USAGE OF GENETIC PROGRAMMING FOR SOLVING ROBOT NAVIGATION AND SYSTEM IDENTIFICATION PROBLEMS

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

ULAŞ BELDEK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

IN

THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2001

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Tayfur ÖZTÜRK
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Fatih CANATAN
Head of the Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Kemal LEBLEBİCİOĞLU
Supervisor

Examining Committee Members

Prof. Dr. Uğur HALICI

Prof. Dr. Kemal LEBLEBİCİOĞLU

Prof. Dr. Mustafa KUZUOĞLU

Assist. Prof. Dr. Lale ALATAN

Assoc. Prof. Dr. Faruk POLAT

ABSTRACT

USAGE OF GENETIC PROGRAMMING FOR SOLVING ROBOT NAVIGATION AND SYSTEM IDENTIFICATION PROBLEMS

Beldek, Ulaş

MS, Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. Kemal Leblebicioğlu

September 2001, 126 pages

Genetic programming is a newly improving phenomenon in recent years. This search technique is being applied to many different concepts ranging from medical researches to control applications. Two important concepts that the search technique is applied are system identification and evolution of rule based adaptive systems. A robot navigation problem is designed for investigating the performance of genetic programming for evolving a rule-based system. Some world patterns are planned, and four robots are inserted into the world patterns. These robots are given the tasks to go to the desired locations in the pattern world. A population of agents composing of rule based 'if then else' statements are constituted in order to control the actions of the robots. The designed agents are evolved to overcome the

difficulties they encounter through their tasks by the methods of genetic programming. Effect of communication between each other is investigated while they are performing their work. For system identification problem performance of genetic programming for identifying and modeling a second order, fourth order and non-linear dynamic systems is considered. For fourth order system, the influence of using steepest descent algorithm with genetic programming to system identification for parameter adjustment and tuning is examined

Keywords: Genetic programming, genetic algorithm, agents, rule-based computing, robot navigation, system identification and modeling, control theory.

ÖZ

ROBOT YÖNLENDİRME VE SİSTEM TANIMLAMA PROBLEMLERİNİN ÇÖZÜMÜNDE GENETİK PROGRAMLAMANIN KULLANIMI

Beldek,Ulaş

Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü

Tez Danışmanı: Prof. Dr. Kemal Leblebicioğlu

Eylül 2001, 126 sayfa

Genetik programlama son yıllarda yeni gelişmekte olan bir olgudur. Bu araştırma tekniği, tıbbi araştırmalardan kontrol uygulamalarına kadar değişen birçok alanda uygulanmaktadır. Araştırma tekniğinin uygulandığı iki önemli konu sistem tanımlama ve kural tabanlı adaptif sistemlerin evrilmesidir. Genetik programlamanın kural tabanlı bir sistemi geliştirmede performansının incelenmesi için bir robot yönlendirme problemi tasarlanmıştır. Bazı dünya modelleri planlanmış ve dört robot bu dünya modellerine yerleştirilmiştir. Bu robotlara dünya modelleri üstünde istenilen konuma gitme görevleri verilmiştir. Robotların hareketlerini kontrol etmek için, kural tabanlı 'eğer yoksa' ifadelerinden oluşan bir nesil ajan oluşturulmuştur. Tasarlanan ajanlar görevlerini yerine getirirken karşılaştıkları zorlukları aşmaları için genetik programlamanın yöntemleriyle

evrilmişlerdir. İşlerini gerçekleştirirken, ajanların aralarındaki haberleşmenin etkisi incelenmiştir. Sistem tanımlama probleminde, genetik programlamanın ikinci derece, dördüncü derece ve doğrusal olmayan dinamik sistemlerin tanımlanması ve modellenmesindeki performansı değerlendirilmiştir. Dördüncü derece sistem için, parametrelerin ayarlanmasında en aşağı azalma algoritmasının genetik programlama ile kullanılmasının sistem tanımlamaya etkisi incelenmiştir.

Anahtar Kelimeler: Genetik Programlama, genetik algoritma, ajanlar, kural tabanlı işleme, robot yönlendirme, sistem tanımlama ve modelleme, kontrol teorisi.

ACKNOWLEDGEMENTS

I would like to express my special thanks to my thesis supervisor Prof. Dr. Kemal Leblebicioğlu for his guidance throughout all stages of this study.

I like to thank my parents Hülya and Mustafa Beldek for their love, and my sister Gözde Arıkol and her husband Namık Arıkol for their support.

I wish to thank my dear friends Koray Akıllı, Vedat Nazik, Utkan Eryılmaz, Suphi Erden, Erdal Kılıç, Ali Erol, Adem Mulayim, Bilge Aydın, Cem Özgür Özdemir, Ayfer Özgür and Feyza Keçeli for their deep understanding.

Finally, I would like to thank everybody helping me in this work.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Historical Background and Literature Survey

In nature, biological structures that are more successful in grappling with their environments survive and reproduce at a higher rate. Biologists interpret the structures they observe in nature as the consequence of Darwinian natural selection operating in an environment over a period of time. In other words, structure is the result of fitness in nature [1].

Computer programs are among the most complex structures created by human. One of the central questions in computer science (attributed to Arthur Samuel in the 1950s) is how the computers can learn to solve problems without being explicitly programmed. In other words, how can computers be made to do what is needed to be done without being told exactly how to do it. Existing methods of machine learning, artificial intelligence, self-improving systems, self-organizing systems, neural networks and induction do not seek solutions in the form of computer programs. Instead these paradigms involve specialized structures, which are nothing like computer programs [1].

The answer to this question began to occur in the 1970s. John Holland's pioneering book *Adaptation in Natural and Artificial Systems* (1975) [14] provided a general framework for viewing all adaptive systems (whether natural or artificial) and then showed how the evolutionary process can be applied to artificial systems.

