTS

I would like to acknowledge the contributions of my advisor Dr. Robert G. Reynolds and my committee members Dr. Jing Hua, Dr. Loren Schwiebert. I also would like to acknowledge the others in my research team without which this work would not have been possible: Thomas Palazzolo, Dustin Stanley, Areej Salaymeh and David Warnke.

TABLE OF CONTENTS

ACKNOWLEDGMENTS ..... **ii**

LIST OF TABLES ..... **vi**

LIST OF FIGURES ..... **vii**

CHAPTER 1: INTRODUCTION **1**

CHAPTER 2: THE CONE'S WORLD: A COMPLEX SYSTEMS TEST BED .. **4**

2.1 Introduction to Complex Systems 4

2.2 The Cone's World Generator 5

CHAPTER 3: THE LEARNING COMPONENT OF THE SIMULATION: CULTURAL ALGORITHMS **10**

3.1 Introduction to the Cultural Algorithm 10

3.2 Belief Space and Knowledge Sources 12

3.2.1 Normative Knowledge 13

3.2.2 Situational Knowledge 14

3.2.3 Domain Knowledge 15

3.2.4 Historical Knowledge 15

3.2.5 Topographical Knowledge 16

3.3 Communication Protocol 18

3.3.1 Acceptance Function 19

3.3.2 Influence Function 19

3.3.3 Update Function 21

CHAPTER 4: SOCIAL FABRIC AND SOCIAL METRICS 23

4.1 Social Fabric 23

4.2 Neighborhood Topology 25

4.3 Agent Decision Making 25

4.4 Social Metrics 29

4.4.1 The Social Tension 31

4.4.2 Minority / Majority Win Scores and Innovation Cost 33

CHAPTER 5: INTRODUCTION OF THE CULTURAL ALGORITHMS TOOLKIT 2.0 SYSTEM 35

5.1 Repast as Development Environment 35

5.2 Cone's World Generation 36

5.3 Main Simulation Loop 38

5.4 Instructions of GUI 40

CHAPTER 6: EXPERIMENTAL FRAMEWORK AND RESULTS 45

6.1 Data and Results Format 45

6.2 Experiment Framework 46

6.3 Experiment Results 48

6.3.1 Performance in Different Dimensions 48

6.3.2 Knowledge Source Performance 50

6.3.3 Social Metrics Summary Tables 52

CHAPTER 7: SUMMARY RESULTS AND ANALYSIS 55

7.1 Introduction 55

7.2 Overall Performance Comparison 55

7.3 Knowledge Source Performance Comparison..... 58

7.4 Social Metrics Summary 62

CHAPTER 8: CONCLUSIONS AND FUTURE WORK 68

8.1 Conclusion 68

8.2 Future Work 69

REFERENCES 70

ABSTRACT 72

AUTOBIOGRAPHICAL STATEMENT 74

LIST OF TABLES

Table 6.1 The Raw Data Example Part 1 46

Table 6.2 The Raw Data Example Part 2 46

Table 6.3 The Test Run Array for the Dimension / Complexity..... 47

Table 6.4 The Performance Comparison in 2 Dimensions..... 48

Table 6.5 The Performance Comparison in 3 Dimensions..... 48

Table 6.6 The Performance Comparison in 4 Dimensions..... 49

Table 6.7 The Performance Comparison in 5 Dimensions..... 49

Table 6.8 The KS Performance Comparison in 2 Dimension..... 51

Table 6.9 The KS Performance Comparison in 3 Dimension..... 51

Table 6.10 The KS Performance Comparison in 4 Dimension..... 52

Table 6.11 The KS Performance Comparison in 5 Dimension..... 52

Table 6.12 The Social Metrics Summary in 2 Dimension 53

Table 6.13 The Social Metrics Summary in 3 Dimension 53

Table 6.14 The Social Metrics Summary in 4 Dimension 54

Table 6.15 The Social Metrics Summary in 5 Dimension 54

Table 7.1 The Summary of Performance Comparisons Part 1 55

Table 7.2 The Summary of Performance Comparisons Part 2 57

Table 7.3 The Summary of Knowledge Source Comparisons 59

Table 7.4 The T-test Results Table 61

Table 7.5 The Summary of Social Metrics Comparisons 63

Table 7.6 The Statistical Expression of Social Tension Cool Down 66

LIST OF FIGURES

Figure 2.1 An Example Landscape In Two-Dimensional Space 7

Figure 2.2 The Logistic Function 8

Figure 2.3 The Logistic Function with Specific A Values 9

Figure 3.1 The Basic Pseudo-code for Cultural Algorithms 11

Figure 3.2 The Schematic of Cultural Algorithms 12

Figure 3.3 The Structure of Normative Knowledge 13

Figure 3.4 The Structure of Situational Knowledge 14

Figure 3.5 The Structure of Topographical Knowledge 16

Figure 3.6 The Pseudo-code for the Topographical Knowledge Influence Function 18

Figure 3.7 The Knowledge Update in the Belief Space 21

Figure 4.1 The Social Fabric Schema 24

Figure 4.2 Some Example Neighborhood Topologies 25

Figure 4.3 Knowledge Source Interaction at the Population Level 26

Figure 4.4 Majority Win Conflict Resolution in the Social Network 27

Figure 4.5 Weighted Majority Win Conflict Resolution in the Social Network 29

Figure 4.6 An Embedded Social Fabric Component in CAT 31

Figure 4.7 The Pseudo-code for Calculating the Social Tension 32

Figure 4.8 The Pseudo-Code for Calculating the Minority Win Score, Majority Win Score, and Innovation Cost Index 34

Figure 5.1 Choosing of A Value in the Logistic Function 36

Figure 5.2 A 2D Landscape Example A = 1.01 37

Figure 5.3 A 2D Landscape Example A = 3.35 38

Figure 5.4 A 2D Landscape Example A = 3.99 38

Figure 5.5 The Repast Control Bar 40

Figure 5.6 Parameter Setting in the GUI 41

Figure 5.7 The Cone's World 2D Landscape Display 42

Figure 5.8 The Best Individual Fitness Graph 43

Figure 5.9 The Overall Social Tension Graph 44

Figure 5.10 The Weighted Majority Win Metrics Graph 44

Figure 7.1 The Social Tension Graph of Run #55 64

Figure 7.2 The Fitness Graph of Run #55 64

Figure 7.3 The Social Tension Graph of Run #28 65

Figure 7.4 The Social Tension Graph of Run #60 65

CHAPTER 1: INTRODUCTION

Evolutionary Computation is a subfield of Artificial Intelligence which is based on the Darwinian principles of evolution. Evolutionary Computation is often applied when solving complex computational problems, especially global optimization problems. Several Evolutionary Computation systems have been proposed, and one of them is the Cultural Algorithms system [Reynolds, 1979, 1994]. The Cultural Algorithm (CA) is a class of computational models that imitate the cultural evolution process occurring in nature. CA has three major components: a population space, a belief space, and a protocol that describes how knowledge is exchanged between the first two components. The population space can support any population-based computational model, such as Genetic Algorithms [Holland 1975], Evolutionary Programming, etc.

Cultural Algorithms have been successfully applied in many disparate problems, and all of these problems have one characteristic in common – they are all complex systems. The complex systems approach studies how relationships between the parts of a system give rise to the collective behavior of the system, and how the system interacts and forms relationships with its environment. The Cones World, developed by Morrison and De Jong [1999], will be used in this thesis as the test environment for the study of complex systems.

Peng [Peng and Reynolds 2004] selected the Cones World to test various CA configurations. Later, Ali [Ali 2008] embedded the CA framework within the Recursive Porous Agent Simulation Toolkit (Repast) [North, Howe et al. 2005]. He produced a toolkit which is now called the Cultural Algorithms Simulation Toolkit (CAT) [Reynolds and Ali

2008]. Ali extended Peng's CA framework in his CAT system, adding a social fabric to enhance the performance of the algorithm. Subsequently, Che [Che 2009] extended the existing models to produce a new version of the Cultural Algorithms Toolkit, namely CAT 2.0.

All the experiments conducted by Ali [Ali 2008] and Che [Che 2009] focused on optimization problems in 2-dimensional landscapes. To build on this existing research, in this paper, our goal is to investigate the influence of problem dimensionality on the performance of Cultural Algorithms. The following list summarizes the major concerns in our research:

1. What is the impact that the increased problem dimensionality has on the effectiveness of the Cultural Algorithm optimization problems?
2. What is the impact that the increased dimensionality has on Cultural Algorithm performance with regard to specific complexity classes?
3. What is the impact that the increased dimensionality has on the effectiveness of the knowledge sources in directing the optimization search process?
4. How does the Social Fabric affect the performance of the population in different dimensionalities?
5. What is the utility of the social metrics that we used as an aid in understanding the behaviors of Cultural Algorithm?

The outline of this thesis is as follows: Chapter 2 describes the complex system environment in which our experiments were conducted. Chapter 3 highlights the design and implementation of the Cultural Algorithms. Chapter 4 introduces the social fabric

used in our cultural system, and describes the social metrics involved that are used to measure the performance of the system. Chapter 5 introduces the Cultural Algorithms Toolkit 2.0. Chapter 6 discusses the experimental framework of the system, and describes the results in detail. Chapter 7 discusses the results. Chapter 8 summarizes our findings, and presents directions for future work.

CHAPTER 2: THE CONE'S WORLD: A COMPLEX SYSTEMS TEST BED

2.1 Introduction to Complex Systems

A complex system is a combination of related components combined through basic interactions. The interaction between these components or basic agents can potentially produce emergent behaviors that cannot be predicted from knowledge of the individual agents alone.

A complex system has the following features [Holland, 1992]:

- Complex systems are non-linear. Small changes in inputs can cause large or very significant changes in outputs.

- Complex systems have feedback loops. Any interaction can direct feedback to itself instantly or after some stages.

- Complex systems are open, i.e. usually far from equilibrium, but they may form pattern stability.

- Complex systems have memory. They evolve, but their past influences their present behavior.

- Complex systems may produce emergent phenomena.

Reynolds [Reynolds, Whallon, et al., 2006] stated that a complex system is one that consists of an organized group of heterogeneous, independent agents. The agents interact with each other and with their environments, and adapt the environment through their feedback. The separate behaviors of the agents, when combined, can cause higher-level behaviors to emerge from the whole group that works together to solve the

problems they face, at the group level. In this chapter, we briefly introduce the complex systems environment that we have used to test Cultural Algorithms.

2.2 The Cone's World Generator

The Cone's World was developed by De Jong and Morrison [Morrison, De Jong, et al. 1999] in order to test the ability of evolutionary algorithms to solve arbitrary complexity problems. The Cones World was an implementation of a complex systems model originally proposed by Christopher Langton [Langton 1992]. Peng [Peng and Reynolds, 2004; Reynolds, Peng, et al. 2005] coined the term "Cone's World" when she tested various Cultural Algorithms configurations. Ali [Ali 2008] made an extension to Peng's Cultural Algorithms framework in his Culture Algorithm Toolkit (CAT) system. He made the Cone's World problem environment available to system users, along with the other traditional benchmark problems. Then, Che [Che 2009] used the Cone's World to examine the new Social Fabric approach in the extended CAT 2.0 system.

The Cone's World Generator creates landscapes of the test problem. In this landscape, a number of cones with different heights and slopes are randomly located in a multi-dimensional space. The Cone's World Generator algorithm has two steps:

1) Initializing a fundamental static landscape with the chosen complexity.

2) Applying the dynamics of the logistic function to adjust the landscape.

The landscape is given by the following formula:

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

In this formula:

k: The number of cones.

n: The dimensionality.

H_j: The height value of cone j.

R_j : The slope value of cone j.

$C_{j,i}$: The coordinate of cone j in dimension i.

The values for each cone (H_j, R_j and $C_{j,i}$) are randomly given by the following user-specified ranges:

$H_j \in$ (Hbase, Hbase + Hrange)

$R_j \in$ (Rbase, Rbase + Rrange)

$C_{j,i} \in$ (-1, 1)

When cones are randomly distributed over the space, they may overlap. When an overlap occurs, the value at that overlap point is computed using the max function. The final height at a point comes from the height of the cone with the largest value at same position, when two cones overlap. This cone generation function can be specified for any number of dimensions. Each time the generator is called, it produces a randomly generated real-valued surface in which random values for each cone are assigned, based on user-specified ranges.

We have used landscapes of multiple dimensions in all of the experiments of this thesis. But first we need to show two-dimensional examples to describe some concepts for simplicity and for visualization purposes, although the patterns and rules discussed also apply to scenarios with more than two dimensions. An example two-dimensional landscape with k = 15, Hbase = 1, Hrange = 9, Rbase = 8, and

Rrange = 12, is given in Figure 2.1 below.

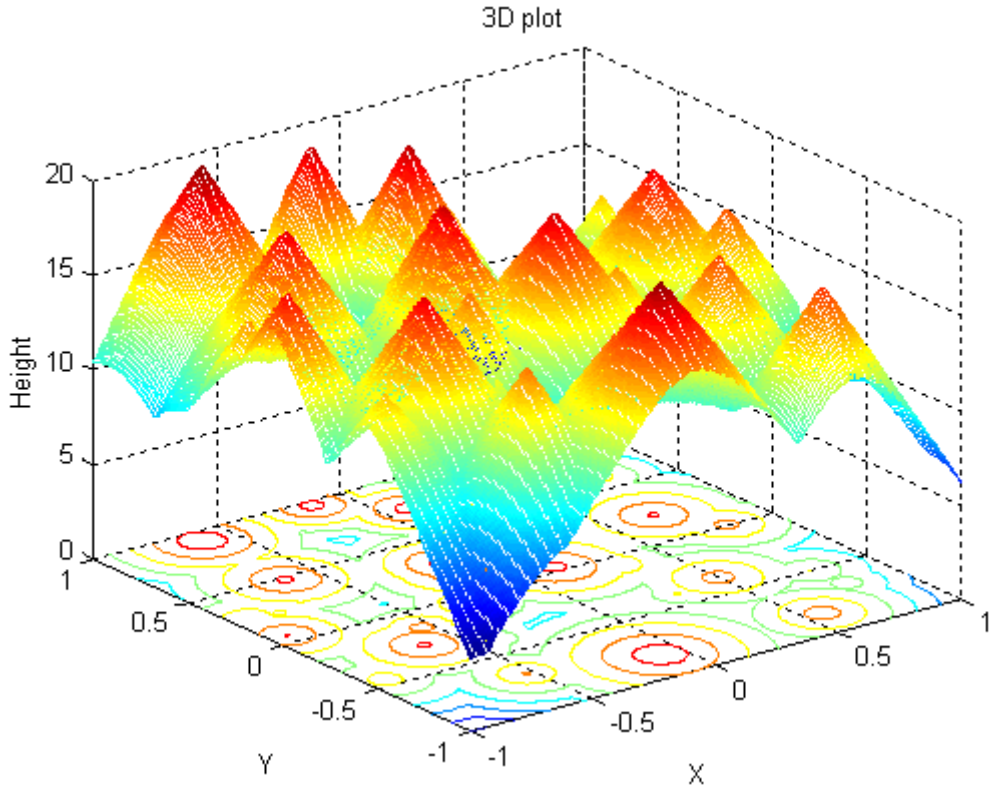

Figure 2.1 An Example Landscape In Two-Dimensional Space

x ∈ (-1.0, 1.0), y ∈ (-1.0, 1.0) with n = 50, H ∈(1, 10), and R ∈ (8, 20)

The next step is to apply the dynamics. Each cone's parameter (coordinate $C_{j,i}$, height value H_j, and slope value R_j) can be modified independently. With the aim of controlling the complexity, the logistics function is used in this step as shown below:

$$Y_i = A \times Y_{i-1} \times (1 - Y_{i-1})$$

In this formula, A is a constant, Y_i is the value at iteration i.

A bifurcation map generated by this function is shown in Figure 2.2. This figure shows that the value Y can be generated in each iteration of the logistic function, if the values of A are in the range of 1.0 to 4.0. The value of A, chosen for each of the dynamic features, identifies whether the movements are same small-sized steps, same

large-sized steps, differently sized steps, or chaotically sized steps.

