SNAKE-IN-THE-BOX PROBLEM USING NATURE INSPIRED SEARCH

by

KARTHEEK ATLURI

(Under the Direction of WALTER D. POTTER)

ABSTRACT

The focus of this project is on three nature inspired search techniques. Inspired by human evolution the Genetic Algorithm (GA) is the most famous and widely used search technique. The second search technique is Particle Swarm Optimization (PSO), which originated with the intent to simulate the graceful but unpredictable choreography of a bird flock. The third and final technique is Simulated Annealing (SA) inspired from annealing in metallurgy. These three nature inspired search techniques are used to find the longest snake in 8-dimensional hypercube. Different snake representations (Node Sequence, Transition Sequence and Bit Representations), various parameter settings and aspects like engineered conditioning are used for each technique in an effort to find the longest snake possible. This project found snakes of length 97 using SA and GA techniques. The longest snake found using the PSO technique is of length 95.

INDEX WORDS: Snake-in-the-box, Genetic Algorithm, Particle Swarm Optimization, Simulated Annealing, Nature inspired search strategies, Node sequence representation, Transition sequence representation, Bit representation

SNAKE-IN-THE-BOX PROBLEM USING NATURE INSPIRED SEARCH

by

KARTHEEK ATLURI

B.Tech. Jawaharlal Nehru Technological University, 2006

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2009

SNAKE-IN-THE-BOX PROBLEM USING NATURE INSPIRED SEARCH

by

KARTHEEK ATLURI

Major Professor: Walter D. Potter

Committee: Khaled Rasheed
Tianming Liu

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2009

DEDICATION

This thesis is dedicated to my grandfather.

ACKNOWLEDGEMENTS

I like to thank Dr. Potter for his support and patience. Without Dr. Potter, this project would not be possible. I would also like to thank Dr. Rasheed for allowing me to do snake-in-the-box problem as my final project for his class and this project is based on the final project done for that class. I would like to thank everyone who helped me with the thesis.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF GRAPHS

Page

CHAPTER 1

INTRODUCTION TO SNAKE-IN-THE-BOX-PROBLEM

1.1 OVERVIEW

This project is the study of three nature inspired search techniques for the Snake-In-The-Box Problem (hereafter referred to as the Snake Problem) using different representations of the snake. The objective of the Snake Problem is to find the longest snake in an 8-dimensional hypercube. A snake is a path containing a sequence of nodes through a n-dimensional hypercube in which each node in the snake is adjacent to exactly two nodes in the snake, one before and one after except for the starting and ending nodes which are adjacent to only one node. The Snake Problem was introduced first by W.H Kautz in 1958 [Kautz58], motivated by the theory of error-correcting codes and is still an open problem.

The Snake Problem belongs to a class of problems that cannot be solved using a deterministic algorithm in polynomial time but a solution can be checked in polynomial time, often referred to as non-deterministic polynomial (NP). Finding the longest snake becomes increasingly difficult as the dimension space increases (dimensions greater than seven) and the search space suffers from combinatorial explosion [WekaComExp09]. This type of problem is often a candidate for heuristic search techniques like the Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Simulated Annealing (SA). All three techniques are inspired by nature and this project focuses on how these three search techniques perform using different snake representations (node, transition, and bit) along with different settings for each technique

and engineered conditioning (local growing, restricting movement in a seeded region, positive mutation) to find the longest snake possible in an 8-dimensional hypercube.

Practical applications for the Snake Problem include error detection, disjunctive normal form simplification [Klee70], electrical engineering and coding theory. In each application the longer the snake, the more effective are its capabilities [WekaSnake09].

When this project was started, the longest snake found in an 8-dimensional hypercube was 97 [Rajan99]. This project is an attempt to find a snake of length 97 or longer using GA, PSO and SA techniques. Three different representations (node sequence representation, transition sequence representation and bit representation) are used in an effort to find the longest snake. All coding is done in the Java programming language, with many built-in functions but taking time when it came to execution. The three search techniques are tested using the longest snake from a 7-dimensional hypercube to seed the initial population, engineered conditioning to restrict movement in a seeded region and simple one step local growing every 5 generations. This project found a snake of length 97 using the GA, using the PSO technique it found a snake of length 95 and a snake of length 97 (without local growing) using the SA technique which is the longest snake found to date with this technique.

1.2 SNAKES

A snake is a sequence of nodes in a hypercube where no nodes are repeated and a node in the snake is adjacent to only two other nodes in the snake (one before that node and one after that node) except for the starting and ending nodes which are adjacent to only one node in the snake. Two nodes are adjacent when the nodes represented in bit form differ by only 1 bit. In an 8-dimensional hypercube, Node 0 (in bit form 00000000) is adjacent to nodes

1(00000001), 2(00000010), 4(00000100), 8(00001000), 16(00010000), 32(00100000), 64(01000000) and 128(10000000). The adjacent nodes when represented in bit form differ by only one bit when compared with node 0s bit form. Figure 1.1 shows a snake of length 4, which is the longest possible snake in a 3-dimensional hypercube. The sequence of nodes is zero, one, five, seven and six. Nodes 2 or 4 cannot be part of the snake because both nodes are adjacent to node 0 and node 0 is already part of snake.

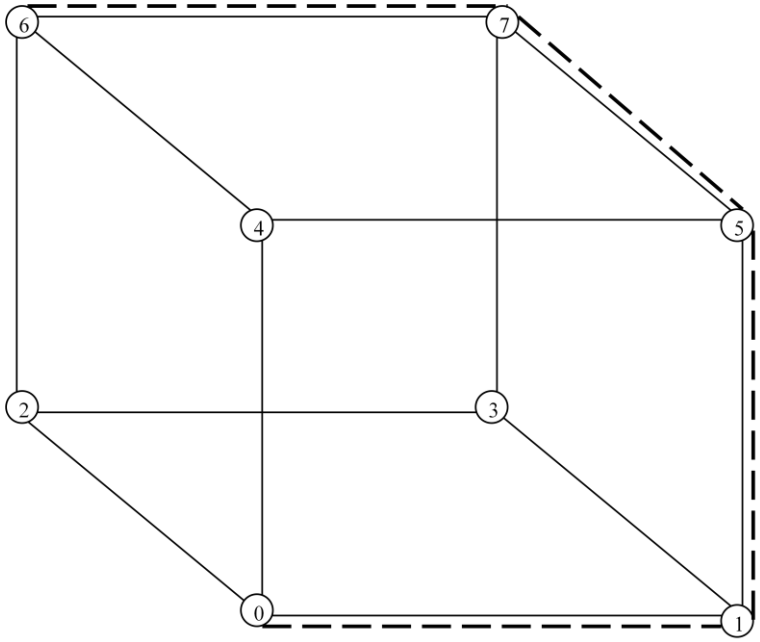


Figure 1.1 3-Dimensional hypercube

The Snake's length is represented in either the number of nodes or the number of edges. Two adjacent nodes in a snake form an edge. The length of the snake is defined as the number of edges existing in the snake. The number of nodes and edges in a snake of length 97 in an 8-dimensional hypercube is 98 and 97 respectively. For each node in a d-dimensional hypercube, there are d connecting nodes. The function s(d) represents snake length.

To find long snakes Potter [Potter94] has formally tried using the GA. There are many published articles regarding the Snake Problem. The articles focus on new bounds for snakes

in varying dimensions and study of the Snake Problem in general [Klee67, Snevily94, Potter94, Kochut94, Kochut96, Rajan99, Bitterman04, Handy08]. All these papers along with the CSCI 8940 Computational Intelligence class and CSCI 8950 Machine Learning class gave a starting point and the ideas are mostly taken from these classes and articles.

1.3 REPRESENTATION

There are three different representations when it comes to representing the snakes. The first one is node sequence representation. This representation consists of integers in the range of 0 to 255 (2^8-1). Each individual consists of a sequence of nodes with no node repeated more than once. Consecutive nodes in the sequence are adjacent to each other in the hypercube (two nodes are adjacent when their integer numbers represented in bit form differ by only 1 bit). This is the easiest representation to calculate the snake length as the snake length can be directly calculated from the representation (the other two representations have to be converted to node sequence representation to calculate the snake length). The sequence of nodes is used to find the longest sub sequence of nodes that satisfy the conditions for a snake.

The second representation is the transition sequence representation, which was introduced by Klee [Klee67]. The individuals represented in this form have a sequence of numbers in the range of 0 to 7 (n-1). The numbers in the sequence indicate the bit position to be flipped to get to the next node. The snake always starts with the first node as 0^{th} node (00000000). The first transition determines which bit is flipped in node 0s binary form to get the binary representation for the next node. The node sequence is thus generated from the transition sequence representation and the snake length and the fitness function are

calculated. One of the advantages of the transition sequence representation over the node sequence representation is that normal crossover and mutation techniques in the GA can be used directly as node adjacency is maintained even during crossover and mutation, which is not the case with node sequence representation.

The third and final representation is bit representation, which was used by Diaz-Gomez [Diaz06] where each individual has a sequence of binary bits (2^8=256) with each bit representing one node in the hypercube. If the node is turned on in the snake then the bit corresponding to that node is set to one and if the node is turned off then the bit value for that node is set to zero. It is probably the most basic representation of the hypercube indicating directly which nodes are turned on and off. Ideas on how to find the snake and snake length from bit representation have been taken from Diaz-Gomez's vector of neighbors [Diaz06] and from [Handy08]. See these two papers for a detailed explanation on how the snake length is calculated from bit representation. For the bit representation used in this project, the snake always starts at node 0 and checks if any of its neighbors are turned on, if so then that node is added to the snake as the next node in the node sequence. If more than two nodes are turned on then the node with a lower node number is always taken first. Once a node sequence is generated, it is then used to find the length of the snake. The paper from [Handy08] is the most influential paper on this project with many ideas taken from that paper and applied to other search techniques. The node representation is used only for the GA, as node adjacency cannot be maintained using other search techniques. The transition sequence representation and bit representation are used for the GA, PSO and SA techniques.

1.4 FITNESS FUNCTION

The fitness function is the most important operator in all the search techniques. The fitness function is used to differentiate a good snake from a bad snake and offers a way for search techniques to converge on good snakes. The fitness function guides how the search techniques move through the search space in finding a good snake. Five fitness functions are used in this project.

The first function, proposed by Potter [Potter94], is the length of the longest snake in an individual (hereafter referred to as fitness function 2a). The next four fitness functions are based on snake length and skin nodes, which measures the tightness of the snake such that it winds itself in such a way that the snake limits the nodes that are not accessible in each step [Touhy07, Handy08]. Skin nodes are essentially nodes adjacent to nodes in the snake but are not part of the snake. If these nodes are shared by more nodes in the snake then the number of nodes available (not part of the snake and not adjacent to any nodes in the snake) for the snake to grow will be more and hence the chances of an individual finding a longer snake also increases with an increase in available nodes. A snake has more potential to grow when the snake is not too loose and not too tight [Handy08]. The fitness functions two, three and four are taken from [Handy08]. Using the GA, Handy was able to find a snake of length 94 [Handy08]. A skin node can be shared by 1, 2, 3, 4, 5, 6 or 7 snake nodes. The second fitness function is (hereafter referred to as fitness function 2b)

$$\frac{\text{Snake Length}}{\text{Total number of Skin Nodes}} \times \sum_{n=3}^{n-1} \text{number of skin nodes shared n times.}$$

The third fitness function is (hereafter referred to as fitness function 2c)

$$\frac{\text{Snake Length}}{\text{Total number of Skin Nodes}} \times \sum_{n=4}^{n-1} \text{number of skin nodes shared n times.}$$

The fourth fitness function is (hereafter referred to as fitness function 2d)

$$\frac{\text{Snake Length}}{\text{Total number of Skin Nodes}} \times \sum_{n=5}^{n-1} \text{number of skin nodes shared n times.}$$

The final fitness function adds some weight to the skin nodes and is (hereafter referred to as fitness function 2e)

$$\text{Snake Length} \times 11 + \sum_{n=1}^{n-1} \text{number of skin nodes shared n times} \times (n + 1) .$$

The fitness functions 2b, 2c and 2d all use a different range for calculating the fitness function in terms of skin nodes shared. Ideally, if nodes are shared more times then the number of nodes available for the snake to grow will be more. One problem with fitness functions 2b, 2c and 2d is that the shared nodes do not have any weight associated with them. There can be two individuals where both have the same snake length, same number of skin nodes, and same summation for the range [3, 7]. One individual can have more nodes shared 6 times, and another with more nodes shared less times. Fitness functions 2b, 2c and 2d do not differentiate between these individuals. To differentiate between nodes shared more times and nodes shared less times fitness function 2e is added that adds some weight to the skin nodes and Snake Length.

CHAPTER 2

GENETIC ALGORITHMS

2.1 INTRODUCTION TO GENETIC ALGORITHMS

The model of Darwin's theory of natural selection or the survival of the fittest guides Genetic Algorithms (GA). The fittest individual is one with the best fitness function. The basic operations involved are selection, crossover and mutation. An individual in the population represents a snake in node, transition or bit representation. Each number in the sequence represents the node number, transition value or if the bit corresponding to a node is turned on or off depending on the snake representation used. The fitness of individuals is calculated (using the fitness function) and the fitness is then used to compare individuals to determine which individual is better. The fitness function credits the individuals with higher values who have a long snake and more potential in terms of skin nodes depending on fitness function used.

The individual selection is done by randomly picking two individuals from the initial population (generated randomly or seeded with snakes from lower dimensions), comparing those two individuals based on their fitness and choosing the individual with the best fitness. This is the first individual used in selection. The same process is repeated to select another individual. The two individuals are collectively referred to as parents. Once the parents are selected, crossover is performed on the parents, which determines which characteristics come from which parent to the child (generated from crossover), and produces a next generation individual (the initial population is the first generation). Various crossover techniques like the two-point

crossover technique and enhanced edge recombination technique are used according to the schema used to represent the snakes as individuals.

The mutation operation introduces new features that are not seen in the parents. It essentially helps the next generation individuals to go to new areas in the search space thereby allowing the GA to explore more search space and therefore increasing its chance of finding a longer snake. Different mutation techniques like naïve mutation, diagonal mutation and positive mutation are used with different representations.

Once crossover and mutation are performed, the individuals are added to the next generation. The process of selection, crossover, and mutation is repeated until the next generation population size is equal to the initial population. The initial population can be generated randomly or can be seeded with snakes from smaller dimensions or from previous runs from dimension 8. The initial population is the starting point for the GA and the performance of the GA depends heavily on where it started in the search space. With seeding the GA starts at a better position in the search space, the chance of finding longer snakes is even higher when compared to initial populations generated randomly.

Once the next generation has enough individuals, the fitness function of each individual is calculated. The process of selection, crossover and mutation are repeated again to generate next generation individuals. This is repeated until the stopping criteria based on a predetermined number of generations or when there is no improvement in the best individual for a certain number of generations is met. The GA stops when the stopping criteria is satisfied and the best individual from the last generation is the output for the GA.

Elitism is a special case in the GA which (when turned on) allows the best individual in each generation to get a free pass to the next generation without any crossover or mutation on

that individual. With elitism, the GA has to find the longest snake only once over all the generations and that will be preserved until the final generation of the GA. With elitism turned off the longest snake found can be lost due to crossover and mutation especially when the number of generations is more than needed. When elitism is turned on, it prevents any negative effect of excess generations.

2.2 USING NODE SEQUENCE REPRESENTATION

2.2.1 INTRODUCTION

To find the longest snake in an 8-dimensional hypercube, a node sequence representation is used to represent snakes. The best snake found using the node sequence representation reported here is 95.

2.2.2 REPRESENTATION

The individual in the GA contains a sequence of node numbers ranging from 0 to 255 (2^8). The node sequence representation makes it difficult to use normal crossover techniques and hence the Enhanced Edge Recombination (EER) technique (also used by Potter [Potter94]) which maintains node adjacency is used [Star91].

In EER, two individuals are taken for crossover and then an edge table is constructed using the nodes in the longest snake from both the individuals instead of using all the nodes in the individual. The edge table contains an entry for all the unique nodes and the nodes that are adjacent to that node are maintained in the edge table. One of the parent's starting nodes is taken as the starting point (node) and one of the nodes adjacent to the starting node is picked from the edge table. The new node is then used to pick the next node by looking at its adjacent nodes in the edge table. This process of picking adjacent nodes is repeated until no new nodes could be added or when the length of the node sequence reaches a predetermined value. No node in the sequence is repeated more than once.

If there is a choice to choose the next adjacent node then the first preference is given to the nodes that are common to both the parents and if there are no such nodes then the next

preference is given to the node that has the least number of adjacent nodes turned on. This way of choosing the nodes that have the least number of adjacent nodes is similar to a narrowest path heuristic (NPH). Choosing the narrowest path causes the snake to compress within the hypercube thus making sure that more nodes remain available to be added to the path in the future thus giving the snake more room to grow.

The upper bound on the longest snake possible for $d \geq 7$ satisfies the following expression: upper bound $\leq 2^{d-1} - (2^{d-1}/ (20d- 41))$ [Snevily94]. The maximum length of the individuals in the population is determined using this equation. In an 8-dimensional hypercube, the upper bound using the above equation is 126.92. Therefore, each individual's length is set to a maximum of 127 nodes. If the initial population is seeded randomly, then to generate the node sequence, a number between 0 and 255 is picked at random and is set as the starting point for that individual. A node adjacent to the first node is picked at random that is not part of the node sequence yet. This is repeated until there are no nodes that can be added to the sequence or when the individual node sequence length reaches 127 (whichever occurs first).

2.2.3 FITNESS FUNCTION

All the five fitness functions are tested individually to see how well each fitness function did in comparison with each other. The five fitness functions used are fitness function 2a, fitness function 2b, fitness function 2c, fitness function 2d and fitness function 2e.

2.2.4 INITIAL POPULATION

The initial population is seeded with a snake of length 50 from the 7-Dimensional hypercube given in [Potter94], and the length is increased by randomly choosing the next

available adjacent nodes. If a node is already part of the node sequence then that node is not used again. Each individual is expanded to a maximum length of 127 nodes. If there are no adjacent nodes after a certain point then the individual terminates at that position, so individuals can be of different length but will not be more than 127.

2.2.5 ENGINEERED CONDITIONING

In an effort to increase efficiency and reliability in finding longer snakes and better average snake length, new strategies are added to the GA. One strategy is to protect the seeded nodes (from 7-dimensional hypercube 0 to 50) from crossover and mutation but the use of EER for crossover does not allow the seed to be protected. In addition, the snake can start at any node and end at any node. Due to this the seeded region is not protected, instead an engineered conditioning operation called positive mutation is introduced. When a node is selected for mutation, that node is mutated and the new fitness for the mutated individual is calculated. If the fitness of the mutated individual is greater than the original individual then the mutated individual replaces the original individual in the population. If the fitness of the mutated individual is less, then the original individual is retained.

A second strategy is that after every 5 generations, each individual is taken and a simple local search technique is used to see if each individual's snake can be increased by one node on either end of the snake. The ending node's adjacent nodes are examined to see if there are any nodes that can be added and still be part of the snake. If so then that node is added. If more than one adjacent node is eligible then one is picked at random. The same procedure is followed to increase the snake at the starting node of the snake.

2.2.6 STOPPING CRITERIA

The stopping criterion is 100 generations with no change in the best individual or a maximum of 500 generations, whichever occurs first.

2.2.7 SELECTION

Binary Tournament Selection is used to select individuals for crossover. To select a parent for crossover two individuals are selected at random and the individual with the best fitness function is selected as parent 1. The same steps are used to select parent 2. Once the parents are selected, crossover is performed to generate the next generation individuals.

2.2.8 CROSSOVER

Once parents are selected, a number is generated at random between zero and one and is compared with the crossover rate to determine if crossover should be performed on the parents. If the generated number is less than the crossover rate then the EER technique is used to perform crossover to generate a new individual. If the generated number is greater than the crossover rate then each parent is directly copied to the new individual without any crossover and the GA moves to the next step (mutation).

2.2.9 MUTATION

Mutation is done for every node in the individual. A random number between 0 and 1 is generated for each node and if the random number is less than the mutation rate then mutation is performed on that node. If the random number is greater than the mutation rate then mutation is not performed on that node. To perform mutation, diagonal flipping is used. If a node connects to

two nodes then there is exactly one more node connecting exactly those same two nodes. Each node selected for mutation is taken (other than the start and end node) and depending on the mutation rate and the random number generated, that node is changed to its diagonal node to see if the fitness function has increased. If it did, it would replace the individual from which it has mutated. If it has the same fitness then the number of available nodes for the two individuals is calculated. If the new individual has more available nodes, then it replaces the individual from which it mutated. This prevents any negative effect of mutation. Since the EER technique and local growing on either side can change the node sequence, restricted mutation on the seeded region is not used, as the place of the seeded sequence cannot be determined within the node sequence of the individual.

2.2.10 ELITISM

When elitism is turned on, the best individual in each generation is added to the next generation without performing any crossover or mutation on it. With elitism turned off, all the individuals go through crossover and mutation to make it to the next generation depending on crossover and mutation rates.

2.2.11 EXPERIMENTAL SETUP

All the five fitness functions are tested using different combinations of population size, crossover rate, mutation rate and elitism.

Population size : 200, 400, 600

Crossover rate : 0.2, 0.4, 0.6, 0.8

Mutation rate : 0.002, 0.0067, 0.008, 0.02, 0.04

Elitism : 1 if turned on and 0 if turned off

The different settings discussed in the thesis proposal are reduced to the above settings as these settings alone for each fitness function for a combination of 120 (3×4×5×2) for 5 trial runs took around 180 hours to run when engineered conditioning was used . One of the main reasons for such an increase in run time is the EER technique and positive mutation. This is significant overhead. Without positive mutation, the same settings ran in around 55 hours. Another important factor is the use of the EER technique to perform crossover, which takes a lot of time to compute when compared to a simple two-point crossover technique. To see how seeding of the initial population (hereafter seeding refers to 'seeding of initial population') and engineered conditioning affected the length of the snakes, all the settings are tested with different combinations. First with no seeding and no local growing, next with no seeding and local growing, next with seeding, no local growing, no restriction on mutation, next with seeding, local growing, no restriction on mutation, next with seeding, no local growing, positive mutation only lastly with seeding, local growing, positive mutation only.

2.2.12 RESULTS

Without seeding and no local growing, fitness function 2c performed better than the other fitness functions. Fitness function 2a is next. See Table 2.1. The longest snake found is of length 81 with fitness function 2c. The best settings are determined using the average snake length for five trial runs. Fitness function 2c is the best with an average snake length of 73.0 with a population size of 600, crossover rate at 0.8, mutation rate at 0.04 and elitism turned off. The top five settings for fitness function 2c favored a crossover rate at either 0.8 or 0.4. For the remaining

fitness functions all different crossover rates made it to the top five suggesting crossover has little to no effect on average snake length.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	600	0.6	0.008	0	71.4	75
	600	0.8	0.04	0	71.0	73
	400	0.4	0.04	0	70.6	78
	400	0.4	0.02	0	70.6	74
	600	0.2	0.02	0	69.8	79
2b	400	0.8	0.0067	1	69.6	74
	400	0.6	0.02	0	69.2	77
	400	0.2	0.008	1	69.2	74
	600	0.6	0.02	0	69.2	74
	600	0.6	0.008	0	69.0	73
2c	600	0.8	0.04	0	73.0	78
	600	0.8	0.002	0	71.0	74
	600	0.8	0.002	1	70.6	76
	600	0.4	0.02	0	70.0	74
	600	0.4	0.008	0	68.0	81
2d	600	0.6	0.04	1	48.4	72
	600	0.4	0.002	1	46.8	52
	600	0.6	0.002	1	46.6	51
	400	0.6	0.002	1	45.6	52
	200	0.6	0.002	1	44.8	46
2e	600	0.6	0.02	0	69.6	73
	600	0.2	0.02	0	69.0	72
	200	0.4	0.008	1	68.4	71
	400	0.8	0.04	1	68.4	71
	600	0.6	0.008	0	68.4	75

Table 2.1: GA using Node Representation with no seeding and no local growing

When it comes to mutation rate, fitness functions 2a and 2e performed better with higher mutation rates. Fitness function 2b preferred mutation rates greater than 0.006 while lower mutation rates did not make it to the top five. Fitness function 2c has the lowest and higher mutation ($\geq$0.008) rates in the top five. Fitness function 2d performed better with either the lowest mutation rate or the highest mutation rate. All the top five settings for fitness function 2d have elitism turned on while the other fitness functions performed better with elitism turned off.

When the settings are ordered (sorted in ascending order) by average snake length, for fitness function 2d the bottom half of the list has elitism turned off. This is not the case with the other fitness functions where a clear pattern is not found. Most of the settings with small populations did not perform well when compared to larger populations and ended up at the bottom of the ordered list. The difference between the top 70 settings in average snake length is around five edges for most fitness functions. So determining which settings are best is not possible as one run could find a snake that is 7 edges longer than the next best and instead of making the top 20 it will make it to the top 5, so making any conclusion on which setting is the best is very difficult.

Table 2.1 gives details on how each fitness function performed in finding the longest snake and the average snake length. Fitness function 2c is first with an average snake length of 73.0. Fitness function 2a is next with 71.4. Fitness functions 2b and 2e are next with 69.6. Fitness function 2d is last with 48.4. Fitness function 2d is expected to perform better when the population is seeded with the longest snake from the 7-dimensional hypercube. The seeded snake is tightly wound when kept in the 8-dimensional hypercube (it used only half of the nodes 0 to 127) and hence leads to more nodes shared more often. Without seeding, finding a snake with a large number of nodes shared 5 or more times is difficult when compared to finding snakes that have skin nodes shared three(fitness function 2b) or four(fitness function 2c) times. Fitness function 2e is able to perform on par with fitness function 2a as each skin node count has a weight associated with it. A snake with four nodes shared 3 times has less fitness than the snake with four nodes shared 5 times (with all other parameters remaining the same). This is not the case with fitness functions 2b, 2c and 2d where they have the same value.

Fitness function 2a performed better with no elitism but when the top 25 settings are considered, settings with no elitism took 14 out of the top 25 spots, which is not as dominant as the top five suggested. It is the same case with fitness functions 2b, 2c and 2e while for fitness function 2d the top 60 have elitism turned on and the bottom 60 have elitism turned off. Fitness function 2d ran in about 4 hours compared to 50 hours for the other fitness functions. It was not able to find snakes longer than 52 more than once. The GA generally performed well with large population sizes for all fitness functions. The longest snakes found using fitness functions 2a, 2b, 2c, 2d and 2e were 79, 77, 81, 72 and 75 respectively.

Without seeding and local growing, fitness function 2b performed better than the other fitness functions. Fitness function 2a is next. See Table 2.2. The longest snake found is of length 84 with fitness function 2a which is an increase of 3 edges when compared with no local growing. The best settings are determined using the average snake length for five trial runs. Fitness function 2b is the best with an average snake length of 74.4 with a population size of 600, crossover rate at 0.8, mutation rate at 0.02 and with elitism turned off. This is a one-edge increase in the average snake length. The difference in performance when compared to no local growing is minimal. The effect of local growing is not seen in the average snake length or the longest snake found. The time taken has increased from 4 to 9 hours for fitness function 2d while for the other fitness functions it increased to around 72 hours.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	600	0.4	0.0200	1	74.2	77
	600	0.2	0.0067	1	74.0	79
	400	0.8	0.0400	0	73.2	78
	200	0.4	0.0020	0	73.0	84
	400	0.2	0.0400	0	73.0	78
2b	600	0.8	0.0200	0	74.4	76
	600	0.8	0.0400	1	74.2	78
	600	0.6	0.0200	1	74.0	77
	400	0.2	0.0080	0	73.8	78
	600	0.8	0.0200	1	73.6	80
2c	600	0.8	0.0200	0	74.0	78
	400	0.4	0.0080	1	73.2	77
	600	0.8	0.0080	0	73.2	82
	600	0.2	0.0067	1	73.2	77
	400	0.2	0.0067	1	73.0	78
2d	600	0.2	0.0067	1	61.0	69
	200	0.2	0.0067	1	60.2	77
	600	0.2	0.0020	1	60.0	75
	600	0.2	0.0400	1	59.6	71
	400	0.2	0.0200	1	57.4	79
2e	600	0.2	0.0200	0	73.4	77
	600	0.8	0.0067	0	72.4	74
	600	0.4	0.0200	0	72.2	79
	600	0.8	0.0200	0	72.0	75
	600	0.2	0.0067	1	71.6	80

Table 2.2: GA using Node Representation with no seeding and local growing

Fitness function 2d performed better with the lowest crossover rate at 0.2 while the other fitness functions did well with any crossover rate. Fitness function 2d again performed better with elitism turned on. When the settings are ordered by average snake length, the settings with elitism turned off are at the bottom of the list although a few settings with elitism turned off performed better than some settings with elitism turned on.

The average snake length for all fitness functions for different settings are extremely close. For example, for fitness function 2a the difference in average snake length between the top 50 in the ordered list is just four edges. The setting that found the best snake out of all fitness

functions is 84 (using fitness function 2a) and has the next best snake at 71. Therefore, instead of 84 if it found a snake of length 71 then the average snake length would be 70.4, which will take the setting to the top 40 in the list instead of the top five. Therefore, the settings should not be taken into account to determine which performs better and which do not. The average snake length and the longest snake found went up a bit with the introduction of local growing. Fitness functions 2a and 2e seemed to favor no elitism. Instead of the top 5, when the top 25 are considered the settings with elitism turned on did well with 14 out of the top 25 for fitness function 2a and for fitness function 2e, 12 out of the top 25 settings performed better with elitism turned on.

Table 2.2 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 84 which is the best snake seen out of all fitness functions. Fitness function 2b found a snake of length 80. Fitness function 2c was able to find a snake of length 82 while fitness function 2d was able to find a snake of length 79. Fitness function 2e was able to find a snake of length 80.

With seeding, no restriction on mutation and no local growing, fitness function 2b performed better than the other fitness functions. Fitness function 2d is next. See Table 2.3. The longest snake found is of length 95 with fitness function 2d, which is an increase of 11 edges when compared with no seeding and local growing. The best settings are determined using the average snake length for five trial runs. Fitness function 2b is the best with an average snake length of 86.4 with a population size of 600, crossover rate at 0.6, mutation rate at 0.008 and with elitism turned on. This is a 12-edge increase in the average snake length when compared to no seeding and local growing. The performance of all fitness functions is much closer with the difference between the best and worst average snake length in the 25 settings (the top 5 from

each fitness function) at 86.4 and 82.6 respectively. With the initial population seeded, fitness function 2d also took around 70 hours along with other fitness functions to complete all the settings.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	600	0.6	0.0020	1	83.4	86
	600	0.6	0.0080	0	83.4	88
	200	0.8	0.0400	1	83.0	88
	600	0.8	0.0080	1	83.0	86
	200	0.6	0.0200	1	80.6	90
2b	600	0.6	0.0080	1	86.4	93
	600	0.6	0.0020	1	85.4	89
	600	0.4	0.0400	0	85.2	89
	400	0.8	0.0067	0	85.0	92
	400	0.6	0.0067	0	85.0	93
2c	200	0.8	0.0400	1	84.4	92
	600	0.8	0.0067	1	84.2	91
	600	0.6	0.0200	1	83.6	85
	600	0.8	0.0020	1	83.4	85
	600	0.8	0.0067	0	83.4	86
2d	600	0.8	0.0080	1	86.2	89
	600	0.8	0.0020	0	85.8	90
	400	0.8	0.0400	0	85.6	90
	600	0.8	0.0067	0	85.4	89
	400	0.8	0.0067	0	83.2	95
2e	400	0.8	0.0200	1	83.8	87
	600	0.8	0.0067	1	83.4	86
	600	0.8	0.0080	1	82.8	89
	400	0.6	0.0400	1	82.6	85
	400	0.6	0.0200	1	82.6	85

Table 2.3: GA using Node Representation with seeding, no restriction on mutation and no local growing

Fitness function 2d improved the most with a second best average snake length and the best in longest snake found with 95. Fitness function 2d performed better with a crossover rate at 0.8 for the top five. Other fitness functions performed well with a high crossover rate (≥0.6) when the top five are considered. Fitness function 2d performed better with elitism turned off but

when the top 25 settings from the ordered list on average snake length are considered the settings with elitism turned on dominated with 15 out of the top 25.

The setting for fitness function 2d in finding the snake of length 95 is 20^{th} in terms of average snake length. The next best for the same settings is 83. If the best snake were 83 instead of 95 then this setting would not even make it to the top 50 of the ordered list. So determining which setting is better is very difficult. There is no clear indication on which setting works and which does not based on crossover, mutation and elitism.

Table 2.3 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 90 compared to 84 without seeding and local growing. Fitness function 2b found a snake of length 93 compared to 80 without seeding and local growing. Fitness function 2c was able to find a snake of length 92 compared to 82 found without seeding and local growing while 2d was able to find a snake of length 95 compared to 79 found without seeding and local growing. Fitness function 2e was able to find a snake of length 89 while fitness function 2d made the most gains with a seeded population.