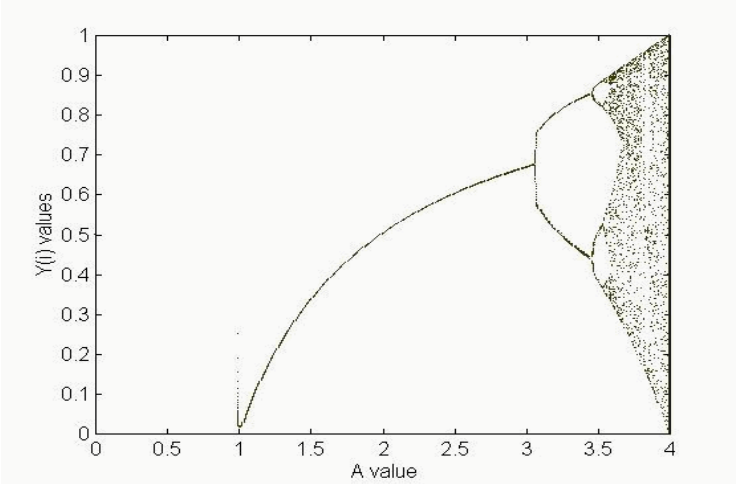

Figure 2.2 The Logistic Function

Che [Che 2009] was interested in a few typical values of A. Figure 2.3 shows the complexities that he selected. He picked A = 1.01, 3.35, and 3.99 for his test environment complexity. A = 1.01 corresponded to one step change, 3.35 corresponded to two steps change and 3.99 corresponded to a totally chaotic step size change. By applying the logistic function to the parameters of the Cone's World Generator, we are able to control the complexity of the generated landscape by providing the A value of the logistics function. Therefore, it is evident that we can generate problem landscapes at different levels of complexities, from static to periodic to chaotic.

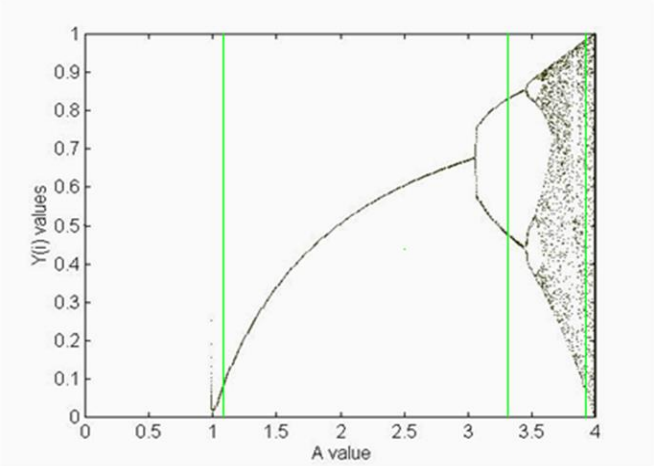

Figure 2.3 The Logistic Function with Specific A Values

This feature enables us to evaluate our model in a more flexible and systematic way. It is also a reasonable facsimile of how resources are spread out within natural environments. From the information theory point of view, the problem environment carrying certain complexities of information could be represented by entropy.

CHAPTER 3: THE LEARNING COMPONENT OF THE SIMULATION: CULTURAL ALGORITHMS

3.1 Introduction to the Cultural Algorithm

In the 1970s, a class of evolution programming models called Cultural Algorithms was developed by Dr. Robert Reynolds [Reynolds, 1979, 1994]. When building Cultural Algorithms, Dr. Reynolds drew an analogy between group learning, the Darwinian theory of natural selection, and the process of the group knowledge acquired in the past influencing current decisions by the individuals in a group. The Cultural Algorithm is a computational model simulating the cultural evolution process occurring in nature.

Cultural Algorithms consists of three main components: the Population Space, the Belief Space and the Communication Protocol. The Population Space is defined as a set of possible solutions to a problem. These individuals are connected by a Social Fabric over which information can be passed. The Belief Space can be defined as the collection of experiential knowledge of individuals within the Population Space, according to their varying degrees of successes. The Belief Space also has the ability to influence the succeeding generations of individuals within the Population Space. The Communication Protocol defines how knowledge is exchanged between the first two components.

The following is a general statement of a generic Cultural Algorithm:

1. The Population Space and the Belief Space are initialized.

2. Individuals in the Population Space are first evaluated and ranked through a fitness function.

3. The function Accept () is used to decide which individuals within the Population

Space are acceptable to update the Belief Space.

4. The function Update () is used to store the experiences of those accepted individuals into the Belief Space.

5. The function Influence () is used the knowledge stored in the Belief Space to influence the selection of individuals for the next generation of the population. Operators are applied to at least some of the children, which transforms them into mutated variants of their parents.

6. Steps 2 to 5 comprise the evolution loop which is repeated until the termination condition is satisfied.

The basic CA process [Reynolds, 1979, 1994] in the pseudo-code is represented in Figure 3.1:

```
Begin
    t = 0
    InitPop(t)          // init population
    InitBelief(t)       // init belief space
    Repeat
        EvaluatePop(t)
        Update(Belief(t), Accept(Pop(t)))
        Generate(Pop(t), Influence(Belief(t)))
        t++
        Select Pop(t) from Pop(t – 1)
    Until (termination condition)
End
```

Figure 3.1 The Basic Pseudo-code for Cultural Algorithms

In the evolution loop, the Population Space and the Belief Space support and interact with each other in an approach similar to the evolution of human cultures.

A visualization of this process [Reynolds, 1979, 1994] can be found in the following

diagram:

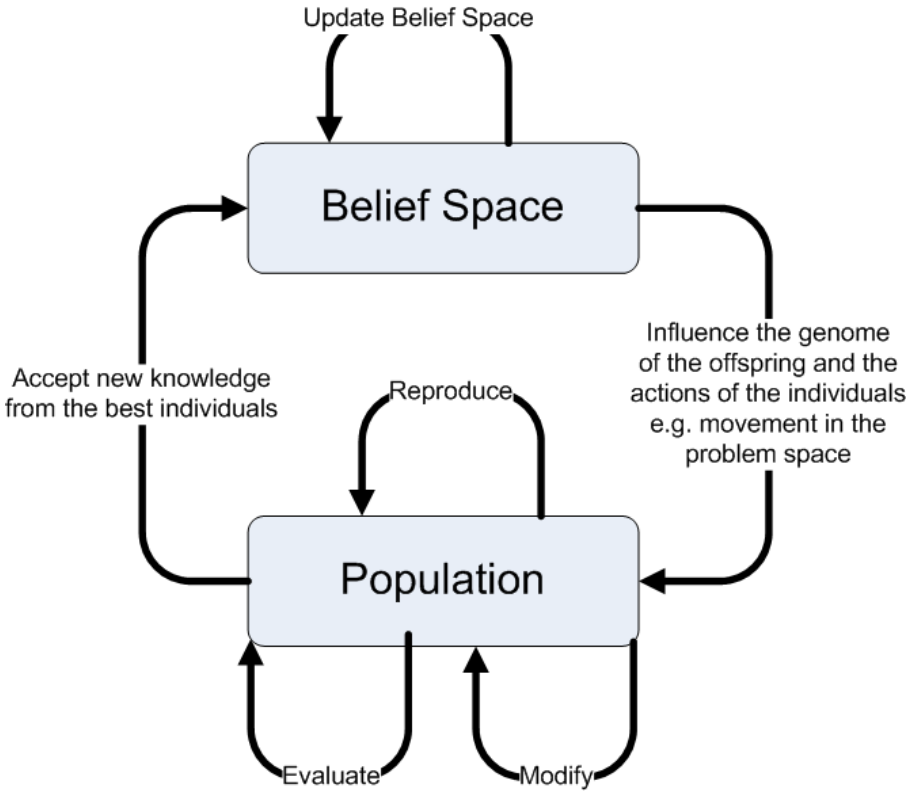

Figure 3.2 The Schematic of Cultural Algorithms

3.2 Belief Space and Knowledge Sources

Cultural knowledge can be subdivided into five basic types. Each of the five basic knowledge types was developed to allow evolution-based optimization for a given domain. For each of them, efficiency and functionality is important for the system to perform well. To accommodate more general situations in our research, we have used all of the KS implementations from previous Cultural Algorithm systems. In this section, we describe each of the five knowledge sources in terms of their definition, data structure, and influence mechanisms. In the following sections, we use some of the mathematical symbols listed below:

The dimension of the optimization problem, n.

Parent individual: X <$x_1,\ x_2, x_n$>

Individual Children: Y <$y_1, y_2,\ y_n$>

3.2.1 Normative Knowledge

Normative knowledge was introduced by Chung [Chung, 1998]. Its data is a set of variable ranges, and each range is expected to be an acceptable solution for a parameter. In the case of function optimization, Normative Knowledge contains a set of ranges for each dimension in the Cone's World. These intervals characterize the range of what are believed to be good areas for searching in each dimension. Consequently, Normative Knowledge provides standards and guidelines for individual behaviors.

The Normative Knowledge data structure that has been used during this thesis is shown in Figure 3.3:

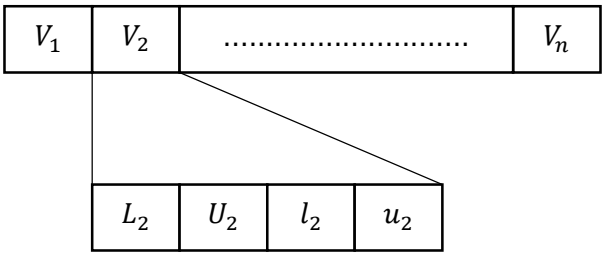

Figure 3.3 The Structure of Normative Knowledge

For each variable,V_i, the data structure holds the upper bounds (u_i) and the lower bounds (l_i), and the performance value of each of the upper bounds and the lower bounds, L_i, and U_i.

3.2.2 Situational Knowledge

The idea of Situational Knowledge was also stated by Chung [Chung and Reynolds, 1998] for problem solving in static environments. Situational knowledge maintains a set of exemplars selected from the Population Space. The data structure of the Situational Knowledge is shown in Figure 3.4.

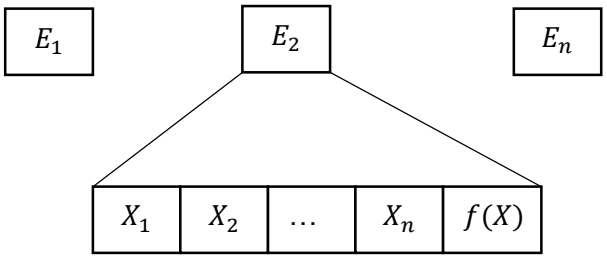

Figure 3.4 The Structure of Situational Knowledge

Each exemplar holds each parameter's value, and its own fitness value in the end. In Che's version, Situational Knowledge will be updated by adding the top ranking individuals to the Situational Knowledge structure that holds the existing elite collection. Cultural Algorithms can take this into account and look for similar solutions that might be even better. Situational Knowledge can contain both positive and negative exemplars. Solutions that score high are considered positive exemplars, and in contrast, solutions that score low are considered negative exemplars. Since our problems concern optimization the situational knowledge structure is elitist and contains only the top performing individuals seen so far.

The Situational Knowledge component keeps track of the best solutions, or positive exemplars, found in each generation. This mechanism allows high performance plans to be present and rewarded in future generations.

After each generation, all the elites in the Situational Knowledge will be distributed to a roulette wheel called the BestCaseWheel, based on their fitness value. When an individual chooses the Situational Knowledge source to influence the next generation, the BestCaseWheel will be spun first, and the resultant outcome will be the lucky elite for the individual to follow. The new location of this individual will be randomly chosen, but close to the selected elites. Previous Cultural Algorithms that were used to benchmark specific problems only had one best case in the knowledge base. Che increased the knowledge base size to enable it to accommodate more complex problems.

3.2.3 Domain Knowledge

The Domain Knowledge component was introduced into the Cultural Algorithm system by Saleem [Reynolds and Saleem, 2001] for solving dynamic resource optimization problems. This improvement allowed them to predict the gradients of resources. The purpose of Domain Knowledge is to characterize relationships between objects in the search space that can be used to predict aspects in the problem landscape. For example, the equation of the cone can be used to predict the value of the performance landscape at a given point. So the equations expressing relationships between cone parameters will constitute domain knowledge here.

3.2.4 Historical Knowledge

Historical Knowledge, too, was introduced into Cultural Algorithms by Saleem [Reynolds and Saleem, 2001]. It stores important events and the general state during the

search, in order to investigate global dynamics and to backtrack or retrace actions. The Historical Knowledge component contains sequences of environmental changes for shifts in the direction or distance of the optimal point in the search space. It can contain a record of good and bad solutions that have occurred in the past, so that the future agents can go towards or avoid those solutions. It is particularly useful if the environment contains a dynamic component that causes it to change over time. The History Knowledge can be used to document patterns in these changes.

3.2.5 Topographical Knowledge

Topographical Knowledge was first devised as a knowledge source which was introduced into Cultural Algorithms by Reynolds and Jin [Jin and Reynolds, 1999], who initially called it “regional schema”. Topographical Knowledge concerns the regional features of the search space. It is able to ignore whole ranges of infeasible solutions, which both reduces the opportunity for error and cuts down on the search time.

The structure of Topographical Knowledge is shown in Figure 3.5.

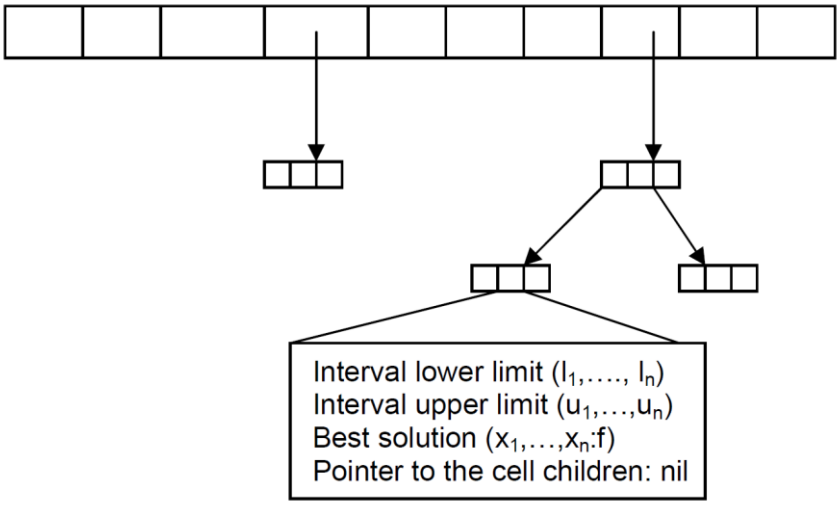

Figure 3.5 The Structure of Topographical Knowledge

Topographical Knowledge is represented here in a multi-dimensional search space, with cells in the grid described as $C_1,\ldots C_i,\ldots C_n$. C_i stands for the cell size of the i^{th} dimension. The data structure of Topographical Knowledge is an array of size n, where n is the quantity of cells in the grid. Each cell contains a lower bound and an upper bound for the n variables ($(l,u)_1,\ldots (l,u)_n$), which indicate the ranges of the best solutions found in that cell so far. And a cell may store pointers for its children.

The implementation detail of Topographical Knowledge is as follows. First the whole search space is first divided into t cells $(C_1, C_2,\ldots C_t)$. During the search process, each cell can be sub-divided into smaller cells recursively, and organized into a hierarchical tree structure. The initial t cells form the top / root level. Cells without sub-cells become leaf cells, and the leaf cells cover the entire search space. Each cell saves the cell-best individual, cellBestInd. Good cells are defined as the top N cells (based on cellBestInd) from the initial cell set.