With seeding, no restriction on mutation and local growing, fitness function 2d performed better than the other fitness functions. Fitness function 2b is next. See Table 2.4. The difference between the best average snake length and the worst average snake length for the 25 runs (the top five runs from each fitness function) is just five edges. The longest snake found is of length 95 with fitness function 2d, there is no improvement when compared with seeding and no local growing but with local growing the GA is able to find a snake of length 95 two times.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	600	0.8	0.0080	1	86.6	90
	600	0.6	0.0400	0	86.4	93
	600	0.6	0.0067	1	85.6	88
	600	0.8	0.0200	0	85.4	88
	600	0.6	0.0400	1	85.4	91
2b	400	0.6	0.0400	1	88.0	89
	600	0.6	0.0080	0	87.6	94
	600	0.6	0.0020	1	86.8	89
	600	0.6	0.0020	0	86.6	88
	600	0.6	0.0200	1	86.6	90
2c	600	0.4	0.0200	1	86.8	90
	200	0.2	0.0080	0	86.4	91
	400	0.8	0.0080	0	86.4	88
	600	0.8	0.0200	1	86.0	90
	600	0.4	0.0067	1	86.0	88
2d	600	0.8	0.0400	1	90.0	95
	600	0.8	0.0020	1	87.6	94
	600	0.6	0.0200	1	87.2	89
	600	0.8	0.0200	0	87.0	90
	400	0.6	0.0400	0	86.4	95
2e	400	0.8	0.0400	1	85.6	87
	600	0.8	0.0067	1	85.6	87
	600	0.6	0.0080	1	85.4	89
	400	0.6	0.0067	1	85.2	87
	400	0.2	0.0400	0	85.0	90

Table 2.4: GA using Node Representation with seeding, no restriction on mutation and local growing

The best settings are determined using the average snake length for five trial runs. Fitness function 2d is the best with an average snake length of 90.0 with a population size at 600, crossover rate at 0.8, mutation rate at 0.04 and with elitism turned on. It is interesting to note that each fitness function is able to find the longest snake when elitism is turned off.

Taking the top five settings (ordered by average snake length), fitness function 2d performed better with high crossover rate and high mutation. When the top 25 settings instead of the top five are taken into account, there is no clear pattern in terms of crossover rate and mutation rate. Out of the top 25 settings, 20 have elitism turned on. The bottom 25 of the ordered

list has most settings with elitism turned off (21 out of 25). The difference in average snake length for the top 40 settings is just five edges. The setting that found one of the longest snakes of length 95 with elitism turned off has the next best snake of length 87. If it found a snake of length 87 again then the average snake length would be 84.8 with which it would be out of the top 40, so making any conclusions on which performed better is not possible based on either the snake length or the average snake length.

When the top 25 settings are considered for the remaining fitness functions the number of settings with elitism turned on is in the range of 12 to 15 indicating that none dominated to show a clear indication that the GA worked better when elitism is turned on. The difference in average snake length for fitness functions 2a, 2b, 2c and 2e for the top 40 settings is between 2 to 3.2 edges while the difference in average snake length between the best and worst settings is in the range of 5 to 8 edges. The time taken to complete the runs for all settings for each fitness function is around 98 hours, which is about 28 hours of overhead due to local growing on either ends of the snake. The performance gain in terms of average snake length is about 2 to 3 edges while there is no improvement in the longest snake found. Considering the overhead of local growing the gain is minimal.

Table 2.4 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 91 compared to 90 with seeding and no local growing; there is an improvement of one edge. Fitness function 2b found a snake of length 94 compared to 93 with seeding and no local growing. Fitness function 2c was able to find a snake of length 91 compared to a snake of length 92 found with seeding and no local growing. Fitness function 2d was able to find a snake of length 95 while fitness function 2e was able to find a snake of length 90.

With seeding, a restriction on mutation through positive mutation and no local growing, fitness function 2d performed better than other fitness functions. Fitness function 2c is next. See Table 2.5. The longest snake found is of length 93 with fitness function 2d, which is a decrease of two edges when compared to the GA with seeding, no restrictions on mutation and no local growing. The use of positive mutation had a negative effect on performance even with high mutation rates at around 0.04, which would allow the GA to explore more search space. The problem comes when the GA is stuck in a local maximum and since it does not take any bad steps in the search space due to positive mutation. The effect of being stuck in a local maximum has nullified any positive effect from positive mutation. This effect is visible on most fitness functions. The settings for fitness function 2a that found the longest snake had a best snake of length 91 and the worst snake of length 75. The GA run with the worst snake is probably stuck at a local maximum and is not able to move from that area in the search space, as positive mutation does not allow any bad steps. This is also evident in the best setting for fitness function 2d where the best snake is of length 93 while the worst snake is of length 69.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	600	0.8	0.0200	1	84.2	87
	600	0.6	0.0080	0	83.6	84
	600	0.8	0.0400	1	83.2	89
	600	0.4	0.0080	0	83.2	90
	200	0.4	0.0200	1	80.4	91
2b	400	0.2	0.0020	1	85.2	88
	600	0.8	0.0400	1	85.2	88
	600	0.8	0.0200	1	85.2	88
	600	0.6	0.0400	1	85.2	90
	600	0.2	0.0400	0	83.8	92
2c	600	0.6	0.0067	1	85.4	90
	600	0.8	0.0020	0	85.0	88
	400	0.8	0.0080	0	84.4	89
	600	0.8	0.0080	1	84.0	88
	400	0.8	0.0200	1	83.0	91
2d	600	0.8	0.0400	0	86.6	91
	400	0.8	0.0020	0	86.2	89
	600	0.8	0.0400	1	84.6	87
	600	0.6	0.0020	0	84.6	92
	400	0.6	0.0080	1	80.8	93
2e	600	0.8	0.0067	0	83.4	86
	600	0.4	0.0400	0	83.4	86
	600	0.6	0.0020	1	83.2	86
	200	0.8	0.0080	1	83.0	89
	200	0.6	0.04	0	81.2	91

Table 2.5: GA using Node Representation with seeding, restriction on mutation and no local growing

The best settings are determined using the average snake length for five trial runs. Fitness function 2d is the best with an average snake length of 86.6 with a population size of 600, crossover rate at 0.8 and mutation rate at 0.04 and with elitism turned off.

The best performing fitness function 2d performed better with high crossover rate and high mutation rates when the top five settings according to average snake length are taken into account. There is no clear pattern in terms of crossover rate and mutation rate when the top 25 settings are taken into account. All settings have crossover rate greater than 0.2 in the top 25. Out of the top 25 settings, 13 have elitism turned on. The bottom 25 of the ordered list have most

settings with elitism turned off (16 out of 25). The difference in average snake length between the top 40 is just 5.2 edges. The setting that found one of the longest snakes of length 93 with elitism turned on is 53rd in the ordered list and has the next best snake of length 85 while the worst snake it found is of length 69. Making any conclusion on which setting performed best is not possible based on either the snake length or the average snake length, as all settings are close.

When the top 25 settings are considered for the remaining fitness functions the number of settings with elitism turned on is in the range of 13 to 17 indicating that the GA worked slightly better when elitism is turned on. The difference in average snake length for fitness functions 2a, 2b, 2c and 2e for the top 40 settings is between 2.6 to 4.2 edges while the difference in average snake length between the best and worst settings is in the range of 7.6 to 10.8 edges. The time taken to complete the runs for all settings for each fitness function is around 157 hours, which is about 60 hours of overhead for checking mutated individuals and comparing with original individuals for each node in the snake. Comparing the time taken with no restriction on mutation and no local growing with seeding it took almost 3 times more time to complete. There is no performance gain in terms of average snake length and longest snake found. Considering the overhead of positive mutation, the GA did better with ordinary mutation.

Table 2.5 gives details on how each fitness function performed when average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 91 compared to 93 with seeding and local growing. Fitness function 2b found a snake of length 92 compared to 94 with seeding and local growing. Fitness function 2c was able to find a snake of length 91 compared to 91 found with seeding and local growing. Fitness function 2d was able to find a snake of length 93 while fitness function 2e was able to find a snake of length 91.

With seeding, restriction on mutation by allowing positive mutation and local growing, fitness function 2d performed better than other fitness functions. See Table 2.6.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	600	0.8	0.0400	0	85.8	88
	600	0.6	0.0020	0	85.8	90
	600	0.4	0.0400	1	85.4	88
	600	0.4	0.0400	0	85.2	86
	600	0.8	0.0080	1	85.0	90
2b	600	0.8	0.0067	1	87.0	90
	600	0.8	0.0200	1	86.6	88
	400	0.6	0.0080	1	86.4	91
	600	0.8	0.0400	1	86.4	89
	200	0.6	0.0200	1	84.2	92
2c	600	0.8	0.0067	1	86.8	89
	600	0.8	0.0400	0	86.2	89
	200	0.6	0.0400	1	86.0	91
	400	0.8	0.0200	1	86.0	88
	600	0.8	0.0400	1	86.0	88
2d	600	0.8	0.0067	1	88.4	92
	600	0.6	0.0400	1	86.8	90
	600	0.6	0.0200	1	86.6	91
	400	0.8	0.0067	1	86.4	90
	600	0.6	0.0080	0	85.0	93
2e	600	0.6	0.0400	1	85.6	87
	600	0.6	0.0020	1	85.6	87
	600	0.6	0.0200	0	85.6	88
	600	0.8	0.0400	1	85.4	89
	200	0.8	0.0200	1	83.6	90

Table 2.6: GA using Node Representation with seeding, restriction on mutation and local growing

The longest snake found is of length 93 with fitness function 2d, which is a decrease of two edges when compared to the GA with seeding, no restrictions on mutation and local growing. One main reason for the lack of finding longer snakes is with positive mutation only, which does not allow the GA to take any bad steps in the search space and therefore does not explore more search space when compared with ordinary mutation. This is evident with a huge difference in snake length between the best and worst snake found (93 and 75 respectively) for

the same settings using fitness function 2d. If the GA is stuck in a local maximum then it will never be able to come out of the local maximum even with the introduction of local growing. This is the main reason for a decrease in performance in finding the longest snake. The addition of local growing did not have much impact on either the average snake length or the longest snake found. The addition of local growing decreased the negative effect of the positive mutation when compared to GA with no local growing with positive mutation.

The best settings are determined using the average snake length for five trial runs. Fitness function 2c is the best with an average snake length at 88.4 with a population size of 600, with crossover rate at 0.8, mutation rate at 0.0067 and with elitism turned on.

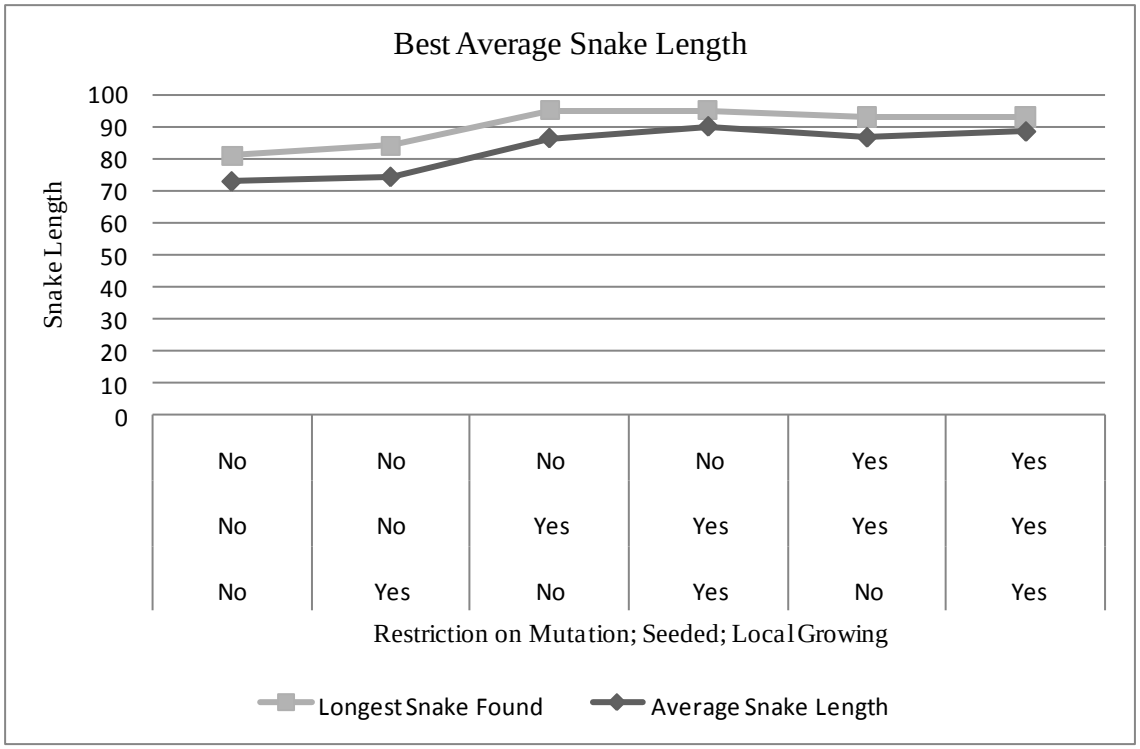
Fitness function 2d performed better with high crossover rate and elitism turned on when the top five are taken into account. There is no clear pattern in terms of a crossover rate and mutation rate when the top 25 settings are taken into account. Most settings have crossover rate greater than 0.2 in the top 25. Out of the top 25 settings, 14 have elitism turned on. The bottom 25 of the ordered list have most settings with elitism turned off (16 out of 25) and are dominated by smaller population sizes and lower crossover rates when compared to the top 25. The difference in average snake length between the top 40 is just 4.2 edges. The setting that found one of the longest snakes of length 93 with elitism turned on is 26^{th} in the ordered list. For the same setting, the next best snake of length 92 while the worst snake it found is of length 75. One more 75 instead of a snake of length 90 would take the setting to the bottom 10 of the ordered list. So making any conclusion on which setting performed best is not possible based on either the snake length or average snake length, as all settings are very close.

When the top 25 settings are considered for the remaining fitness functions the number of settings with elitism turned on is in the range of 12 to 17 indicating that the GA worked slightly

better when elitism is turned on. The difference in average snake length for fitness functions 2a, 2b, 2c and 2e for the top 40 settings is between 2 to 3.4 edges while the difference in average snake length between the best and worst settings is in the range of 7 to 9 edges. The time taken to complete runs for all settings for each fitness function is around 188 hours, which is about 30 hours of overhead of local growing when compared with seeding, no local growing and restriction on mutation with positive mutation. There is no performance gain in terms of longest snake. The average snake length improved by about 1.8 edges and is about 1.6 edges shorter when compared to the GA with seeding, local growing and no restriction on mutation. Considering the overhead of positive mutation, the GA does better with ordinary mutation.

Table 2.6 gives details on how each fitness function performed when average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 90. Fitness function 2b found a snake of length 92. Fitness function 2c was able to find a snake of length 91. Fitness function 2d was able to find a snake of length 93 while fitness function 2e was able to find a snake of length 90.

Graph 2.1 shows how the best average snake length and best longest snake found varied with seeding, local growing and restriction on mutation through positive mutation. The GA using a node representation performed poorly without seeding and no local growing which found a longest snake of length 81 and the best average snake length at 73. There is improvement in terms of both average snake length and longest snake found when local growing is added. It found a longest snake of length 84 and the best average snake length at 74.4.



Graph 2.1: GA using Node Representation.

When the initial population is seeded, with no local growing, the best average snake length went up by 12 edges to 86.4 while the longest snake found is 95 which is an 11 edge increase, which is the longest snake found using the node representation for the GA. When local growing is used along with seeding, the average snake length increased to 90 while the longest snake found is 95. With seeding, restriction on mutation with positive mutation and no local growing the average snake length went down by 3.4 edges to 86.6 and the longest snake found is 93. With the introduction of local growing, the average snake length went up to 88.4 and the longest snake found is 93 but is still a decrease from the GA with no restriction on mutation. The introduction of positive mutation only did not work, as it does not allow the GA to get out of local maximum in the search space, as bad moves are never taken. Using a node representation the GA was able to find a longest snake of length 95 using fitness function 2d.

2.3 USING TRANSITION SEQUENCE REPRESENTATION

2.3.1 INTRODUCTION

To find the longest snake in an 8-dimensional hypercube, a transition sequence representation is used to represent the snakes. The best snake found using the transition sequence representation reported here is 95.

2.3.2 REPRESENTATION

The snake contains a sequence of numbers ranging from 0 to 7 (0 to n-1 dimensions) and each number in the sequence corresponds to the binary bit being flipped (from 1 to 0 or 0 to 1) in the previous node's binary representation (00000000) in order to generate the current node's binary representation. The hypercube is symmetric and so one can start at any node say at zero and use it to get the remaining nodes by using the transition sequence.

The upper bound on the longest snake possible for $d \geq 7$ satisfies the following expression: upper bound $\leq 2^{d-1} - (2^{d-1}/ (20d- 41))$ [Snevily94]. The length of the individuals in the population is determined using this equation. In an 8-dimensional hypercube, the upper bound using the above equation is 126.92. Therefore, each individual's length is set to 127. If the initial population is seeded randomly, then to generate the transition sequence, a number between zero and seven is picked at random and used as the starting transition from node 0. To generate the next transition a number other than the present transition is picked at random. This process is repeated until the individual's transition sequence reaches a length of 127 (127 transitions generated randomly for each individual in the population).

2.3.3 FITNESS FUNCTION

To calculate the fitness, the individual in transition sequence representation is converted to node sequence representation. This node sequence is then used to calculate the fitness of the individual. All the five fitness functions are tested individually to see how well each fitness function did in comparison with each other. The five fitness functions used are fitness function 2a, fitness function 2b, fitness function 2c, fitness function 2d and fitness function 2e.

2.3.4 INITIAL POPULATION

The individuals are seeded with the longest snake from the 7-dimensional hypercube given in [Potter94] by converting the snake from node sequence representation to transition sequence representation. Each individual is expanded to a maximum length of 127 by randomly generating transition numbers.

2.3.5 ENGINEERED CONDITIONING

In an effort to increase efficiency and reliability in finding longer snakes and better average snake length, new strategies are added to the GA. One strategy is to protect the seeded transition sequence (0 to 49 in the individual, called the seeded region) throughout the GA approach by not performing crossover and mutation on the transitions in the seeded region. A second strategy is that after every 5 generations, each individual is taken and a simple local search technique is used to see if each individual's snake length can be increased by one at the end of the snake. To do this, the transition sequence representation is first converted to the node sequence representation and then the ending node's adjacent nodes are examined to see if there is any node that can be added and still is part of the snake, if so then that node is added. The node

sequence is then converted back to a transition sequence representation. If more than one adjacent node is eligible then one node is picked at random.

2.3.6 STOPPING CRITERIA

The stopping criterion is 200 generations with no change in the best individual or a maximum of 1000 generations, whichever occurs first.

2.3.7 SELECTION

Binary Tournament Selection is used to select parents for crossover. To select a parent for crossover two individuals are selected at random and the individual with the better fitness function is selected as parent 1. The same steps are used to select parent 2. Once the parents are selected, crossover is performed to generate the next generation individuals.

2.3.8 CROSSOVER

The Two-Point Crossover technique is used to generate the next generation individuals. A random number between zero and one is generated. If the generated number is greater than the crossover rate then the crossover is not performed. The transition sequence in parents is directly copied to the next generation individuals and they go to the next step (mutation). If the generated number is less than or equal to the crossover rate then the crossover is performed on the parents by picking two numbers at random between 0 and 127 and the transition sequence between the two points is exchanged between the parents to generate the next generation individuals. No crossover in the seeded region is accomplished by picking two numbers between 50 and 127 instead of 0 and 127.

2.3.9 MUTATION

To perform mutation each transition in the sequence (0 to 127) is taken and determined if mutation is to be performed by generating a random number. If the random number is less than the mutation rate then the transition in that position is modified such that the transition number is not the same as its adjacent transitions on either side of the transition. If the transition sequence is 2 4 6 and if transition 4 is selected for mutation then the new value for the mutated transition cannot be 2, 4 or 6. So out of remaining transitions (0, 1, 3, 5 and 7) one is picked at random. No mutation in the seeded region is accomplished by performing mutation from 50 to 127.

2.3.10 Elitism

When elitism is turned on, the best individual in each generation is added to the next generation without performing any crossover or mutation on it. With elitism turned off, all the individuals go through crossover and mutation to make it to the next generation depending on crossover and mutation rates.

2.3.11 EXPERIMENTAL SETUP

All the five fitness functions are tested using different combinations of population size, crossover rate, mutation rate and elitism.

Population size : 200, 400, 800

Crossover rate : 0.2, 0.4, 0.6, 0.8

Mutation rate : 0.002, 0.005, 0.0067, 0.008, 0.02

Elitism : 1 if turned on and 0 if turned off

To see how different representations affected the GA, the same settings (used for the node sequence representation) are used for the transition sequence representation even though it took considerably less time (around 12 hours) to run all the settings. Due to the large number of settings (about 120), only the top five settings determined by the average snake length from each fitness function are discussed. If the best snake found is not within the top five settings then the setting with the least average snake length in the top five is replaced by the setting that found the longest snake in the table.

To see how seeding and engineered conditioning affected the length of the snakes, all the settings are tested with different combinations. First with no seeding and no local growing, next with no seeding and local growing, next with seeding, no local growing, no restriction in seeded region, next with seeding, local growing, no restriction in seeded region, next with seeding, no local growing, restricted movement in seeded region lastly with seeding, local growing, restricted movement in seeded region.

2.3.12 RESULTS

Without seeding and no local growing, fitness function 2b performed better than other fitness functions. Fitness function 2a is next. See Table 2.7. The longest snake found is of length 72 with fitness function 2b. The best settings are determined using the average snake length for five trial runs. Fitness function 2b is the best with an average snake length of 64.2 with a population size of 400, crossover rate at 0.8, mutation rate at 0.005 and with elitism turned off. The top five settings for fitness function 2b had the highest crossover rate at 0.8 while for other fitness functions the top five settings mostly had crossover rates between 0.2 and 0.6.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	800	0.2	0.0067	1	61.0	64
	800	0.6	0.0050	1	61.0	66
	800	0.4	0.0080	0	60.6	63
	400	0.4	0.0080	0	59.8	64
	200	0.6	0.0067	0	59.0	68
2b	400	0.8	0.0050	0	64.2	68
	800	0.8	0.0067	1	64.0	69
	800	0.8	0.0080	1	63.2	66
	400	0.8	0.0050	1	62.8	67
	800	0.8	0.0067	0	60.6	72
2c	800	0.2	0.0080	1	39.8	53
	400	0.8	0.0080	1	38.8	61
	800	0.4	0.0067	1	38.6	59
	400	0.2	0.0067	1	37.8	49
	200	0.2	0.0067	1	37.4	68
2d	800	0.4	0.0050	1	36.4	41
	800	0.4	0.0200	1	35.8	38
	400	0.2	0.0067	1	35.6	40
	800	0.2	0.0080	1	35.6	41
	400	0.6	0.0080	1	35.4	44
2e	400	0.2	0.0050	0	59.4	64
	800	0.4	0.0080	1	59.4	66
	800	0.4	0.0067	1	58.8	61
	400	0.6	0.0020	1	58.2	63
	200	0.6	0.0067	0	55.2	67

Table 2.7: GA using Transition Representation with no seeding and no local growing

When it comes to mutation rate, most fitness functions performed well with mutation rates between 0.005 and 0.008. The settings with the mutation rate at 0.02 (which is the highest mutation rate the GA is tested on) performed poorly with almost all fitness functions and the settings with high mutation rates are at the very end when the settings are ordered by the average snake length. The settings with mutation rate at 0.002 (which is the lowest mutation rate the GA is tested on) also did not make it to the top five of the list. These settings performed better than the settings with 0.02 mutation rate. One reason why the GA performed well with lower mutation rate is that the GA checks each transition (all 127 transitions in each individual) and

decides if it should mutate that particular transition or not by generating a random number. With high mutation rate, the snake found can be destroyed unless elitism is used, so the GA did well with lower mutation rates.

For fitness functions 2c and 2d, all the settings in the bottom half of the ordered list have elitism turned off. This is not the case with other fitness functions where a clear pattern is not found. When the top 25 settings of the remaining fitness functions are considered, the settings with elitism turned on slightly dominated the top 25 with 12 to 16 out of 25. There is no clear pattern seen with population sizes but generally, settings with bigger population dominated the top 25. The average crossover rate (for the top 25 settings) for all fitness functions varied from 0.46 to 0.55. Settings with different crossover rates made it to the top 25 and no crossover rate dominated completely.

Table 2.7 gives the details on how each fitness function performed in finding the longest snake and the average snake length. Fitness function 2b is first with an average snake length of 64.2. Fitness function for 2a is next with 61.0. Fitness function 2e is next with 59.4. Fitness function 2c is next with 39.8 while fitness function 2d is last with 36.4. Fitness functions 2c and 2d did not perform well when compared to other fitness functions. The longest snake found using fitness function 2a is 68, using fitness function 2b is 72, using fitness function 2c is 68, using fitness 2d is 44 and using fitness function 2e is 67.

Without seeding and local growing, fitness functions 2b and 2c performed better than other fitness functions. Fitness function 2a is next. See Table 2.8.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	800	0.8	0.0080	1	69.6	76
	800	0.2	0.0050	1	68.2	70
	200	0.2	0.0067	0	67.4	73
	800	0.8	0.0067	1	67.4	71
	800	0.2	0.0020	0	67.2	68
2b	800	0.6	0.0067	1	74.8	83
	400	0.6	0.0080	1	74.4	79
	200	0.8	0.0050	0	74.0	77
	800	0.2	0.0080	1	73.6	75
	800	0.6	0.0080	1	73.2	77
2c	400	0.6	0.0067	1	74.8	82
	800	0.6	0.0080	0	74.8	81
	400	0.4	0.0050	0	74.4	81
	400	0.6	0.0080	1	74.0	78
	800	0.4	0.0050	1	74.0	76
2d	800	0.2	0.0020	0	68.2	71
	800	0.4	0.0050	1	67.8	81
	800	0.2	0.0020	1	66.6	69
	400	0.2	0.0020	1	65.2	72
	200	0.2	0.0020	1	62.8	76
2e	400	0.8	0.0080	1	69.4	72
	800	0.6	0.0080	1	68.8	73
	800	0.8	0.0080	1	68.2	76
	800	0.8	0.0080	0	67.4	74
	400	0.2	0.0067	1	67.2	70

Table 2.8: GA using Transition Representation with no seeding and no local growing

The longest snake found is of length 83 with fitness function 2b which is an increase of 11 edges when compared with no seeding and no local growing. The best settings are determined using the average snake length for five trial runs. Fitness functions 2b and 2c are best with an average snake length of 74.8 with both having a crossover rate of 0.6, a mutation rate 0.0067 and with elitism turned on. This is a 10-edge increase in the average snake length when compared with no seeding and no local growing. Fitness functions 2c and 2d gained the most with fitness

function 2c being almost at the bottom with no local growing at 39.8 to 74.8 with local growing which an increase of 35 edges. Fitness function 2d also gained about 30 edges when the average snake length is taken into account. The use of local growing on each individual after every 5 generations can be attributed to these performance gains.

When the top five settings ordered by the average snake length are taken into account, fitness function 2a performed well with large populations with either the least crossover rate or the highest crossover rate. Fitness function 2a preferred low mutation rates and it performed better when elitism is turned on. The best setting for fitness function 2b and 2c have the same crossover rate, mutation rate and with elitism turned on with both having the same average snake length. Fitness function 2d performed well with low crossover rates and low mutation rates of 0.2 to 0.4 and 0.002 to 0.005 respectively. Fitness function 2e slightly favored higher crossover and higher mutation rates when compared to the other fitness functions.

When the settings are ordered by average snake length, for fitness function 2d the settings in the bottom half of the ordered list have elitism turned off. This is not the case with other fitness functions where a clear pattern is not found. When the top 25 settings of the remaining fitness functions are considered, the settings with elitism turned on dominated with 13 to 18 out of the top 25 settings. There are no clear pattern for population and crossover with all different population sizes and crossover rates making to the top 25. The difference in the average snake length between the top 40 settings for all fitness functions except fitness function 2d is 5.2 to 6.4. The difference in the average snake length for fitness function 2d for top 40 is 15.7, which is more when compared to the other fitness functions. The best setting for fitness function 2d produced a best snake of length 81 and worst snake of length 49, a difference of 32 edges. For other fitness functions, the difference is about 13 edges.

Table 2.8 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 76. Fitness function 2b found a snake of length 83 which is the best snake seen out of all fitness functions. Fitness function 2c was able to find a snake of length 82 while fitness function 2d was able to find a snake of length 81. Fitness function 2e was able to find a snake of length 76.

With seeding, no restriction on movement in the seeded region and no local growing, fitness function 2b performed better than other fitness functions. Fitness function 2e is next. See Table 2.9.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	400	0.2	0.0050	0	80.0	89
	400	0.6	0.0050	1	79.4	80
	800	0.2	0.0050	0	79.4	83
	800	0.4	0.0067	1	78.6	81
	800	0.4	0.0020	1	78.0	84
2b	800	0.2	0.0080	1	81.4	86
	200	0.2	0.0050	1	79.4	82
	400	0.6	0.0067	1	79.4	84
	800	0.6	0.0067	1	79.2	86
	200	0.6	0.0080	1	78.0	87
2c	800	0.4	0.0080	1	79.8	85
	400	0.6	0.0067	1	78.8	83
	400	0.2	0.0080	1	78.6	83
	800	0.2	0.0067	1	78.6	82
	800	0.2	0.0080	1	78.6	82
2d	800	0.6	0.0067	1	78.8	82
	800	0.2	0.0050	1	78.0	83
	800	0.6	0.0050	1	77.8	80
	800	0.4	0.0080	1	77.2	80
	800	0.4	0.0050	1	75.0	88
2e	800	0.8	0.0080	1	80.4	82
	800	0.4	0.0020	1	78.8	81
	800	0.4	0.0067	1	78.2	81
	800	0.8	0.0067	1	77.6	82
	400	0.2	0.0050	1	77.4	86

Table 2.9: GA using Transition Representation with seeding, no restricted movement in seeded region and no local growing

The longest snake found is of length 89 with fitness function 2a which is an increase of 6 edges when compared with no seeding and local growing. The best settings are determined using the average snake length for five trial runs. Fitness function 2b is the best with an average snake length of 81.4 with a population size of 800, crossover rate at 0.2, mutation rate at 0.008 and with elitism turned on. This is a 7-edge increase in average snake length. The performance of all fitness functions increased in terms of the average snake length and the longest snake found. All fitness functions performed better when elitism is turned on. Settings with mutation rate between 0.005 and 0.008 gave better results. Settings with large populations generally gave better results.

When the settings are ordered by the average snake length, for all fitness functions (irrespective of population, crossover and elitism), the settings with high mutation rates performed poorly and are at the bottom of the ordered list. All the bottom 24 settings have mutation rate at 0.02 and within the bottom 24, the settings with elitism on performed better than the settings with elitism turned off. When it comes to population and crossover, there is no clear pattern. The top 25 settings in all fitness functions are dominated by settings with elitism turned on with 15 to 20 out of the top 25 settings. The difference in the average snake length for the top 40 from each fitness function varied from 4.4 to 9.8 edges. The difference between the best performing and the worst performing run for the best setting is 15 edges (89 to 74). The difference between the best and second best for the best setting is eight edges.

Table 2.9 gives details on how each fitness function performed when average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 89 compared to 76 without seeding and local growing. Fitness function 2b found a snake of length 87 compared to 83 without seeding and local growing. Fitness function 2c was able to find a snake of length 85 compared to a snake of length 82 found without seeding and local

growing while fitness function 2d was able to find a snake of length 88 compared to a snake of length 81 found earlier without seeding and local growing. Fitness function 2e was able to find a snake of length 86. Fitness function 2e made the most gains with a seeded population both in terms of average snake length and longest snake found.

With seeding, no restriction on movement in the seeded region and local growing, fitness function 2b performed better than the other fitness functions. Fitness function 2d is next. See Table 2.10.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	800	0.8	0.0067	1	85.4	86
	200	0.2	0.0067	0	84.4	88
	400	0.6	0.0067	1	84.4	88
	400	0.6	0.0020	1	84.2	87
	800	0.6	0.0050	0	83.2	89
2b	800	0.2	0.0080	1	87.2	88
	800	0.8	0.0067	1	86.4	88
	400	0.4	0.0050	1	85.8	89
	800	0.8	0.0050	1	85.6	88
	400	0.2	0.0067	0	84.6	91
2c	400	0.8	0.0067	1	85.2	88
	800	0.4	0.0080	1	85.2	86
	400	0.4	0.0080	1	84.6	88
	800	0.2	0.0067	1	84.6	89
	800	0.2	0.0080	1	84.6	88
2d	800	0.8	0.0050	1	85.8	90
	800	0.4	0.0067	1	85.6	88
	400	0.8	0.0020	1	85.2	87
	800	0.2	0.0080	1	85.2	88
	800	0.6	0.0020	1	85.2	87
2e	800	0.4	0.0050	1	84.4	89
	800	0.6	0.0050	1	84.4	88
	800	0.2	0.0020	0	84.2	88
	800	0.2	0.0080	1	84.2	86
	800	0.6	0.0020	1	82.6	90

Table 2.10: GA using Transition Representation with seeding, no restricted movement in seeded region and local growing

The difference between the best average snake length and the worst snake length for the 25 runs (the top five runs from each fitness function) is just six edges. The longest snake found is of length 91 with fitness function 2b which is an increase of 2 edges when compared to the GA with seeding and no local growing. The best settings are determined using the average snake length for five trial runs. Fitness function 2b is the best with an average snake length of 87.2 with a population size of 800, crossover rate at 0.2 and mutation rate at 0.008 and with elitism turned on. For all fitness functions, the GA performed poorly with higher mutation rate especially with no elitism. When the settings are ordered by average snake length, the settings with higher mutation rates are at the bottom of the list. One of the main reasons high mutation rates did not perform well is that the mutation is considered for each transition so it is likely that more than one transition for each individual is changed in each generation. Even a simple change in transition sequence changes the snake completely from the point of change as each transition depends on the previous transition to generate the node number when node sequence is generated. With elitism turned on and mutation rates high the GA performed relatively well with average snake length in the high 60s to low 70s when compared to the GA with elitism turned off and high mutation rates, which performed poorly with the average snake length in the high 30s to mid 40s.