The pseudo-code of Topographical Knowledge influence function is shown below.

```
for each (parent cell X)
    Find X's host cell C_h and C_h's best individual cb_h
    if(search is in progress / improving)
        if(C_h is a good cell)
            Y = mutate (X)
        else
            Pick one good cell k, 1≤ k ≤ t
            Y = mutate (cb_k)
        endif
    else
        // no progress
        If(parent X is better than cb_h)
            Y = mutate (X)
        else
            Select one cell C_s from the top level cells
            Y = mutate (cb_s)
        endif
    endif
endfor
```

Figure 3.6 The Pseudo-code for the Topographical Knowledge Influence Function

When the Topographical Knowledge structure is initialized, a solution point in every cell is sampled, and a list of the best cells was generated. Topographical Knowledge was updated when a cell was divided into several sub-cells, or if an accepted individual was better than the best solution in that cell. Updates also occurred when the fitness value of the cell's best solution had increased after a change-inducing event.

3.3 Communication Protocol

The five knowledge sources described above presented interesting behaviors regarding different roles in the search process. All these knowledge sources have been updated and integrated by the communication protocol.

The Communication Protocol of a Cultural Algorithm System has three major components: the Acceptance function, the Influence function and the Update function. The Acceptance function determines which individuals from the current population are

able to update the Belief Space. The Influence function determines how the Belief Space influences the Population Space when generating new solutions. The Update function manages the update actions of each individual knowledge source. We begin our discussion with the Acceptance function.

3.3.1 Acceptance Function

The Acceptance function determines which individuals from the current population are acceptable to impact the Belief Space. The Acceptance function in this project will compare the fitness value of each individual in the population and select a subset of the best individuals in the population space to update the Belief Space. The updated Belief Space will then influence the Population Space, and direct the decision-making for the next generation as described in the following section.

3.3.2 Influence Function

In this chapter, we have introduced the five basic knowledge sources that are used in the basic Cultural Algorithm system. It is important to find a method to integrate the basic knowledge sources over the population when multiple knowledge sources are used together. The earliest Influence function was an arbitrary integration function employed by Saleem [Reynolds and Saleem, 2001]. Then, the Marginal Value Theorem was developed by Peng [Peng and Reynolds, 2004], which allowed a simple interaction between the knowledge categories to make use of the co-evolutionary relationship between the Belief and the Population spaces. Peng used a co-evolutionary analogy, the

predator-prey relationship, as the basis for extending the Influence function to integrate the influence of all of the knowledge source categories via a roulette wheel model. The Knowledge sources were the predators and the individuals in the population space the prey. The better a Knowledge Source was at improving performance in the population the larger the area it occupied on a roulette wheel. The wheel was spun for each individual in the population every time step to see which Knowledge Source would influence them. Next, Ali employed the Social Fabric, which extended the Influence function to certain individuals selected through the Marginal Value Theorem method. A basic majority voting scheme was then applied to the individuals to determine which knowledge source will impact them. Ali's initial version was dynamic in that individual connections were not continued although the network topology type did not change. Following Ali's work, Che extended the Social Fabric to let the fixed communication links between individuals in the population support the spread of influence of the knowledge sources through the network. The Social Fabric is described in greater detail in Chapter 4.

Individual knowledge sources were selected at random in a basic way that was to normalize the performance of each of the knowledge sources, and assign each to a portion of the wheel, relative to their performance. The knowledge source that had greater fitness value would have more opportunities to guide the next generation. At each generation, the roulette wheel was to be updated and the average performance of each knowledge source was to be recalculated. At the start of each new generation, the roulette wheel was spun for each individual in order to assign them a knowledge source that would reflect their direct influence. Next, they received the direct influence of their

neighbors and pooled together their influences to make a decision regarding what knowledge source will control them in the same time. Here, the pooling process has been based upon a majority voting decision.

3.3.3 Update Function

An example of the knowledge update process in the Belief Space is summarized in Figure 3.7.

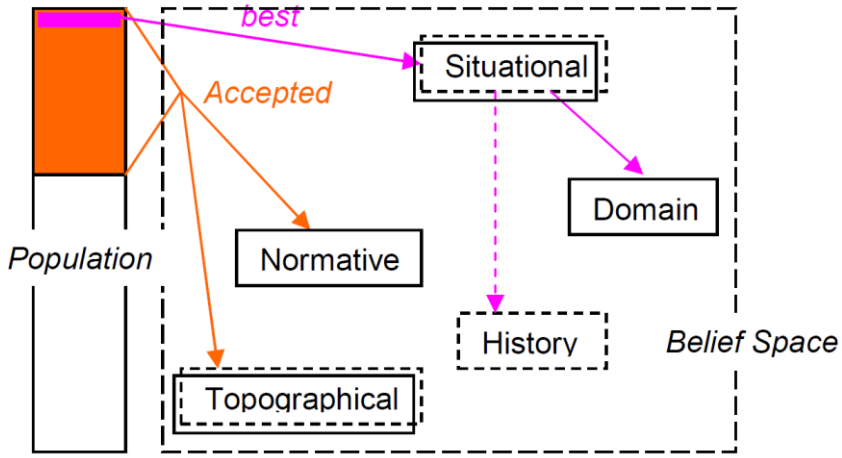

Figure 3.7 The Knowledge Update in the Belief Space

All accepted individual experiences in the current generation are used to update Normative and Topographical Knowledge, which is indicated in orange in Figure 3.7. The three other knowledge sources are updated based only on the best performer, which is indicated by red.

Although each individual knowledge source is updated depending on new knowledge from the current generation of the individual, and its own accumulation of knowledge from previous generations, some knowledge sources will also use other knowledge sources in their updating procedures. This means that some effects will be

spread to other knowledge sources as well. Figure 3.7 shows that Situational Knowledge is necessary when Domain and Historical knowledge sources are updated.

CHAPTER 4: SOCIAL FABRIC AND SOCIAL METRICS

4.1 Social Fabric

Ali [Ali, 2008] introduced a new version of the Influence function, which is based on the notion of collaboration and the Social Fabric, into the Cultural Algorithm framework. In previous Cultural Algorithm frameworks, individuals did not interact with each other in problem solving. Then, Ali proposed a communication topology, the Social Fabric, which specified the communication connections between the problem solvers in the population space. The topology type used was constant, but the positions of the individuals within the network were randomly selected at each time step. Using this method, he could evaluate the influence of just adding communication links to the search process.

The concept of Social Fabric is illustrated as a schema in Figure 4.1, with five different networks shown as five vertical lines of different colors, one for each of the five knowledge sources. Horizontal lines represent individuals. Nodes of individuals stand for their participation in each network. The nodes darkened with a network's color represent a problem solver participating in the network, and the darkened and circled nodes refer to a frequent participant.

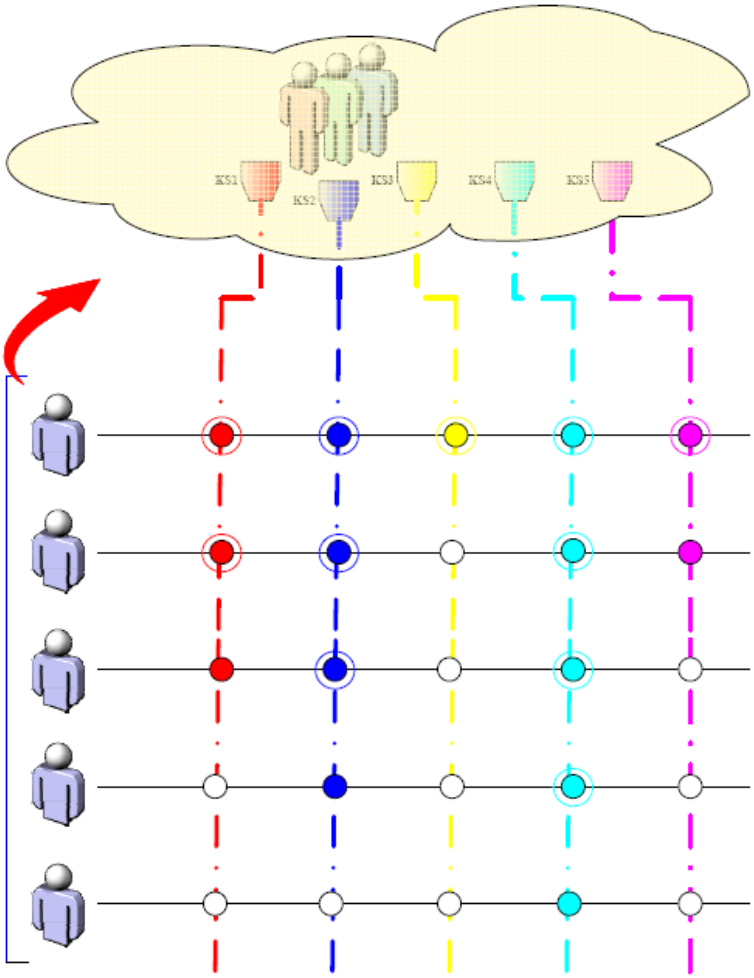

Figure 4.1 The Social Fabric Schema

Notice that the red group has a small but active set of participants. In the light blue group, everyone participates in the network. For those individuals who participate in more than one group, activities in one group can constrain activities in another. So, a knowledge source can influence an individual, and at the same time, this influence can be spread to the neighbors of the individuals. In this way, an individual can potentially be influenced by multiple knowledge sources. The integration of these knowledge sources will be at the individual level, and the knowledge source which has the strongest influence can be selected.

4.2 Neighborhood Topology

Neighborhood Topology is the method used to control the distribution of information through the Social Fabric in order to expedite the search within a given environment. In terms of the Population Space, the network of the Social Fabric can reflect a relationship. In terms of the Belief Space, the network is accessible to the knowledge sources.

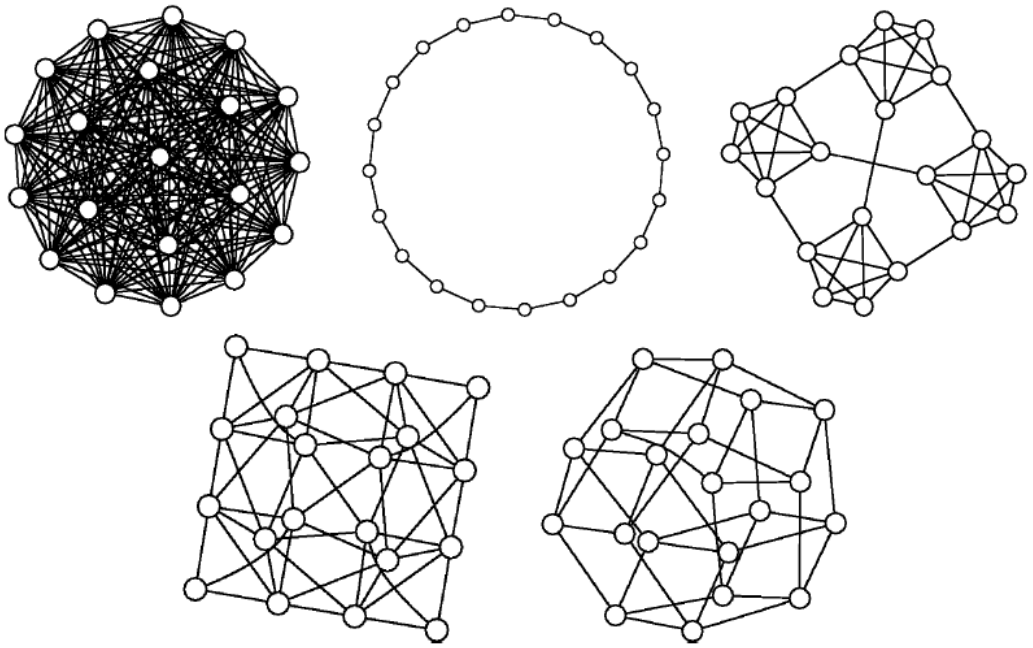

Figure 4.2 Some Example Neighborhood Topologies

Ali and Che employed a series of typical neighborhood topologies taken from the Swarm Intelligence Literature, in order to investigate in detail how topology impacts the optimization performance for different landscapes. There are many ways of constructing a neighborhood topology. The existing topologies included Lbest, Square, Hexagon, Octagon, Hexadecagon and Global. In Figure 4.2, Global, Lbest and some well-known variations of the square are shown.

4.3 Agent Decision Making

After the addition of the Social Fabric topology, each individual in the network was

not only influenced by several knowledge sources, but also now received influence from its immediate neighbors. There should be a mechanism to select one of the knowledge sources from this set of alternatives. Ali employed an un-weighted majority win scenario and Che [Che 2009] modified that decision-making approach to allow an incentive-based scheme. Here, we introduce this decision making schema applied in CAT, which is called the Incentive based Majority Win. In this rule, each vote received by an individual has a weight. The selected knowledge source should have greatest total weight.

When each individual calls the Influence function, the latter will have a direct knowledge source for this individual by spinning the knowledge wheel. However, this individual can also receive information from its neighbors as shown in Figure 4.3.

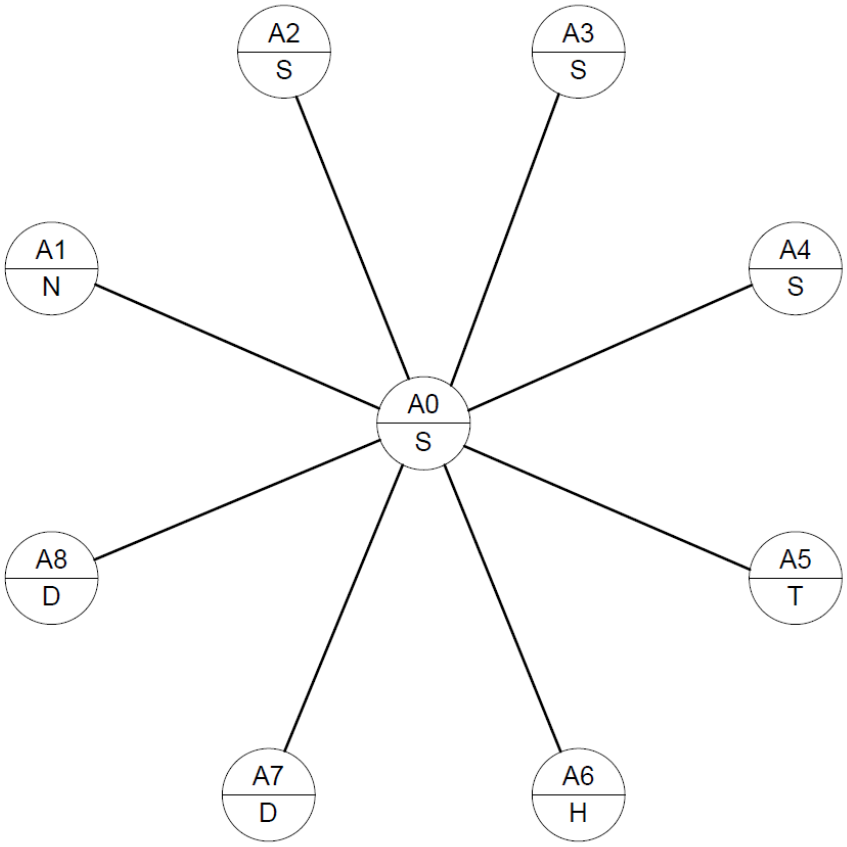

Figure 4.3 Knowledge Source Interaction at the Population Level

In this figure, we have individual A0 that is directly controlled by S, which stands for Situational Knowledge source. A0 has 8 neighbors, from A1 to A8, and each of them has a controlling Knowledge Source (KS). Here T stands for Topographical KS, D stands for Domain KS, N stands for Normative KS, and H stands for History KS. In the Population Space, the previous CAT system used the majority win based decision making in order to decide which Knowledge Source to select from the current Social Fabric. Figure 4.4 shows the majority win process.