The settings with higher mutation rates and with elitism turned on performed better than some settings with lower mutation rates and with elitism turned off while settings with high mutation rates and with elitism turned off performed the worst for all fitness functions. When it comes to population and crossover, there is no clear pattern. The top 25 settings in all fitness functions are dominated by settings with elitism turned on with 16 to 24 out of the top 25 settings. The difference in average snake length for the top 40 settings varied from 2.6 to 4.2

edges when all the fitness functions are considered. The difference between the best run and the worst run for the best setting is nine edges (89 to 74). The difference between the best run and the second best run for the best setting is seven edges.

Table 2.10 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 89 compared to 89 with seeding and no local growing, there is no improvement. Fitness function 2b found a snake of length 91 compared to 87 with seeding and no local growing. Fitness function 2c was able to find a snake of length 89 compared to a snake of length 85 found with seeding and no local growing. Fitness function 2d was able to find a snake of length 90 while fitness function 2e was able to find a snake of length 90. The performance gain in terms of longest snake using local growing is around 3 to 5 edges. The difference in the average snake length decreased and almost all the fitness functions performed well with different settings. The difference in the best snake found using the different fitness functions is just three edges (89 to 91).

With seeding, no local growing and restriction on the movement of individuals by not performing any crossover and mutation in the seeded region. Fitness function 2a performed better than the other fitness functions. Fitness function 2b is next. See Table 2.11.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	400	0.8	0.0200	1	81.8	86
	400	0.6	0.0067	0	80.8	84
	400	0.8	0.0080	0	80.0	85
	800	0.2	0.0200	0	80.0	83
	400	0.8	0.0020	1	78.2	89
2b	200	0.6	0.0067	1	81.6	90
	400	0.2	0.0080	0	81.2	89
	200	0.2	0.0067	0	80.6	86
	400	0.4	0.0080	1	80.6	83
	800	0.2	0.0080	1	80.4	84
2c	800	0.4	0.0200	1	80.6	84
	800	0.6	0.0200	1	80.4	85
	400	0.8	0.0080	0	80.2	87
	400	0.4	0.0200	1	79.4	84
	400	0.8	0.0050	1	79.0	81
2d	800	0.2	0.0200	1	80.2	82
	400	0.2	0.0200	0	80.0	83
	800	0.4	0.0067	1	79.2	89
	400	0.8	0.0080	1	79.0	90
	800	0.6	0.0200	1	79.0	85
2e	400	0.2	0.0067	1	80.0	82
	800	0.8	0.0080	0	80.0	85
	800	0.6	0.0080	1	79.4	83
	800	0.2	0.0200	1	79.0	82
	800	0.6	0.0080	0	78.6	80

Table 2.11: GA using Transition Representation with seeding, restricted movement in seeded region and no local growing

The longest snake found is of length 90 with fitness functions 2b and 2d. This is a decrease of one edge when compared with seeding and local growing without any restrictions on crossover and mutation. The effect of local growing is more than the effect of restricted movement. The best settings are determined using the average snake length for five trial runs. Fitness function 2a is the best with an average snake length of 81.8 with a population size of 400, crossover rate at 0.8, mutation rate at 0.02 and with elitism turned on. This is a decrease of five edges in the average snake length when compared to the GA with seeding, no restrictions on crossover, mutation and local growing.

With restricted crossover and mutation, the settings with higher mutation rates were able to make it to the top five for the first time in most fitness functions. When the top five settings from each fitness function are taken into account, all but one setting with mutation rate at 0.02 has elitism turned on and only when elitism is turned on did the GA with high mutation rate perform better. With elitism turned on, it has to hit the best snake only once and that snake will be preserved through all the generations due to elitism. With restricted crossover and mutation in the seeded region, the GA has less search space to search and with high mutation rates, it can search more search space as a change in one transition number changes the snake completely after that point, and the GA can potentially look at more places than with low mutation rates. This is the main reason why high mutation rates worked well with the introduction of restricted crossover and mutation as whatever the mutation rate the seed will remain the same from starting generation to ending generation.

When the settings are ordered by the average snake length, the settings with low mutation rates were at the end of the ordered list. These settings were still able to get an average snake length in low 70s using fitness functions 2a and 2e which is high compared to high 30 and low 40s generated by the GA with seeding, local growing and no restricted crossover and mutation. When it comes to population and crossover, there is no clear pattern. The top 25 settings in fitness functions 2a and 2b are dominated by settings with elitism turned off with 13 and 14 respectively out of the top 25 settings. For the remaining fitness functions, 16 to 19 settings of the top 25 settings have elitism turned on. The difference in average snake length for the top 40 settings varied from 3.6 to 5.4 edges for all fitness functions. The difference between the best performing and the worst performing run for the best setting is 13 edges (90 to 77). The difference between the best and the second best for the best setting is eight edges.

Fitness function 2a found a snake of length 89. Fitness function 2b found a snake of length 90 compared to 91 with seeding and local growing. Fitness function 2c was able to find a snake of length 87 compared to a snake of length 89 found with seeding and local growing. Fitness function 2d was able to find a snake of length 90 while fitness function 2e was able to find a snake of length 85.

With seeding, restriction on movement of individual in the seeded region and local growing, fitness function 2b performed better than the other fitness functions. Fitness function 2d is next. See Table 2.12.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	800	0.6	0.0200	1	83.4	87
	800	0.8	0.0020	1	83.2	88
	800	0.4	0.0067	1	83.0	87
	800	0.6	0.0050	0	82.6	84
	800	0.6	0.0050	1	81.6	90
2b	800	0.2	0.0200	1	87.4	90
	800	0.6	0.0080	1	86.8	90
	400	0.6	0.0200	1	86.4	91
	800	0.4	0.0200	1	86.0	90
	400	0.6	0.0067	0	84.6	93
2c	800	0.8	0.0080	1	85.6	89
	400	0.8	0.0080	1	85.4	88
	400	0.4	0.0080	1	85.0	89
	800	0.8	0.0020	0	85.0	92
	200	0.4	0.0067	1	83.4	93
2d	400	0.2	0.0050	1	86.8	92
	800	0.6	0.0050	1	85.8	88
	800	0.8	0.0200	1	85.8	89
	800	0.2	0.0080	0	85.4	89
	400	0.8	0.0080	1	81.0	95
2e	200	0.8	0.0067	0	84.2	91
	800	0.4	0.0020	1	82.8	85
	800	0.2	0.0020	0	82.6	89
	800	0.8	0.0050	0	82.6	86
	200	0.6	0.0080	0	82.0	86

Table 2.12: GA using Transition Representation with seeding, restricted movement in seeded region and local growing

The longest snake found is of length 95 with fitness function 2d, which is an increase of five edges when compared to the GA with seeding, restricted movement in seeded region and local growing. The best settings are determined using the average snake length for five trial runs. Fitness function 2b is the best with an average snake length of 87.40 with a population size of 800, crossover rate at 0.2, mutation rate at 0.02 and with elitism turned on. The longest snake of length 95 found using fitness function 2d does not reflect the best setting, as the next best in terms of snake length for the same setting is 82, so there is a variation of 17 edges between the top two runs. This is one of the main reasons why longest snake length is not taken as the performance criteria to identify the best settings even though the goal is to find the longest snake.

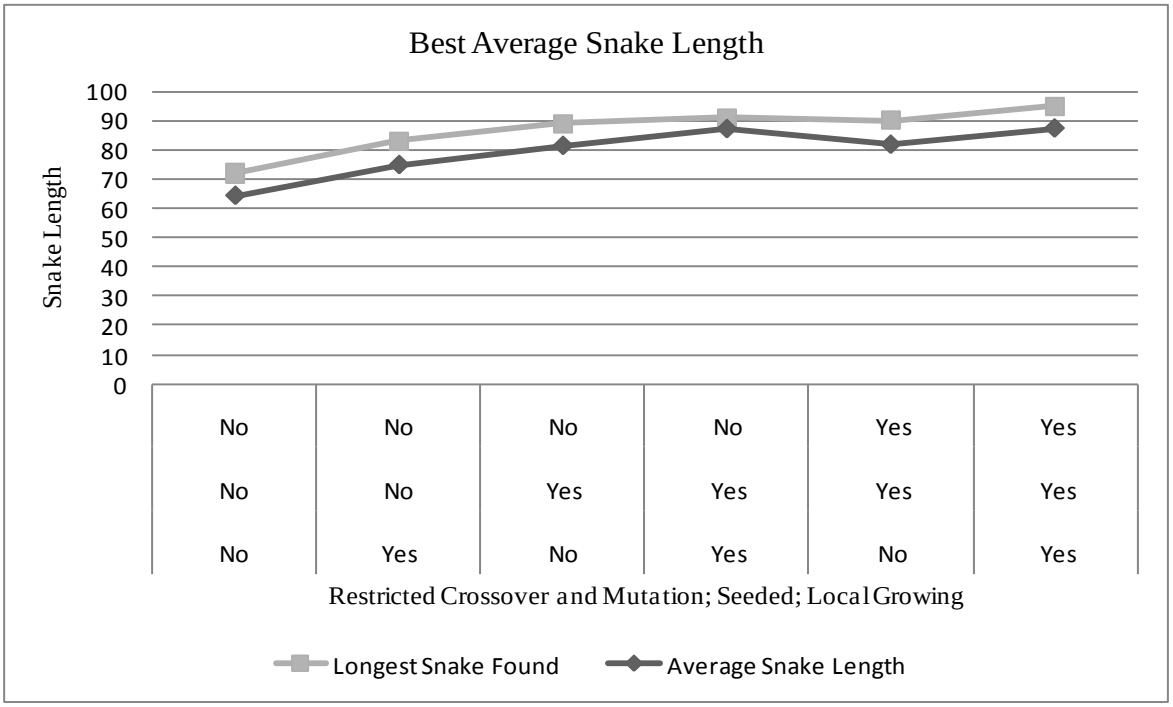
Most of the top settings with high mutation rates performed well when elitism is turned on. With elitism turned on, the GA has to hit the best snake only once and that snake will be preserved through all the generations due to elitism. The crossover rate does not seem to have much effect on either the average snake length or the longest snake found. All the different crossover settings from 0.2 to 0.8 made to the top 25 of the list. When all the settings are ordered by the average snake length, the settings with small population, low mutation rates and high mutation rates with elitism turned off performed worse with the average snake length in high 70s. This increase is due to local growing.

When it comes to population and crossover, there is no clear pattern. The top 25 settings in fitness function 2e are dominated by settings with elitism turned off with 16 out of the top 25 settings. The remaining fitness functions performed well when elitism is turned on with 15 to 20 out of the top 25 settings. The difference in the average snake length for the top 40 varied from 2.6 to 4.4 edges when all fitness functions are taken. The difference between the best performing and the worst performing run for the best setting is 25 edges (95 to 70). The difference between

best and second best for the best setting is 13 edges. The setting that found a snake of length 95 is 87th out of a total of 120 settings in terms of average snake length for fitness function 2d.

Table 2.12 gives details on how each fitness function performed when average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 90 compared to 89 with seeding, restricted movement and no local growing. Fitness function 2b found a snake of length 93 compared to 90 with seeding, restricted movement and no local growing. Fitness function 2c was able to find a snake of length 93 compared to a snake of length 87 found with seeding, restricted movement and no local growing. Fitness function 2d was able to find a snake of length 95 while fitness function 2e was able to find a snake of length 91.

Graph 2.2 shows how the best average snake length and longest snake found varied with seeding, local growing and restricted crossover, mutation in the seeded region. The GA using a transition sequence representation performed poorly with no seeding, no local growing and no restriction of crossover and mutation with a snake of length 72 and an average snake length of 64.2. There is improvement in terms of both the average snake length and the longest snake when local growing is added with an average snake length at 74.8 and longest snake of length 83, which is the largest gain seen both in terms of the average snake length and the longest snake found.



Graph 2.2: GA using Transition sequence Representation.

When the initial population is seeded, the average snake length went up by seven edges to 81.4 while the best snake found is 89, which is an increase of six edges. When local growing is used along with seeding the average snake length increased to 87.2 while the longest snake found is 91, which a two edge increase. With restricted crossover and mutation over the seeded region and no local growing, the average snake length went down by six edges to 81.8 and the longest snake is down by one edge to 90. The effect of local growing can be seen here. With the introduction of local growing, the average snake length went up to 87.4 and the longest snake found is 95, both are the best seen in the GA using a transition sequence representation.

2.4 USING BIT REPRESENTATION

2.4.1 INTRODUCTION

To find the longest snake in an 8-dimensional hypercube, a bit representation is used to represent the snakes. The best snake found using the bit representation reported here is 97.

2.4.2 REPRESENTATION

The individual in the GA contains a sequence of 256 bits (one for each node in 8-dimensional hypercube 2^8=256). If a node is used then its corresponding bit would be set to 1 indicating that the node is turned on. If the node is not used then its corresponding bit would be set to zero indicating that the node is turned off. The first node (Node 0) is always turned on, i.e., set to one. To generate a node sequence using the bit representation, the node sequence always starts at node 0 and then looks to see if any of its adjacent nodes (any of the 8 possible nodes adjacent to node 0) are turned on. If more than one node is turned on then the node with the lowest node number is always chosen. For node 0, if nodes 1, 2 and 4 are turned on then the sequence will always choose node 1 as it has the lowest node number. Once the node sequence is generated, the length of the snake and its fitness function are calculated.

The length of each individual is 256 bits, one for each node. If the initial population is seeded with the snake from the 7-dimensional hypercube then for each individual, only the bits corresponding to nodes in the seeded sequence are turned on. The remaining bits (>127) are randomly turned on or off. To restrict the movement, the bits from zero to 127 (the seeded region) are kept unchanged by restricting crossover and mutation to bits greater than 127. If the initial population is randomly seeded then all bits (0 to 255) are randomly turned on or off.

2.4.3 FITNESS FUNCTION

To calculate the fitness function, the individual in the bit representation is converted to a node sequence representation. This node sequence is then used to calculate the fitness of the individual. All the five fitness functions are tested individually to see how well each fitness function did in comparison with each other. The five fitness functions used are fitness function 2a, fitness function 2b, fitness function 2c, fitness function 2d and fitness function 2e.

2.4.4 INITIAL POPULATION

The individuals are seeded with the longest snake from the 7-dimensional hypercube given in [Potter94]. It is converted from node sequence representation to bit representation. First, all 256 bits in the individual are turned off (set to 0) and then the bits with the corresponding nodes in the node sequence are turned on. The bits corresponding to nodes greater than 127 are randomly turned on.

2.4.5 ENGINEERED CONDITIONING

In an effort to increase efficiency and reliability in finding longer snakes and better average snake length, new strategies are added to the GA. One strategy is to protect the seeded bits (0 to 127, called the seeded region) throughout the GA approach by not performing crossover and mutation on the bits in the seeded region. The GA behaves as if it is working on a 128-dimension search space and operates only in those dimensions. A second strategy is that after every 5 generations, each individual from the GA is taken and a simple local search technique is used to see if its snake length can be increased by one node at the end of the snake. To do this, the bit representation is first converted to the node sequence representation and then

the ending node's adjacent nodes are examined to see if there are any nodes that can be added and still be part of the snake. If so then that node is added. The node sequence is then converted back to the bit representation by turning the added node's corresponding bit to 1. If more than one adjacent node is eligible then the node with the lowest node number is picked.

2.4.6 STOPPING CRITERIA

The stopping criterion is 200 generations with no change in the best individual or a maximum of 1000 generations, whichever occurs first.

2.4.7 SELECTION

Binary Tournament Selection is used to select parents for crossover. To select a parent for crossover two individuals are selected at random and the individual with better fitness is selected as parent 1. The same steps are used to select parent 2. Once the parents are selected, crossover is performed to generate the next generation individuals.

2.4.8 CROSSOVER

The Two-Point Crossover technique is used to generate the next generation individuals. A random number between 0 and 1 is generated. If the generated number is greater than the crossover rate then crossover is not performed. The bits in the parents are directly copied to the next generation individuals and they go to next step (mutation). If the generated number is less than or equal to the crossover rate then crossover is performed on the parents by picking two numbers at random between 0 and 255 and the bits between the two points are exchanged between the parents to generate the next generation individuals. No crossover in the seeded

region is accomplished by picking the two crossover points between 128 and 255 instead of 0 and 255.

2.4.9 MUTATION

To perform mutation each bit (0 to 255) is taken and determined if mutation is to be performed by generating a random number. If the random number is less than the mutation rate then the bit in that position is modified. If the node is turned on (1) then it is turned off (0) and if it is turned off (0), it is turned on (1). No mutation in seeded region is accomplished by performing mutation from 128 to 255 bits.

2.4.10 ELITISM

When elitism is turned on, the best individual in each generation is added to the next generation without performing any crossover or mutation on it. With elitism turned off, all the individuals go through crossover and mutation to make it to the next generation depending on crossover and mutation rates.

2.4.11 EXPERIMENTAL SETUP

All the five fitness functions are tested using different combinations of population size, crossover rate, mutation rate and elitism.

Population size : 100, 200, 400, 600, 800

Crossover rate : 0.2, 0.4, 0.6, 0.8

Mutation rate : 0.002, 0.005, 0.0067, 0.008, 0.01517, 0.02, 0.04

Elitism : 1 if turned on and 0 if turned off

The different settings discussed in the thesis proposal are reduced to above settings as these settings alone for each fitness function for a combination of 280 (5×4×7×2) for 5 trial runs took around 260 hours to run. With the same combination of settings (120) as transition sequence representation the GA, using bit representations ran for 92 hours to complete 120 settings compared to 12 hours for the GA using transition sequence representation. After some initial runs and predicting how long it took, for the GA's with no seeding and no restrictions in the seeded region, the number of settings is reduced to 96 by decreasing population size to 400, 600 and 800 as the GA performed better with larger population. The GA with higher mutation rates did not perform well with transition sequence representation with no seeding and no restrictions in seeded, so the mutation range is decreased to 0.005, 0.002 or 0.0067, 0.008 and 0.01517. With restriction on the seeded region, the population size and mutation range are increased to include small population size and larger mutation rates. The total number of settings increased from 96 to 280 and it took around 260 hours (about 11 days) to complete all settings.

To see how seeding and engineered conditioning affected the length of the snakes, all the settings are tested with different combinations. First with no seeding and no local growing, next with no seeding and local growing, next with seeding, no local growing, no restriction in seeded region, next with seeding, local growing, no restriction in seeded region, next with seeding, no local growing, restricted movement in seeded region lastly with seeding, local growing, restricted movement in seeded region.

2.4.12 RESULTS

Without seeding and no local growing, Fitness function 2e performed better than the other fitness functions. Fitness function 2a is next. See Table 2.13.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	600	0.8	0.0050	1	58.6	60
	600	0.2	0.0050	1	58.4	62
	400	0.6	0.0050	1	58.2	62
	800	0.4	0.0050	1	58.2	62
	600	0.4	0.0050	1	57.0	63
2b	400	0.2	0.0050	1	14.8	48
	400	0.8	0.0152	1	7.4	13
	600	0.8	0.0152	1	7.2	9
	400	0.2	0.0067	1	6.6	8
	600	0.2	0.0080	1	6.6	24
2c	800	0.2	0.0050	1	50.0	57
	800	0.6	0.0050	1	45.6	54
	800	0.4	0.0050	1	45.0	48
	800	0.4	0.0067	1	44.6	48
	600	0.4	0.0050	1	38.6	61
2d	600	0.2	0.0152	1	9.6	26
	600	0.8	0.0050	1	8.6	28
	800	0.4	0.0067	0	6.6	12
	600	0.2	0.0050	1	6.4	12
	400	0.8	0.0067	0	6.2	11
2e	800	0.2	0.0050	1	59.8	65
	600	0.2	0.0050	1	59.2	61
	600	0.4	0.0050	1	59.0	65
	400	0.2	0.0050	1	58.8	61
	600	0.6	0.0050	1	58.8	66

Table 2.13: GA using Bit Representation with no seeding and no local growing

The longest snake found is of length 66 with fitness function 2e. The best settings are determined using the average snake length for five trial runs. Fitness function 2e is the best with an average snake length of 59.8 with a population size of 800, crossover rate at 0.2, mutation rate at 0.005 and with elitism turned on. The top five settings for fitness function 2e mostly favored a low crossover rate of 0.2 to 0.4.

When it comes to mutation rate most settings favored mutation rates between 0.005 and 0.008. The mutation rate of 0.01517, which is the highest mutation rate tested performed poorly with most fitness functions and the setting with high mutation rates at the very end when the settings are sorted by average snake length especially when elitism is turned off. One reason why the GA favored lower mutation rates is that the GA checks each of the 256 bits and decides if it should mutate that particular bit or not by generating a random number for each of the 256 bits. With a higher mutation rate, the snake found can be destroyed unless elitism is used, so the GA favored settings with lower mutation rates and with elitism turned on. With even one bit turned off, the snake is cut in to two parts and if the bit is near the 0 node (starting node for calculating snake length and fitness function) then the snake length would be reduced. Although the bit representation does not change the node sequence like the transition sequence representation, a small change in bit representation can affect the snake length severely.

Except for fitness function 2d, all other fitness functions did well only when elitism is turned on. The average snake length for fitness functions 2b and 2d are so low that making any conclusion on which settings work and which do not is very difficult. This is evident with all the top five settings for each fitness function having elitism turned on. Table 2.13 gives the details on how each fitness function performed in finding the longest snake and the average snake length. Fitness function 2e is first with an average snake length of 59.8. Fitness function for 2a is next with 58.6. Fitness function 2c is next with 50.0. Fitness function 2b is next with 14.8 while fitness function 2d is last with 9.6. Fitness functions 2b and 2d did not perform well. Fitness function 2e is able to perform on par with fitness function 2a. The GA favored a bigger population size for all fitness functions. The longest snake found using fitness function 2a is 63,

using fitness function 2b is 48, using fitness function 2c is 61, using fitness function 2d is 28 and using fitness function 2e is 66.

Without seeding and local growing, fitness function 2a performed better than the other fitness functions. Fitness function 2e is next. See Table 2.14.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	800	0.8	0.002	1	73.2	76
	800	0.2	0.002	0	72.6	80
	800	0.8	0.002	0	72.6	75
	600	0.8	0.002	0	71.6	74
	800	0.6	0.002	1	71.6	76
2b	800	0.8	0.002	1	71.0	76
	400	0.4	0.002	1	70.4	76
	600	0.8	0.002	1	70.4	82
	800	0.2	0.002	1	70.4	73
	400	0.8	0.002	1	65.2	83
2c	600	0.2	0.005	1	53.4	59
	800	0.2	0.002	1	50.8	72
	800	0.6	0.002	1	50.8	67
	800	0.4	0.005	1	50.6	69
	600	0.6	0.002	1	48.6	76
2d	600	0.2	0.015	1	9.4	25
	600	0.8	0.005	1	8.6	16
	600	0.2	0.005	1	7.2	17
	800	0.8	0.015	0	6.8	14
	800	0.6	0.005	1	6.4	12
2e	400	0.8	0.002	1	72.2	78
	600	0.6	0.002	1	71.8	78
	800	0.2	0.002	0	71.8	77
	800	0.4	0.002	1	71.8	80
	800	0.6	0.002	1	71.8	77

Table 2.14: GA using Bit Representation with no seeding and no local growing

The longest snake found is of length 83 with fitness function 2b which is an increase of 17 edges when compared with no seeding and no local growing. The best settings are determined using the average snake length for five trial runs. Fitness function 2a is the best with an average snake length of 73.2 with population size of 800, crossover rate of 0.8, mutation rate 0.002 and

with elitism turned on. This is a 14-edge increase in the average snake length. For fitness functions 2a and 2e, all the top five settings have a mutation rate of 0.002. For the GA with no local growing the lowest mutation rate used is 0.005 (using 0.002 might have given better results). So the difference between local growing and no local growing is a bit exacerbated. Fitness function 2b gained the most in terms of average snake length, from almost at the bottom with no local growing at 14.4 to 71.0 with local growing which is an increase of 57 edges. Fitness functions 2c and 2d did not have much impact with the introduction of local growing.

Fitness function 2a performed well with a large population. Crossover rate has almost no impact on either the longest snake found or the average snake length. The general trend is higher crossover rates outperformed lower crossover rates with other parameters remaining same. This is true in most but not all cases. Fitness function 2a preferred low mutation rates and it performed better with elitism turned on, though settings with elitism turned off also made it to the top of the list. Fitness functions 2b and 2e performed better with low mutation rates as all the top 5 settings have mutation rates at 0.002. Fitness function 2c also performed better with low mutation rates (0.002 to 0.005).

For fitness functions 2a, 2b and 2e, the most influential parameter is mutation. The settings with the lowest mutation rates performed better than the remaining settings irrespective of population size, crossover rate and elitism turned on or off. The lower the mutation rate, the better the GA performed in finding longer snakes and higher average snake lengths. For fitness function 2c, the top 25 settings are dominated by settings with lower mutation (0.002 to 0.008) and with elitism turned on. The bottom 25 are dominated by settings with elitism turned off. The settings with lower mutation rates and elitism turned off performed poorly. There is no clear pattern seen for settings using fitness function 2d.

Fitness function 2a found a snake of length 80. Fitness function 2b found a snake of length 83 which is the best snake seen out of all fitness functions. Fitness function 2c was able to find a snake of length 72 while 2d was able to find a snake of length 25. Fitness function 2e was able to find a snake of length 80.

With seeding, no restrictions in seeded region and no local growing, fitness function 2a performed better than the other fitness functions. Fitness function 2e is next. See Table 2.15.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	800	0.4	0.005	1	69.4	72
	600	0.2	0.005	1	68.4	72
	600	0.4	0.005	1	68.2	71
	600	0.6	0.005	1	67.0	69
	600	0.8	0.005	1	65.4	74
2b	600	0.2	0.005	1	68.8	72
	800	0.2	0.005	1	67.4	70
	400	0.4	0.005	1	67.0	71
	400	0.2	0.005	1	66.6	73
	600	0.6	0.005	1	66.0	69
2c	800	0.2	0.005	1	68.2	71
	800	0.4	0.005	1	66.8	70
	600	0.4	0.005	1	66.2	72
	600	0.6	0.005	1	66.0	67
	400	0.4	0.005	1	64.6	73
2d	800	0.6	0.005	1	63.0	67
	800	0.4	0.005	1	61.6	67
	800	0.2	0.005	1	61.0	67
	400	0.2	0.005	1	60.4	64
	600	0.8	0.0067	1	57.6	70
2e	600	0.2	0.005	1	69.0	71
	800	0.2	0.005	1	67.8	71
	400	0.2	0.005	1	67.6	71
	400	0.4	0.005	1	66.6	70
	600	0.4	0.005	1	66.6	72

Table 2.15: GA using Bit Representation with seeding, no restrictions in seeded region and no local growing

The longest snake found is of length 74 with fitness function 2a, which is a decrease of nine edges when compared with no seeding and local growing. The best settings are determined

using the average snake length for five trial runs. Fitness function 2a is the best with an average snake length of 69.4 with a population size of 800, crossover rate at 0.4, mutation rate at 0.005 and with elitism turned on. This is a decrease of four edges in terms of average snake length. The performance of all fitness functions is much closer in terms of both the average snake length and the longest snake found with the best and the worst (average snake length) in the 25 settings (the top 5 settings from each fitness function) at 69.4 and 57.6 respectively. All the fitness functions performed better when elitism is turned on. None of the settings with elitism turned off made the top five for all fitness functions. All the fitness functions performed better with a lower mutation rate at 0.005 which is the lowest mutation rate used for this GA. The decrease in performance is due to no local growing and not using the lower mutation rate of 0.002, which gave better results than 0.005 mutation rate with local growing. Large populations generally gave better results.

When the settings are ordered by the average snake length for each fitness function, the top half settings for all fitness functions have elitism turned on and the bottom half have elitism turned off. Within the top half, the settings with lower mutation rates again dominated as most settings with higher mutation rates were at the bottom of the top half. The same pattern is seen in the bottom half. Crossover did not have much impact on either the average snake length or the longest snake found.

Table 2.15 gives details on how each fitness function performed when average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 74 compared to 80 without seeding and local growing. Fitness function 2b found a snake of length 73 compared to 83 without seeding and local growing. Fitness function 2c was able to find a snake of length 73 compared to 76 without seeding and local growing while 2d was able to find a snake of length 70 compared to 25 earlier without any seeding and local growing. Fitness

function 2e was able to find a snake of length 72. Fitness function 2d made the most gains with seeding both in terms of the average snake length and in terms of the longest snake found which is mainly due to seeding the initial population.

With seeding, no restrictions in seeded region and local growing, fitness function 2b performed better than the other fitness functions. Fitness function 2a is next. See Table 2.16.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	800	0.4	0.002	0	84.8	90
	400	0.6	0.002	0	84.0	86
	600	0.4	0.002	1	83.8	88
	600	0.6	0.002	0	83.8	91
	400	0.8	0.002	1	83.4	87
2b	600	0.2	0.002	1	86.8	91
	600	0.8	0.002	1	86.0	91
	600	0.4	0.002	0	85.8	90
	800	0.6	0.002	1	85.8	91
	400	0.4	0.002	0	85.0	94
2c	400	0.6	0.002	0	82.8	91
	800	0.4	0.002	1	80.8	89
	600	0.8	0.002	1	79.8	88
	800	0.6	0.002	1	79.0	85
	600	0.4	0.002	0	78.6	86
2d	600	0.2	0.002	1	78.6	86
	800	0.6	0.002	1	78.0	92
	800	0.4	0.002	1	76.8	86
	600	0.4	0.002	1	76.0	92
	400	0.2	0.002	0	75.4	90
2e	600	0.6	0.002	0	84.2	87
	400	0.8	0.002	1	83.6	88
	800	0.6	0.002	0	83.2	89
	400	0.4	0.002	0	82.8	86
	600	0.4	0.002	1	82.8	85

Table 2.16: GA using Bit Representation with seeding, no restrictions in seeded region and local growing

The difference between the best average snake length and the worst average snake length for the top 25 runs (the top five runs from each fitness function) selected is nine edges. The longest snake found is of length 94 with fitness function 2b which is an increase of 20 edges

when compared with seeding and no local growing. The best settings are determined using the average snake length for five trial runs. Fitness function 2b is the best with an average snake length of 86.8 with a population size of 600, crossover rate at 0.2 and mutation rate at 0.002 and with elitism turned on.

For all fitness functions, the GA performed poorly with a higher mutation rate especially when elitism is turned off. All the settings with higher mutation rates are at the bottom when ordered by average snake length. Change in single transition changes the whole snake after the point of change when transition sequence representation is used. This is not the case with bit representation but with a change in one bit, the snake length could be affected, as the link could be broken when a bit is changed. Therefore, the snake length can be affected severely even with a change in one bit especially when the bit is near node 0.

The settings with elitism turned on and low mutation rate performed relatively well with the average snake length in high 60s to low 70s when compared to the settings with elitism turned off and high mutation rates which performed poorly with the average snake length in low 30s to mid 40s. The seeded snake from the 7-dimensional hypercube is disturbed due to high mutation rate and hence the snake length is less than 50.

For all fitness functions, the top 24 settings in each fitness function have mutation rate at 0.002, (the lowest mutation rate) with elitism either turned on or turned off. For the remaining settings, the ones with elitism turned on dominated while the bottom 25 settings of the list have elitism turned off. The settings with mutation rate greater than 0.005 and elitism turned off performed poorly.

Table 2.16 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length

91 compared to 74 with seeding and no local growing; there is an improvement of 17 edges. Fitness function 2b found a snake of length 94 compared to 73 with seeding and no local growing. The next best snake found for the same settings using fitness function 2b is 85. Fitness function 2c was able to find a snake of length 91 compared to a snake of length 73 found with seeding and no local growing. Fitness function 2d was able to find a snake of length 92 while fitness function 2e was able to find a snake of length 89. The performance gain in terms of longest snake using local growing is around 15 to 20 edges. The difference in average snake length decreased and almost all fitness functions performed well for different settings. The difference in the best snakes found using different fitness functions is five edges (89 to 94).