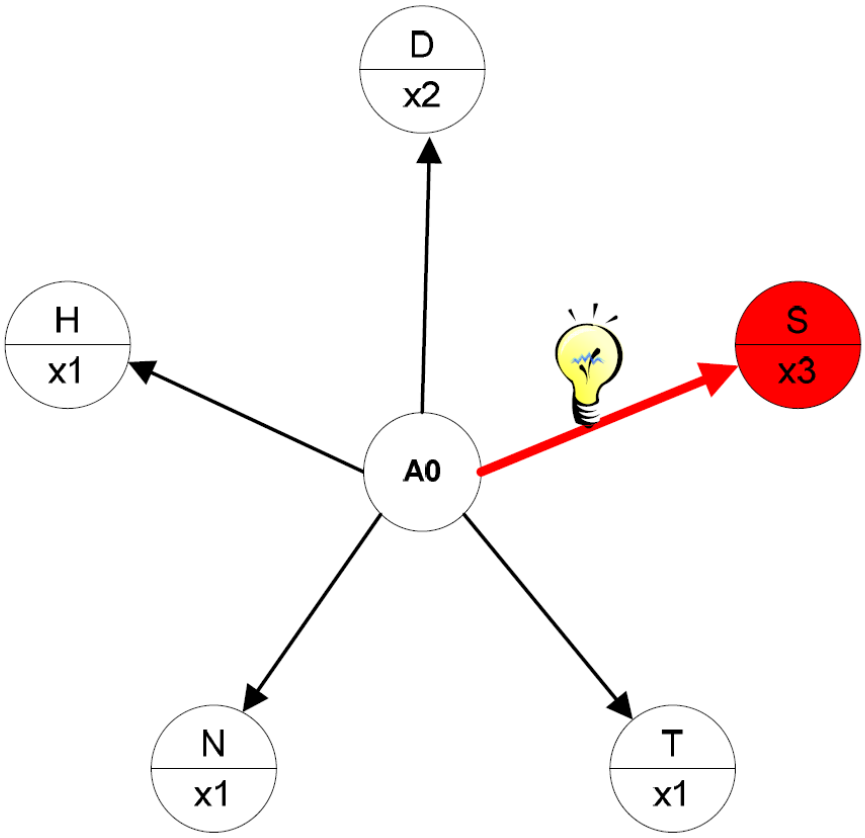

Figure 4.4 Majority Win Conflict Resolution in the Social Network.

Every individual is influenced by one of the knowledge sources at each time step. In the current version, the process is a double blind—the knowledge sources know nothing about the network and the selected individuals' positions in it. First, the individual sends the name of the influencing knowledge source to its neighbors. Next, each individual

counts the number of knowledge source bids collected from its neighbors. It will have the direct influence from the knowledge source that selected it, plus the influence of the knowledge sources transmitted to it by its neighbors. The knowledge source that has the most votes is the winner, and will direct the individual for that time step. In case of a tie, there are several tie-breaking rules embedded in the implementation of the system.

In Figure 4.4, Individual A0 has the following count of votes:

3 neighbors (including itself) votes for Situational Knowledge Source.

2 votes for Domain Knowledge Source.

1 vote for Topographical Knowledge Source.

1 vote for Normative Knowledge Source.

1 vote for History Knowledge Source.

So, Situational Knowledge Source wins the votes.

Che employed the weighted approach as an extension to the basic approach. He used the current average fitness value of each Knowledge Sources as the weight of each Knowledge Sources count, and then did majority win based on the weighted count as shown in Figure 4.5.

It is clear that after the weighted count adjustment, shown beside the arrow, Domain Knowledge becomes the winner.

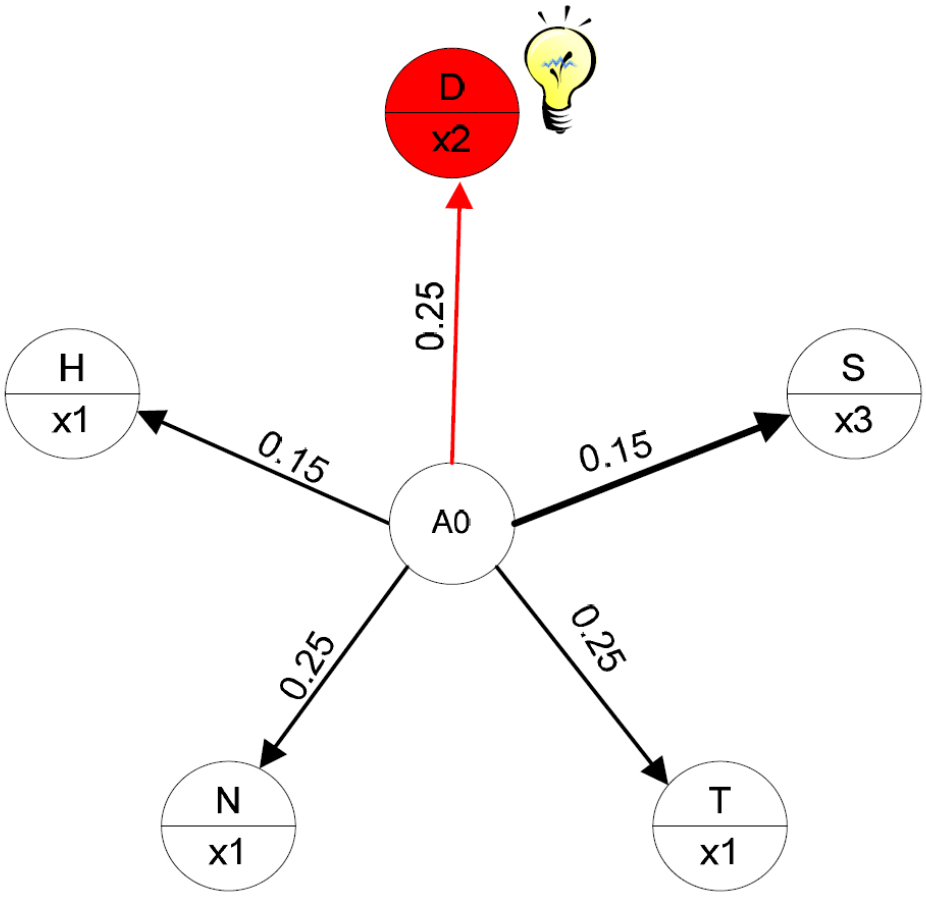

Figure 4.5 Weighted Majority Win Conflict Resolution in the Social Network

In this modified majority win rule, each knowledge source is a vector and wants to decide where the individual needs to go. The average fitness value of the current generation is the key to winning in this bidding game. If a lesser used knowledge source can find a good solution, its average fitness can rise dramatically, and therefore this approach will tend to spread its influence in the network.

4.4 Social Metrics

In this section, we describe the three metrics that we have used to display the Cultural Algorithm's vital signs in a given environment. The metrics make it possible to watch the diversity produced by the Influence function at each step. The extended influence function has the following components:

1. The update function adjusts the knowledge sources based on agent experiences to increase the diversity of the Situational Knowledge Source, whose data has also influenced other knowledge sources.

2. The Marginal Value Theorem assigns a knowledge source to each population agent based upon the relative performances of the Knowledge Sources. That knowledge source is the agent's direct influence.

3. The direct influence for each agent is distributed to its neighbors.

4. In the weighted vector voting scheme, the KS with the highest weight total is the winner. It is then able to control the behavior of the individuals at that time step.

Here, we use two metrics to assess the vital signs of the system in steps 2 and 4 above. The metric related with step 4 is named Social Tension. It reflects the distance on the functional landscape over which directly connected individuals in the population space are spread out. This reflects the diversity or entropy in the population space.

And there are several metrics associated with step 2. They are used to assess the entropy in the Belief Space based upon the relative performance of the Knowledge Sources. They are:

1. Majority Win Score: the average value of the score when the majority knowledge source wins the bidding game in a time step.

2. Minority Win Score: the average score for the time period when a minority knowledge source wins the bidding.

3. The Innovation Cost: The difference between metrics 2 and 1 above. This represents a drop in the performance associated with the need to experiment with new

solutions.

4.4.1 The Social Tension

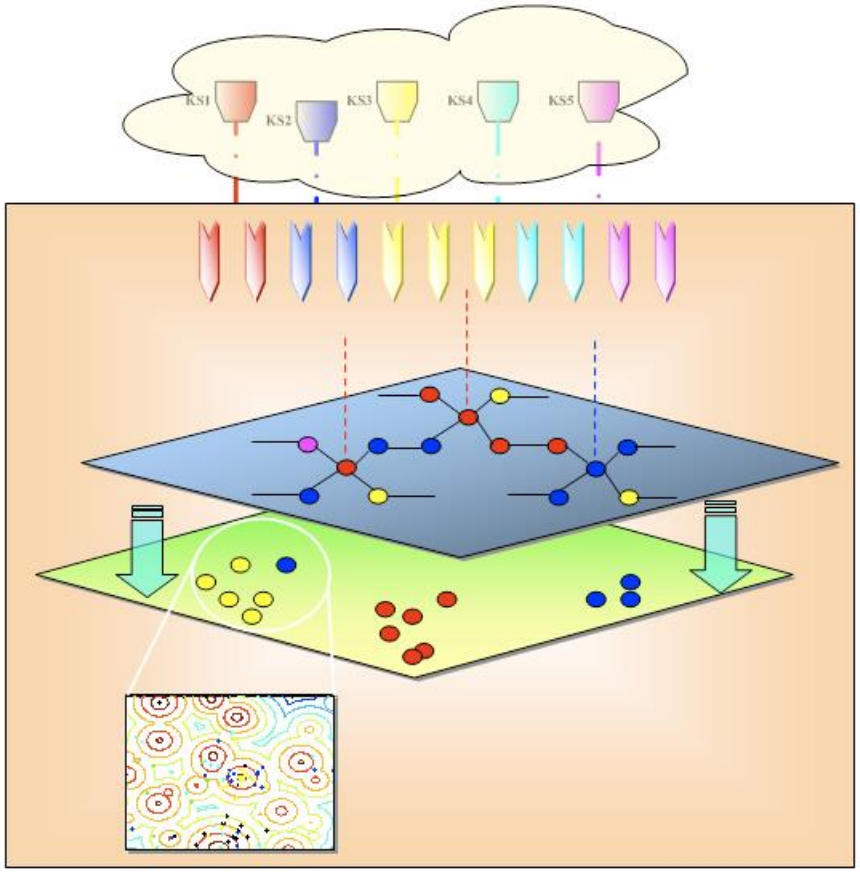

Figure 4.6 An Embedded Social Fabric Component in CAT

Figure 4.6 shows the connection between the network of agents and their layout on the 2 dimension functional landscape. The five knowledge sources compete with each other to influence individuals. Each individual has the same color as the knowledge source that influences it. Each knowledge source has a bounding box where the majority of individuals influenced by it.

The Social Tension is the sum of the Euclidean distances between the directly connected neighbors in the network. If the Social Tension is 0, then they are all located at

a fixed point. The definition of the Social Tension for one generation in a certain social environment needs the following parameters:

N: The total number of individuals,

Dim: The total number of dimensions of this environment,

M: The number of neighbors directly adjacent to each individual,

$X_{i,k}$: The coordinate on dimension k for individual i,

$a_{i,j,k}$: The coordinate of j^{th} neighbor of individual i on dimension k.

Then the Social Tension (ST) is defined as the mean of the Euclidean distances between each individual (X_1, $X_{2...}X_{Dim}$) and its immediate neighbors in the Social Fabric (a_1, $a_{2...}a_M$). It is describe

$$ST=\frac{1}{M\cdot N}\sum_{i=1}^{N}\sum_{j=1}^{M}\sqrt{\sum_{k=1}^{Dim}(X_{i,k}-a_{i,j,k})^2}$$

The pseudo-code for calculating the Social Tension is given below, where ST is the Social Tension for each generation, ESum is sum of the Euclidean distances between all individuals and its neighbors, E_{ij} is the Euclidean distance between i and j, LNum is total number of links between all individuals and their neighbors. K is the number of the neighbors for this topology.

```
Initialize ESum, LNum and ST;
For each individual Ai in a generation
    Find Ai's neighbors Aj [A1, ... Ak]
        ESum = ESum + Eij
    LNum = LNum + K
ST = ESum/ LNum
```

Figure 4.7 The Pseudo-code for Calculating the Social Tension

4.4.2 Minority / Majority Win Scores and Innovation Cost

A minority knowledge source win will happen when a knowledge source with few individuals finds a new promising region, and as a result, its average performance for that time period is high enough to beat the sum of the majority influences. The basic indices are given below:

Minority Win Score: for each generation, it is the average fitness of the winning KS when the minority wins case occurs.

Majority Win Score: for each generation, it is the average fitness of the winning KS when the majority win case occurs.

The 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. The score reflects the cost of innovation in terms of the reduction in performance that is caused when the majority does not win in a given situation.