With seeding, no local growing and restrictions with no crossover and no mutation in seeded region, fitness function 2b performed better than the other fitness functions. Fitness function 2a is next. See Table 2.17. All 280 combinations of settings are used.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	800	0.6	0.0050	0	87.0	89
	400	0.8	0.0050	1	86.6	94
	800	0.2	0.0050	0	85.6	89
	200	0.2	0.0050	1	85.2	92
	200	0.2	0.0080	1	85.0	88
2b	800	0.8	0.0067	1	90.0	91
	800	0.6	0.0050	1	89.8	94
	600	0.8	0.0050	0	89.2	91
	800	0.4	0.0067	1	88.8	92
	800	0.8	0.0050	1	88.8	91
2c	600	0.4	0.0050	1	89.0	97
	800	0.4	0.0050	1	87.4	91
	800	0.2	0.0067	1	86.8	91
	600	0.4	0.0080	1	86.6	91
	400	0.6	0.0050	0	86.2	92
2d	600	0.2	0.0080	1	85.2	91
	800	0.4	0.0020	0	84.6	89
	800	0.2	0.0050	1	83.6	91
	800	0.6	0.0050	1	83.4	91
	800	0.2	0.0067	1	81.8	93
2e	600	0.2	0.0067	1	86.2	91
	600	0.4	0.0067	1	86.0	89
	400	0.4	0.0050	1	85.6	88
	800	0.2	0.0080	1	85.6	89
	800	0.2	0.0067	1	85.4	88

Table 2.17: GA using Bit Representation with seeding, restricted movement in seeded region and no local growing

The longest snake found is of length 97 using fitness function 2c which is an increase of 3 edges when compared with seeding and local growing with no restrictions. With the addition of restricted crossover and mutation, the GA performed better even without local growing both in terms of the average snake length and in terms of the longest snake found.

The best settings are determined using the average snake length for five trial runs. Fitness function 2b is the best with an average snake length of 90.0 with a population size of 800, crossover rate at 0.8, mutation rate at 0.0067 and with elitism turned on. The GA performed better with the largest population and the highest crossover rate given to the GA. All the fitness

functions performed better with low mutation rate (0.002 to 0.008), large population size and when elitism is turned on.

When the settings are ordered by the average snake length, the settings with high mutation rates were at the end of the ordered list especially settings with 0.02, 0.04 mutation rates and with elitism turned off. The average snake length for these settings is in the low 60s. The settings with mutation rates higher than 0.008 did not make it to the top 25 even when elitism is turned on. Settings with higher population size performed better than the settings with lower populations on most occasions. No particular pattern is observed with the crossover rate. Most of the top 25 settings for all fitness functions performed well when elitism is turned on (18 to 20 out of 25). The average mutation rate for the top 25 settings in each fitness function varied between 0.0057 to 0.0061 indicating that the GA did well with mutation rates mostly between 0.005 and 0.008. Settings with a mutation rate at 0.002 also made it to the top 25. The difference in the average snake length for the top 40 settings in each fitness function varied from 3.6 to 8.2 edges indicating that all the settings are closely matched. The setting that found a snake of length 97 has the next best snake at 90 and the worst snake at 84. So making any conclusion on which setting gives the longest snake is very difficult.

Table 2.17 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 94 compared to 91 with seeding and local growing. Fitness function 2b found a snake of length 94. Fitness function 2c was able to find a snake of length 97 compared to a snake of length 91 found with seeding and local growing. Fitness function 2d was able to find a snake of length 93 while fitness function 2e was able to find a snake of length 91.

With seeding, restrictions with no crossover and no mutation in the seeded region and local growing, fitness function 2c performed better than the other fitness functions. Fitness function 2b is next. See Table 2.18.

Fitness Function	Size	Crossover	Mutation	Elitism	Average Length	Longest Snake
2a	600	0.4	0.0067	1	89.8	91
	800	0.4	0.0020	0	89.0	91
	800	0.8	0.0050	1	89.0	92
	800	0.2	0.0050	1	88.2	91
	800	0.2	0.0067	1	86.8	94
2b	800	0.8	0.0050	1	91.0	91
	800	0.2	0.0050	1	90.0	94
	800	0.8	0.0020	1	89.8	91
	600	0.4	0.0050	1	89.6	92
	800	0.4	0.0067	1	86.6	97
2c	600	0.8	0.0050	1	91.8	93
	800	0.6	0.0050	1	91.2	97
	800	0.4	0.0067	1	90.4	93
	800	0.2	0.0050	1	89.4	93
	800	0.2	0.0050	0	89.2	95
2d	400	0.2	0.0067	1	89.2	93
	800	0.4	0.0067	1	86.4	91
	800	0.2	0.0050	0	85.2	88
	800	0.2	0.0050	1	85.0	90
	800	0.4	0.0050	1	84.6	91
2e	800	0.6	0.0080	1	88.2	91
	600	0.2	0.0050	1	87.2	88
	600	0.4	0.0020	0	87.2	89
	200	0.4	0.0080	1	87.0	91
	800	0.4	0.0050	1	86.4	93

Table 2.18: GA using Bit Representation with seeding, restricted movement in seeded region and local growing

The longest snake found is of length 97 with fitness functions 2b and 2c. The best settings are determined using the average snake length for five trial runs. Fitness function 2c is the best with an average snake length of 91.8 with a population size of 600, crossover rate at 0.8, mutation rate at 0.005 and with elitism turned on.

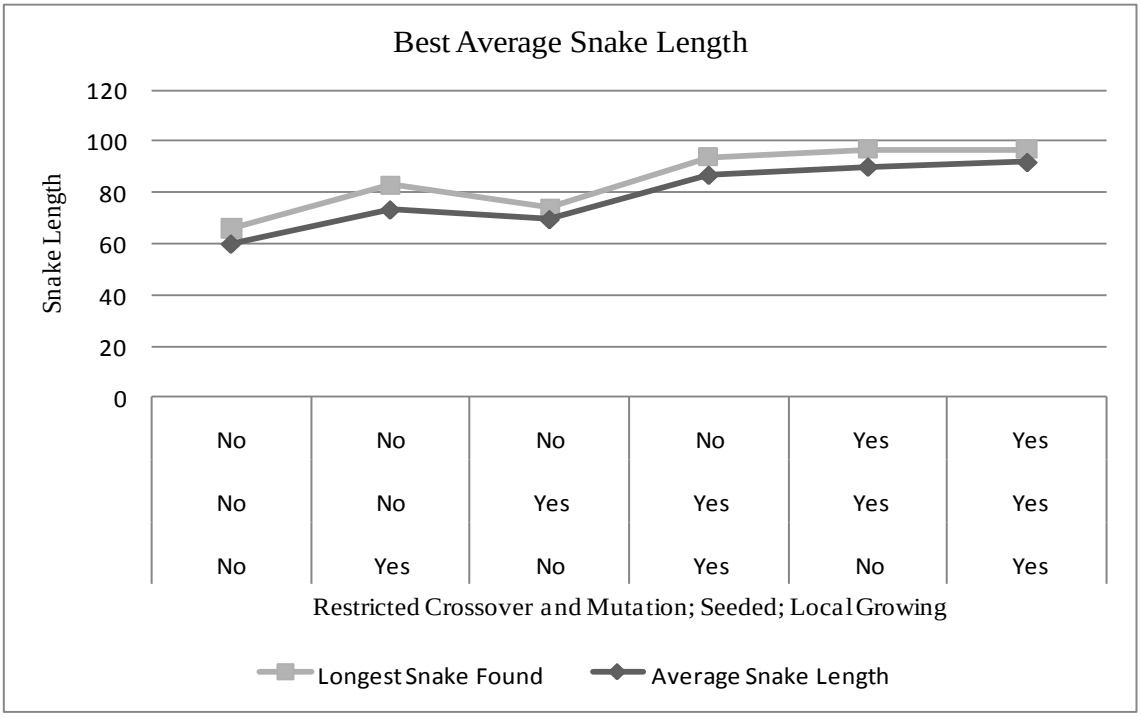
The crossover rate does not seem to have much effect on either the average snake length or the longest snake found. All the different crossover rates from 0.2 to 0.8 made the top 25 (the top five from each fitness function). When all the settings are ordered by the average snake length, the settings with small populations, high mutation rates, and with elitism turned off performed worse with the average snake length in mid 60s. The settings with high mutation rates (greater than 0.008) performed poorly. The performance decreased with an increase in mutation rate especially when the settings have elitism turned off. The bottom half of the ordered list is dominated by the settings with elitism turned off. Most of the settings with low population size ($\leq$200) did poorly. When the top 25 settings from each fitness function are taken, the settings with elitism turned on dominated with 16 to 19 out of the top 25 settings. The average mutation rate for the top 25 settings for all the fitness functions varied from 0.0058 and 0.0065 indicating that low mutation rates dominated with most settings having mutation rate around 0.005. The average snake length for the top 40 settings in each fitness function varied from 3.2 to 9.2 edges. The setting that found the best snake of length 97 using fitness function 2c has the next best snake at 91 while the worst run found a snake of length 86.

Table 2.18 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 94. Fitness function 2b found a snake of length 97. Fitness function 2c was able to find a snake of length 97. Fitness function 2d was able to find a snake of length 93 while fitness function 2e was able to find a snake of length 93.

All the 280 combinations of settings are used for the GA and unlike the GA using a transition sequence representation where the best settings are ones with higher mutation rates, the GA using a bit representation performed better with lower mutation rates. The effect of local

growing is not as much as in the GAs using a transition sequence representation. The fact that the GA was able to find a snake of length 97 without local growing shows how effectively the GA works when given the right representation. The only problem is that instead of having individuals of length 127, the bit representation needs individuals of length 256. This slows the GA considerably and also changing transition sequence to node representation is easier than changing bit representation to node representation.

Graph 2.3 shows how the best average snake length and the best longest snake found varied with seeding, local growing and restriction in seeded region. The GA using a bit representation performed poorly without seeding of the initial population along with no local growing and no restriction in the seeded region with a snake of length 66 and the average snake length at 66. There is improvement in terms of both the average snake length and the longest snake when local growing is used with no seeding and no restrictions in seeded region with the average snake length at 73.2 and the longest snake of length 83. This is the largest gain in terms of both the average snake length and the longest snake found.



Graph 2.3: GA using Bit Representation.

When the initial population is seeded, the average snake length went down by four edges to 69.4 while the best snake found is 74, which is a 9-edge decrease when compared to no seeding and local growing. When local growing is used along with seeding the average snake length increased to 86.8 while the longest snake found is 94 which a 20 edge increase. With restricted crossover and mutation over the seeded region and no local growing, the average snake length went up by three edges to 90 and the longest snake found is 97. With the introduction of local growing the average snake length went up to 91.8 and the longest snake found is 97, both are the best seen with bit representation using the GA and are the best results using any representation for the GA.

2.5 CONCLUSION

The GA approach was able to find a snake of length 97 using bit representation with seeded individuals and restricted crossover and mutation in the seeded region without local growing and with local growing.

CHAPTER 3

PARTICLE SWARM OPTIMIZATION

3.1 INTRODUCTION TO PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is similar to the Genetic Algorithms (GA) in that the system is initialized with a population of random solutions. It is unlike the GA; however, in that, each potential solution is assigned a randomized velocity and the potential solutions, called particles are then flown through the multi-dimensional search space [Eberhart07]. In the snake-in-the-box problem each particle flies in a 128 or 256 (128 for the transition sequence representation and 256 for the bit representation) dimensional search space having independent velocity in each dimension that is constantly updated by the particle's own experience and the experience of the particle's neighbors.

The main idea behind the development of the PSO is the social sharing of information among individuals of a population that is not seen in the GA. In the PSO, search is conducted by using a population of particles, corresponding to individuals as in the case of the GA. Unlike the GA, there is no natural evolution where the next generation is used to find new solutions, the PSO particles go through the search space updating their position in the multi-dimensional search space after every iteration. Iteration in the PSO is similar to generation in the GA except that in PSO the same particles are used in the next iteration unlike the GA where each generation has new individuals.

The key difference between the PSO and the GA is the exchange of information. In the GA, the individuals do not exchange information except when there is a crossover, which is

limited to just two individuals participating in the crossover and not the whole population. The PSO is based on the exchange of information between the individuals as the entire individuals are compared and the best individual is identified as global best. The global best individual is then used to adjust each particle's position along with the particle's own experience from previous iterations. This way all particles are exchanging information in the form of global best.

Keeping track of its own best position establishes the particle's experience implying a local search along with global search emerging from the experience of the whole swarm in the form of global best by keeping track of the best individual in the swarm (population).

3.1.1 PARTICLE SWARM OPTIMIZATION ALGORITHM

The velocity of the particle is calculated using the equations suggested by [Tasgetiren03]. The velocity of the initial population's particle is calculated using the below formula. This velocity is calculated for each dimension of the particle. $v^0_{i,\,d} = V_{min} + (V_{max} - V_{min}) \times rand()$. In the equation, i is the i^{th} particle and d is the d^{th} dimension and 0 says that it is the initial population. The power in v^0 typically represents the iteration and rand() is the function that returns a random number between zero and one.

The velocity of the particle in each dimension is restricted between V_{max} and V_{min}. V_{max} is set to +4 while V_{min} is set to -4. These values are arbitrary values. These restrictions on velocity help in keeping the swarm (particles) together (not stretched over a large area). For the transition sequence representation, discrete PSO is used to handle the discrete values (0 to 7) for the Snake Problem while for the bit representation binary PSO is used. In the discrete PSO, particle velocity in each dimension is used to determine the change in value in that dimension for

the next iteration while in the binary PSO, a sigmoid function is used to determine if the node is turned on or off (1 or 0).

For the discrete PSO, the calculated velocity in each dimension is converted into an integer and then clamped between V_{max} and V_{min} and this velocity replicates the job of creep mutation, that is, the velocity determines how much the transition sequence value will change in that dimension and in what direction (increase or decrease in the transition value). If the new transition sequence value is greater than 7, then the new value is clamped to 7 or if the new value is less than 0 then the new value is clamped to 0.

For the binary PSO, a sigmoid value is calculated using the sigmoid function $1/ (1+ e^{-1\times \text{particle velocity}})$. After finding the sigmoid value using the sigmoid function, a random number between 0 and 1 is generated and if the number is less than the sigmoid value then the bit is set to one and if the number is not less than the sigmoid value then the bit is set to zero.

Each Particle stores its personal best (pb) seen so far. In addition, the population keeps track of the global best (gb) they have seen so far.

In the next iteration, the change in velocity is calculated for each dimension of the particle in the population using the following formula (see below). $Pb_{i,d}$ is the personal best value for the i^{th} particle and d^{th} dimension. gb_d is the global best value for the d^{th} dimension. $x^{k-1}_{i,d}$ is the value from previous iteration of the i^{th} particle's d^{th} dimension. The values of pb, gb and x are either 1 or 0 for the binary PSO and between 0 and 7 for the discrete PSO. The values of C_1 and C_2 influence the local search verses global search. If C_1 is greater than C_2 then the algorithm gives more emphasis on the local search space and a greater C_2 value than C1 value makes the algorithm emphasize more on global search space and less emphasis on local search. C1 is exploitation of the search space and C2 is exploration of the search space [Eberhart07].

$$\Delta v^{k-1}_{i,d} = C_1 \times rand() (pb_{i,d} - x^{k-1}_{i,d}) + C_2 \times rand() (gb_d - x^{k-1}_{i,d})$$

$$v^{k}_{i,d} = h(inertia \times v^{k-1}_{i,d} + \Delta v^{k-1}_{i,d})$$

The new velocity for the k^{th} iteration is calculated using the above equation. The inertia in the equation restricts how much the velocity from pervious iteration has influence over the next iteration. The higher the inertia the more influence the previous iteration velocity has on next iteration velocity. It essentially helps the PSO to get out of a local maximum in the search space for the Snake Problem.

The h function restricts the velocity between V_{max} and V_{min}. If the velocity is greater than V_{max} then the velocity is clamped to V_{max}. If the velocity is less than V_{min} then the velocity is clamped to V_{min}. The velocity remains as it is when it is in between V_{max} and V_{min}. This clamping is done so that the individuals in the swarm stay together in the search space and do not disperse over all the search space. The velocity is then used to update the particle's transition value or bit value according to the representation used.

This is done for all the dimensions of all the particles. Once new $x^{k}_{i,d}$ and the new fitness functions are calculated, the particle best and global best are updated if the fitness value of the particle increases. The whole process is repeated until there is no change in the global best for a certain number of iterations or a maximum predetermined number of iterations, whichever occurs first.

3.2 USING TRANSITION SEQUENCE REPRESENTATION

3.2.1 INTRODUCTION

To find the longest snake in an 8-dimensional hypercube, a transition sequence representation is used to represent the snakes. The best snake found using the transition sequence representation reported here is 92.

3.2.2 REPRESENTATION

The particle in the PSO contains a sequence of numbers in the range 0 to 7 (0 to n-1 dimensions) and each number in the sequence corresponds to the binary bit being flipped from 1 to 0 or 0 to 1 in the previous node's binary form (00000000) in order to generate the current node's binary form. The hypercube is symmetric and so one can start at any node say at zero and use it to get the remaining nodes by using the transition sequence.

The upper bound on the longest snake possible for $d \geq 7$ satisfies the following expression: upper bound $\leq 2^{d-1} - (2^{d-1}/ (20d- 41))$ [Snevily94]. The length of the particles in the swarm is determined using this equation. In an 8-dimensional hypercube, the upper bound using the above equation is 126.92. Therefore, each particle's length is set to 127. If the initial population is seeded randomly then to generate the transition sequence, a number between zero and seven is picked at random and is set as the starting transition from node 0. To generate the next transition a number other than the present transition is picked at random. This process is repeated until the particle's transition sequence reaches a length of 127 (127 transitions generated randomly for each particle in the population).

3.2.3 FITNESS FUNCTION

To calculate the fitness, the transition sequence is converted to node sequence representation. This node sequence is then used to calculate the fitness of the particle. All the five fitness functions are tested individually to see how well each fitness function did in comparison with each other. The five fitness functions used are fitness function 2a, fitness function 2b, fitness function 2c, fitness function 2d and fitness function 2e.

3.2.4 INITIAL POPULATION

The particles are seeded with a snake of length 50 from the 7-Dimensional hypercube given in [Potter94], which is converted from node sequence representation to transition sequence representation, and the length is increased by randomly generating numbers between 0 to 7 to get a sequence of transitions of length 127.

3.2.5 ENGINEERED CONDITIONING

In an effort to increase efficiency and reliability in finding longer snakes and better average snake length, new strategies are added to the PSO. One strategy is to protect the seeded transition sequence (0 to 49 in the particle, called the seeded region) throughout the PSO approach by not performing any velocity updates in the seeded region. The PSO behaves as if it is working on a dimension 78 (50 to 127) search space and updates are made only in those dimensions. A second strategy is that after every five iterations, each particle is taken and a simple local search technique is used to see if each particle's snake length can be increased by one edge at the end of the snake. To do this, the transition sequence representation is first converted to node sequence representation and then the ending node's adjacent nodes are

examined to see if there are any nodes that can be added and still be part of the snake. If so then that node is added. The node sequence is then converted back to transition sequence representation. If more than one adjacent node is eligible then one is picked at random.

3.2.6 STOPPING CRITERIA

The stopping criterion is 200 iterations with no change in global best or a maximum of 1000 iterations, whichever occurs first.

3.2.7 DISCRETE PSO

The transition sequence representation has more than two values for each dimension. So for each of 127 dimensions, the velocity in each dimension is used to determine how much the transition value changes and in what direction. The velocity is clamped between +4 and -4 so the transition value can at most change by 4.

3.2.8 EXPERIMENTAL SETUP

All the five fitness functions are tested using different combinations of population size, C1, C2 and inertia values.

Population size : 100, 200, 400, 800

C1 : 1, 1.4, 1.8, 2, 4, 5

C2 : 1, 1.4, 1.8, 2, 4

Inertia : 0.1, 0.2, 0.4, 0.6, 0.8

The different settings discussed in the thesis proposal are reduced to the above settings as these settings alone for each fitness function for a combination of 600 (4×6×5×5) for 5 to 10 trial

runs took around 50 hours to run. Due to the large number of combinations, only the top five settings determined by the average snake length for each fitness function are discussed. If the best snake found is not within the top five settings then the setting with the least average snake length in the top five is replaced by the setting that found the longest snake in the tables discussed in this section.

To see how seeding and engineered conditioning affected the length of the snakes, all the settings are tested with different combinations. First with no seeding and no local growing, next with no seeding and local growing, next with seeding, no local growing, no restriction in seeded region, next with seeding, local growing, no restriction in seeded region, next with seeding, no local growing, restricted movement in seeded region lastly with seeding, local growing, restricted movement in seeded region.

3.2.9 RESULTS

Without seeding and no local growing, fitness function 2a performed better than the other fitness functions. Fitness function 2e is next. See Table 3.1. The longest snake found is of length 63 with fitness function 2a and 2e. The best settings are determined using the average snake length for 10 trial runs. Fitness function 2a is the best with an average snake length of 55.40 with a population size of 800, C1 at 4, C2 at 1.4 and inertia at 0.1. With low inertia values and higher value for C1 than C2, the PSO with no seeding and no local growing was able to find better snakes with exploitation. The concept of inertia weight was developed to better control exploration and exploitation [Eberhart07]. Lower values tend to favor exploitation and higher values tend to favor exploration. For the Snake Problem the PSO with no seeding and no local

growing seems to favor exploitation over exploration. Table 3.1 gives the details on how each fitness function performed in finding the longest snake and the average snake length.

Fitness Function	Size	C1	C2	Inertia	Average length	Longest Snake
2a	800	4	1.4	0.1	55.40	58
	800	5	1.4	0.1	52.70	63
	800	4	1.8	0.1	52.10	58
	800	4	1	0.1	51.90	60
	800	4	2	0.1	51.80	57
2b	800	1.4	2	0.1	32.80	48
	800	1	1.8	0.4	32.70	39
	400	1.4	2	0.4	30.40	44
	800	1.8	1.4	0.1	30.40	47
	400	2	1.8	0.4	29.10	50
2c	200	1.8	1.4	0.1	10.70	23
	800	1	2	1	10.40	25
	400	5	4	0.8	10.20	24
	800	5	1	0.1	10.10	25
	200	4	2	0.4	8.90	32
2d	400	1	1.8	1	10.30	21
	200	5	1.4	1	10.00	27
	100	4	2	1	9.50	19
	200	5	4	1	9.50	22
	200	4	2	0.4	9.50	19
2e	800	4	1.4	0.4	54.20	61
	800	4	1.4	0.1	53.60	57
	800	1.8	1.8	0.4	52.60	63
	400	2	1.4	0.4	52.10	60
	800	2	1.4	0.4	52.00	57

Table 3.1: PSO using Transition Representation with no seeding and no local growing

Fitness function 2a is first with an average snake length of 55.40. Fitness function 2e is next with 54.20. Fitness function 2a favored exploitation over exploration with comparatively larger values for C1 over C2 and low inertia values. Fitness function 2e had similar values for C1 and C2 and slightly higher inertia values when compared to fitness function 2a. The remaining fitness function's performance is very low and the settings for these functions do not show any clear pattern. Getting a good snake using fitness functions 2b, 2c and 2d is more of a chance.

When the top 25 and the bottom 25 settings from fitness function 2a are taken, the average population size for the top 25 settings is 704 suggesting that PSO did well with large populations while the average population size for the bottom 25 settings is 136 suggesting that PSO did poorly with small population sizes. The C1 and C2 average for the top 25 is 3.13 and 1.57 respectively suggesting that PSO did well with exploitation while the C1 and C2 average for the bottom 25 is 2.84 and 2.64 respectively suggesting that a balanced PSO did poorly. The average inertia for the top 25 is 0.24 suggesting the PSO did well with exploitation (as lower inertia values favor exploitation). The average inertia for the bottom 25 is 0.98 suggesting that PSO did poorly with exploration (higher inertia values favor exploration). For fitness function 2e, average population size, average C1 value, average C2 value and the average inertia for the top 25 are 680, 3.2 1.58 and 0.23 respectively. The values are similar to values seen with fitness function 2a. The bottom 25 for fitness function 2e also had a low average population size of 120 and a very high average inertia value of 0.98.

Fitness function 2b is next with C1 and C2 values comparatively nearer to each other and inertia values slightly higher than minimum. It found a snake of length 48. Fitness functions 2c and 2d did not perform well as compared. Since the particles are not seeded, it is easier to find nodes that are shared fewer times than to find nodes that are shared more times by the nodes in the snake. Therefore, fewer shared nodes dominate the more shared nodes and this leads to not being able to find larger snakes using these fitness functions. These fitness functions are expected to perform on par with the remaining fitness functions when the particle is seeded and restricted movement on particle is enforced as the seed is tightly wrapped which means that more share the same skin nodes.

Fitness function 2e is able to perform on par with fitness function 2a as each skin node count has a weight associated with it. A snake with four nodes shared 3 times has less fitness than the snake with four nodes shared 5 times (with all other parameters remaining the same), which is not the case with fitness functions 2b, 2c and 2d where both have the same value.

Without seeding and local growing, fitness function 2a performed better than the other fitness functions. Fitness function 2e is next. See Table 3.2.

Fitness Function	Size	C1	C2	Inertia	Average length	Longest Snake
2a	800	1.8	1.8	0.4	58.80	61
	800	1.8	1.8	0.2	58.00	64
	800	1.4	1	0.6	57.20	63
	800	5	1.8	0.2	57.00	61
	800	1.4	1.8	0.4	56.40	67
2b	800	1	2	0.4	43.40	62
	800	1	1.4	0.4	41.80	51
	400	1	1.8	0.2	40.80	50
	800	1	1.8	0.6	40.20	63
	800	1.4	1.4	0.6	39.80	52
2c	400	4	4	0.6	13.60	23
	200	1.8	2	0.2	12.60	26
	800	4	4	0.2	12.40	22
	100	1	1.4	0.8	12.20	35
	200	1	1.4	0.8	11.60	16
2d	400	1.8	2	0.4	13.20	20
	800	5	4	0.1	11.80	15
	100	4	4	0.4	11.60	19
	800	1.4	4	0.8	11.40	18
	800	2	4	0.4	10.00	24
2e	800	1.8	1.4	0.6	57.00	59
	800	1.4	1.4	0.6	56.80	60
	800	2	1.4	0.1	56.40	61
	200	1.8	1.4	0.2	56.00	67
	800	1	1.4	0.6	56.00	69

Table 3.2: PSO using Transition Representation with no seeding and no local growing

The longest snake found is of length 69 with fitness function 2e which is an increase of 6 edges when compared with no local growing. The best settings are determined using the average

snake length for five trial runs instead of ten trial runs as local growing creates overhead and thus takes more time to run. Fitness function 2a is the best with an average snake length of 58.80 with a population size of 800, C1 at 1.8, C2 at 1.8 and inertia at 0.4. This is a 3-edge increase in the average snake length and the PSO seems to favor a balanced exploitation and exploration with the same C1 and C2 values and inertia close to 50% dominating the top five settings. Even though some settings with higher C1 values than C2 did better, the difference between C1 and C2 values is not much. The same observation is made for fitness function 2e where the C1 and C2 values are almost the same with higher inertia. Fitness functions 2a, 2b and 2e found the longest snake a with C1 value less than C2 and inertia value around 50%. With exploration, the PSO searches more search space and this increases its chance of finding longer snakes.

The best improvement is seen in fitness function 2b where the average length of the snake went up by almost 10 edges and it performed better in finding the longest snake possible. It favored a balanced exploitation, exploration with identical values for C1 and C2 giving better results with inertia values around 50%. There is no major improvement in the average snake length or the longest snake found using fitness functions 2c and 2d. The PSO favored large population sizes for finding better snakes.

When the top 25 and the bottom 25 settings from fitness function 2a are taken, the average population size for the top 25 settings is 672 suggesting that the PSO did well with large populations while the average population size for the bottom 25 is 168 suggesting that PSO did poorly with small population sizes. The C1 and C2 average for the top 25 is 2.2 and 1.5 respectively, which are closer to each other when compared to the PSO with no local growing while the C1 and C2 average for the bottom 25 settings is 4.1 and 2.6. The average inertia for the top 25 is 0.42 suggesting the PSO did well with balanced exploitation and exploration. The

average inertia for the bottom 25 is 0.78 suggesting that the PSO did poorly with exploration (higher inertia values favor exploration). For the bottom 25 settings, with C1 and C2 values the PSO favored exploitation while with inertia values the PSO favored exploration. For fitness function 2e, average population size, average C1 value, average C2 value and the average inertia for the top 25 are 592, 1.8, 1.4 and 0.48 respectively. These values are similar to values seen with fitness function 2a. The bottom 25 for fitness function 2e also had a low average population size of 160 and a high average inertia value of 0.78. The average C1 and C2 values are 4.2 and 2.7, similar to values seen with fitness function 2a. Fitness function 2b average values are also similar.

Table 3.2 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 67. Fitness function 2b found a snake of length 63 with higher C2 value than C1 and higher inertia value. Fitness function 2c was able to find only a snake of length 35 while fitness function 2d was able to find only a snake of length 24. Fitness function 2e was able to find a snake of length 69, which is the best snake found with no seed and local growing every five iterations.

With seeding, no restriction on movement in the seeded region and no local growing, fitness function 2a and fitness function 2b performed better than the other fitness functions. Fitness function 2e is next. See Table 3.3. The longest snake found is of length 87 with fitness function 2c which is an increase of 18 edges when compared with no seeding and local growing. The best settings are determined using the average snake length for 10 trial runs. Fitness function 2a is the best with an average snake length of 78.10 with a population size of 100, C1 at 4, C2 at 1 and inertia at 0.1. This is an increase of 20 edges in the average snake length and the PSO

seems to favor exploitation over exploration with a higher value of C1 than C2 and inertia value at 0.1 or 0.4 when the top five settings are considered for fitness functions 2a and 2b.

Fitness Function	Size	C1	C2	Inertia	Average length	Longest Snake
2a	100	4	1	0.1	78.10	85
	400	5	1.4	0.4	78.00	84
	800	5	1	0.4	77.80	82
	400	4	1.4	0.4	77.70	82
	400	2	1.4	0.4	74.30	86
2b	200	5	1.4	0.4	78.10	84
	800	5	1.4	0.1	77.30	82
	800	5	1.4	0.4	77.10	85
	800	4	1.4	0.4	77.10	83
	800	5	2	0.4	76.70	86
2c	800	5	1.8	0.4	74.70	82
	400	5	1.8	0.4	74.10	79
	800	4	1.4	0.1	74.10	87
	400	5	1.4	0.4	73.70	78
	800	5	1.8	0.1	73.50	79
2d	800	5	1.4	0.1	73.30	80
	800	4	1.8	0.1	71.10	79
	800	2	2	0.4	70.30	77
	800	2	1.4	0.8	69.80	77
	800	1.8	1.4	0.1	65.80	85
2e	400	5	1	0.1	77.40	85
	800	5	1.8	0.1	77.30	82
	200	4	1.4	0.1	77.10	82
	400	4	1	0.4	77.10	81
	800	4	1	0.1	76.80	82

Table 3.3: PSO using Transition Representation with seeding, no restricted movement in seeded region and no local growing

The PSO settings for the best average snake length are similar to the settings for PSO with no seeding and no local growing. It seems like with no local growing the PSO is favoring exploitation over exploration while using local growing the PSO is favoring exploration as probably local growing is doing the job of exploitation. The same observation can be made for the remaining fitness functions. The performance of all the fitness functions increased considerably in finding the longest snake as well as the average snake length. Even though most

fitness functions favored settings with large populations, settings with small populations were also able to find longer snakes.

When the top 25 and the bottom 25 settings from all fitness functions are taken, the average population size for the top 25 varied from 548 to 680 suggesting that the PSO did well with large populations while the average population size for the bottom 25 varied from 100 to 124 suggesting that the PSO did poorly with small population sizes. The population size is the most important factor that affects the performance of the PSO. The C1 and C2 average for the top 25 varied from 3.2 to 4.2 and from 1.4 to 1.5 respectively, which suggests that PSO did well with exploitation. The average values for C1 and C2 for the bottom 25 varied from 2.6 to 3.1 and 2.0 to 2.2 suggesting that the PSO with a small difference between exploitation and exploration did poorly. The performance is influenced heavily by small population sizes. The average inertia for the top 25 is 0.26 to 0.35 suggesting the PSO favored exploitation. The average inertia for the bottom 25 is 0.95 to 0.97 suggesting that the PSO did poorly with exploration (higher inertia values favor exploration). Table 3.3 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 86 compared to 67 without seeding and local growing. Fitness function 2b found a snake of length 86 compared to 63 without seeding and local growing. Fitness function 2c was able to find a snake of length 87 compared to a snake of length 35 found without seeding and local growing. Fitness function 2c's average snake length is also up with other fitness functions. Fitness function 2d was able to find a snake of length 85 compared to a snake of length 24 found without any seeding. Fitness function 2e was able to find a snake of length 85. Fitness functions 2b, 2c and 2d made the most gains with seeding of initial population.