The following pseudo-code in the Influence function in Belief space produces the three winning scores mentioned above.

```
For each individual Ai in population
    Spin the beliefWheel to get direct influence KSdirect
    Find all neighbors of Ai
    For each 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 4.8 The Pseudo-Code for Calculating the Minority Win Score, Majority Win Score, and Innovation Cost Index

CHAPTER 5: INTRODUCTION OF THE CULTURAL ALGORITHMS TOOLKIT 2.0 SYSTEM

In this chapter we discuss how the version of Cultural Algorithms used for the experiments is implemented here. Ali added the Social Fabric into the influence function of Cultural Algorithm Framework and embedded it into the Repast agent-based simulation system. He called this system the Cultural Algorithm Toolkit (CAT). Che adjusted and improved the framework of CAT later, which was subsequently called CAT 2.0. He generalized the knowledge sources, implemented a comprehensive, stable homogeneous social network structure, and employed new social metrics to measure the performance of the social system.

5.1 Repast as Development Environment

CAT 2.0 is embedded in the Recursive Porous Agent Simulation Toolkit (Repast), which is an open source library developed by Sallach, Collier, Howe, North and some others in University of Chicago [Collier 2003]. Repast has been released in different mainstream programming languages and can be run on all modern computing platforms (e.g., Windows, Mac OS, and Linux). For the purpose of this research, the system used the Repast J package, a version based on Java, as a development environment of CAT. Repast is fully object-oriented and supports software modularization. It is a powerful integrated development environment that can create, run, display, and collect data for agent based simulations. Repast allows for the development of extremely flexible models of living social agents. Developers can build customized simulations by using the Repast library components in their programs. Repast provides some standardized features like:

GUI for managing parameters, generating output data, and displaying agent interaction. These tools have proved to be really helpful in simulation modeling.

5.2 Cone's World Generation

Since we want to investigate how the Cultural Algorithm performance relates to the dimensionality and complexity of a problem, we needed to generate the Cone's World problem for our experiments.

Che found that the following alpha values represented landscapes with different levels of complexity, as shown in Figure 5.1.

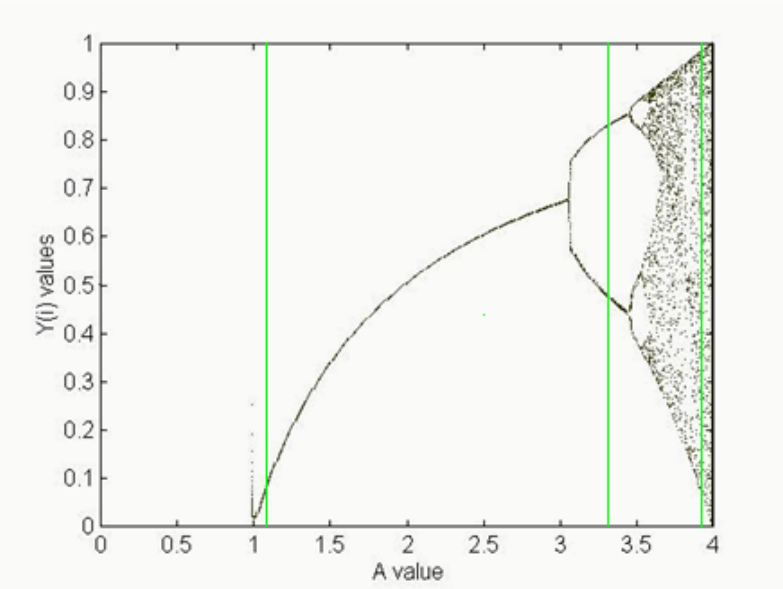

Figure 5.1 Choosing of A Value in the Logistic Function

In Figure 5.1, when A = 1.01, A has one corresponding Y value; when A = 3.35, A has two corresponding Y values; and when A = 3.99, A has random, chaotic corresponding Y values. These correspond to three classes of complexity: fixed, periodic, and chaotic, respectively. We generate five different landscapes for each complexity (A) value for each dimension tested. Therefore, we have 15 landscapes of three different complexities that are available for each dimension tested. And we have tests for four different dimensions:

2D, 3D, 4D, and 5D. Since each landscape runs only once, we have a total of 4 times 15 or 60 Landscapes.

We used 2D, 3D, 4D, and 5D landscapes in our tests, but the GUI only shows 2 dimensional graphs. Therefore, it was decided that 2D landscapes would be used as examples to explain how complexity values affect differences on landscapes. Figure 5.2 through Figure 5.4 are some of the examples of 2D landscapes. Figure 5.2 is for A = 1.01, Figure 5.3 is for A = 3.35, and Figure 5.4 is for A = 3.99. The graphs clearly show that the landscape with the higher complexity value has more ridges and plateaus.

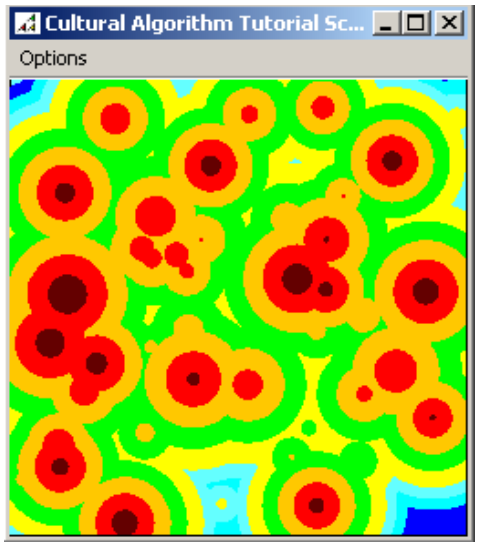

Figure 5.2 A 2D Landscape Example A = 1.01

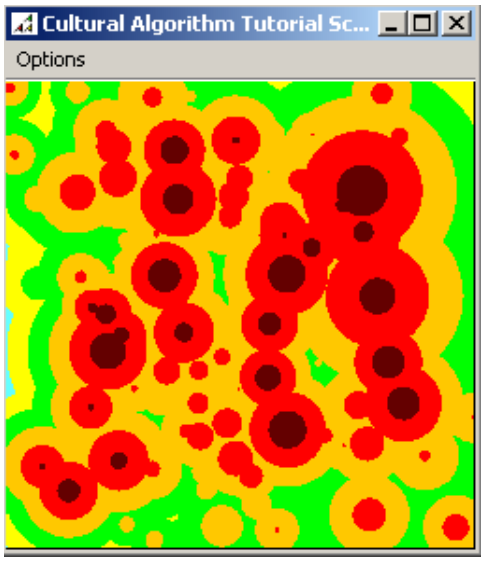

Figure 5.3 A 2D Landscape Example A = 3.35

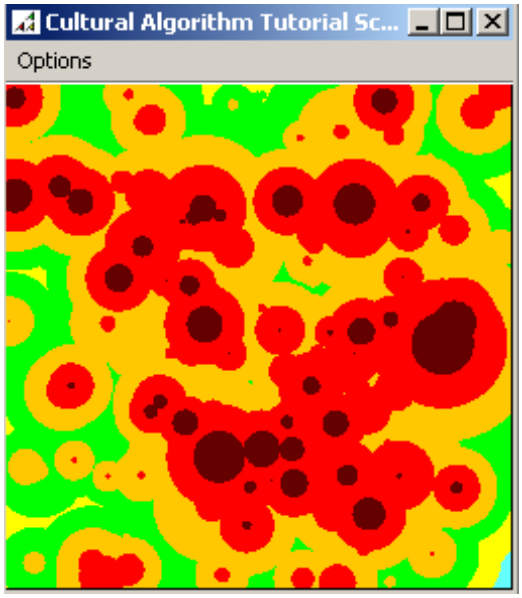

Figure 5.4 A 2D Landscape Example A = 3.99

5.3 Main Simulation Loop

Events were scheduled by setting up method calls on objects in the Repast framework. There are several basic events in the evolutionary process of CAT 2.0. We called upon a Cultural Algorithms object to process one generation, then incremented the year and displayed all of the resultant data on the screen after processing each year. Certain codes have been employed to store each generation's details in a results file, as

will be explained in greater detail later.

Precision and exit conditions are the most important factors in an optimization system. The exit condition is in some sense related to the precision and the outcome of the overall system performance. This is a combination of computational cost, time, precision, and success rate, among others. In the real world, humans or other agents only have a certain amount of time to solve certain kinds of problems, and it is not possible to produce a system that necessarily allows the solution of these problems within a restricted time frame or generations.

We defined the exit conditions to be either the finding of a solution, or the reaching of a year limit. If a run ends at the year limit, then the best result will be compared with the optimum result that has been estimated based on the fitness value. Using phenotype values as the criteria for finding a solution is sufficient in many optimization problems, especially in the case of maximization and minimization problems. We are interested in seeking a combination of parameters to reach the maximum or minimum value of the fitness.

As with many generic optimization systems, the goal of our system is to find a global optimal solution, instead of finding locally optimal solutions. False peaks always make problems more complex. We can say that a system has found the optimum solution only when it is actually on the right cone, and close enough to the optimum peak. We can calculate the ε value, which is the Difference between the current best fitness value and the global optimum fitness value.

Here, we also applied an exit condition to be able to compare the performance of the

system in terms of the success rate, time and cost of each run. The results presented in this thesis are all based on a ε value of 0.05. Although the ε value is very easy to reach for 2D tests, it is challenging in 4D and 5D situations.

5.4 Instructions of GUI

In this section, we describe the GUI of the CAT 2.0 system. Since the system is embedded within Repast, most of the GUI components are built based on the Repast toolkit. The basic Repast GUI control bar is shown in Figure 5.5:

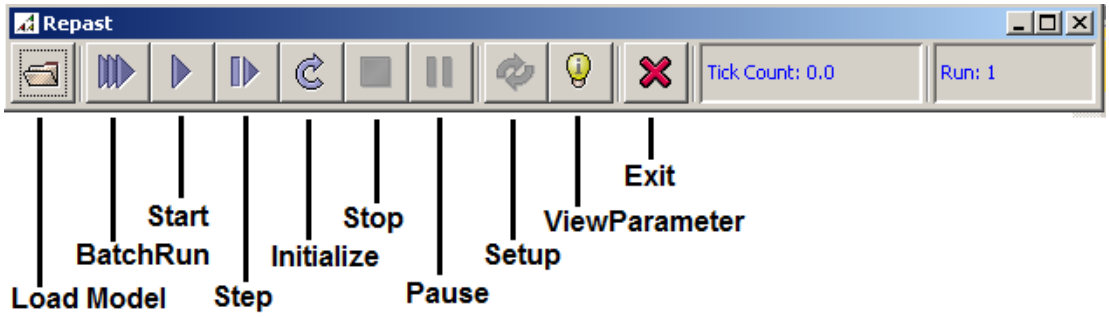

Figure 5.5 The Repast Control Bar

The unction of each of the buttons on the control bar is as follows:

Load Model: Pops up a dialog allowing the user to choose any model.

BatchRun: Executes the simulation in batches.

Start: Starts the simulation, or resumes it after the pause button has been used.

Step: Runs the simulation through a single iteration of the scheduled activities.

Initialize: Executes only the initializing code.

Stop: Stops the simulation.

Pause: Pauses the simulation.

Setup: Executes the setup code. Usually used for changing some parameters.

ViewParameter: Views the Repast parameter settings.

Exit: Exits from the simulation.

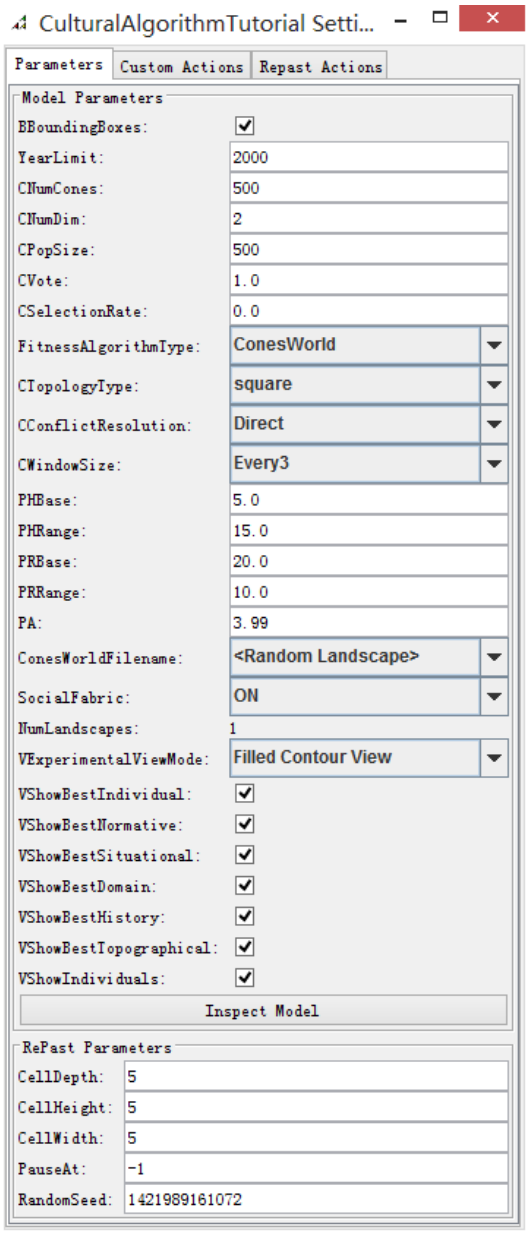

Figure 5.6 Parameter Setting in the GUI

Figure 5.6 shows the setup GUI, where the following important parameters can be modified by the user for a given run:

YearLimit: The maximum generations allowed before the optimum value is reached.

CNumCones: The total number of cones used to generate the surface to be explored.

CNumDim: The dimensionality of the landscape.

CPopSize: The total number of individuals in the population space.

PHBase, PHRange, PRBase, PRRange, PA: The parameters for the generation of Cone's World. PA is the complexity index comprising A values. They provide the range of values from which to generate the cones, with a different range of heights and widths.

CTopologyType: the topology of the Social Fabric network to be used. The choices here are Lbest, Square, Hexagon, Octagon, Hexadecagon, and Global. Here we select the square topology.

CWindowSize: the frequency with which to distribute Knowledge Source information between neighbors. This can range from 0 to the maximum run length.

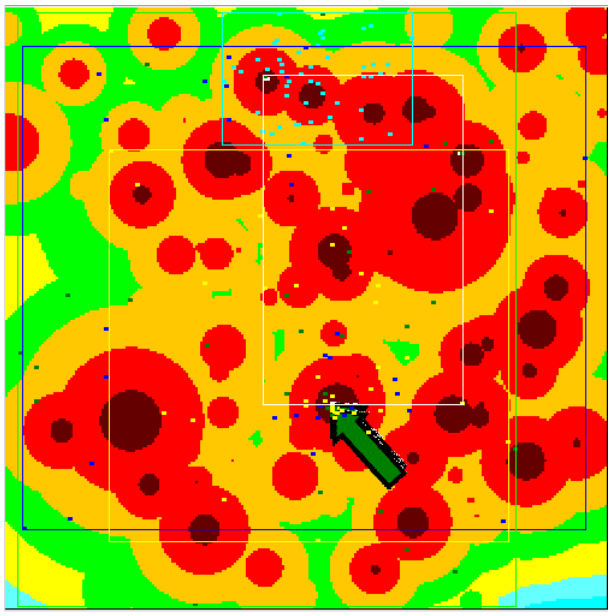

Figure 5.7 The Cone's World 2D Landscape Display

Figure 5.7 shows a visualization of the Cone's World landscape as seen at run time in the GUI, which is only available for 2D landscapes. Each individual agent is represented as a color-coded dot in the landscape. The colors represent the Knowledge Source currently influencing the individual. Cones are displayed as circles, with their

heights represented by contour lines that are filled with colors reflecting the cone's height. Blue signifies the lowest value, and dark red the highest. The green arrow shows the best individual in the current generation, as well as in the situational knowledge base which records the elites of all generations. The colored bounding box reflects the range of generated individuals within one standard deviation from the mean value for a given knowledge source. Both the agents and the bounding boxes are similarly colored for a given knowledge source. The following color code has been used here: blue stands for Normative, white stands for Situational, green stands for Domain, yellow stands for History, and light blue stands for Topographical.

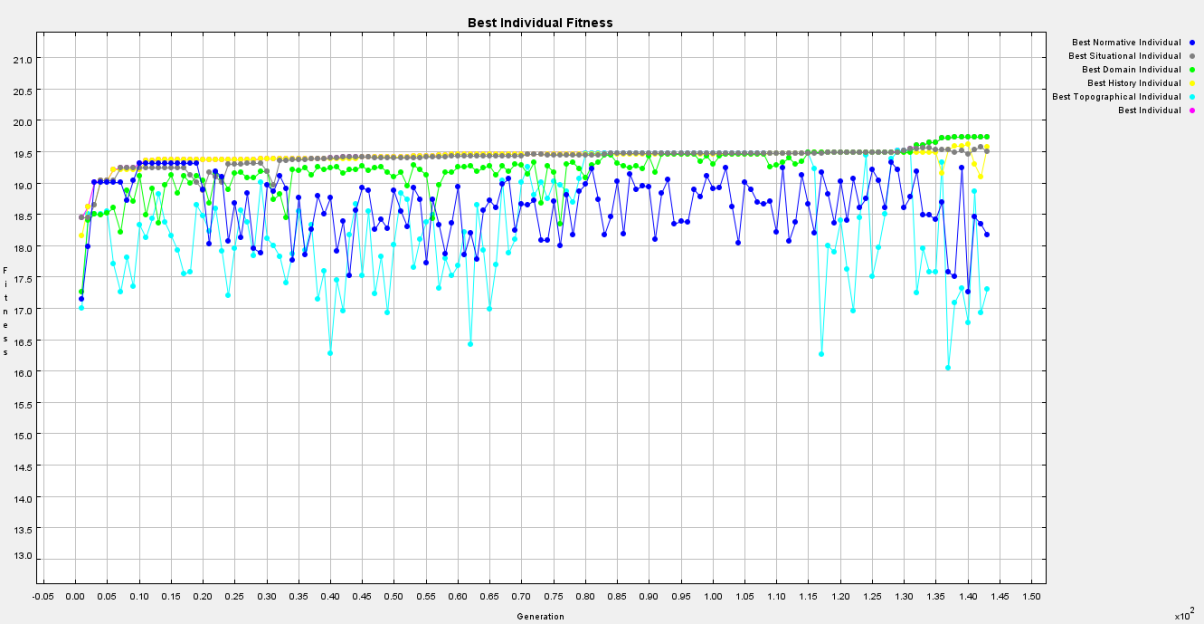

Figure 5.8 The Best Individual Fitness Graph

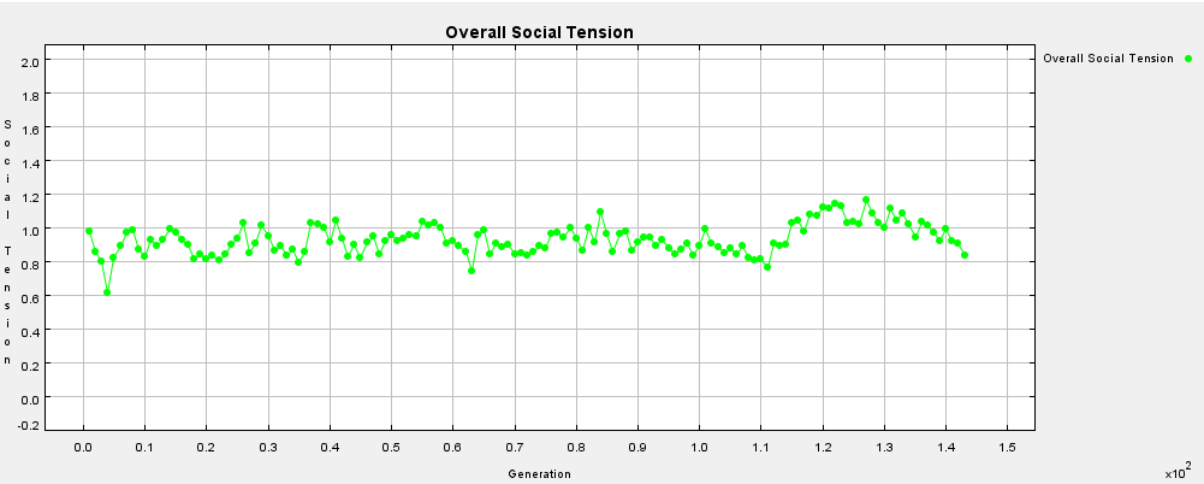

Figure 5.9 The Overall Social Tension Graph

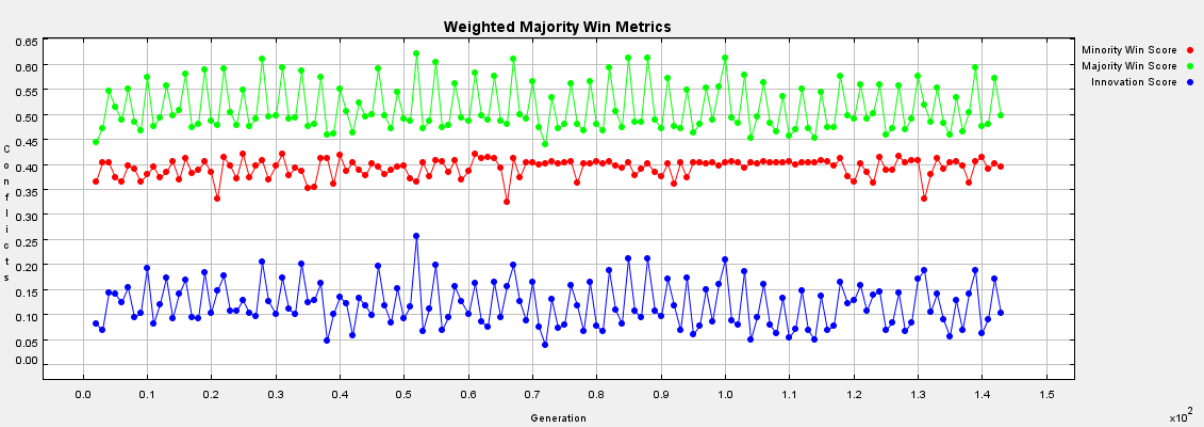

Figure 5.10 The Weighted Majority Win Metrics Graph

Figure 5.8 through Figure 5.10 show the best individual fitness, overall social tension, and weighted majority win metrics graphs generated. These three screenshots are from a test run with in 2 dimension, a 3.99 complexity value, with 500 cones, 500 population size, 2,000 generations limit, and square neighborhood topology. We can display the Cone's World landscape and the output in graphs of social metrics and knowledge source best fitness, updated in real-time.

CHAPTER 6: EXPERIMENTAL FRAMEWORK AND RESULTS

6.1 Data and Results Format

The results of our experiment have been recorded for each generation and stored in result files for later analysis. The data structure of the raw data file is as follows:

RunID: The ID of the recorded run.

Dimension: The dimension of this run.

Population: The population size of this run.

Topology: The topology type used in the Population Space.

PA: The complexity coefficient used to generate the performance landscape.

Generation: The ID of the generation.

Social Tension: The overall social tension for this generation.

MinorityWinScore: The minority win score of this generation.

MajorityWinScore: The majority win score of this generation.

InnovationCost: The difference between the minority and majority win scores.

BestFitness: The fitness of the best individual in this generation.

NormativeBest: The fitness of the best Normative influenced individual.

SituationalBest: The fitness of the best Situational influenced individual.

DomainBest: The fitness of the best Domain influenced individual.

HistoryBest: The fitness of the best History influenced individual.

TopographicalBest: The fitness of the best Topographical influenced individual.

Based on the raw data detailed above, we can produce tables of statistics relating to the relative performance of the topologies, the knowledge source activity

and performance, and the social metrics dynamics. The raw data of Run #19 is shown below as an example:

Table 6.1 The Raw Data Example Part 1

Run ID	Dimension	Population	Topology	PA	Generation	Best Fitness	Normative Best	Situational Best	Domain Best	History Best	Topographical Best
19	3	500	square	1.01	1	19.03	17.80	19.03	17.42	18.73	16.95
					2	19.39	18.61	19.03	18.81	19.39	17.92