With seeding, no restriction on movement in the seeded region and local growing, fitness function 2b performed better than the other fitness functions. Fitness function 2e is next. See Table 3.4.

Fitness Function	Size	C1	C2	Inertia	Average length	Longest Snake
2a	400	5	1.4	0.2	78.20	86
	800	4	1.4	0.2	78.20	85
	800	5	2	0.2	78.00	83
	400	5	1	0.2	77.80	82
	200	4	1.8	0.2	77.50	80
2b	800	4	1.4	0.4	80.70	92
	800	5	1.8	0.1	78.60	84
	800	2	1.8	0.6	78.60	83
	800	5	1	0.2	78.40	85
	400	2	1.4	0.4	77.80	85
2c	800	1.4	1	0.6	77.10	85
	400	1.8	1	0.4	75.30	85
	800	2	1.4	0.4	75.00	89
	800	2	1.4	0.2	74.50	80
	800	1.8	1.4	0.6	74.00	84
2d	800	1.8	1	0.6	73.00	88
	400	4	1.4	0.1	70.20	82
	800	2	1.4	0.6	70.10	82
	800	2	1.4	0.4	70.00	78
	400	1.4	1.4	0.4	69.90	78
2e	400	4	1	0.2	78.60	89
	400	5	1	0.2	78.40	82
	200	2	1	0.6	78.30	84
	400	1	1.4	0.6	78.30	83
	200	5	1	0.1	78.20	83

Table 3.4: PSO using Transition Representation with seeding, no restricted movement in seeded region and local growing

The longest snake found is of length 92 with fitness function 2b which is an increase of 5 edges when compared with seeding and no local growing. The best settings are determined using the average snake length for 10 trial runs. Fitness function 2b is the best with 80.70 with a population size of 800, with C1 at 4, C2 at 1 and inertia at 0.4. This is an increase of two edges in the average snake length. The top five settings from each fitness function suggest that the PSO

seems to favor exploitation over exploration with a higher value of C1 than C2. This is a reversal from what is seen in the PSO with no seed and local growing which did well with almost equal values for C1 and C2. With the initial population seeded, the PSO did well with exploitation.

When the top 25 and the bottom 25 settings from all fitness functions are taken, the average population size for the top 25 varied from 532 to 708 suggesting that PSO did well with larger populations while the average population size for the bottom 25 varied from 132 to 196 suggesting that PSO did poorly with small population sizes. The C1 and C2 average for the top 25 varied from 2.5 to 3.4 and from 1.2 to 1.5 respectively, which suggests that PSO did well with exploitation. The average for C1 and C2 values for the bottom 25 varied from 3.3 to 3.9 and 2.6 to 3.1 suggesting with higher values for C1, C2, and with a small difference between C1 and C2 values, the PSO did poorly. The average inertia for the top 25 is 0.29 to 0.32 suggesting the PSO favored exploitation. The average inertia for the bottom 25 is 0.78 suggesting that PSO did poorly with exploration (higher inertia values favor exploration). With seeding the initial population and local growing, it is very difficult to find which setting works well and which does not by taking C1, C2 and inertia values. The only parameter that remained consistent is population size. With large population sizes, the PSO performs well.

Table 3.4 gives details on how each fitness function performed when average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 86. Fitness function 2b found a snake of length 92 compared to 86 with seeding and no local growing. The next best snake found for the same setting using fitness function 2b is 82. Fitness function 2c was able to find a snake of length 89 compared to 86 found with seeding and no local growing. Fitness function 2d was able to find a snake of length 88 while fitness function 2e was

able to find a snake of length 89. The performance gain in terms of longest snake using local growing is around 3 to 5 edges.

The difference in the average snake length decreased and almost all fitness functions performed well for different settings. The difference in average snake length for the top 40 settings in each fitness function is in the range of 2.1 to 5.9. All the settings are very close in terms of average snake length and it is not possible to determine which setting works and which does not. The best setting that found a snake of length 92 using fitness function 2b, found a second best snake of length 82 and the worst run found a snake of length 76. The difference between the best and second best is 10 edges while the difference the best and worst snake is 16 edges. The difference is huge and therefore making any conclusion based on the performance is not possible.

With seeding, restriction on movement of individuals in the seeded region and no local growing, fitness function 2b performed better than the other fitness functions. Fitness function 2a is next. See Table 3.5. The longest snake found is of length 87 with fitness function 2b which is a decrease of 5 edges when compared with seeding and local growing with no restrictions. The best settings are determined using the average snake length for five trial runs. Fitness function 2b is the best with an average snake length of 80.40 with a population size of 800, C1 at 5, C2 at 1.4 and inertia at 0.4. This is a slight decrease in the average length of snake when compared to the PSO with seeding, no restrictions and local growing and the PSO seems to favor exploitation over exploration with a higher value of C1 than C2. This is consistent with what is seen in the PSO with seeding and no local growing. The restriction of movement of the individual in the seeded region has no effect on either the average length of the snake or the longest snake found. It was just able to perform faster when compared to the PSO with seeding, no restriction on

movement and no local growing as the dimension size is reduced from 127 to 78. With restrictions on movement in the seeded region, the PSO was able to finish running all settings in 30 hours compared to 50 hours taken for the PSO with no restrictions on movement in the seeded region.

Fitness Function	Size	C1	C2	Inertia	Average length	Longest Snake
2a	400	5	1.8	0.1	79.60	84
	400	5	1	0.1	78.60	81
	400	5	1.8	0.4	78.60	85
	800	5	1	0.1	78.60	83
	400	4	1	0.4	78.40	85
2b	800	5	1.4	0.4	80.40	87
	400	4	1.4	0.4	79.80	82
	800	4	1.4	0.1	78.40	84
	800	5	1	0.4	78.00	84
	400	4	1	0.4	77.80	82
2c	800	4	1.4	0.4	77.40	82
	400	5	1.4	0.1	75.60	81
	400	5	2	0.4	75.20	81
	800	5	2	0.4	75.20	83
	400	4	1	0.4	72.80	84
2d	800	2	2	0.4	74.40	82
	800	4	1.4	0.4	73.40	79
	800	4	1.8	0.1	71.80	79
	200	2	1.8	0.4	71.20	78
	200	4	1.4	0.4	69.80	85
2e	800	5	1	0.1	79.00	80
	400	5	1	0.1	78.80	84
	400	4	1	0.1	78.20	79
	200	5	1	0.4	78.00	80
	400	5	1.4	0.4	77.80	79

Table 3.5: PSO using Transition Representation with seeding, restricted movement in seeded region and no local growing

When the top 25 settings from each fitness function are taken, the average population size varied from 488 to 616 while for the bottom 25 the average population size is 136 to 256. The average inertia for the top 25 settings is 0.18 to 0.39 suggesting that the PSO did well with exploitation while the bottom 25 settings have an average inertia of 1.0 suggesting that PSO did

not work well with exploration. Here, inertia is the biggest factor in deciding which settings work and which do not work for the PSO. The population is next showing a pattern that smaller population sizes does not work well. The average C1 value for the top 25 settings varied from 3.0 to 4.5 while the average C2 value for the top 25 settings varied from 1.4 to 1.7. The average C1 value for the bottom 25 settings varied from 2.1 to 3.0 while the C2 values varied from 2.4 to 3.3.

Table 3.5 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 86. Fitness function 2b found a snake of length 87 compared to 92 found by the PSO with seeding and local growing. Fitness function 2c was able to find a snake of length 84 compared to a snake of length 89 found by the PSO with seeding and local growing. Fitness function 2d was able to find a snake of length 85 while fitness function 2e was able to find a snake of length 84. The performance loss in terms of the longest snake using local growing is around 3 to 5 edges. Fitness function 2b was able to top all other fitness functions both in terms of the average snake length and in finding the longest snake possible.

With seeding, restriction on movement of individuals in the seeded region and local growing, fitness functions 2b and 2e performed better than the other fitness functions. Fitness function 2a is next. See Table 3.6. The longest snake found is of length 88 with fitness function 2e which is an increase of 1 edge when compared to the PSO with seeding, restricted movement in the seeded region and no local growing. The best settings are determined using the average snake length for five trial runs.

Fitness Function	Size	C1	C2	Inertia	Average length	Longest Snake
2a	800	5	1.4	0.1	78.80	83
	800	1.8	1	0.4	78.80	82
	800	4	1.4	0.4	78.40	80
	100	4	1.8	0.1	77.80	79
	200	1.8	1	0.8	77.00	85
2b	800	4	1	0.4	80.20	81
	800	4	1	0.1	78.40	81
	800	4	1.8	0.4	78.40	84
	800	1.4	1	0.8	78.40	87
	400	1.8	1.4	0.4	78.00	87
2c	800	2	2	0.4	78.00	81
	800	5	1.4	0.1	76.40	81
	800	4	1.4	0.4	75.20	82
	800	1.8	1	0.4	75.20	83
	400	1.8	1.8	0.4	72.60	86
2d	800	1.8	2	0.1	73.00	80
	800	2	1.4	0.1	71.40	83
	800	1.4	1	0.8	70.80	80
	400	2	1.4	0.4	70.60	80
	200	2	1.4	0.1	67.40	84
2e	200	4	1.4	0.1	80.20	83
	800	1.8	1.4	0.4	79.40	81
	800	2	1	0.4	78.80	80
	400	5	1	0.4	78.60	81
	400	1.4	1.8	0.8	76.60	88

Table 3.6: PSO using Transition Representation with seeding, restricted movement in seeded region and local growing

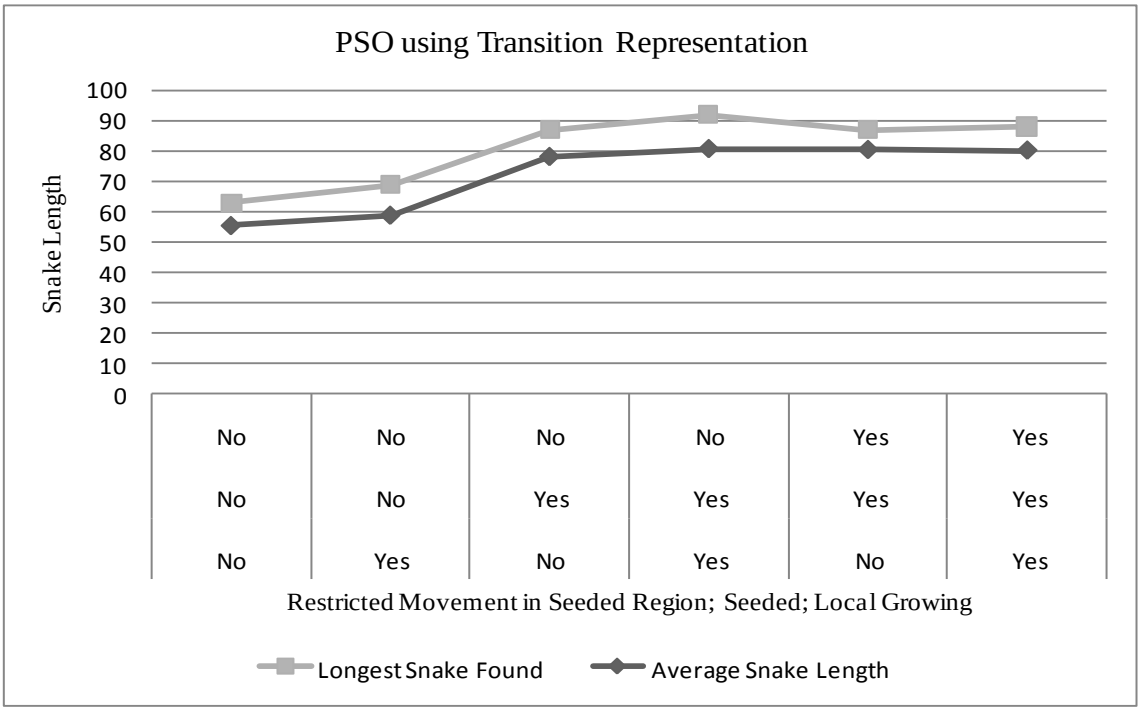
Fitness functions 2b and 2e are best with an average snake length of 80.20 with a population size of 800, C1 at 4, C2 at 1 and inertia at 0.4 for fitness function 2b and for fitness function 2e a population size of 200, C1 value 4, C2 value 1.4 and a very low inertia of 0.1. The PSO seems to favor exploitation over exploration with a higher value of C1 than C2. The restriction on movement of the individuals in the seeded region has no effect on either the average length of the snake or the longest snake found. It was just able to perform faster when compared to the PSO with seeding, no restriction on movement in seeded region and local growing as the dimension size is potentially reduced from 127 to 78. The use of local growing

also has no effect on either performance criteria. With restrictions on movement of particles in the seeded region, the PSO was able to finish running all the settings in around 44 hours. It took more time as local growing takes more computation time.

Table 3.6 gives details on how each fitness function performed when average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 85 compared to 86 found by the PSO with seeding, restricted movement and no local growing. Fitness function 2b found a snake of length 87 compared to 87 found by the PSO with seeding, restricted movement and no local growing, there is no improvement. Fitness function 2c was able to find a snake of length 86 compared to snake of length 84 found by the PSO with seeding, restricted movement and no local growing. Fitness function 2d was able to find a snake of length 84 while fitness function 2e was able to find a snake of length 88.

Graph 3.1 shows how the best average snake length and the best longest snake found varied with seeding, local growing and restricted movement in the seeded region. The PSO using a transition sequence representation performed poorly without seeding of the initial population along with no local growing and no restriction on movement of particles in the seeded region with the longest snake of length 63 and average snake length of 55.4. There is a slight improvement both in terms of the average snake length and in terms of the longest snake when local growing is used with no seeding and no restricted movement in the seeded region. The biggest gain is seen when the initial population is seeded. The average snake length went up by 23 edges to 78.1 while the longest snake found is 87, which is a 24-edge increase with no local growing. When local growing is used along with seeding the average snake length increased to 80.7 while the longest snake found is 92 which a 5 edge increase and is the best result both in terms of average snake length and longest snake found using PSO with a transition sequence

representation. For reasons not known, the introduction of restricted movement in the seeded region had a negative effect on the longest snake found and had no improvement in the average snake length.



Graph 3.1: Particle Swarm Optimization using Transition sequence Representation.

3.3 USING BIT REPRESENTATION

3.3.1 INTRODUCTION

To find the longest snake in an 8-dimensional hypercube, a bit representation is used to represent the snakes. The best snake found using bit representation reported here is 95.

3.3.2 REPRESENTATION

The individuals in PSO contain 256 bits (one for each node in the 8-dimensional hypercube 2^8=256). If a node is used then its corresponding bit would be set to 1 indicating that the node is turned on. If the node is not used then its corresponding bit would be set to zero indicating that the node is turned off. The first node (Node 0) is always turned on, i.e., set to one. To generate the node sequence using the bit representation, the node sequence always starts at node 0 and then looks to see if any of its adjacent nodes (any of the 8 possible nodes adjacent to node 0) are turned on. If more than one node is turned on then the node with the lowest node number is always chosen. For node 0, if node 1, 2 and 4 are turned on then the sequence will always choose node 1 as it has the lowest node number. Once the node sequence is generated, the length of the snake and its fitness function are calculated.

The length of the individuals is 256 bits, one for each node. If the initial population is seeded with the snake from the 7-dimensional hypercube then for each individual, only the bits corresponding to the nodes in the seeded sequence are turned on. The remaining bits (>127) are randomly turned on or off. To restrict movement, the bits from 0 to 127 (the seeded region), are kept unchanged by not performing any velocity updates or updates to bit values. Velocity is calculated only for nodes >127 and the bit values are updated using the sigmoid function.

3.3.3 FITNESS FUNCTION

To calculate the fitness function the individual in the bit representation is converted to a node sequence representation. This node sequence is then used to calculate the fitness of the individual. All the five fitness functions are tested individually to see how well each fitness function did in comparison with each other. The five fitness functions used are fitness function 2a, fitness function 2b, fitness function 2c, fitness function 2d and fitness function 2e.

3.3.4 INITIAL POPULATION

The individuals are seeded with the longest snake from the 7-dimensional hypercube given in [Potter94]. It is converted from node sequence representation to bit representation. First, all 256 bits in the individual are turned off (set to 0) and then the bits with the corresponding nodes in the node sequence are turned on. The bits corresponding to nodes greater than 127 are randomly turned on.

3.3.5 ENGINEERED CONDITIONING

In an effort to increase efficiency and reliability in finding longer snakes and better average snake length, new strategies are added to the PSO. One strategy is to restrict the movement of individuals. Change in bits less than 128 (0 to 127, called the seeded region) is not performed throughout the PSO approach by not performing any velocity updates in the seeded region. The PSO behaves as if it is working on a dimension 128 (256-128) search space and updates the individuals in those dimensions only.

A second strategy is that after every 5 generations, each individual is taken and a simple local search technique is used to see if each individual snake length can be increased by

one node at the end of the snake. To do this the bit representation is first converted to node sequence representation and then the ending node's adjacent nodes are examined to see if there are any nodes that can be added and still be part of the snake. If so then that node is added. The node sequence is then converted back to bit representation by turning the added node's corresponding bit to 1. If more than one neighbor is eligible then the node with the lowest node number is picked.

3.3.6 STOPPING CRITERIA

The stopping criterion is 200 iterations with no change in global best or a maximum of 1000 iterations, whichever occurs first.

3.3.7 BINARY PSO

The bit representation has two values for each dimension. So for each of the 256 dimensions, the sigmoid value is calculated using the sigmoid function $1/ (1+ e^{-1\times \text{particle velocity}})$. After that, a random number is generated and if it is less than the sigmoid value then the bit for that dimension is set to 1 and if not then the bit is set to 0. This is done for all the 256 dimensions in all the individuals.

3.3.8 EXPERIMENTAL SETUP

All the five fitness functions are tested using different combinations of population size, C1, C2 and inertia values.

Population size : 100, 200, 400, 600 or 800

C1 : 1, 1.4, 1.8, 2, 4, 5

C2 : 1, 1.4, 1.8, 2, 4

Inertia : 0.4, 0.6, 0.8, 1.0, 1.1, 1.2

The different settings discussed in the thesis proposal are reduced to the above settings as these settings alone for each fitness function for a combination of 720 (4×6×5×6) for 5 to 10 trial runs took around 50 to 150 hours (2 to 6 days) to run. For one fitness function with all combinations, including lower inertia values and five different population sizes with 20 trial runs for each setting took around 500 hours (about 20 days) to complete all the runs for all the settings. When the results are examined, it is observed that using lower inertia values did not produce as good snakes as it did using a transition sequence representation so lower inertia values were not used in other fitness functions to save time. Due to time constraints, either a population size of 600 or 800 is used but not both. Due to the large number of combinations, only the top five settings determined by the average snake length for each fitness function are discussed. If the best snake found is not within the top five settings then the setting with the least average snake length in the top five is replaced by the setting that found the longest snake in the tables discussed in this section.

To see how seeding and engineered conditioning affected the length of the snakes, all the settings are tested with different combinations. First with no seeding and no local growing, next with no seeding and local growing, next with seeding, no local growing, no restriction in seeded region, next with seeding, local growing, no restriction in seeded region, next with seeding, no local growing, restricted movement in seeded region lastly with seeding, local growing, restricted movement in seeded region.

3.3.9 RESULTS

Without seeding and no local growing, fitness function 2e performed better than the other fitness functions. Fitness function 2a is next. See Table 3.7.

Fitness Function	Size	C1	C2	Inertia	Average Length	Longest Snake
2a	600	2	5	1	65.40	69
	400	2	5	1	62.00	65
	400	2	1.8	1	61.40	71
	400	1	2	1	61.40	66
	600	1.4	1.4	1	61.40	66
2b	600	1.4	1.8	1	60.00	65
	400	4	5	1	57.20	60
	600	4	5	1	57.20	64
	600	5	4	1	55.60	61
	600	1.8	4	1	55.20	66
2c	400	1.4	1.8	1.1	31.00	38
	600	1	1	1.1	30.40	33
	600	1.4	1.8	1	29.40	37
	600	1.4	5	1	29.20	35
	600	1.8	2	1	29.00	43
2d	400	1	2	1	12.20	27
	200	1.4	1	1.1	10.80	26
	400	1.4	1.4	1.1	10.60	30
	200	1.4	2	0.4	9.20	25
	600	5	4	0.1	9.00	32
2e	600	1.8	2	1.1	66.60	71
	400	4	4	1	65.20	69
	600	1.4	5	1	65.00	72
	600	1	1	1	65.00	71
	600	1.8	1.4	1	64.80	70

Table 3.7: PSO using Bit Representation with no seeding and no local growing

The longest snake found is of length 72 with fitness function 2e compared to the longest snake 63 found using a transition sequence representation without seeding and no local growing is a nine edge increase in the longest snake found. The best settings are determined using the average snake length for five trial runs. Fitness function 2e is the best with an average snake length of 66.60 with a population size of 600, C1 at 1.8, C2 at 2 and inertia at 1.1. The high

inertia value and higher value for C2 compared to C1 shows that the PSO with no seeding and no local growing was able to find better snakes by exploring the search space more than exploiting the search space. The PSO using the transition sequence representation on the other hand favored exploitation over exploration. The drawback of a transition sequence representation is minimized using the bit representation in that a simple change in one bit will not change the snake drastically as it did using the transition sequence representation.

Table 3.7 gives the details on how each fitness function performed in finding the longest snake and the average snake length. Fitness function 2e is first with an average snake length of 66.60. Fitness function 2a is next with 65.40. Fitness function 2a mostly favored exploration over exploitation with C2 value greater than C1 or having both values the same indicating a balance of exploitation and exploration and high inertia values suggesting that the PSO favored exploration. Fitness function 2e also favored exploration over exploitation with higher values for C2 than C1 in three out of the top five settings and higher inertia values for all the top five settings.

Fitness function 2b is next with C1 and C2 values nearer to each other and higher inertia values. It found a snake of length 66. Fitness functions 2c and 2d did not perform well as compared. These fitness functions are expected to perform on par with the remaining fitness functions when the individual is seeded and restricted movement on the individual is enforced as the seed is tightly wrapped which means that more nodes in the snake share the same skin nodes.

All fitness functions favored larger populations. The larger the population the better the PSO performed. The complexity of the Snake Problem search space can be attributed to this trend. When the top 25 and the bottom 25 settings from fitness function 2e are taken, the average population size for the top 25 is 492 suggesting that PSO did well with large populations

(400 and 600) while the average population sizes for the bottom 25 is 120 suggesting that PSO did poorly with low population size (100 and 200). The C1 and C2 average for the top 25 is 2.2 and 2.7 respectively suggesting that PSO did well with exploration while the C1 and C2 average for the bottom 25 is 2.1 and 1.8 suggesting that PSO did poorly with exploitation. The average inertia for the top 25 is 1.05 suggesting the PSO did well with exploration (as higher inertia values favor exploration). The average inertia for the bottom 25 is 0.20 suggesting that PSO did poorly with exploitation (lower inertia values favor exploitation). For fitness function 2a, average population size, average C1 value, average C2 value and the average inertia for the top 25 are 504, 1.9, 2.4 and 1.04 respectively. The values are similar to the values seen with fitness function 2e. The bottom 25 for fitness function 2a also had a low average population size of 136 and a very low average inertia value of 0.17. For this PSO, the population size and inertia are the most important parameters in determining if a setting works or not. Fitness function 2b also had similar values.

Fitness function 2a found a longest snake of length 71. Fitness function 2b found a snake of length 66. Fitness function 2c found a longest snake of length 38. Fitness function 2d found a snake of length 32. Fitness function 2e found a snake of length 72, which is the longest snake found without seeding and no local growing for the PSO using the bit representation.

Without seeding and local growing, fitness function 2e performed better than the other fitness functions. Fitness function 2a is next. See Table 3.8. The longest snake found is of length 75, this is an increase of six edges when compared to the longest snake of length 69 found using the PSO with the transition sequence representation with no seeding and local growing. This is also an increase of three edges when compared to the PSO with no seeding and no local growing using the bit representation.

Fitness Function	Size	C1	C2	Inertia	Average Length	Longest Snake
2a	600	2	2	1.2	64.60	70
	100	4	4	1.2	64.20	71
	400	1.4	2	1	63.60	67
	600	1.8	1.4	1	63.60	66
	200	4	4	1	60.40	72
2b	600	4	5	1	63.40	69
	600	2	2	1	59.00	63
	600	5	4	1.2	57.60	66
	400	1.4	1.4	1.2	57.40	61
	600	5	5	1	51.60	73
2c	600	1.4	2	1	35.80	42
	400	1	2	1	35.20	39
	600	1.8	2	1.2	32.80	41
	600	5	5	1.2	32.60	40
	200	1.8	2	1.2	27.80	54
2d	600	4	4	0.8	13.40	34
	600	4	5	0.1	12.40	30
	400	2	4	0.1	11.20	24
	600	5	2	1	10.80	25
	200	1.4	2	0.8	10.60	31
2e	600	1	5	1	67.80	72
	600	4	4	1	67.20	71
	600	1	1	1	66.80	69
	600	1.8	2	1	66.80	68
	400	2	1.4	1	64.40	75

Table 3.8: PSO using Bit Representation with no seeding and no local growing

The best settings are determined using the average snake length for five trial runs. Fitness function 2e is the best with an average snake length of 67.80 with a population size of 600, C1 at 1, C2 at 5 and inertia at 1. Fitness functions 2a, 2b, 2c and 2e mostly favored balanced exploitation and exploration with identical values for C1, C2 or with C2 values higher than C1 and higher inertia values favoring exploration. Fitness function 2d did not perform well. The PSO favored larger populations for finding better snakes. The performance improvement gained using local growing is low when compared to PSO without local growing.

When the top 25 and the bottom 25 settings from fitness function 2e are taken, the average population size for the top 25 is 504 suggesting that PSO did well with large populations (400 and 600) while the average population size for the bottom 25 is 124 suggesting that PSO did poorly with low population sizes (100, 200). The C1 and C2 average for the top 25 is 2.1 and 2.8 (either C1 and C2 values are equal or C2 value is large and C1 value is small) respectively suggesting that PSO did well with exploration while the C1 and C2 average for the bottom 25 is 2.1 and 1.9 suggesting that PSO did poorly with exploitation. The average inertia for the top 25 is 1.05 suggesting the PSO did well with exploration (as higher inertia values favor exploration). The average inertia for the bottom 25 is 0.20 suggesting that PSO did poorly with exploitation (lower inertia values favor exploitation). For fitness function 2a, average population size, average C1 value, average C2 value and the average inertia for the top 25 are 486, 2.8, 3.2 and 1.1 respectively. The values are similar to values seen with fitness function 2e and the same analysis can be made for fitness function 2a. The bottom 25 for fitness function 2a also had a low average population size of 112 and a very low average inertia value of 0.21. For this PSO, the population size and inertia are the most important parameters in determining if a setting works or not. Fitness function 2b also had similar values. There is no clear pattern seen in the other two fitness functions.

Table 3.8 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 72. Fitness function 2b was able to find a snake of length 73 and was consistent with good average snake length. Fitness function 2c was able to find a snake of length 54 but was not able to find good snakes consistently. The low average snake length for fitness function 2c is an indication that it is not consistent in finding good snakes. Fitness function 2d did worse than

fitness function 2c in terms of finding the longest snake at just 34 and the best average snake length of about 13, which is very low. Fitness function 2e is the best performing fitness function both in terms of the average snake length with 67.80 as well as the longest snake found at 75.

With seeding, no restriction on movement in the seeded region and no local growing, fitness function 2e performed better than the other fitness functions as far as average snake length is concerned while fitness function 2b found the longest snake of length 90. There is a lot of luck involved in finding this snake, as the next best snake for the same settings is 72, which is not even close. See Table 3.9.

Fitness Function	Size	C1	C2	Inertia	Average Length	Longest Snake
2a	800	5	1.4	1	77.80	83
	400	2	1.4	1	77.00	79
	800	5	1.8	1.1	76.40	80
	800	5	1.8	1	76.40	80
	400	2	1	1	76.20	82
2b	800	5	2	1	77.20	85
	400	1.4	1	1	77.00	82
	800	1.8	1.4	1.1	77.00	81
	800	5	1.8	1	76.80	85
	800	5	5	1.1	74.00	90
2c	800	4	1.8	1.1	75.80	81
	800	5	4	1	73.40	83
	800	1.4	1.4	1	73.40	81
	800	2	2	1	73.00	80
	800	1.8	5	1	71.20	85
2d	800	2	1.4	1.1	69.80	73
	400	2	1.4	1	69.60	71
	800	1.4	1	1	69.60	72
	800	1.4	2	1	69.60	72
	800	1	1.8	1	66.80	76
2e	800	5	5	1	79.60	86
	800	4	5	1	79.00	83
	800	1.8	1.4	1	78.80	83
	800	5	5	1.1	78.40	83
	800	5	2	1.1	77.60	80

Table 3.9: PSO using Bit Representation with seeding, no restricted movement in seeded region and no local growing

Fitness function 2e was able to find a snake of length 86 which is the second best snake. The best settings are determined using the average snake length for five trial runs. Fitness function 2e is the best with an average snake length of 79.60 with a population size of 800, C1 at 5, C2 at 5 and inertia at 1. This is an increase of 13 edges in the average length of snake compared to the PSO using the bit representation with no seeding and local growing. The PSO using fitness function 2e seems to favor a balanced exploitation and exploration with near equal values for C1 and C2 in most settings and a higher value for inertia suggesting the PSO favored exploration. The settings with higher C1 values than C2 also made it to the top five settings for fitness function 2e.

Fitness function 2a favored exploitation with higher values for C1 than C2 while without seeding it favored a balanced exploitation and exploration with equal values of C1 and C2. Fitness function 2b also followed the same trend as fitness function 2a. No particular pattern can be seen in fitness function 2c and fitness function 2d from the top five settings as far as C1 and C2 are concerned. All fitness functions did well with large populations and high inertia.

When the top 25 and the bottom 25 settings from all fitness functions are taken, the average population size for the top 25 varied from 640 to 704 suggesting that PSO did well with large populations (mostly 800). The average population size for the bottom 25 varied from 108 to 160 suggesting that PSO did poorly with low population sizes (100). The C1 and C2 average for the top 25 varied from 2.8 to 3.1 and 2.1 to 2.7 (either C1 and C2 values are equal or C1 value is slightly bigger than C2 value) respectively suggesting that PSO did well with exploitation. The C1 and C2 average for the bottom 25 varied from 1.8 to 2.4 and 1.8 to 2.3 respectively. A clear pattern is not seen as all different combinations of C1 and C2 made the bottom 25. The population size heavily influenced how the PSO performed. The average inertia for the top 25

varied from 1.02 to 1.04 suggesting the PSO favored exploration as far as inertia is concerned (as higher inertia values favor exploration). The average inertia for the bottom 25 varied from 0.40 to 0.51 (inertia values from 0.1 to 0.8 are seen in the bottom 25) suggesting that PSO did poorly with exploitation (lower inertia values favor exploitation) but population size influenced the performance.

Fitness function 2b found a snake of length 90, which is the longest snake found, using any fitness function. The next best snake for the same setting is 72. There is a difference of 18 edges between the best and second best snake. If the same setting found a snake of length 72 instead of 90 then the setting would be out of the top 20 and would also be out of the top 50. For this reason, making any decisions on which setting works and which does not is very difficult. The average snake length for the top 40 settings in all fitness functions is very close. For all fitness functions, the difference between the top 40 settings varied from 4.4 to 5.8 edges.

Table 3.9 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 83 compared to 72 without seeding and local growing. Fitness function 2b found a snake of length 90 compared to 73 without seeding and local growing. Fitness function 2c was able to find a snake of length 85 compared to a snake of length 54 found without seeding and local growing. Fitness function 2c's average snake length is also up with the other fitness functions, while fitness function 2d was able to find a snake of length 76 compared to a snake of length 34 found earlier without any seeding. Fitness function 2e was able to find a snake of length 86. Fitness functions 2c and 2d made the most gains with a seeded population.

With seeding, no restriction on movement in the seeded region and local growing, fitness functions 2a and 2e performed better than the other fitness functions when average snake length is taken. See table 3.10.

Fitness Function	Size	C1	C2	Inertia	Average Length	Longest Snake
2a	800	2	1	1	80.80	89
	800	2	1.4	1	80.80	89
	800	4	1.8	1	79.60	84
	400	4	1.4	1	79.40	86
	800	1.4	1	1	79.00	80
2b	600	5	1.8	1	80.60	84
	400	1.4	1	1	79.40	83
	400	1.4	1.4	1	78.00	89
	400	1.4	1.8	1.2	77.80	84
	600	5	2	1.2	77.60	80
2c	400	1	1	1	74.80	81
	400	1	1.4	1	74.80	84
	400	1.8	1.4	1	74.20	82
	400	5	4	1.2	74.00	83
	400	1.8	1.8	1	73.00	86
2d	400	4	1.8	1	70.40	72
	400	4	5	1	70.40	75
	600	5	4	1.2	70.40	80
	400	1.4	1.4	1.2	70.20	74
	400	1.8	1.4	1	70.20	73
2e	400	5	4	1	80.80	83
	400	2	4	1	79.20	84
	400	5	2	1	78.60	83
	400	4	1.4	1	78.60	80
	400	1.8	1.8	1	77.20	86

Table 3.10: PSO using Bit Representation with seeding, no restricted movement in seeded region and local growing

Fitness functions 2a and 2b found the longest snake of length 89, which is an edge less than what the PSO was able to find without local growing but using local growing the fitness functions performed better in terms of average snake length. The best settings are determined using the average snake length for five trial runs. Fitness function 2a and 2e are the best with an average snake length of 80.80. This is an increase in the average snake length and the PSO seems

to favor exploitation over exploration in most settings, which is similar to what is observed for PSO with seeding and no local growing. This trend is seen in all fitness functions.

All the fitness functions favored high inertia values. When the top 25 and the bottom 25 settings from all fitness functions are taken, the PSO generally performed well with large populations while the PSO with small populations performed poorly. The C1 and C2 average for the top 25 varied from 2.3 to 3.3 and 1.9 to 2.7 (either C1 and C2 values are equal or C1 value is bigger than C2 value) respectively suggesting that PSO did well with exploitation. The C1 and C2 average for the bottom 25 varied from 1.9 to 2.2 and 1.8 to 2.3 respectively. A clear pattern is not seen as all different combinations of C1 and C2 made the bottom 25. The population size heavily influenced how the PSO performed. The average inertia for the top 25 varied from 1 to 1.06. The average inertia for the bottom 25 varied from 0.39 to 0.51. For all fitness functions, the difference in average snake length for the top 40 settings varied from 3 to 7.2 edges.

Table 3.10 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 89 compared to 83 found using PSO with seeding and no local growing. Fitness function 2b found a snake of length 89 compared to 90 found using PSO with seeding and no local growing. Fitness function 2c was able to find a snake of length 86 compared to a snake of length 85 found using PSO with seeding and no local growing. Fitness function 2d was able to find a snake of length 80 while fitness function 2e was able to find a snake of length 84.

With seeding, restriction on movement of individual in the seeded region and no local growing, fitness functions 2c and 2e performed better than the other fitness functions. See Table 3.11.

Fitness Function	Size	C1	C2	Inertia	Average Length	Longest Snake
2a	800	1.4	1.4	1	84.00	87
	800	4	2	1	83.80	94
	800	1.4	1	1	83.60	87
	800	4	1	1	82.80	86
	800	2	1	1	82.60	89
2b	800	4	1	1	84.00	90
	800	4	2	1	82.40	91
	800	1.8	2	1	82.20	84
	400	4	1	1	82.00	89
	400	4	1.8	1	81.80	88
2c	800	4	1.8	1	84.40	89
	800	4	1	1	82.20	89
	800	4	1.4	1	81.80	87
	200	4	1.4	1	81.40	89
	400	2	2	1	77.20	95
2d	800	4	2	1	76.60	88
	800	4	1.4	1	74.60	79
	800	4	1	1	74.00	78
	800	1.8	1	1	73.60	75
	400	4	1.8	1	73.40	79
2e	400	4	1.4	1	84.40	89
	800	2	1	1	83.20	87
	400	4	1	1	82.80	86
	800	1.8	1.4	1	82.80	87
	400	2	1	1	82.00	86

Table 3.11: PSO using Bit Representation with seeding, restricted movement in seeded region and no local growing

The longest snake found is of length 95 with fitness function 2c which is an increase of 6 edges when compared to the PSO using the bit representation with seeding and local growing but the next best snake found using the same setting is 76. The best settings are determined using the average snake length for five trial runs. Fitness functions 2c and 2e are the best with an average snake length of 84.40, which is a 4-edge increase when compared to the PSO using the bit

representation with seeding, local growing and no restriction in seeded region. All the fitness functions performed better with higher C1 values than C2 suggesting the PSO performed better when it favored exploitation over exploration and with higher inertia values.

When the top 25 and the bottom 25 settings from each fitness function are taken, the most important factors in deciding if a setting is at the top of the list or at the bottom of the list are inertia and population size. Settings with large populations performed well while the settings with small populations coupled with small inertia (0.1) did poorly irrespective of C1 and C2 values. High inertia values dominated the top of the list. The average snake length for the top 40 settings in all fitness functions varied between 4.6 and 9.8

Table 3.11 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 94 compared to 89 found with seeding and local growing, which is a 6-edge increase in longest snake seen. Fitness function 2b found a snake of length 91 compared to 89 with seeding and local growing. Fitness function 2c was able to find a snake of length 95 compared to a snake of length 86 found with seeding and local growing. This is the longest snake found using particle swarm optimization so far. Fitness function 2d was able to find a snake of length 88 while fitness function 2e was able to find a snake of length 89.

With seeding, restriction on movement of individual in the seeded region and local growing, fitness function 2e performed better than the other fitness functions. Fitness function 2b is next. See Table 3.12.

Fitness Function	Size	C1	C2	Inertia	Average Length	Longest Snake
2a	100	1.8	1	1.1	84.40	87
	800	4	1	1	84.10	92
	800	2	1	1	83.50	93
	800	4	1.4	1	83.30	90
	800	4	2	1	83.10	92
2b	200	4	2	1	85.60	92
	800	1.4	1	1	85.20	91
	400	1.4	1	1	85.00	88
	400	4	1	1	85.00	88
	800	1.4	1	1	83.60	94
2c	400	4	1	1	83.80	91
	400	1.4	1	1	83.40	86
	200	1.8	1	1	82.60	91
	800	1	1	1	82.00	88
	200	1	2	1	76.40	93
2d	800	1	1	1	77.80	85
	800	1.4	1	1	76.20	86
	800	1.4	1	1	76.20	80
	200	4	1.8	1	76.00	80
	400	1.4	1	1	74.40	87
2e	800	1.4	1	1	86.00	90
	400	4	1	1	84.80	87
	400	1.8	1.4	1	84.20	87
	100	4	1.4	1	83.40	87
	800	1	2	1	83.00	92

Table 3.12: PSO using Bit Representation with seeding, restricted movement in seeded region and local growing

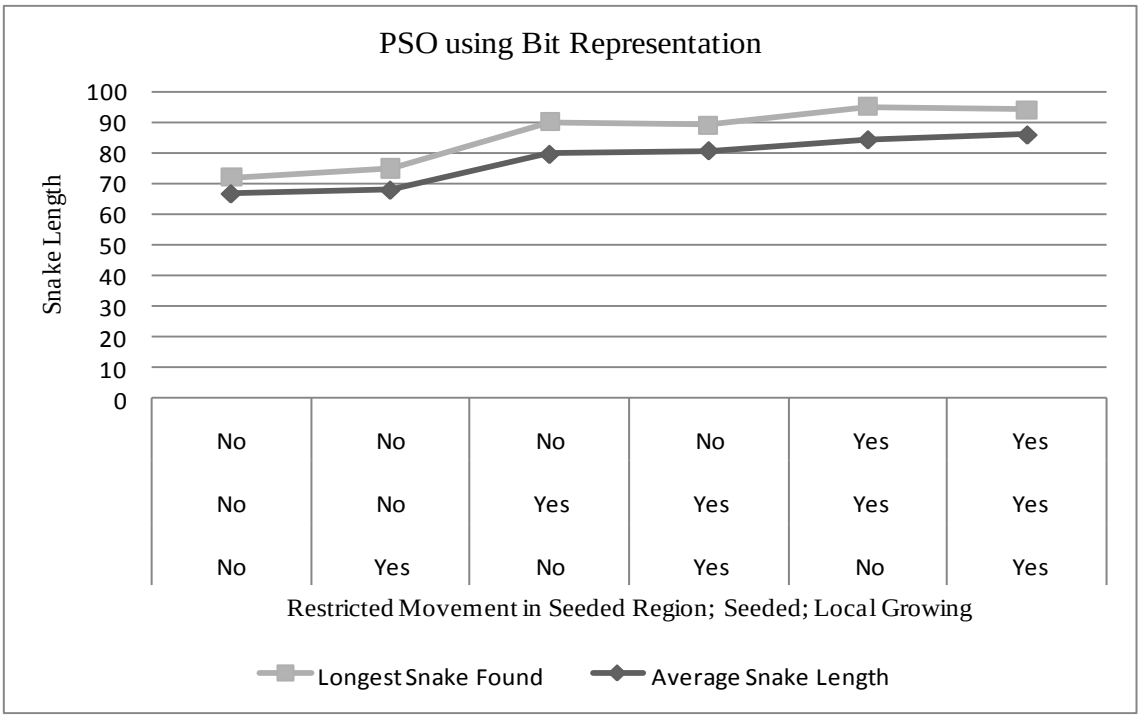
The longest snake found is of length 94 with fitness function 2b which is a decrease of 1 edge when compared with seeding, restricted movement in the seeded region and no local growing. The best settings are determined using the average snake length for five trial runs. Fitness function 2e is the best with an average snake length of 86.0 with a population size of 800, with C1 at 1.4, C2 at 1 and inertia at 1.

The PSO favored exploitation over exploration with higher values of C1 than C2. The use of local growing every 5 generations has no effect on the longest snake found but the average snake length increased by about two edges. The population size and inertia values determined if a setting is at the top of the list (ordered by average snake length) or at the bottom of the list. Settings with large populations and higher inertia values did well and dominated the top 25 for each fitness function. Settings with small populations and lower inertia values dominated the bottom 25 of the list. There is no clear pattern seen in C1 and C2 values. The average snake length for the top 40 settings for all fitness functions varied between 4.5 and 7.6 edges.

Table 3.12 gives details on how each fitness function performed when the average snake length and finding the longest snake are considered. Fitness function 2a found a snake of length 93 compared to 94 with seeding, restricted movement and no local growing. Fitness function 2b found a snake of length 94 compared to 91 with seeding, restricted movement and no local growing. Fitness function 2c was able to find a snake of length 93 compared to a snake of length 95 found with seeding, restricted movement and no local growing. Fitness function 2d found a snake of length 87 while fitness function 2e found a snake of length 92.

Graph 3.2 shows how the best average snake length and longest snake found varied with seeding, local growing and restricted movement in the seeded region. The PSO using a bit representation performed poorly without seeding of initial population along with no local growing and no restriction on movement with a longest snake of length 72 and an average snake length of 66.6. This is an increase of about 10 edges in both longest snake found and average length of snake when compared to the PSO using a transition sequence representation with no seeding and no local growing. There is a slight improvement both in terms of average snake

length and in terms of longest snake when local growing is used with no seeding and no restricted movement.



Graph 3.2: Particle Swarm Optimization using Bit Representation.

When the initial population is seeded, the average snake length went up by 13 edges to 79.6 while the best snake found is 90. When local growing is used along with seeding, the average snake length increased to 80.8 while the longest snake found is 89. The introduction of restricted movement in the seeded region had a positive effect on the longest snake found and there is a slight improvement in terms of average snake length. Without local growing, the best snake seen is 95. When local growing is added, the best snake seen is 94. The average snake length also went up to 84.4 and 86 with no local growing and local growing respectively.

The PSO using a bit representation performed better than the PSO using a transition sequence representation both in terms of average snake length with 86 compared to 80.7 and the longest snake found with 95 compared to 92.

3.4 CONCLUSION

The PSO approach was able to find a snake of length 95 using a bit representation with seeded individuals and restricted movement on individuals with no local growing.

CHAPTER 4

SIMULATED ANNEALING

4.1 INTRODUCTION TO SIMULATED ANNEALING

This technique stems from thermal annealing which aims to obtain perfect crystallizations by a slow enough temperature reduction to give atoms the time to attain the lowest energy state. This search tries to avoid local minima by jumping out of them early in the computation [Black09]. Each step of the Simulated Annealing (SA) algorithm replaces the current solution by a random nearby solution, chosen with a probability that depends on the difference between the corresponding fitness function values and a global parameter T (temperature) that is gradually decreased during the process. The dependency is such that the current solution changes almost randomly when T is large, but increasingly downhill as T goes to zero. The allowance for uphill moves saves the method from becoming stuck at local minima [wikiSA09]. Towards the end of the computation when the temperature or probability of accepting a worse solution is nearly zero, this simply seeks the local minimum. The chance of getting a good solution can be traded off with computation time by slowing down the cooling schedule (i.e., having higher values for cooling rate such that the temperature is decreased in smaller increments). With 0.98, the temperature decreases the cooling rate by 2% while 0.95 decreases it by 5%.

4.1.1 Algorithm

There are two loops in the SA algorithm. The outer loop is the temperature loop and the inner loop is the Metropolis loop. The initial temperature is set and the inner loop is executed a certain number of times determined by the user input (nloop). During each loop in the inner loop, a neighbor is generated. If the fitness of the neighbor is greater than the present individual fitness, then the search simply moves to the neighbor position by replacing the present individual with the neighbor individual. The Snake Problem is about finding the maximum (longest snake) while the SA is a minimizing technique, so the search space is flipped to "hill climb" by changing the accepting probability from $P= e^{(\text{present fitness - neighbor fitness})/\text{temperature}}$ to $P= e^{((-1)*(\text{present fitness - neighbor fitness})/\text{temperature})}$. If the neighbor has less fitness then by using the accepting probability ($P= e^{((-1)*(\text{present fitness - neighbor fitness})/\text{temperature})}$) it is determined if the individual should move to the neighbor's position or not. A random number is generated and if the generated number is less than P then the neighbor replaces the present individual. If the random number is greater than or equal to P then no changes are made to the present individual. Once the algorithm comes out of the inner loop the temperature is decreased using the formula $T_{new}=T_{old} \times$ cooling rate. The algorithm enters the inner loop after making changes to temperature. The outer loop is repeated until there is no change in individual fitness for a predetermined number of outer loop iterations.

4.2 USING TRANSITION SEQUENCE REPRESENTATION

4.2.1 INTRODUCTION

To find the longest snake in an 8-dimensional hypercube, a transition sequence representation is used to represent the snakes. The best snake found using the transition sequence representation reported here is 91.

4.2.2 REPRESENTATION

The snake contains a sequence of numbers in the range 0 to 7 (0 to n-1 dimensions) and each number in the sequence corresponds to the binary bit being flipped from 1 to 0 or 0 to 1 in the previous node's binary form (00000000) in order to generate the current node's binary form. The hypercube is symmetric and so one can start at any node say at zero and use it to get the remaining nodes by using the transition sequence.

The upper bound on the longest snake possible for $d \geq 7$ satisfies the following expression: upper bound $\leq 2^{d-1} - (2^{d-1}/ (20d- 41))$ [Snevily94]. The length of the individual is determined using this equation. In an 8-dimensional hypercube, the upper bound using the above equation is 126.92. Therefore, the individual's length is set to 127. If the initial individual is seeded randomly, then to generate the transition sequence, a number between zero and seven is picked at random and is set as the starting transition from node 0. To generate the next transition a number other than the present transition is picked at random. This process is repeated until the individual's transition sequence reaches a length of 127.

4.2.3 FITNESS FUNCTION

To calculate the fitness, the transition sequence representation is converted to the node sequence representation. This node sequence is then used to calculate the fitness of the individual. All the five fitness functions are tested individually to see how well each fitness function did in comparison with each other. The five fitness functions used are fitness function 2a, fitness function 2b, fitness function 2c, fitness function 2d and fitness function 2e.

4.2.4 GENERATING A NEIGHBOR

To get a neighbor for the individual, different mutation schemes are used. One scheme is picked at random to generate the neighbor of the present individual in the Metropolis loop (nloop). In one scheme, the naïve mutation is used where a transition is selected (between 0 and 127) and then a transition value is given at random (between 0 and 7) such that it is not the same as its adjacent transitions or its previous value.

The second scheme is diagonal mutation where a transition is selected at random. The selected transition is exchanged with one of the nodes on either side. This essentially does the diagonal mutation using the transition sequence representation.

The third scheme, a number between 2 and 8 is generated and then a point in the individual is selected at random. Now from the selected point, 2 to 8 consecutive nodes (depending on the random number generated) are selected and reordered by selecting one node randomly until all nodes are selected to generate the new neighbor.

4.2.5 INITIAL INDIVIDUAL

The individual is seeded with the longest snake of length 50 from the 7-Dimensional hypercube given in [Potter94], which is converted from the node sequence representation to the transition sequence representation, and the length is increased by randomly generating numbers between 0 to 7 to get a sequence of transitions of length 127.

4.2.6 ENGINEERED CONDITIONING

In an effort to increase efficiency and reliability in finding longer snakes and better average snake length, new strategies are added to the SA. The strategy used for the SA is to protect the seeded transition sequence (0 to 49 in the individual, called the seeded region) throughout the SA approach by not performing any mutation in the seeded region to generate a neighbor. The mutation is performed on the remaining nodes (50 to 127) to generate a neighbor.

4.2.7 STOPPING CRITERIA

The stopping criterion is 100 temperature loop iterations with no change in the present individual fitness. This is an arbitrary choice. For initial runs the stopping criterion is kept at 200 temperature loop iterations (similar to GA and PSO) but this took more time to run with no improvements in snake length. Therefore, the stopping criterion is set at 100 temperature loop iterations with no change in the present individual fitness.

4.2.8 ACCEPTING PROBABILITY

If the neighbor has more fitness than the present individual does, then the neighbor replaces the present individual. If the neighbor has less fitness than the present individual does, then a probability is generated using the following formula

$$P = e^{((-1)* (\text{present fitness - neighbor fitness})/\text{temperature})}$$

A random number is generated and if the generated number is less than P then the neighbor replaces the present individual. If the random number is greater than or equal to P then no changes are made to the present individual. With higher temperature, the probability of accepting a bad neighbor is greater.

4.2.9 EXPERIMENTAL SETUP

All the five fitness functions are tested using different combinations of Metropolis loop iterations, cooling rates (rate of decrease) and starting temperature. After each Metropolis loop, the temperature is changed to Temperature=Temperature*Cooling Rate.

Metropolis Loop : 20000, 40000

Cooling Rate : 0.95, 0.98

Temperature : 50, 500

The different settings discussed in the thesis proposal are reduced to the above settings as these settings alone for each fitness function for a combination of 8 (2×2×2) for 20 trial runs took around 140 hours (close to 6 days) to run. To see how seeding and engineered conditioning effected the length of the snakes, all the settings are tested by not seeding at all then tested again by just seeding and no engineered conditioning and lastly with seeding and engineered conditioning by restricting the movement of the individual in the seeded region.

4.2.10 RESULTS

Without seeding, fitness function 2a performed better than the other fitness functions. Fitness function 2e is next. The longest snake found is of length 83 with fitness functions 2a and 2e. The best settings are determined using the average snake length for 20 trial runs. When the average snake length is considered, fitness function 2a is the best with an average snake length of 78.1 with the Metropolis loop at 40000 iterations, cooling rate at 0.98 and with the starting temperature at 500. Fitness function 2e is next with 77.65. The results for fitness function 2a are in Table 4.1 and the results for fitness function 2e are in Table 4.2. Only the results for the best two fitness functions are shown. In general, with an increase in the Metropolis loop, the average fitness went up but due to time constraints, large values for the Metropolis loop are not taken. Fitness functions 2b, 2c and 2d did not perform well.

nloop	cooling rate	temperature																Runs	avg length
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
20000	0.95	50		1	1	2	4	4	2	6								20	76.95
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
20000	0.95	500	2		2	4	3	4	3	1	1							20	76.05
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
20000	0.98	50			1	2	3	2	5	4	1	1	1					20	77.75
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
20000	0.98	500		1		1	5	5	5	1	1	1						20	77.15
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
40000	0.95	50		1		3	4	3	3	3		3						20	77.35
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
40000	0.95	500		1	1		4	5	3	4	1		1					20	77.40
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
40000	0.98	50				1	5	5	3	2		2	2					20	77.90
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
40000	0.98	500		1		4	1		4	3	5	1		1				20	78.10

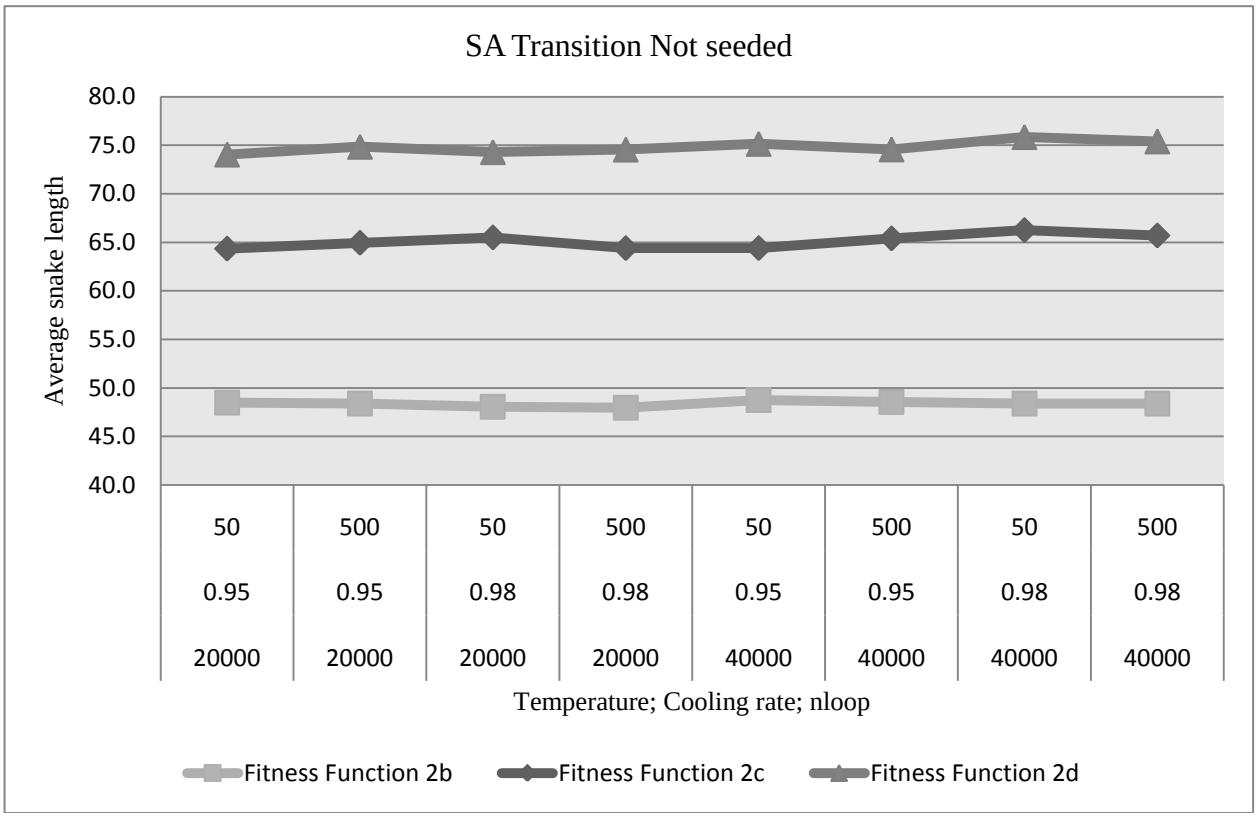
Table 4.1: SA using Transition sequence Representation without Seeding for Fitness Function 2a

nloop	cooling rate	temperature																Runs	avg length
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
20000	0.95	50	1	1	4	4	3	4	2	1								20	75.60
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
20000	0.95	500	2	3	4	3	1	1	2	1	1	1	1					20	75.75
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
20000	0.98	50		1	3			3	6	1	6							20	77.65
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
20000	0.98	500		2		6	3	2	3	3	1							20	76.45
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
40000	0.95	50		2	2	2	5	3	2	1	1	2						20	76.60
			71	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
40000	0.95	500	1	1	1	2	1	4	1	5	2	1	1					20	77.40
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
40000	0.98	50			1	3	3	5	6		1			1				20	77.15
			72	73	74	75	76	77	78	79	80	81	82	83	84	85	86		
40000	0.98	500				3	2	6	5	2	1	1						20	77.40

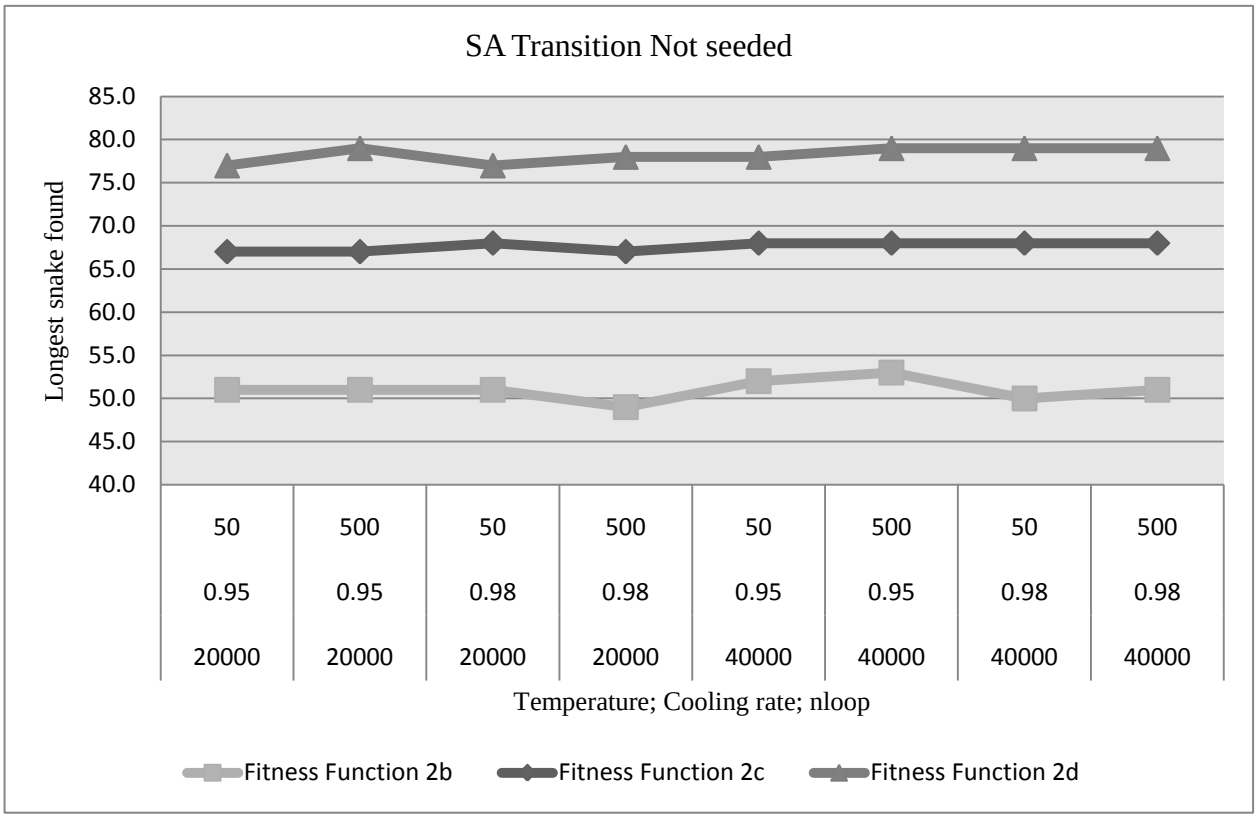
Table 4.2 SA using Transition sequence Representation without Seeding for Fitness Function 2e

Graph 4.1 shows the average snake length for different settings for fitness functions 2b, 2c and 2d. Fitness function 2d outperformed fitness function 2c for all settings and fitness function 2b is outperformed by fitness function 2c for all settings. As the range for skin nodes taken into account in fitness function is reduced from [3, 7] to [5, 7] the snake length increased and the fitness functions with lower range performed better.

Graph 4.2 shows the best snakes found for fitness functions 2b, 2c and 2d using various settings. The performance of fitness functions in finding the longest snake is similar to how fitness functions performed with the average snake length. The longest snake found using fitness function 2b is 53. The longest snake found using fitness function 2c is 68 and the longest snake found using fitness function 2d is 79.



Graph 4.1 SA using Transition Sequence without Seeding: Average Snake Length

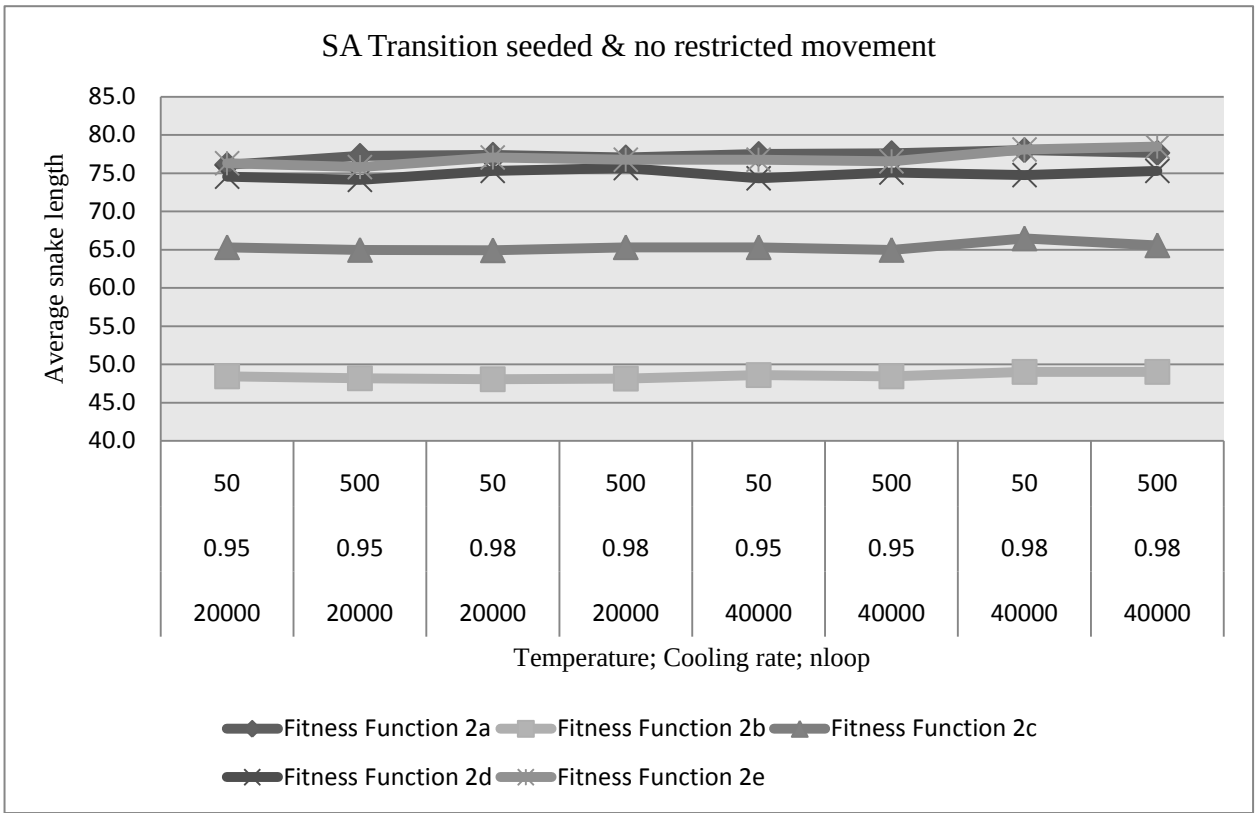


Graph 4.2 SA using Transition Sequence without Seeding: Longest Snake Found

With seeding and no restrictions on the movement of the individual, fitness function 2e performed better than the other fitness functions when the average snake length is considered. When finding the longest snake possible is considered fitness function 2a outperformed the other fitness functions with the longest snake of length 84. The best settings are determined using the average snake length for 20 trial runs. Fitness function 2e is the best with an average snake length of 78.84 with the Metropolis loop for 40000 iterations, cooling rate at 0.98 and with starting temperature 500 compared to 78.1 with no seeding and the same settings, it performed better. Fitness function 2a is next with 78. The performance of fitness functions 2b, 2c and 2d are almost identical to how they performed without seeding.