					3	19.80	18.28	19.46	18.84	19.80	16.03
					4	19.80	18.86	19.70	19.24	19.80	16.17
					5	19.80	18.80	19.70	19.53	19.80	17.79
					6	19.80	18.82	19.72	19.51	19.80	18.53
					7	19.82	18.18	19.72	19.57	19.82	16.38
					8	19.85	18.98	19.76	19.59	19.85	17.90
					9	19.88	19.47	19.60	19.61	19.88	17.28

Table 6.2 The Raw Data Example Part 2

Run ID	Dimension	Population	Topology	PA	Generation	Social Tension	Conflicts	Agreement	No Majority Win
19	3	500	square	1.01	1	0.84	NaN	NaN	NaN
					2	0.73	0.38	0.48	0.10
					3	0.61	0.38	0.49	0.11
					4	0.75	0.41	0.60	0.19
					5	0.68	0.38	0.49	0.11
					6	0.70	0.39	0.49	0.10
					7	0.68	0.41	0.60	0.19
					8	0.62	0.39	0.48	0.09
					9	0.67	0.39	0.49	0.10

6.2 Experiment Framework

We chose the **Square Topology** for the social network, from the several topologies discussed in Chapter 4 (Lbest, Square, Hexagon, Octagon, Hexadecagon, and Gbest). In Che's research [Che, 2009], he found that the Square Topology solved most of the problems and used the lowest mean number of generations of the homogeneous

topologies tested.

We collected the data for processing and statistical summary tables. Here, we chose the maximum number of generations to be 2,000. If the number is too small, the result would not reflect the pattern. Overall, we needed sufficient variability in the results, so as to be able to observe the patterns of interest.

We tested and compared various population sizes, including 50, 100, 200, and 500, and we finally selected the 500 population size for all the runs. Population sizes of 50, 100, and 200 are good for 2 and 3 dimensions problems, but in high dimensional situations, their efficiency is much less than when the size is 500. By using a population size of only 500, five dimensional problems can be solved in 2,000 generations.

Each dimension (2, 3, 4, 5) has three complexities (1.01, 3.5, 3.99). For each dimension / complexity combination we did five runs, and in each run we randomly generated landscapes. So each dimension has a total of 15 runs and whole experiment has 60 runs. The maximum number of generations for each run is 2,000. The test schema has been summarized in Table 6.3.

Table 6.3 The Test Run Array for the Dimension / Complexity

Dimension	Complexity	Runs
2	1.01	5
	3.5	5
	3.99	5
	Total	15
3	1.01	5
	3.5	5
	3.99	5
	Total	15
4	1.01	5
	3.5	5
	3.99	5
	Total	15
5	1.01	5
	3.5	5
	3.99	5
	Total	15
Total		60

6.3 Experiment Results

6.3.1 Performance in Different Dimensions

We can produce performance comparison tables that compare the various dimensions with each other, over all the three complexity classes as shown below:

Table 6.4 The Performance Comparison in 2 Dimensions

ID	Dimension	Complexity	Generations	Best Fitness	Time	Optimal	Difference from Optimal
1	2	1.01	17	19.81	30s	19.83	0.02
2			52	19.90	74s	19.94	0.04
3			9	19.93	16s	19.96	0.03
4			31	19.95	42s	20.00	0.05
5			7	19.96	11s	20.00	0.04
6		3.5	4	19.91	6s	19.95	0.04
7			24	19.90	36s	19.93	0.03
8			35	19.80	48s	19.85	0.05
9			3	19.93	4s	19.97	0.04
10			51	19.95	62s	19.97	0.02
11		3.99	56	19.67	75s	19.71	0.04
12			23	19.88	39s	19.92	0.04
13			6	19.95	11s	19.99	0.04
14			3	19.94	4s	19.99	0.05
15			104	19.92	142s	19.97	0.05

Table 6.5 The Performance Comparison in 3 Dimensions

ID	Dimension	Complexity	Generations	Best Fitness	Time	Optimal	Difference from Optimal
16	3	1.01	17	19.94	48s	19.99	0.05
17			362	19.96	829s	20.00	0.04
18			80	19.90	221s	19.93	0.03
19			9	19.88	27s	19.92	0.04
20			160	19.96	399s	20.00	0.04
21		3.5	74	19.92	155s	19.97	0.05
22			16	19.95	56s	20.00	0.05
23			164	19.90	393s	19.92	0.02
24			2000	19.89	8660s	19.99	0.10
25			142	19.87	364s	19.89	0.02
26		3.99	56	19.97	203s	19.99	0.02
27			91	19.96	302s	19.99	0.03
28			1491	19.89	6635s	19.94	0.05
29			2000	19.67	9054s	19.90	0.23
30			8	19.85	22s	19.89	0.04

Table 6.6 The Performance Comparison in 4 Dimensions

ID	Dimension	Complexity	Generations	Best Fitness	Time	Optimal	Difference from Optimal
31	4	1.01	79	19.91	256s	19.95	0.04
32			2000	19.86	9430s	19.93	0.07
33			289	19.92	1084s	19.97	0.05
34			337	19.90	1117s	19.93	0.03
35			119	19.86	402s	19.91	0.05
36		3.5	2000	19.79	8924s	19.99	0.20
37			2000	19.93	11632s	20.00	0.07
38			82	19.90	276s	19.93	0.03
39			2000	19.81	8058s	19.97	0.16
40			2000	19.39	9230s	19.95	0.56
41		3.99	243	19.74	836s	19.78	0.04
42			2000	19.82	9660s	20.00	0.18
43			485	19.79	1637s	19.83	0.04
44			121	19.96	403s	20.00	0.04
45			2000	19.94	7440s	20.00	0.06

Table 6.7 The Performance Comparison in 5 Dimensions

ID	Dimension	Complexity	Generations	Best Fitness	Time	Optimal	Difference from Optimal
46	5	1.01	2000	19.54	21078s	19.94	0.40
47			2000	19.75	15798s	19.93	0.18
48			2000	19.23	22185s	19.95	0.72
49			932	19.88	8237s	19.93	0.05
50			1308	19.88	13296s	19.93	0.05
51		3.5	221	19.91	1247s	19.96	0.05
52			2000	19.91	19396s	20.00	0.09
53			2000	19.40	18712s	19.96	0.56
54			2000	19.78	13793s	19.97	0.19
55			2000	19.56	15699s	19.95	0.39
56		3.99	2000	19.52	11700s	19.91	0.39
57			2000	19.78	13813s	19.94	0.16
58			2000	19.79	12857s	19.98	0.19
59			2000	19.76	11638s	19.94	0.18
60			429	19.92	2153s	19.97	0.05

Tables 6.4 to 6.7 above are the overall system performance comparison tables. For each dimension, we have its overall performance:

RunID: The ID of the run.

Dimension: The dimension of this run.

Complexity: The complexity coefficient.

Generations: The number of generations used in this run (max = 2,000).

Best fitness: The final fitness value of this run.

Time: The computation time used in this run.

Optimal: The optimal fitness value in this run.

Difference from Optimal: The difference between fitness and optimal values.

6.3.2 Knowledge Source Performance

The second group of statistics reflects the performance of the knowledge sources. Tables 6.8 to 6.11 provide the knowledge source performance information for each dimension. We now explain what the data in each table represents:

Normative Average: The average performance of those individuals influenced by the Normative Knowledge Source. We recorded the value for each generation, and gave the average fitness of one run.

Situational Average: The average performance of those individuals influenced by the Situational Knowledge Source. We recorded the value for each generation, and gave the average fitness of one run.

Domain Average: The average performance of those individuals influenced by the Domain Knowledge Source. We recorded the value for each generation, and gave the average fitness of one run.

History Average: The average performance of those individuals influenced by the History Knowledge Source. We recorded the value for each generation, and gave the average fitness of one run.

Topographical Average: The average performance of those individuals influenced by the Topographical Knowledge Source. We recorded the value for each generation, and gave the average fitness of one run.

Table 6.8 The KS Performance Comparison in 2 Dimension

ID	Dimension	Complexity	Normative Average	Situational Average	Domain Average	History Average	Topographical Average
1	2	1.01	19.42	19.68	19.65	19.66	19.07
2			19.51	19.82	19.81	19.85	19.30
3			19.20	19.47	19.40	19.42	19.33
4			19.45	19.76	19.87	19.88	19.27
5			19.57	19.55	19.42	19.59	18.55
6		3.5	19.29	19.66	19.68	19.67	18.53
7			19.18	19.72	19.43	19.74	18.97
8			19.16	19.46	19.45	19.65	19.06
9			19.72	19.70	19.18	19.85	19.33
10			19.49	19.52	19.84	19.82	19.06
11		3.99	19.15	19.56	19.47	19.54	18.85
12			19.48	19.71	19.68	19.84	18.69
13			19.09	19.61	19.49	19.76	19.01
14			19.59	19.44	19.23	19.77	19.12
15			19.58	19.89	19.83	19.88	19.41

Table 6.9 The KS Performance Comparison in 3 Dimension

ID	Dimension	Complexity	Normative Average	Situational Average	Domain Average	History Average	Topographical Average
16	3	1.01	19.00	19.32	19.45	19.81	17.56
17			18.97	19.80	19.80	19.81	18.86
18			18.39	19.77	19.81	19.79	18.09
19			18.64	19.53	19.12	19.65	17.22
20			18.88	19.75	19.72	19.79	18.36
21		3.5	18.79	19.73	19.78	19.77	18.10
22			18.93	19.59	19.52	19.66	17.44
23			18.80	19.74	19.78	19.78	18.70
24			19.23	19.89	19.80	19.89	19.25
25			18.99	19.71	19.52	19.72	18.09
26		3.99	19.09	19.59	19.43	19.78	17.47
27			18.92	19.77	19.79	19.82	17.99
28			18.90	19.88	19.73	19.85	18.99
29			18.77	19.66	19.65	19.66	18.85
30			18.58	19.20	19.21	19.39	17.67

Table 6.10 The KS Performance Comparison in 4 Dimension

ID	Dimension	Complexity	Normative Average	Situational Average	Domain Average	History Average	Topographical Average
31	4	1.01	18.36	19.60	19.46	19.69	15.18
32			18.25	19.82	19.82	19.82	18.42
33			18.14	19.61	19.60	19.62	17.17
34			18.24	19.54	19.36	19.55	17.39
35			17.65	19.07	19.11	19.18	16.23
36		3.5	18.30	19.69	19.68	19.69	18.24
37			18.22	19.90	19.64	19.90	18.35
38			17.27	18.93	18.96	19.12	15.77
39			18.29	19.78	19.70	19.79	18.33
40			17.98	19.35	19.14	19.34	18.02
41		3.99	18.14	19.13	19.01	19.17	16.37
42			18.08	19.70	19.54	19.49	18.01
43			18.78	19.67	19.61	19.72	18.45
44			18.23	19.25	19.23	19.39	15.68
45			18.20	19.79	19.71	19.79	18.33

Table 6.11 The KS Performance Comparison in 5 Dimension

ID	Dimension	Complexity	Normative Average	Situational Average	Domain Average	History Average	Topographical Average
46	5	1.01	17.70	19.48	19.26	19.50	17.57
47			17.77	19.71	19.59	19.73	17.86
48			17.42	19.18	18.91	19.19	17.31
49			17.68	19.82	19.57	19.82	17.47
50			17.67	19.84	19.59	19.84	17.63
51		3.5	17.70	19.37	19.33	19.43	16.20
52			18.03	19.74	19.62	19.84	17.54
53			17.34	19.31	19.00	19.32	17.01
54			18.23	19.76	19.39	19.76	18.02
55			17.52	19.51	19.43	19.52	17.10
56		3.99	17.72	19.50	19.37	19.51	17.61
57			17.72	19.74	19.45	19.76	17.57
58			17.65	19.73	19.52	19.73	17.39
59			17.09	19.72	19.43	19.68	17.12
60			17.85	19.78	19.60	19.79	17.00

6.3.3 Social Metrics Summary Tables

These tables give the statistics for the social metrics that were used to generate the vital signs for a given run. We produced the Social Metrics Tables 6.12 to 6.15, in order to

present the results of all the runs. Here, we explain what the data in each table means:

Social Tension Average: The average social tension for each run.

Majority Win Score Average: The average winning score when everyone conforms, i.e. the average fitness value of the winning KS when everybody agrees with each other.

Minority Win Score Average: The average fitness of the winning KS when there is a conflict between an individual and its neighbors.

Innovation Cost Index Average: The difference between the conformity mean and the conflict mean reflects the opportunity for innovation.

Table 6.12 The Social Metrics Summary in 2 Dimension

ID	Dimension	Complexity	Social Tension Average	MinorityWinScore Average	MajorityWinScore Average	Innovation Cost Index Average
1	2	1.01	0.89	0.39	0.51	0.12
2			0.83	0.39	0.51	0.13
3			0.90	0.39	0.51	0.12
4			0.80	0.39	0.52	0.12
5			0.71	0.39	0.52	0.13
6		3.5	0.82	0.39	0.52	0.13
7			0.72	0.39	0.50	0.11
8			0.74	0.39	0.50	0.11
9			0.62	0.37	0.48	0.11
10			0.76	0.40	0.51	0.11
11		3.99	0.66	0.39	0.51	0.13
12			0.76	0.38	0.51	0.13
13			0.83	0.40	0.48	0.08
14			0.76	0.39	0.48	0.09
15			0.52	0.40	0.51	0.11

Table 6.13 The Social Metrics Summary in 3 Dimension

ID	Dimension	Complexity	Social Tension Average	MinorityWinScore Average	MajorityWinScore Average	Innovation Cost Index Average
16	3	1.01	0.66	0.39	0.52	0.13
17			0.61	0.40	0.51	0.11
18			0.66	0.40	0.51	0.11
19			0.70	0.39	0.52	0.12
20			0.74	0.39	0.52	0.12
21		3.5	0.73	0.40	0.52	0.12
22			0.73	0.40	0.52	0.12
23			0.49	0.41	0.51	0.10
24			0.38	0.41	0.50	0.10
25			0.59	0.40	0.52	0.12
26		3.99	0.67	0.40	0.52	0.12
27			0.73	0.40	0.52	0.12
28			0.81	0.40	0.51	0.11
29			0.47	0.41	0.50	0.10
30			0.64	0.39	0.52	0.12

Table 6.14 The Social Metrics Summary in 4 Dimension

ID	Dimension	Complexity	Social Tension Average	MinorityWinScore Average	MajorityWinScore Average	Innovation Cost Index Average
31	4	1.01	0.76	0.43	0.54	0.12
32			0.68	0.40	0.53	0.13
33			0.80	0.41	0.53	0.12
34			0.77	0.41	0.52	0.11
35			0.57	0.41	0.53	0.12
36		3.5	0.47	0.41	0.51	0.10
37			0.75	0.41	0.51	0.10
38			0.76	0.42	0.54	0.12
39			0.41	0.41	0.51	0.10
40			0.40	0.40	0.51	0.10
41		3.99	0.81	0.41	0.53	0.11
42			0.42	0.41	0.51	0.10
43			0.47	0.41	0.51	0.10
44			0.87	0.42	0.53	0.12
45			0.62	0.40	0.52	0.12