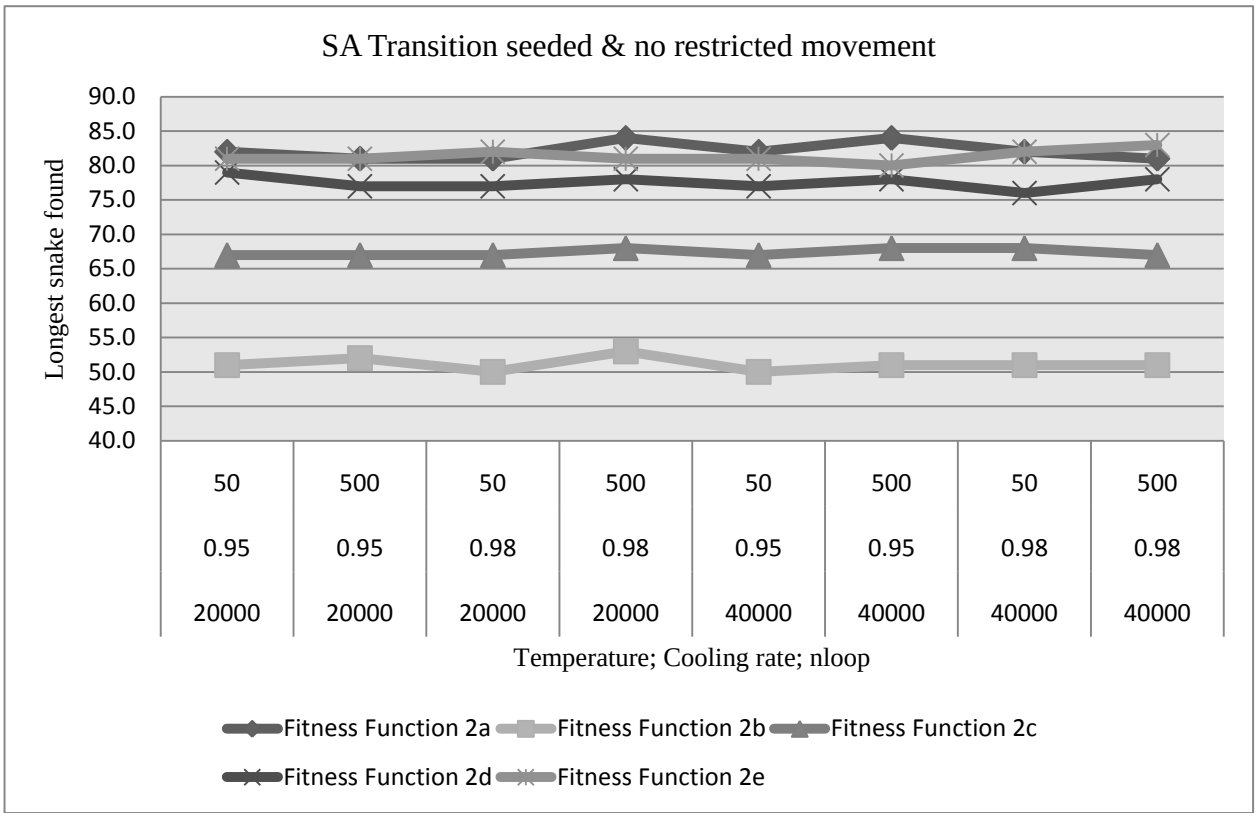
Due to the large Metropolis loop iterations, the seeded sequence is lost due the movement during the early stages. With high initial temperature, the chance of moving to a location with lower fitness (and most of the time smaller snake) is more likely. Therefore, whatever seeding did at the beginning is lost during the initial stages itself and therefore the effect of seeding on the individual is not seen in the results. This drawback is countered by using engineered conditioning where the movement of the individual is restricted by not allowing the seeded region to be disrupted during the generation of a neighbor by only changing the transition sequence after the seeded sequence (from 50 to 127).

Graph 4.3 shows average snake length for the different settings for all fitness functions. The average snake length for fitness functions 2a, 2d and 2e is in the high 70s while for fitness functions 2b and 2c, the average snake length is in the high 40s and mid 60s respectively.



Graph 4.3 SA using Transition Sequence: Seeding and no restricted movement: Average Snake Length

Graph 4.4 shows the best snake found for all fitness functions with different settings. The best snake found using fitness function 2a is 84 with initial temperature at 500, cooling rate at 0.98, Metropolis (nloop) at 20000 and for another setting with temperature at 500, cooling rate at 0.95, and Metropolis loop at 40000.



Graph 4.4 SA using Transition Sequence: Seeding and no restricted movement: Longest Snake Found

With seeding and restrictions on movement of the individual by not allowing any changes in the seeded region, fitness function 2a performed better than the other fitness functions when average snake length is considered. When finding the longest snake possible is considered fitness function 2b outperformed other fitness functions with a longest snake of length 92.

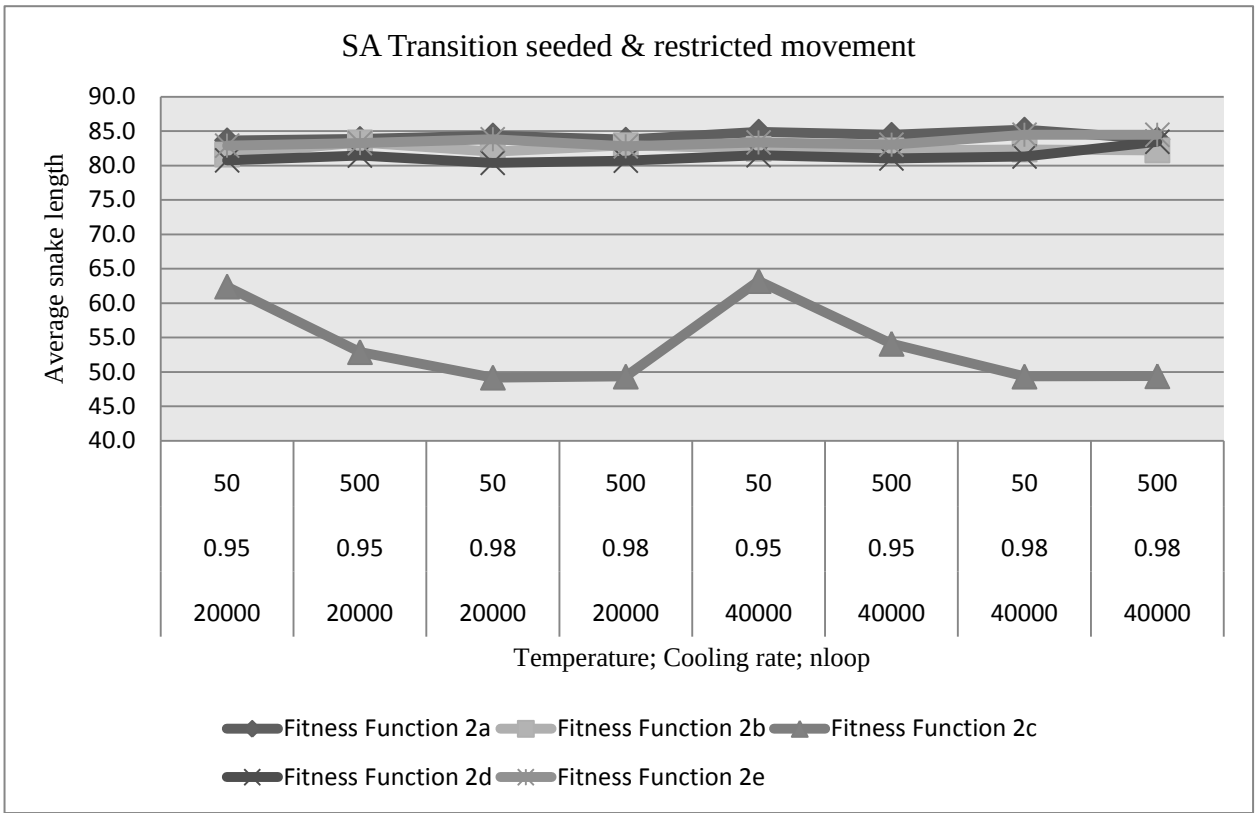
The best settings are determined using the average snake length for 20 trial runs. Fitness function 2a is the best with an average snake length of 85.2 with the Metropolis loop at 40000 iterations, cooling rate at 0.98 and with starting temperature at 50. The average snake length increased by about seven edges compared to the SA with seeding and no restrictions. The average snake length of all fitness functions except 2c is in the 80s. The average snake length for fitness function 2c is in the range of 50s to 60s. Table 4.3 shows the numbers for the average snake length and the longest snake for all fitness functions using different settings.

nloop	Cooling Rate	Temperature	Average Snake Length					Longest Snake				
			2a	2b	2c	2d	2e	2a	2b	2c	2d	2e
20000	0.95	50	83.7	81.6	62.4	80.8	82.9	88	88	87	88	87
20000	0.95	500	83.9	83.4	52.9	81.5	83.3	88	91	90	89	86
20000	0.98	50	84.4	82.0	49.2	80.4	83.8	88	92	50	89	89
20000	0.98	500	83.8	83.0	49.4	80.7	82.8	90	89	52	87	88
40000	0.95	50	84.9	82.4	63.2	81.5	83.4	91	88	86	88	90
40000	0.95	500	84.5	82.2	54.1	81.1	83.1	89	85	85	89	88
40000	0.98	50	85.2	82.4	49.4	81.3	84.5	90	87	53	90	88
40000	0.98	500	83.8	82.2	49.4	83.5	84.5	87	89	50	91	90

Table 4.3 SA using Transition Sequence Seeded & Restricted Movement

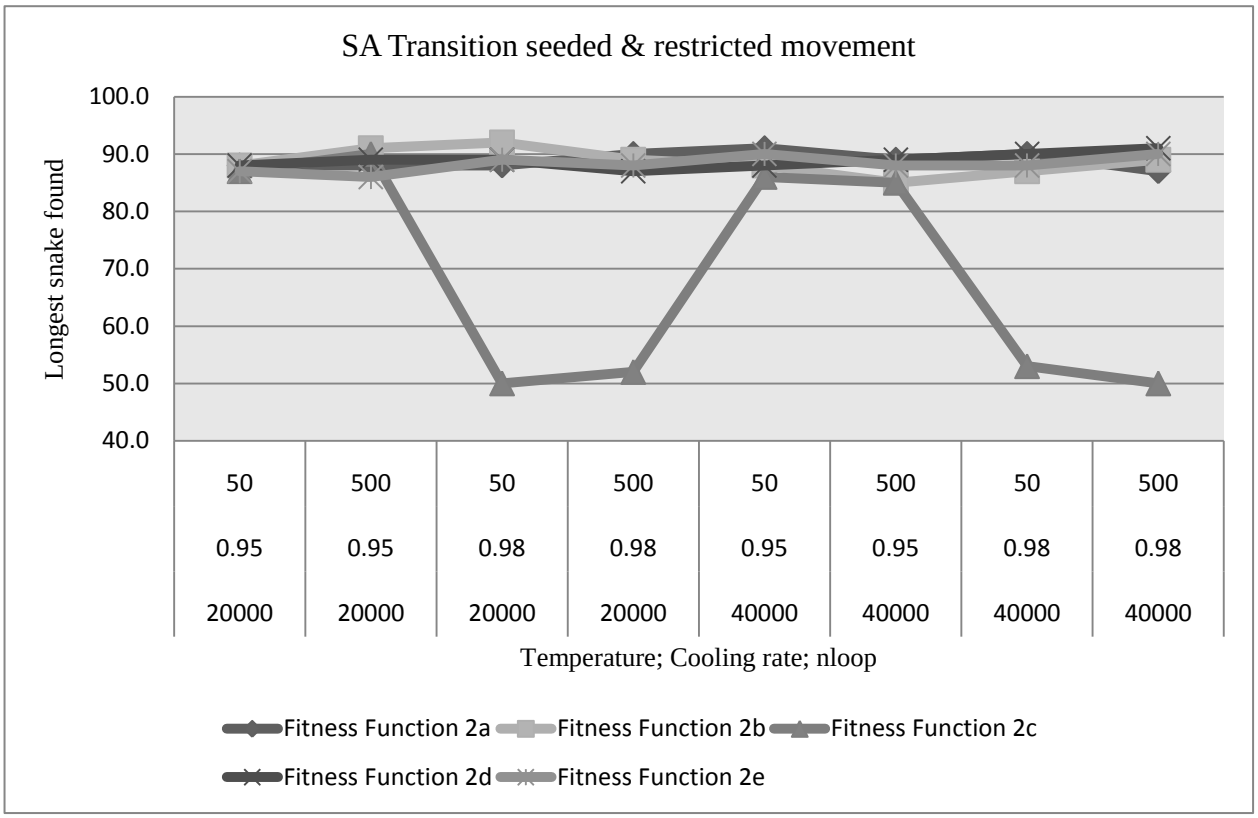
Fitness function 2c is able to find a longest snake of length 90 but during the 20 trial runs for the same settings, it found most snakes in the low 50s. The other fitness functions found longer snakes in the high 80s to low 90s consistently.

Graph 4.5 shows the average snake length for different settings for all fitness functions. The average snake length for fitness functions 2a, 2b, 2d and 2e varied from 80.7 to 85.2 while for fitness function 2c, the average length varied from 49.2 to 63.2. It is interesting to note that fitness functions 2a, 2b, 2d and 2e did not have any effect in performance with change in settings while fitness function 2c showed an effect in performance with a change in settings especially the cooling rate. It performed much better at lower cooling rates than at higher cooling rates and the same effect is observed with changes in temperature. Having high temperature decreased the average snake length. This can be seen prominently at cooling rates 0.95 while an increase in the Metropolis loop seems to have a little negative effect as well.



Graph 4.5 SA using Transition Sequence: Seeding and restricted movement: Average Snake Length

Graph 4.6 shows the best snake found for each setting in all fitness functions. The best snake found is of length 92 using fitness function 2b with an initial temperature of 50, cooling rate at 0.98, and Metropolis (nloop) at 20000. The effect of different settings using fitness function 2c can also be seen using the longest snake found using different settings, which are consistent to the pattern seen with average snake length. This indicates that the shorter the run, the chance of finding a long snake is greater, at least in the case of fitness function 2c. Having a lower cooling rate decreases the time taken for the run to complete as the temperature is decreased at a much faster rate. A change in settings for the remaining fitness functions did not have much impact on their performance in finding the longest snake possible.



Graph 4.6 SA using Transition Sequence: Seeding and restricted movement: Average Snake Length

4.3 USING BIT REPRESENTATION

4.3.1 INTRODUCTION

To find the longest snake in an 8-dimensional hypercube, a bit representation is used to represent the snake. The best snake found using the bit representation reported here is 97.

4.3.2 REPRESENTATION

The individual in Simulated Annealing contains 256 bits (one bit for each node in 8-dimensional hypercube 2^8=256). If a node is used then its corresponding bit would be set to 1 indicating that the node is turned on, if not then it would be set to zero indicating that the node is turned off. The first node (Node 0) is always turned on, i.e., set to one. To generate the node sequence using the bit representation, the sequence always starts at node 0 and then looks to see if any of its adjacent nodes (any of the 8 possible nodes adjacent to node 0) are turned on. If more than one node is turned on then the node with the lowest node number is always chosen. For node 0, if node 1, 2 and 4 are turned on then the sequence will always choose node 1 as it has the lowest node number. Once the node sequence is generated, the length of the snake and its fitness are calculated.

The length of the individuals is 256 bits, one for each node. If the initial individual is seeded then only the bits corresponding to nodes in the seeded sequence are turned on. The remaining bits (>127) are randomly turned on or off. To restrict the movement in the seeded region, the bits from zero to 127 are kept unchanged by restricting mutation (to generate a neighbor) to bits greater than 127. If the initial individual is randomly seeded then all bits (0 to 255) are randomly turned on or off.

4.3.3 FITNESS FUNCTION

To calculate the fitness function the individual in the bit representation is converted to a node sequence representation. This node sequence is then used to calculate the fitness of the individual. All the five fitness functions are tested individually to see how well each fitness function did in comparison with each other. The five fitness functions used are fitness function 2a, fitness function 2b, fitness function 2c, fitness function 2d and fitness function 2e.

4.3.4 GENERATING NEIGHBORS

To get a neighbor for the individual, a number is generated at random between 1 and 255. If that node is turned on (set to 1) then it is turned off (set to 0) and if the node is turned off then it is turned on. This generates a neighbor to the present individual.

4.3.5 INITIAL INDIVIDUAL

The individual is seeded with the longest snake from the 7-dimensional hypercube given in [Potter94]. It is converted from a node sequence representation to bit representation. First, all 256 bits are turned off (set to 0) and then the bits with the corresponding nodes in the node sequence are turned on. Bits corresponding to the nodes greater than 127 are randomly turned on. If the individual is not seeded then all the bits are turned on or off randomly.

4.3.6 ENGINEERED CONDITIONING

In an effort to increase efficiency and reliability in finding longer snakes and better average snake length, a strategy to restrict the movement of an individual is used. When the individual is seeded, the seeded individual's bits <128 are not changed during the entire SA

approach. So all the nodes from 0 to 127 that are turned on during initial seeding will remain on until the end and the nodes that are turned off will remain off until the end. In restricted movement, bits less than 128 are not changed to generate neighbors.

4.3.7 STOPPING CRITERIA

The stopping criterion is 100 temperature loop iterations with no change in the present individual fitness. This is an arbitrary choice.

4.3.8 ACCEPTING PROBABILITY

If the neighbor has more fitness than the present individual does, then the neighbor replaces the present individual. If the neighbor has less fitness than the present individual does, then a probability is generated using the following formula

$$P = e^{((-1)*(\text{present fitness} - \text{neighbor fitness})/\text{temperature})}$$

A random number is generated and if the generated number is less than P then the neighbor replaces the present individual. If the generated number is greater than or equal to P then the present individual remains unchanged.

4.3.9 EXPERIMENTAL SETUP

All the five fitness functions are tested using different combinations of Metropolis loop iterations, cooling rates and starting temperature. After each Metropolis loop, the temperature is changed to Temperature=Temperature*Cooling Rate.

Metropolis Loop : 20000, 40000

Cooling Rate : 0.95, 0.98

Temperature : 50, 500

The different settings discussed in the thesis proposal are reduced to the above settings as these settings alone for each fitness function for a combination of 8 (2×2×2) for 20 trial runs took around 310 to 370 hours (12 to 15 days) to run.

To see how seeding and engineered conditioning effected the length of the snake, all the settings are tested first by not seeding at all then tested again by just seeding and no engineered conditioning and lastly with seeding and engineered conditioning by restricting the movement of the individual in the seeded region.

4.3.10 RESULTS

Without seeding, fitness function 2e performed better than the other fitness functions. Fitness function 2a is next. See Table 4.4.

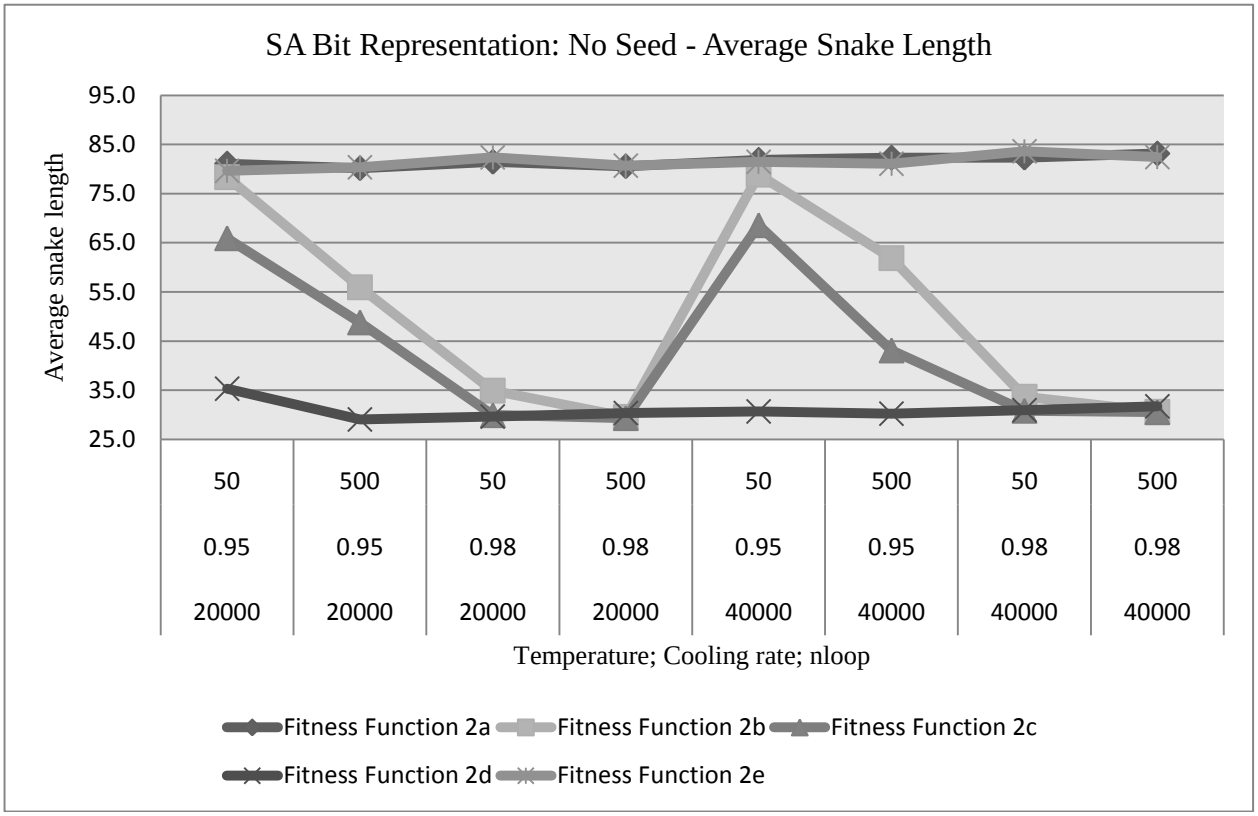
			Average Snake Length					Longest Sanke				
nloop	cooling rate	temperature	2a	2b	2c	2d	2e	2a	2b	2c	2d	2e
20000	0.95	50	81.10	78.35	65.90	35.30	79.65	86	83	85	81	83
20000	0.95	500	80.15	55.95	48.80	29.05	80.35	83	81	86	32	86
20000	0.98	50	81.45	34.90	29.95	29.65	82.35	86	81	33	32	87
20000	0.98	500	80.55	29.60	29.35	30.35	80.70	84	33	32	33	85
40000	0.95	50	81.85	78.85	68.55	30.70	81.45	87	84	85	34	86
40000	0.95	500	82.30	61.90	43.10	30.26	81.05	86	84	84	35	86
40000	0.98	50	82.30	33.70	30.85	30.95	83.60	88	78	35	34	88
40000	0.98	500	83.15	30.60	30.55	31.75	82.45	87	33	36	37	87

Table 4.4: SA using Bit Representation without Seeding

The longest snake found is of length 88 with fitness functions 2e and 2a. The best settings are determined using the average snake length for 20 trial runs. Fitness function 2e is the best with an average snake length of 83.60 compared to 78.1 found by the SA using a transition sequence representation with no seeding for an increase of 5.5 edges. The SA setting for the best snake is with a Metropolis loop at 40000 iterations, cooling rate at 0.98 and with the starting

temperature 50. Fitness function 2a is next with an average snake length of 83.15. Table 4.4 gives details on how each fitness function performed using different settings with the average snake length and the longest snake found in 20 trial runs. Fitness functions 2a and 2e responded to an increase in temperature, cooling rate and Metropolis loop by finding longer snakes and an increase in average length of snakes. The remaining three fitness functions 2b, 2c and 2d varied differently and do not seem to respond positively to an increase in temperature, cooling rate and Metropolis loop. Fitness functions 2b, 2c, and 2d did not perform well. The fitness function performance is similar to the performance seen by the SA using a transition sequence representation.

Graph 4.7 shows the average snake length for all fitness functions. The performance of fitness functions 2a and 2e are identical for all the settings.

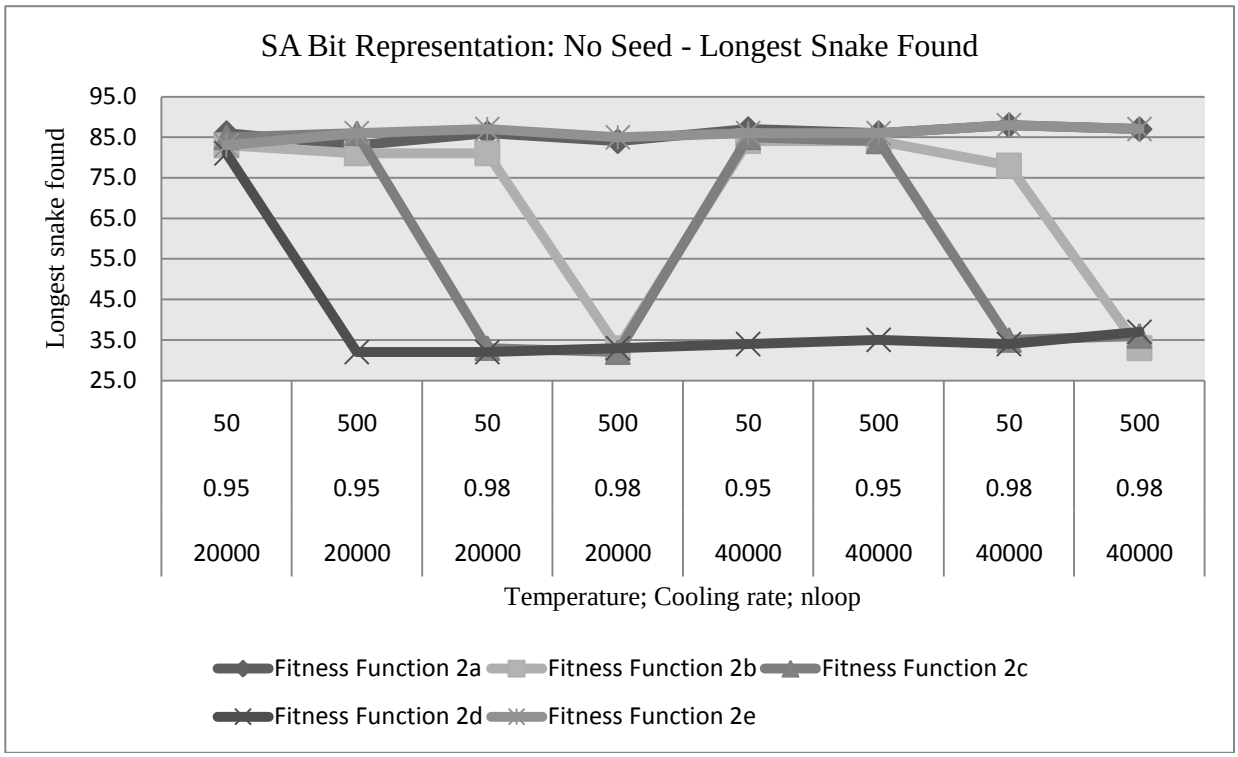


Graph 4.7 SA using Bit Representation without Seeding: Average Snake Length

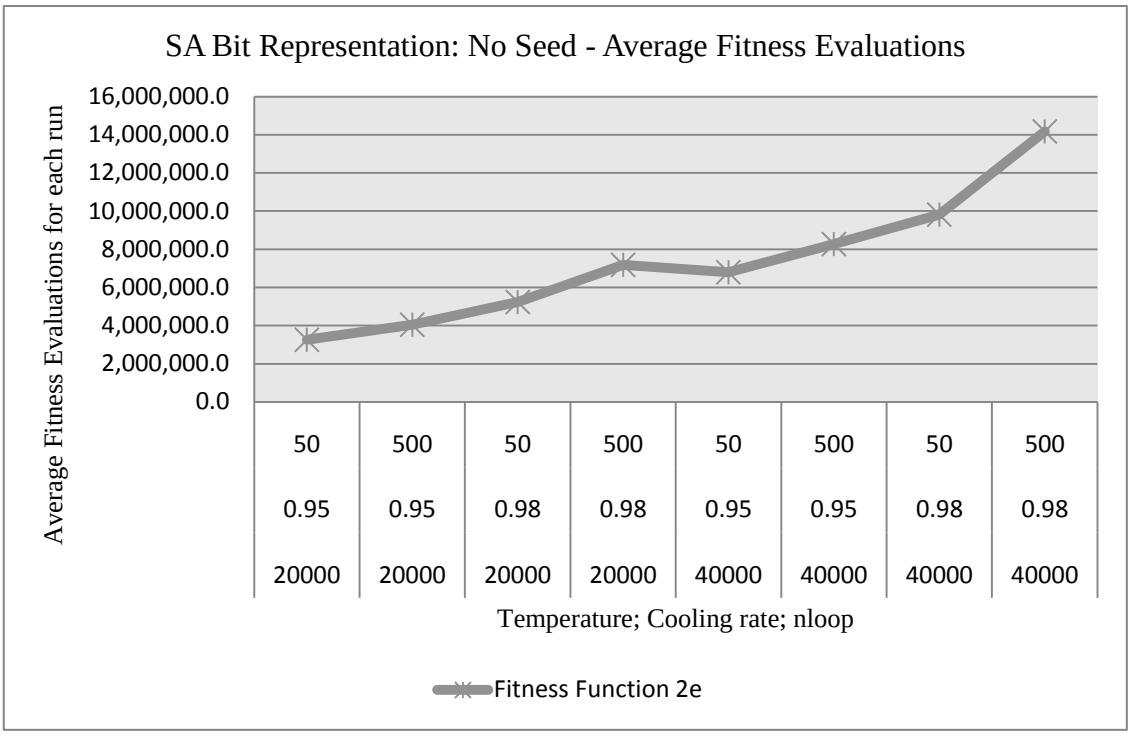
Fitness function 2d is outperformed by all fitness functions. The reason behind this is that even for a snake of length 97 the number of nodes shared 6 times is 6 to 7 while the number of nodes shared 7 times is zero. Therefore, the only factor that counts for fitness function 2d is the number of nodes shared 5 times. Therefore, the fitness function tries to find snakes that suit the fitness function well but a snake of length 97 might not have better fitness than smaller snakes as far as fitness function 2d is concerned. The count for nodes shared 3, 4 and 5 times are almost identical in the range of mid 30s to mid 40s for long snakes. So using all these should give better results. Fitness functions 2b and 2c performed poorly as well. With lower temperature and lower cooling rate, fitness functions 2b and 2c performed better. Higher temperature and higher cooling rate had a negative effect on the length of the snake, as the likelihood of moving to a bad location is more when compared.

Graph 4.8 shows the longest snake found for all fitness functions. The performance of the fitness function in finding the longest snake is similar to how fitness functions performed with average snake length. The longest snake found using fitness function 2a is 87, 2b is 84, 2c is 86, 2d is 81 and for 2e is 88.

Graph 4.9 shows how different settings affect the average number of fitness evaluations performed during each run. The average fitness evaluations are collected for fitness function 2e. The trial runs using fitness function 2e runs are longer than the trial runs using any other fitness functions. For the best performing setting (temperature 50, cooling rate 0.98 and Metropolis loop 40000), it took 9.8 million fitness evaluations on average while the setting with temperature 500, cooling rate 0.98 and Metropolis loop 40000 took 14.2 million fitness evaluations.



Graph 4.8 SA using Bit Representation without Seeding: Longest Snake Found



Graph 4.9 SA using Bit Representation without Seeding: Average Fitness Evaluations

With seeding and no restrictions on movement of the individual, fitness function 2a performed better than the other fitness functions when average snake length is considered. When finding the longest snake possible is considered, fitness function 2b outperformed the other fitness functions with the longest snake of length 90, compared to a snake of length 84 found by SA using a transition sequence representation with seeding and no restrictions, there is an increase of 6 edges in the longest snake found. See Table 4.5.

			Average Snake Length					Longest Sanke				
nloop	cooling rate	temperature	2a	2b	2c	2d	2e	2a	2b	2c	2d	2e
20000	0.95	50	80.00	77.60	51.55	51.45	70.85	87	82	53	53	87
20000	0.95	500	80.70	51.15	50.60	52.00	80.20	86	54	53	58	85
20000	0.98	50	81.85	51.05	51.75	51.15	81.15	87	56	55	54	87
20000	0.98	500	82.55	51.20	51.30	51.40	81.35	85	55	54	58	86
40000	0.95	50	82.25	78.10	51.30	51.50	81.85	86	90	55	55	85
40000	0.95	500	81.10	51.75	50.95	51.25	81.40	87	56	54	54	85
40000	0.98	50	84.25	51.05	51.35	52.15	82.15	87	54	55	55	85
40000	0.98	500	82.95	51.25	51.00	51.05	82.75	85	54	54	54	87

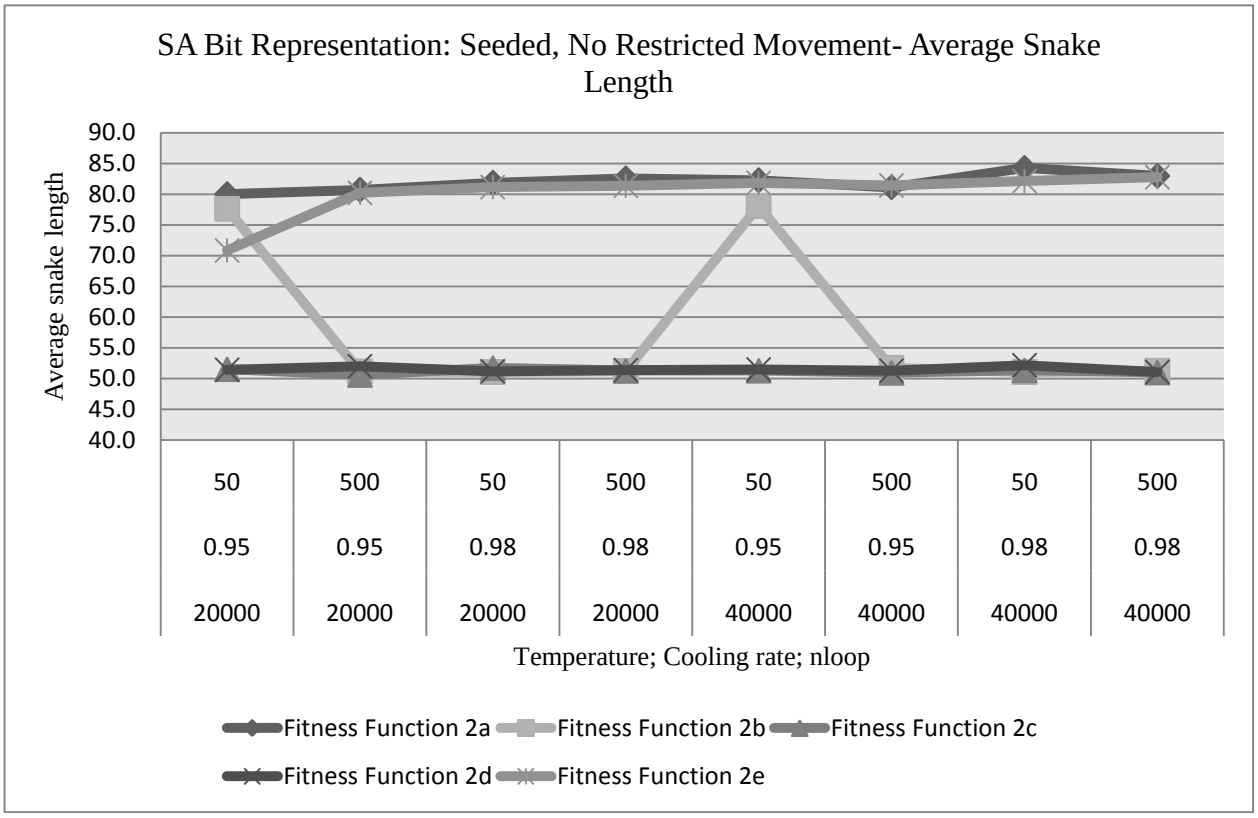
Table 4.5: SA using Bit Representation with Seeding and No Restricted Movement

The best settings are determined using the average snake length for 20 trial runs. Table 4.5 gives details on how each fitness function performed using different settings with the average snake length and the longest snake found in 20 trial runs. Fitness function 2a is the best with an average snake length of 84.25 with Metropolis loop at 40000 iterations, cooling rate at 0.98 and with the starting temperature at 50. Fitness function 2e is next with 82.75. The performance of fitness functions 2b, 2c and 2d are almost identical and compared to the SA with no seeding the average snake length is mostly unaffected by changes in settings for the SA while all the three equations showed plenty of variation in average snake length when the individual is not seeded.