Table 6.15 The Social Metrics Summary in 5 Dimension

ID	Dimension	Complexity	Social Tension Average	MinorityWinScore Average	MajorityWinScore Average	Innovation Cost Index Average
46	5	1.01	0.54	0.42	0.52	0.11
47			0.46	0.42	0.52	0.10
48			0.77	0.42	0.55	0.13
49			0.55	0.42	0.53	0.11
50			0.36	0.42	0.53	0.11
51		3.5	0.64	0.42	0.53	0.10
52			0.55	0.42	0.52	0.11
53			0.49	0.41	0.53	0.13
54			0.51	0.40	0.52	0.12
55			0.26	0.42	0.55	0.12
56		3.99	0.60	0.41	0.52	0.11
57			0.36	0.41	0.53	0.12
58			0.71	0.41	0.53	0.12
59			0.32	0.41	0.51	0.10
60			0.41	0.42	0.52	0.10

CHAPTER 7: SUMMARY RESULTS AND ANALYSIS

7.1 Introduction

In this chapter, we compare the performances of the Cultural Algorithm in terms of solving problems in each of the four dimensions with the three complexity classes in each. We start by comparing their overall problem solving performances, and then observe the differences in terms of how the knowledge sources work in the problem solving process for various dimensions. The question of concern is the extent to which an increase in problem dimensionality has on the solution of problems of different complexity classes.

7.2 Overall Performance Comparison

Problem landscapes with various dimensionalities make huge difference of computation resource cost. Table 7.1 gives the statistical comparison of the performances, from 2 dimension problems to 5 dimension problems. Here, we explain what the data in the table means:

Dimension: The dimensionality of this group of runs.

Average generation needed: The average generations used for all experiments.

Average time cost: The average time used for all experiments.

Average Difference: The average difference between the fitness and optimal values.

Table 7.1 The Summary of Performance Comparisons Part 1

Dimension	Average generation needed	Average time cost	Average Difference
2	28.33	40s	0.039
3	444.67	1825s	0.054
4	1050.33	4692s	0.108
5	1659.33	13440s	0.243

Obviously, the number of generations needed to solve the problems is related to the number of dimensions that the problem space has. High-dimension problems need more generations and time. As the problem's dimensions grow, it will become increasingly harder to reach the optimal fitness value.

For 2 dimensional problems, the population size of 500 makes it very easy to solve each problem. All 15 runs were successfully completed in a very short time, and the differences from the optimal values were very low.

For 3 dimensional problems, we find that the average number of generations needed to find a solution increases markedly in the result. The average difference from the optimal value 0.054 is higher than the exit ε value, 0.05.

For 4 dimensional problems, we notice that only half the runs can be solved in 2,000 generations, and the average difference from optimal is double that observed in the case of 3 dimension problems.

For 5 dimensional problems, where the 500 population size reaches the limit of its ability, only 33% of all runs reach their optimal solutions before 2,000 generations. The average difference from the optimal solution here is more than twice of what is observed in 4 dimension problems. It used many more generations and greater time for finding the optimal solutions. In successful cases, its average CPU time to get a good result is 2 hours.

Table 7.2 The Summary of Performance Comparisons Part 2

Dimension	Complexity	Average generation needed	Average time cost	Average Difference
2	1.01	23.2	35s	0.036
	3.5	23.4	31s	0.036
	3.99	38.4	54s	0.044
3	1.01	125.6	305s	0.040
	3.5	479.2	1926s	0.048
	3.99	729.2	3243s	0.074
4	1.01	564.8	2458s	0.048
	3.5	1616.4	7624s	0.204
	3.99	969.8	5995s	0.072
5	1.01	1648	16119s	0.280
	3.5	1644	13769s	0.256
	3.99	1686	10432s	0.194

We then summarized how the complexity of landscapes influenced performance in each group of problems. The results are presented in Table 7.2.

2 dimensional problems were easily solved in each complexity class for the landscape. All 15 runs were processed in a very short time, and random error covered the influence of the complexity. So, there is no obvious trend in the performance results obtained for 2 dimensional problems.

For 3 dimensional problems, as the complexity increased, the number of generations and the time needed to solve the optimization problems, and the difference of the resultant value from the optimal value all increased in general between the three complexity classes.

For 4 dimensional problems, the relation between the complexity and performance is not sufficiently clear. When the complexity is 3.5, the difference from the optimal value is 0.2, which is very high. But when the complexity is 3.99, it drops down to 0.07.

For 5 dimensional problems, the complexity doesn't influence the performance, which is similar to our observations in the case of 4 dimensional problem results. The

performance differences are lesser than in the case of 4 dimensional problems.

In summary, within the limitations of our experiment parameters (population = 500, using the square topology), when the problem is easy (dim=2) or very hard (dim≥4), complexity is not the most important factor affecting the results.

7.3 Knowledge Source Performance Comparison

In the previous section, we observed how problem environments influence the general performance of each run as the dimensionality and complexity increases. In this section, we wish to discuss the performance differences observed relative to the knowledge source used in the search process.

Table 7.3 gives the knowledge source performance statistics. Here, we explain what the data in the table means:

Dimension: The number of dimensions in this group of runs.

Total Normative Average: The average performance of those individuals influenced by the Normative Knowledge Source for all the runs of the group.

Total Situational Average: The average performance of those individuals influenced by the Situational Knowledge Source for all the runs of the group.

Total Domain Average: The average performance of those individuals influenced by the Domain Knowledge Source for all the runs of the group.

Total History Average: The average performance of those individuals influenced by the History Knowledge Source for all the runs of the group.

Total Topographical Average: The average performance of those individuals influenced by the Topographical Knowledge Source for all the runs of the group.

Table 7.3 The Summary of Knowledge Source Comparisons

Dimension	Total Normative Average	Total Situational Average	Total Domain Average	Total History Average	Total Topographical Average
2	19.39	19.64	19.56	19.73	19.04
3	18.86	19.66	19.61	19.74	18.18
4	18.14	19.52	19.44	19.55	17.33
5	17.67	19.61	19.40	19.63	17.36

Table 7.3 shows two trends in our experiments. First the total Normative average fitness and total Topographical average fitness decrease when the problem's dimensionality increases. The other three knowledge sources, Situational, Domain and History, do not exhibit a statistically significant change with the increase in the number of dimensions.

To verify the hypothesis that the Normative and the Topographical knowledge sources are affected but the other three knowledge sources are not, we did the t-tests.

A t-test is a type of statistical hypothesis test. It can be used to determine if two sets of data have significantly difference from each other as a result of coming from different statistical populations.

The t statistic to test whether the means are different can be calculated as follows:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{s_{X_1 X_2} \cdot \sqrt{\frac{1}{n}}}$$

In this formula, $s_{X_1 X_2}$ is the grand standard deviation, 1 stand for group one, 2 stand for group two. The denominator of t is the standard error of the difference between two means.

When a t value is determined, a p-value can be calculated using a table of values from Student's t-distribution. The p-value is the probability of observing an effect given that the null hypothesis is true. A threshold value is chosen when the test is performed,

called the significance level of the test, usually 0.05 or 0.01. If the p-value is below the threshold chosen for statistical significance, then the null hypothesis is rejected.

We conducted the t-test to test the influence of dimensionality on each Knowledge Source performance. We suggested the following null hypothesis:

“Dimensionality doesn’t affect the performance of the certain Knowledge Source.”

For each Knowledge Source, we have 15 samples in data set of each dimensionality. Each data set from a dimensionality will be compared with other data sets from different dimensionalities. So we have six tests (2D-3D, 2D-4D, 2D-5D, 3D-4D, 3D-5D and 4D-5D) for observing the influence of dimensionality in a Knowledge Source.

We tried two threshold values, 0.05 and 0.01, in our t-tests. If the p-value is less than 0.05, it implies that two data sets are different. If p-value is less than 0.01, it means these two data sets are definitely different and have no possibility of correlation.

The t-test results table 7.4 is shown below:

Table 7.4 The T-test Results Table

Knowledge Source	Test object	p-value	Result(p <0.05)	Result(p <0.01)
Normative KS	2D and 3D	7.87E-08	Reject	Reject
	2D and 4D	6.25E-13	Reject	Reject
	2D and 5D	4.83E-18	Reject	Reject
	3D and 4D	1.16E-07	Reject	Reject
	3D and 5D	9.47E-14	Reject	Reject
	4D and 5D	0.000238	Reject	Reject
Situational KS	2D and 3D	0.687146	Accept	Accept
	2D and 4D	0.185944	Accept	Accept
	2D and 5D	0.711615	Accept	Accept
	3D and 4D	0.139779	Accept	Accept
	3D and 5D	0.510345	Accept	Accept
	4D and 5D	0.336123	Accept	Accept
Domain KS	2D and 3D	0.579693	Accept	Accept
	2D and 4D	0.194917	Accept	Accept
	2D and 5D	0.056194	Accept	Accept
	3D and 4D	0.080452	Accept	Accept
	3D and 5D	0.016224	Reject	Accept
	4D and 5D	0.708904	Accept	Accept
History KS	2D and 3D	0.74002	Accept	Accept
	2D and 4D	0.025523	Reject	Accept
	2D and 5D	0.121365	Accept	Accept
	3D and 4D	0.013395	Reject	Accept
	3D and 5D	0.06521	Accept	Accept
	4D and 5D	0.376752	Accept	Accept
Topographical KS	2D and 3D	5.1E-05	Reject	Reject
	2D and 4D	7.11E-06	Reject	Reject
	2D and 5D	5.57E-13	Reject	Reject
	3D and 4D	0.020176	Reject	Accept
	3D and 5D	0.000334	Reject	Reject
	4D and 5D	0.921916	Accept	Accept

As we can see in the table, all the results from Normative Knowledge Source are “Reject” and most of results from Topographical Knowledge Source are “Reject”, only three are “Accept”. In the other hand, for other three Knowledge Sources, they totally have three “Reject”.

Obviously the results of the t-test proved that only Normative and Topographical Knowledge Sources have very significant difference between different dimensionality results.

The reason for this is that the Normative and Topographic are exploratory knowledge, and the other three knowledge sources are exploitative knowledge sources. The new

knowledge is produced by the exploratory knowledge sources. The Normative knowledge source and Topographic knowledge source are the explorers. Their role is to search and explore the landscape. We know that higher dimensional landscapes have many more points to search over than lower dimensional landscapes. Therefore, exploring a higher dimensional map can take more computational time than searching a lower dimension map. Therefore, it is reasonable that the fitness of Normative and Topographic knowledge sources decreases when the number of dimensions in the landscapes increase. On the other hand, three exploitative knowledge sources, Situational, Domain and History, only focus on some specific localized regions in space. They take the information from good individuals from the exploratory knowledge sources. Thus, these three knowledge sources act locally and are not directly influenced by the number of dimensions of landscapes. That is why their average fitness value appears to have no correlation with the dimension number.

7.4 Social Metrics Summary

In this section, we investigate how the Social Metrics are affected by increase in dimensionality. Table 7.5 gives the statistical comparison of the results. Here, we explain what the data in the table means:

Total Social Tension Average: The average social tension for all the runs in a certain dimensionality.

Total Majority Win Score Average: The average winning score for all the runs in a certain dimensionality when every individual conforms.

Total Minority Win Score Average: The average fitness of the winning KS for all

the runs in a certain dimensionality when there is a conflict between an individual and its neighbors.

Total Innovation Cost Index Average: The difference between the conformity mean and the conflict mean reflects the opportunity for innovation.

Table 7.5 The Summary of Social Metrics Comparisons

Dimension	Total Social Tension Average	Total Minority Win Score Average	Total Majority Win Score Average	Total Innovation Cost Index Average
2	0.75	0.39	0.50	0.12
3	0.64	0.40	0.51	0.12
4	0.64	0.41	0.52	0.11
5	0.50	0.42	0.53	0.11

We find an obvious trend here—the total average Social Tension decreases with the increase in dimension number. Since the Social Tension is the sum of the Euclidean distances between the directly connected neighbors in the network, this trend implies that each individual is more likely to be close to its immediate neighbors in higher dimension problems. The reason behind this phenomenon will be discussed later.

Some of the runs did not reach the optimal solution (terminated on the 2,000th generation). In them, the Social Fabric works well in the early stages of a run in a large scale problem, but later seems less effective. As the social tension is reduced the amount of variability within the population is reduced. For example, in Run #55, the last fluctuations occurred around generation 470. The graph of this example is shown in Figure 7.1, the red line marking the cool down generation. In the same time the best fitness value no longer increase after the cool down generation as shown in Figure 7.2.

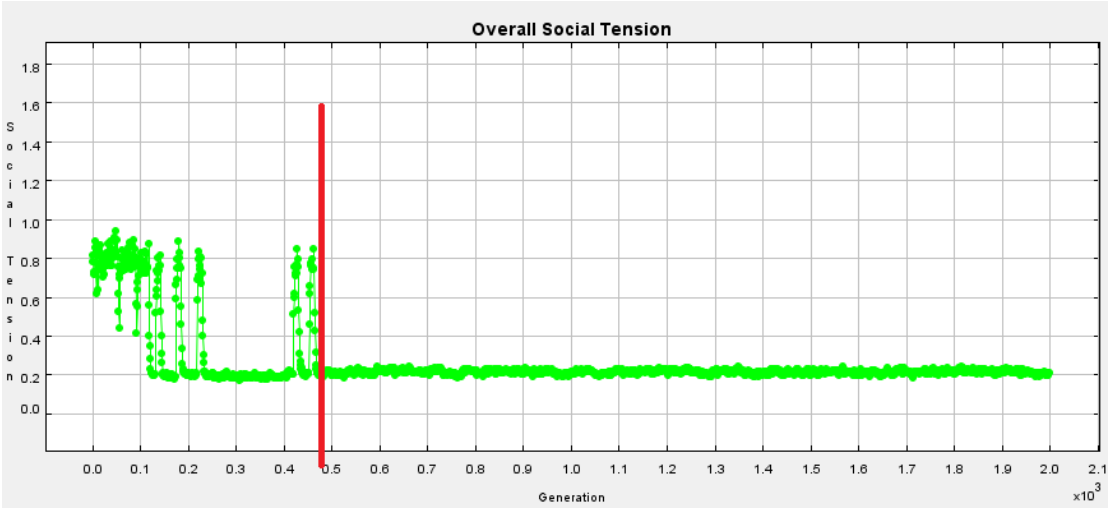

Figure 7.1 The Social Tension Graph of Run #55

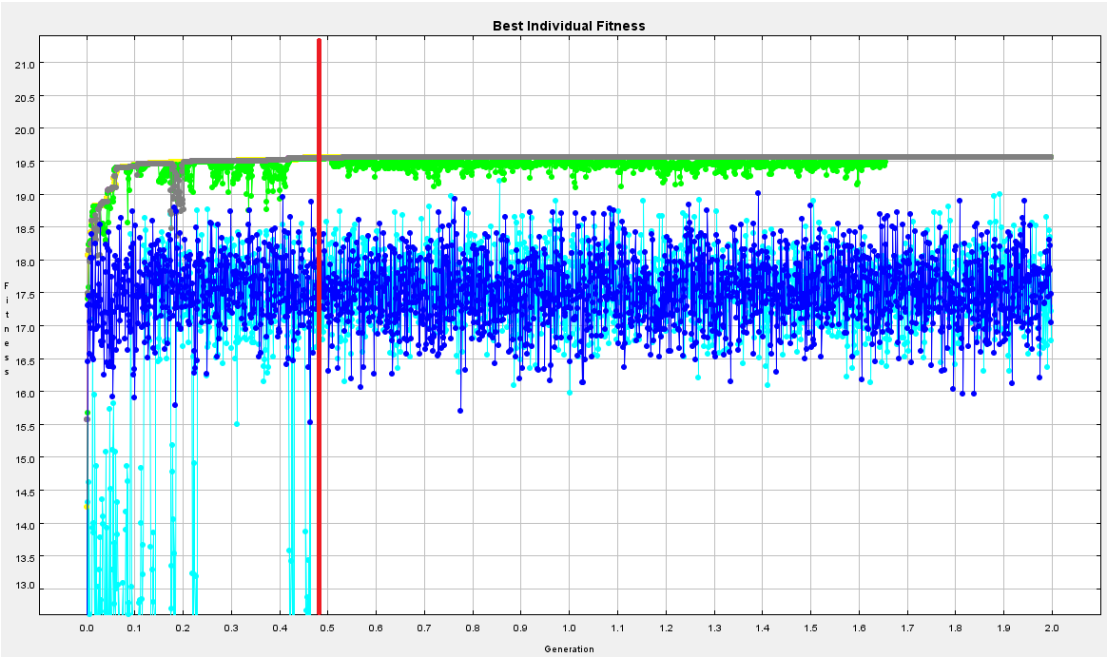

Figure 7.2 The Fitness Graph of Run #55

To compare with the unsuccessful run, here we present a Social Tension graph for some successful runs. The graph of successful examples from run #28 and run #60 are shown in Figure 7.3 and 7.4 below:

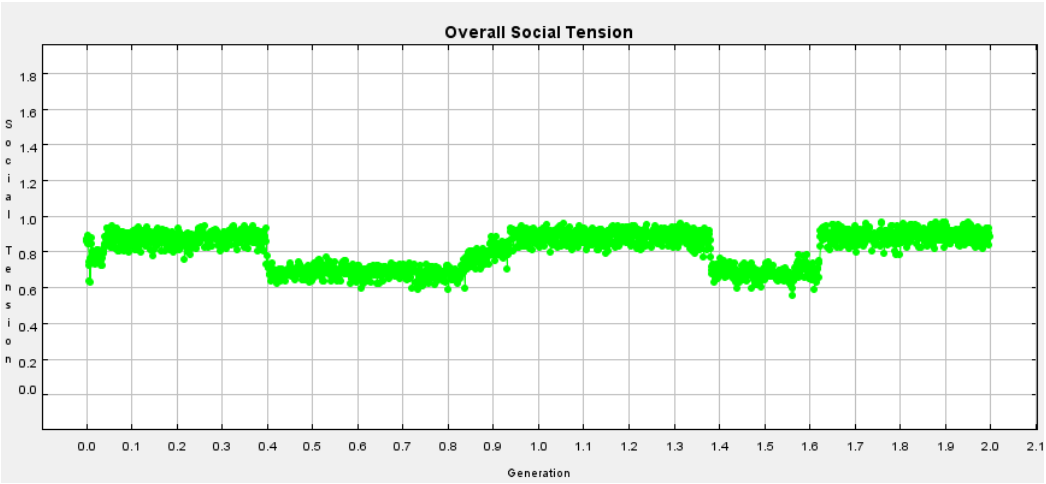

Figure 7.3 The Social Tension Graph of Run #28

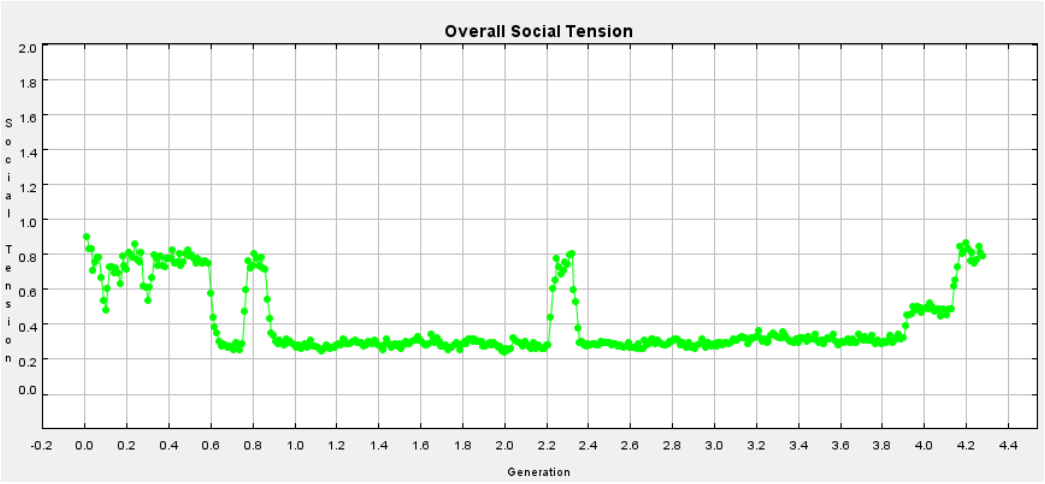

Figure 7.4 The Social Tension Graph of Run #60

As we can see, Social Tension of successful runs can have significant fluctuations from start to end. This implies that generating sufficient diversity in the population is an important point to achieve success in the run.

In the previous part of this section, we found that the total average social tension decreases with increasing dimension number. To explain this phenomenon, we looked over the record of each run, and recognized that the reduction in performance was reflected by the reduction in Social Tension. This suggests that while the Belief Space is continuing to generate diversity, the social fabric topology is unable to effectively

distribute it throughout the population. These results suggest that a heterogeneous social fabric may be needed in order to more effectively distribute the innovations throughout the population.In these situations, optimization is seldom finished before 2,000 generations. In Table 7.6, we list the generation number at which the Social Tension began to be reduced for those runs that did not get the solution by 2000 generations..

Table 7.6 The Statistical Expression of Social Tension Cool Down

Run ID	Dimension	Generation when Social Tension coosl down
24	3	60
29		110
32	4	1000
36		200
37		290
39		350
40		480
42		280
45		990
46	5	1300
47		1150
48		240
52		650
53		500
54		140
55		470
56		140
57		240
58		160
59		180
Average		447

As we can see in Table 7.6, the Social Tension value cools down at an average value of around 500 generations in those runs that did not find the optimal by 2000. Of course, this does not mean that they cannot find the optimum only that the search process has been slowed.

In these experiments, most of the individuals are not spread out over the search landscape, and they tend to cluster around several cones which have good fitness, but not optimal. When the landscape’s dimension number goes up, the search space becomes sparser, and the gap between peaks can become deeper and wider. For

individuals who are influenced by exploratory knowledge sources, their work is much more difficult in higher dimension problems, so they can hardly get out of the ‘desert’ and often go back to the original cone. This suggests that a homogeneous topology is not able to generate sufficient diversity in the population over time. Thus, a more flexible and heterogeneous topology may be necessary with increasing dimensionality. This will be studied in future work.

CHAPTER 8: CONCLUSIONS AND FUTURE WORK

8.1 Conclusion

In the previous chapters, we used the Cultural Algorithms Toolkit to examine the performance of Cultural Algorithms in solving multi-dimensional optimization problems.

We applied experiments from 2-dimensional problems to 5-dimensional problems. Having conducted statistical analyses and summarized the findings, we can present several conclusions with regard to our overall objectives, which are listed below:

1. As the landscape dimensionality increases, the Cultural Algorithm needs more computation resource to reach an optimal solution in terms of the number of generations used and the overall time cost.

2. As the landscape dimensionality increases, more diversity in the population is needed to exploit the larger search space.

3. As the landscape dimensionality increase, there is more pressure on the social fabric to distribute innovations throughout the population.

4. As landscape dimensionality increase, the average social tension of individuals will be lower and social tension will cool down more frequently. This is because the homogeneous topology employed (square) is not sufficient to create diversity in the population.

5. A homogeneous social fabric is not sufficient to handle increases in problem dimensionality after a certain point. It is sufficient for 2 dimensions, but falls off quickly after that. It suggests that a dynamic heterogeneous social fabric will be more useful for problems of higher dimensionality.

These conclusions are not entirely independent. The summary of social tension and the phenomena in the third conclusion have a reasonable, logical relationship. Together they all explain the primary reason that landscape dimensions have a considerable impact on the performance of this optimization problem.

8.2 Future Work

The results presented here suggest that the following future work:

1. While static homogeneous topologies are be sufficient for low dimensional problems, dynamic heterogeneous topologies might be effective for higher dimensional problems. In future work, the relationships between problem dimensionality and social dynamics will be studied.
2. The social metrics employed here were useful in understanding aspects of Cultural Algorithm performance in all dimensionalities. In future, work additional metrics will be introduced in order to extract more information about the system performance.

REFERENCES

Che, Xiangdong, Reynolds. Weaving the Social Fabric: Optimization Problem solving in Cultural Algorithms using the Cultural Engine. (2009)

Chung, C. J., Reynolds, R. G. A test bed for solving optimization problems using cultural algorithms, Evolutionary Programming, pp. 225-236, (1996)

Chung, C. J., Reynolds, R. G. "CAEP: An evolution-based tool for real-valued function optimization using cultural algorithms." International Journal on Artificial Intelligence Tools 7(3). (1998)

Collier, N. "Repast: An extensible framework for agent simulation." The University of Chicago's Social Science Research. (2003)

Jin, X. and Reynolds, R. G. Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: a cultural algorithm approach. Proceedings of the 1999 Congress on Evolutionary Computation. (1999)

J. Jin. Path Planning in Reality Games Using Cultural Algorithm: The Land Bridge Example. (2011)

Langton, C. Life at the Edge of Chaos, Artificial life II, pp: 41–91. (1992)

Morrison, R., De Jong, K., Inc, G., & Vienna, V. A test problem generator for non-stationary environments. (1999)

Peng, B. and Reynolds, R. G. Cultural algorithms: knowledge learning in

dynamic environments. (2004)

Reynolds, R. G. An Introduction to Cultural Algorithms in Proceedings of the 3rd Annual Conference on Evolutionary Programming. (1994)

Reynolds, R. G., Saleem, S. M. The impact of environmental dynamics on cultural emergence. In Perspectives on Adaptations in Natural and Artificial Systems. Oxford University Press, (2001)

Reynolds, R. G., Peng, B., Whallon, R. Emergent Social Structures in Cultural Algorithms. (2005)

Reynolds, R. G and Ali, M. "Computing with the social fabric: The evolution of social intelligence within a cultural framework." IEEE Computational Intelligence Magazine 3(1): 18-30. (2008)

Stanley, Samuel Dustin, “Analyzing Environmental Change And Prehistoric Hunter Behavior Through A 3d Time-Lapsed Model With Level Auto-Generation And Cultural Algorithms”, Master Thesis, Wayne State University, Detroit. (2013)

Saleem, S., and Reynolds, R.G., "Cultural Algorithms in Dynamic Environments," in IEEE Congress on Evolutionary Computation, San Diego. (2000)

Vitale, Kevin. “Learning Group Behavior in Games Using Cultural Algorithms and the Land Bridge Simulation Example”. Master’s Thesis, Wayne State University, Detroit. (2009)

ABSTRACT

THE IMPACT OF INCREASED OPTIMIZATION PROBLEM DIMENSIONALITY ON CULTURAL ALGORITHM PERFORMANCE

by

YANG YANG

August 2015

Advisor: Dr. Robert Reynolds

Major: Computer Science

Degree: Master of Science

In this thesis, we investigate the performance of Cultural Algorithms when dealing with the increasing dimensionality of optimization problems. The research is based on previous cultural algorithm approaches with the Cultural Algorithms Toolkit, CAT 2.0, which supports a variety of co-evolutionary features at both the knowledge and population levels. In this project, the system was applied to the solution of 60 randomly generated problems that ranged from 2-dimensional to 5-dimensional problem spaces.

As a result, we were able to produce the following conclusions with regard to our overall objectives:

1. As the landscape dimensionality increases, the Cultural Algorithm needs more computation resource to reach an optimal solution in terms of the number of generations used and the overall time cost.

2. As the landscape dimensionality increases, more diversity in the population is needed to exploit the larger search space.

3. As the landscape dimensionality increase, there is more pressure on the social fabric to distribute innovations throughout the population.

4. As landscape dimensionality increase, the average social tension of individuals will be lower and social tension will cool down more frequently. This is because the homogeneous topology employed (square) is not sufficient to create diversity in the population.

5. A homogeneous social fabric is not sufficient to handle increases in problem dimensionality after a certain point. It is sufficient for 2 dimensions, but falls off quickly after that. It suggests that a dynamic heterogeneous social fabric will be more useful for problems of higher dimensionality.

Keywords: Cultural Algorithm, Optimization, Social Fabric, Co-evolution

AUTOBIOGRAPHICAL STATEMENT

Yang Yang received a Bachelor's degree from Xidian University in China in 2012 with a major in Electronic Engineer. He began to study for a Master's degree in Computer Science in Wayne State University in 2013. He is interested in the areas of Artificial Intelligence and Game Programming. Yang Yang has worked as an intern at several companies including Autodesk where he investigated Cloud Computing applications. He currently lives in Northville, Michigan. He is now working for the Deluxe Corporation in Plymouth, Michigan.