Due to large Metropolis loop iterations, the seeded sequence in bit representation is lost due to movement during the early stages. With a high initial temperature, the chance of moving to a location with lower fitness (and most of the time smaller snake) is more likely. Therefore,

whatever seeding did at the beginning is lost during the initial stages itself and therefore the effect of seeding the individual is not seen in the results. This drawback is countered by using engineered conditioning where the movement of an individual is restricted in the seeded region by not allowing the seed to be disrupted during the generation of a neighbor by only changing the bits after the seeded sequence (from 128 to 256).

Graph 4.10 shows the average snake length for all fitness functions. The performance of fitness functions 2a and 2e is identical for all the settings.

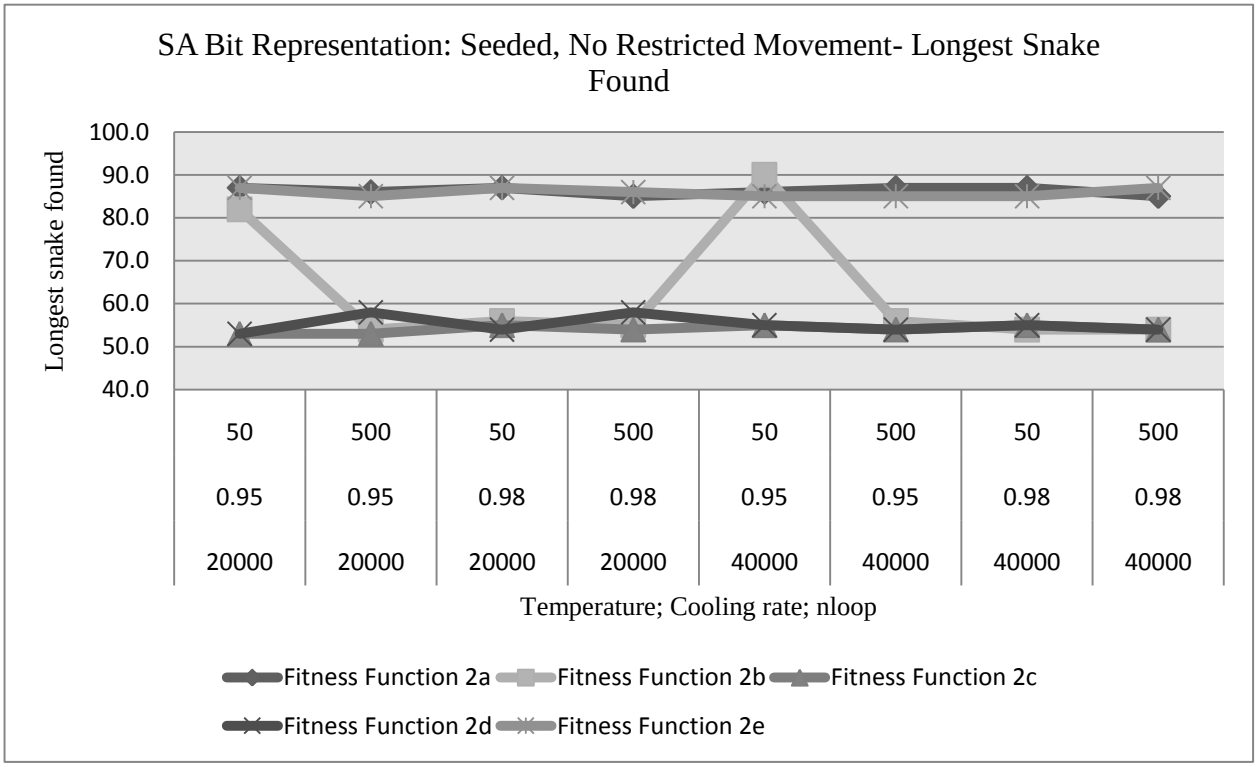


Graph 4.10 SA using Bit Representation: Seeding and No Restricted Movement: Average Snake Length

Except for Fitness Function 2b, all other fitness functions have no effect on average snake length with changes in settings. Fitness functions 2a and 2e found snakes in the 80s for almost all settings while fitness functions 2c and 2d found snakes in the 50s for all settings. Fitness function 2b is the only one to respond with changes in average snake length with changes

in settings. At a lower temperature and lower cooling rate, the average snake length went up to the 70s irrespective of the Metropolis loop while for all other settings it performed just like fitness functions 2b and 2c with average snake length in the 50s.

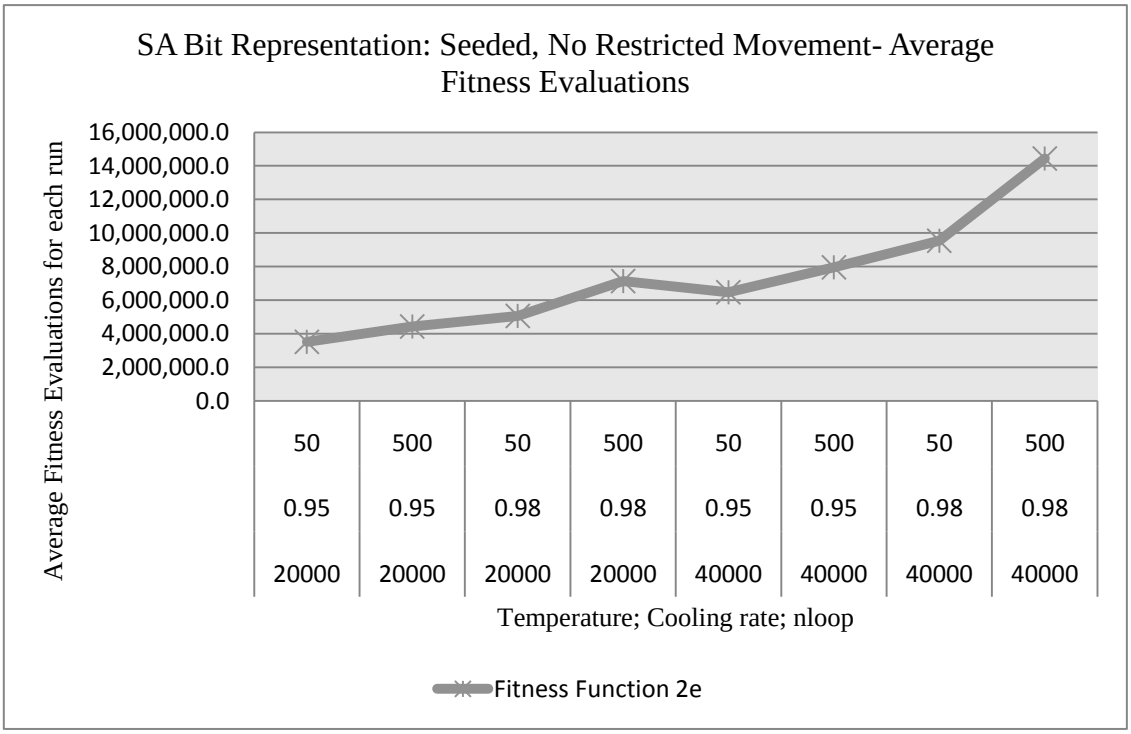
Graph 4.11 shows the best snake found for all fitness functions. The performance of the fitness function in finding the longest snake is similar to fitness function performance with average snake length. The longest snake found using fitness function 2a is 87, 2b is 90, 2c is 55, 2d is 58 and for 2e is 87. The snake of length 90 found using fitness function 2b is more of a chance as the next best snake found is of length 82.



Graph 4.11 SA using Bit Representation: Seeded and no Restricted Movement: Longest Snake Found

Graph 4.12 shows how different settings effect the average fitness evaluations performed during each run. The average fitness evaluations are collected for fitness function 2e. With temperature 500, cooling rate at 0.98 and Metropolis loop at 40000 it took 14.4 million fitness

evaluations on average, which is almost similar to the number of average fitness evaluations done without seeding.



Graph 4.12 SA using Bit Representation: Seed and no Restricted Movement: Average Fitness Evaluations

With seeding and restrictions on movement of the individual by not allowing any changes in the seeded region (0 to 127 bits), fitness function 2e performed better than the other fitness functions when average snake length is considered. Fitness function 2a is next but when finding the longest snake possible is considered fitness function 2a performed better with various settings. See Table 4.6.

			Average Snake Length					Longest Sanke				
nloop	cooling rate	temperature	2a	2b	2c	2d	2e	2a	2b	2c	2d	2e
20000	0.95	50	90.70	90.30	89.90	89.15	89.70	95	97	97	92	95
20000	0.95	500	90.95	84.30	79.95	76.35	90.10	95	96	97	93	94
20000	0.98	50	92.45	81.90	70.65	68.30	91.40	96	97	92	89	95
20000	0.98	500	92.60	66.95	66.15	65.85	90.45	97	89	70	71	97
40000	0.95	50	92.40	91.75	89.50	90.20	90.65	97	96	95	94	96
40000	0.95	500	92.05	88.25	82.85	78.95	92.15	97	95	96	92	96
40000	0.98	50	93.25	79.60	72.80	72.45	93.40	97	95	91	93	97
40000	0.98	500	93.40	68.80	66.20	66.30	93.50	97	94	68	69	97

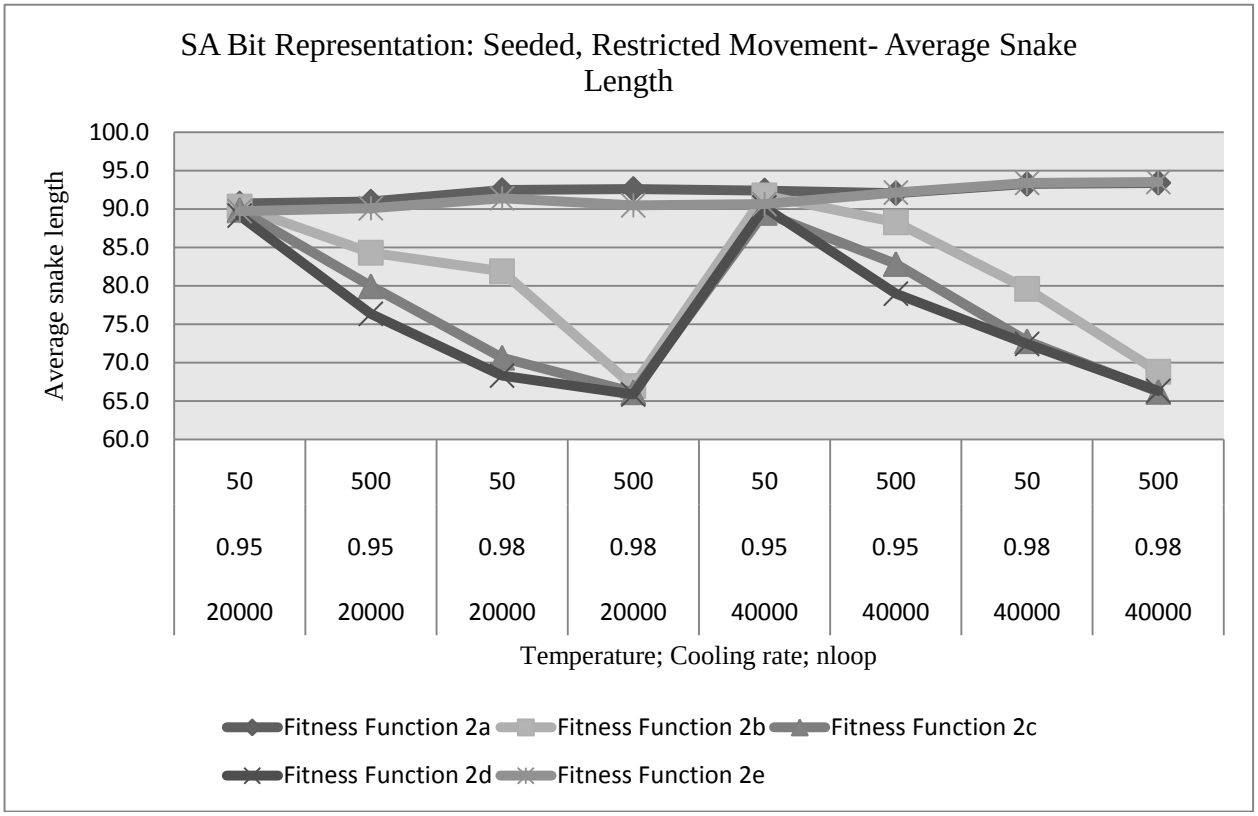
Table 4.6: SA using Bit Representation with Seeding and Restricted Movement

Fitness function 2a found the longest snake of length 97 for 5 out of 8 settings. Fitness function 2e also found the longest snake of length 97 but for only 3 out of 8 settings. The longest snake found by SA using a transition sequence representation with seeding and restricted movement is 92. The average length of the snake increased by almost 10 edges when compared. The use of a bit representation took more time to run but gave better results.

The best settings are determined using the average snake length for 20 trial runs. Table 4.6 gives details on how each fitness function performed using different settings with the average snake length and the longest snake found in 20 trial runs. Fitness function 2e is the best with an average snake length of 93.5 for Metropolis loop at 40000 iterations, cooling rate at 0.98 and with the starting temperature at 500. The average snake length increased by about seven edges compared to the SA using the bit representation with seeding and no restrictions. Fitness function 2a is next with 93.4. Fitness functions 2b, 2c and 2d were outperformed by fitness functions 2a

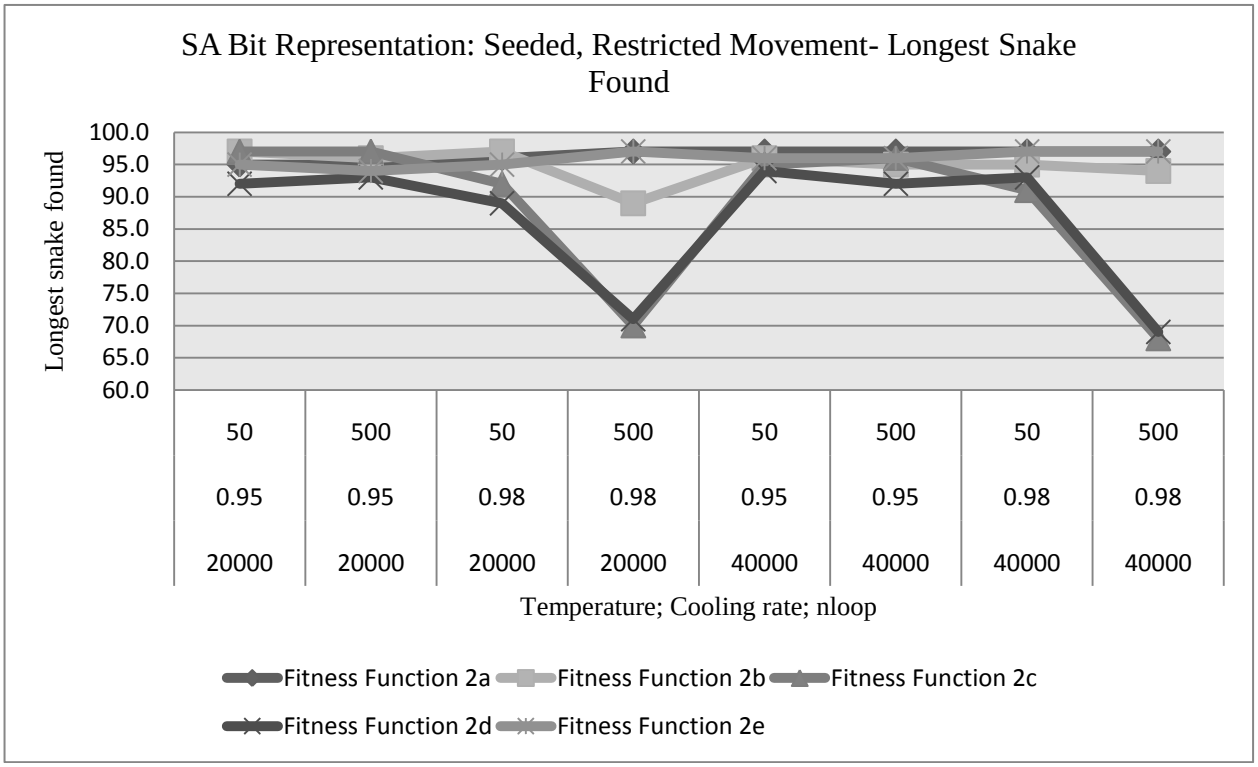
and 2e. Fitness functions 2b, 2c and 2d showed much variation in the average snake length for different settings similar to how they performed when they are not seeded. All three fitness functions (2b, 2c and 2d) performed poorly with high temperature and high cooling rate.

Graph 4.13 shows the average snake length for all fitness functions. The performance of fitness functions 2a and 2e are identical for all the settings with both showing an increase in snake length for an increase in temperature and cooling rate. Fitness functions 2a and 2e found snakes in the 90s consistently. The remaining three fitness functions (2b, 2c and 2d) performed similarly to changes in settings with the average snake length peaking at a low temperature and low cooling rate. Any increase in temperature and cooling rate had a negative effect on the snake length. The difference in the average snake length for the low settings and high settings is about 20 edges. An increase in Metropolis loop has a slight increase in performance but not by much.



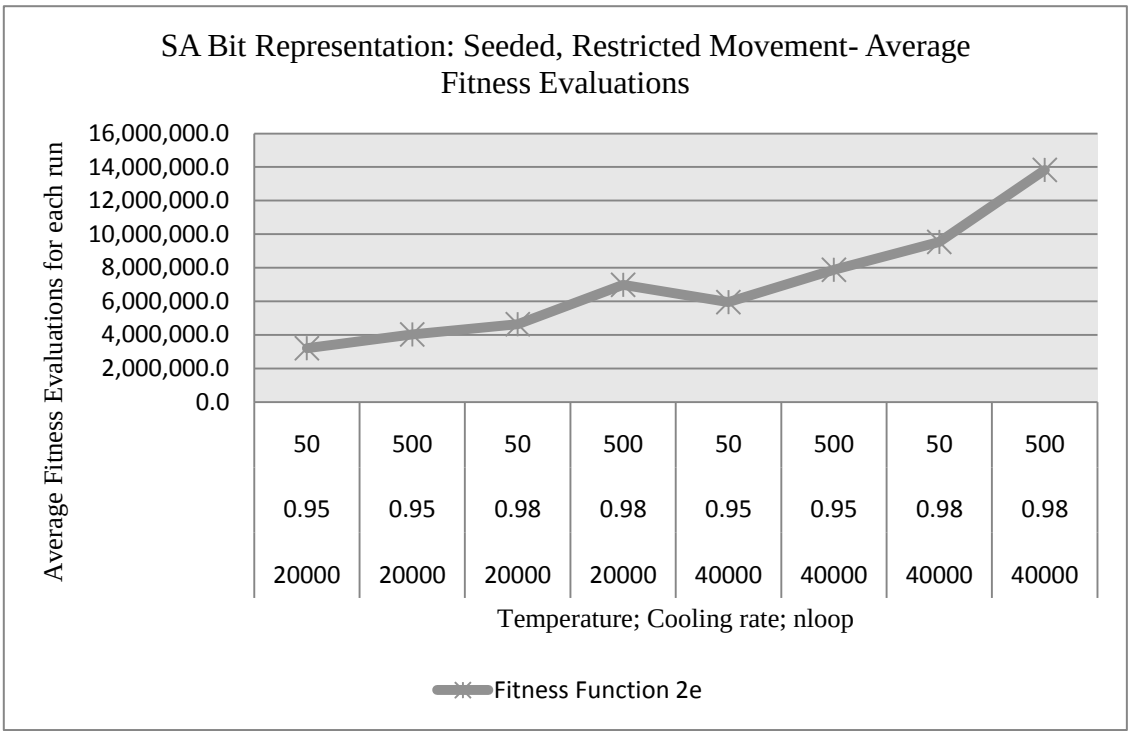
Graph 4.13 SA using Bit Representation: Seeding and Restricted Movement: Average Snake Length

Graph 4.14 shows the best snake found for all fitness functions. The performance of the fitness functions in finding the longest snake is similar to how fitness functions performed with average snake length for fitness functions 2a, 2e, 2c and 2d. Fitness function 2b performed better when compared to fitness functions 2c and 2d in finding the longest snake. For average snake length, fitness function 2b's performance is similar to fitness functions 2c and 2d but for longest snake found it performed similar to fitness functions 2a and 2e. All fitness functions except fitness function 2d are able to find a snake of length 97 at least once. Fitness function 2d was only able to find a snake of length 94.



Graph 4.14 SA using Bit Representation: Seeded and Restricted Movement: Longest Snake Found

Graph 4.15 shows how different settings effect the average fitness evaluations performed during each run. The average fitness evaluations are collected for fitness function 2e. The setting with temperature at 500, cooling rate at 0.98 and Metropolis loop at 40000 took 13.8 million fitness evaluations on average.



Graph 4.15 SA using Bit Representation: Seed and Restricted Movement: Average Fitness Evaluations

The average fitness evaluations when compared with the GA and the PSO are very large. For the GA using the bit representation the setting that found a snake of length 97 took on average (over 5 trial runs) 690720 fitness evaluations. For the PSO using the bit representation the setting that found the longest snake of length 95 took on average (over 5 trial runs) 123280 fitness evaluations. When compared to SA these are very small numbers. The stopping criteria for SA with Metropolis loop at 40000 iterations itself takes about 4 million (40000×100 temperature loop no change in individual) fitness function evaluations to decide to stop (there has been no change in individual fitness during the 4 million fitness evaluations).

The sequence of zeros and ones below is the bit representation of one of the snakes of length 97 found using fitness function 2e. There is only one available node (25). There are 12 edges that are shared once, 16 nodes that are shared twice, 30 nodes shared 3 times, 45 nodes shared 4 times, 48 nodes shared 5 times, 6 nodes shared 6 times, and there are no nodes that are

shared 7 times. The remaining 98 nodes are part of the snake. It took around 5 hours to find this snake and about 16.84 million fitness evaluations are done during the process.

Bit Representation: 1 1 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 1 1 0 0 1 1 0 0 1 1 0 1 1 0 1 0 0 0 1 1 0 0 0 1 0 0 1 0 0 1 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 0 1 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 1 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1

Converted to Node Representation: 0 1 3 7 15 31 63 55 51 49 48 56 60 28 20 22 18 26 10 42 46 38 36 37 45 41 105 104 72 88 80 81 83 91 123 122 126 118 116 117 125 93 77 69 68 70 66 98 99 103 111 239 235 234 202 218 222 220 212 213 215 247 243 242 240 248 249 217 201 193 225 161 163 162 130 134 132 133 141 157 189 181 180 182 190 186 187 155 147 145 144 152 136 168 172 236 228 230 198

4.4 CONCLUSION

The SA approach was able to find a snake of 97 using a bit representation with seeding and restricted movement on the individual in the seeded region. Even though it took around 14 million fitness evaluations on average, which compared to 2^{128} (= 3.4×10^{38}) is considerably smaller. Using restricted movement the number of combinations for the snake reduced from 10^{77} (2^{256}) to 10^{38} and due to a narrowed search the SA was able to find a snake of length 97.

CHAPTER 5

CONCLUSION AND FUTURE WORK

All search techniques performed better using the bit representation. The GA and SA are able to find snakes of length 97 using the bit representation. The PSO is able to find a snake of length 95. The snakes of length 97 and 95 found using the SA and the PSO respectively are the best snakes found using those search techniques. With proper representation, SA and the PSO are able to perform on par with the GA approach. The average snake length of 93.4 using SA is the best result obtained in this project. The best fitness function varied depending on the search technique used and the snake representation used.

The SA is generally used to find near optimal solutions. The SA was able to find snakes of length 97 in 4 out of 20 trial runs. If the longest snake in a 7-dimensional hypercube does not lead to the longest snake in 8-dimensional hypercube then with restricted movement in the seeded region, the search techniques would never be able to find longer snakes. With restricted movement, this project will not be able to find a snake (possibly longer than 97) that does not have the longest snake from 7-dimensional hypercube as its sub sequence.

For the GA, there are too many settings and some parameters are not fully explored. For Selection, only the Tournament Selection is used. Use of other Selection techniques like the Roulette Wheel Selection in combination with the Tournament Selection might help in giving better results. Use the Tournament Selection in the initial generations and in the later generations switch to the Roulette Wheel Selection. Use of new fitness functions that combine more factors might help in giving better results. For Crossover, only the Two-Point Crossover technique is used with the transition sequence and the bit representation. The One-point crossover technique

might help in giving better results. Local growing is done only every 5 generations, probably other combinations like using local growing every other generation or even every generation might give better results. Large population sizes (1000 or more) might give better results for all the three search techniques. Local growing was not used in the SA. The use of local growing might help the SA in finding better results.

A hybrid search technique like combining the GA and the SA might give better results. One idea is to have a normal GA but after every 5 or 10 generations take each individual and use the SA on that individual or once the GA is done, each individual is taken and the SA is used on each individual. The results from one run can be used as the seed for the next run. Only the longest snake in dimension 7 is used as the seed in this thesis. Using smaller snakes or shortening the longest snake randomly on either end might give longer snakes. The only drawback is that protecting the seed becomes a bit difficult. It came as a surprise when the SA found a snake of length 97, so depending on the problem, if one has the correct settings, the techniques that did not perform well before might do better. Other search techniques with the correct settings for the problem can be explored that were not given as much focus as the GA for the snake problem.

All engineered conditioning approaches and fitness functions can be easily modified to work in higher dimension snakes. One can use the longest snake from dimension 8 as the seed for a 9 dimensional hypercube. For a transition sequence representation, the seeded region will be 0 to 96 instead of 0 to 49. For a bit representation, the seeded region will be 0 to 256 (2^{n-1}). All fitness functions can be applied to any dimensional hypercube greater than seven.

Various settings for each parameter led to a huge number of combinations which took an extensive amount of time to execute. It might be better to use some kind of adaptive parameters

that can adjust themselves as the techniques progress. One can set values for some parameters directly which are known to work at certain values. For parameters that are uncertain, use of adaptive parameters might do a better job in finding longer snakes and in less time.

The type of representation used by the search techniques played a huge role on how the search techniques performed. More emphasis should be given to snake representation as this thesis showed that with a better representation like bit representation all search techniques were able to perform better. The use of various settings for different parameters showed how the results vary and how dependent the techniques are on the type of representation used. For example, for the PSO using transition sequence representation, inertia values around 0.4 gave better results while for the PSO using a bit representation inertia values around 1.0 gave better results. Therefore, if one setting worked for a certain representation, it may not necessarily work well when a different representation is used. The use of adaptive parameters might be the best way to go forward from here.

REFERENCES

[Kautz58] Kautz, W.H., "Unit-Distance Error-Checking Codes", *IRE Trans. Electronic Computers*, Vol. 7, pp 179-180, 1958.

[Klee67] Klee, V., "A method for constructing circuit codes", *J. Assoc. Comput. Mach.*, Vol. 14, pp 520-538, 1967.

[Klee70] Klee, V., "What is the maximum length of a d-dimensional snake?", *Amer. Math. Monthly*, Vol. 77, pp 63-65, 1970.

[Rajan99] Rajan D.S., and Shende A.M., "Maximal and Reversible Snakes in the Hypercube", in *Proceedings of the Annual Australian Conference on Combinatorial Mathematics and Combinatorial Computing*, 1999.

[Potter94] Potter, W.D., Robinson R.W., Miller J.A., and Kochut, K.J., "Using the Genetic Algorithm to Find Snake- In-The-Box Codes", *In Proc of the 7th International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems*, pp 421-426. Austin, Texas, 1994.

[Star91] Starkweather, T., McDaniel. S., Mathias, K., Whitley, D., and Whitley, C., "A Comparison of Genetic Sequencing Operators", *In Proc of the Fourth International Conference on Genetic Algorithm*, pp. 69-76, 1991.

[Snevily94] Snevily, H. S., "The Snake-in-the-Box Problem: A New Upper Bound*". Discrete Mathematics*. Vol. 133(1-3), pp 307-314, 1994.

[Kochut94] Kochut, K.J., Potter, W.D., Robinson, R.W., Miller, J.A., "Hunting For Snake-In-The-Box Codes." *Submitted to Annals of Mathematics and Artifical Intelligence*, 1994.

[Kochut96] Kochut, K.J., "Snake-In-The-Box Codes for Dimension 7." *Journal of Combinatorial Mathematics and Combinatorial Computing*, Vol. 20: pp 175-185, 1996.

[Goldberg89] Goldberg, D., "Genetic Algorithms in Search Optimization and Machine Learning" Addison Wesley Longman, Inc. Boston.MA, 1989.

[Diaz06] Diaz-Gomez, P., Hougen, D., "Hunting for Snakes in Hypercubes using Genetic Algorithms." *International Journal of Computer and Information Science.* Vol. 8, No. 1 March 2007.

[Touhy07] Touhy, D.R., Potter, W.D., Casella, D.A., "Searching for Snake-in-the-box Codes with Evolved Pruning Models", *Proceedings of the 2007 Int. Conf. on Genetic and Evolutionary Methods*, Las Vegas, Nevada: 3-9, 2007.

[Tasgetiren03] Tasgetiren, M.F., Liang, Y., "A Binary Particle Swarm Optimization Algorithm for Lot Sizing Problem." *Journal of Economic and Social Research*, Vol. 5(2), pp 1-10, 2003.

[Bitterman04] Bitterman, D.S., "New Lower Bounds For The Snake-In-the-Box Problem: A Prolog Genetic Algorithm and heuristic Search Approach", MSAI thesis, Univ. of Georgia, 2004.

[Chevalier] Chevalier, R.F., Using Genetic Algorithms to Search for Snake-in-the-Box Codes. Technical report from Institute of Artificial Intelligence, Univ. of Georgia. http://chevy76.myweb.uga.edu/pdf/Snake.pdf (June. 18 2009)

[Handy08] Handy, G., "Apply Simple Genetic Algorithms to the Snake-in-the Box Problem in Dimension 8".
Technical report from Institute of Artificial Intelligence, Univ. of Georgia, 2008.

[Black09] Black, P.E., "Simulated Annealing", in *Dictionary of Algorithms and Data Structures* online, Paul E. Black, ed., U.S. National Institute of Standards and Technology. http://www.itl.nist.gov/div897/sqg/dads/HTML/simulatedAnnealing.html (March. 30 2009)

[WekaSnake09] Wikipedia.org (2009) *Snake-in-the-box.* http://en.wikipedia.org/wiki/Snake-in-the-box (June. 18 2009)

[WekaComExp09] Wikipedia.org (2009) *Combinatorial explosion.* http://en.wikipedia.org/wiki/Combinatorial_explosion (June. 18 2009)

[wikiSA09] Wikipedia.org (2009) *Simulated Annealing* http://en.wikipedia.org/wiki/Simulated_annealing (June. 20 2009)

[Eberhart07] Eberhart, R.C., Shi, Y., Computational Intelligence: Concepts to Implementations, *Morgan Kaufmann Publishers*, Burlington, MA, 2007.