Any problem in adaptation can generally be formulated in genetic terms. Once formulated in those terms, such problems can often be solved by what is called a 'genetic algorithm'. But genetic algorithm was not sufficient to answer the problem totally. First of all although it can solve problems without being explicitly constituted, genetic algorithm was for constant length strings and the string structures were just sets of parameters, which were still away from computer programs. In 1980s need for more powerful representation for genetic algorithms has been recognized (De Jong [15-17]). At first, some new forms of algorithm appeared. These new algorithms used variable length strings and higher order representations [18-19]. After that, conditional statements, Boolean relations and Boolean operators were used [20] in a hierarchical way to constitute a tree like structure, which can be told a computer program. These hierarchical computer programs constituted the base for genetic programming. In 1990s genetic programming become another discipline of search method after these discoveries [1], [21]. Today genetic programming is a machine learning model which is applicable to various issues like pattern recognition, medical researches, control theory and so on. There are various search interfaces in Internet about genetic programming. One of the best web pages about genetic programming is in [29]. A genetic programming bibliography is available via anonymous ftp in [30]. There is also a mailing list in Internet [31]. Expert people about the issue can give information about related works about genetic programming.

One drawback about genetic programming is that continuous function evaluations take huge amount of time, which slows down the search method. Parallel processing of computers is important for decreasing time consumption. As the power and processing time of computers increase, genetic programming will become a more effective search technique in the future

In the literature, it is possible to find many applications of genetic programming. For example, Koza [1] investigated nearly all types of problems that genetic programming can be used. Angelina and Kinnear [2] opened a new argument about this issue and they deepened thoughts about the ways and styles the

genetic programming can be applied. [1] and [2] are two basic books for learning and understanding genetic programming.

When the two types of applications in this thesis are considered, most of the ideas for the first type were from Iba [3], which gives the fundamentals of evolutionary learning of communicating agents. The evolutionary strategies used depend on the basis of the strategies used in [4-10]. These strategies are main frameworks for the agents systems. Werner and Dyer [11] showed that cooperation is important in solving the multi-agent system problems. In [12] Ito, Iba and Kimura showed that robustness is another important feature of a program evolved by genetic programming. These important works constituted the background of the first application.

Second application depends on the work done by Gray, Smith, Li, Sharman and Weinbrenner [13]. Application of genetic programming to system identification of non-linear systems is considered in this work. Works related about the same subject can be encountered through [22-28].

1.2 The Thesis

In this thesis two applications of genetic programming is considered. The applications are chosen from two different subjects in order to investigate and compare the performance of genetic programming in these two different areas.

The first application is about evolving a rule-based system by the methods of genetic programming. Different world patterns are constituted and four robots are inserted into these structures. All the robots are controlled by different agents whose structures depend on rule-based 'if then else' statements. The 'if then else' statements are evolved according to the methods of genetic programming. Also the performance of the agents is considered if communication between agents is provided by some means or operations. Originality in this application comes from the method applied while performing genetic programming. An example of this problem is solved by Iba [3]. But evolved agents used vector information to

overcome the tasks given to them. In this work rule-based structures are evolved in order to overcome the tasks given. Another important aspect about the application is that the validation scenarios are chosen from various world patterns. Some validation scenarios even do not have any similarity with the training scenarios. Robustness of the trained agents is examined in different validation scenarios.

Second application is a system identification task. The performance of genetic programming for system modeling and identification is considered in this subject. As the order of the system to be identified increases or as the system begins to show non-linearity, optimization of system parameters becomes a must. Parameter tuning of the obtained systems by genetic programming constitutes the main body of this application.

The organization of the thesis is as follows. In Chapter 2 general information about genetic programming is given and methods of genetic programming are discussed. In Chapter 3, an evolving a rule-based system for the multi-agent control robot problem is implemented by genetic programming. Then in Chapter 4, genetic programming is applied to system identification tasks on different systems. Finally in Chapter 5 two applications discussed in Chapter 3 and in Chapter 4 are compared in sense of feasibility when genetic programming is applied to them.

CHAPTER 2

DEFINITION AND STRUCTURE OF GENETIC PROGRAMMING

2.1 What is Genetic Programming?

Genetic programming (GP) is a kind of search technique applicable on a special class of problems. Its structure depends on the model of genetic transition in the living organisms (Darwinian genetic reproduction principles). The search technique starts with composing an initial population of tree like structures called functions or S-expressions or individuals which are potential solution strategies for the particular problem deal with. Tree like structures are formed from two different kinds of objects. First kind of objects are called terminals which are the variables or constants for a particular S-expression, and second kind of objects are operations which are used to combine the terminals or other operations to form a meaningful function. After obtaining an initial population, a fitness value is assigned for all the S-expressions in the population according to their liability to solve the particular problem. Fitness of the S-expressions is the main driving force that provides the search to continue and reach a result. The S-expressions are ordered with respect to their fitness. A selection procedure is progressed according to fitness and some of the S-expressions are chosen to be used in genetic operations like reproduction, crossover and mutation. After applying genetic means new S-expressions are formed which constitute the new generation. New generation takes the place of the previous one. From this point same procedures are applied to the new generation

until a newer one is obtained. As generations pass liability of the S-expressions to solve the problem efficiently increases. The search is stopped when an S-expression in a generation produces a reasonable result for the problem or the search is terminated after a predefined number of generations.

The problems that can be solved by genetic programming are various. GP generally gives structural results or it defines a system or method of solution for a particular problem.

2.2 What is a Genetic Algorithm?

A genetic algorithm (GA) is a kind of optimization algorithm. As in genetic programming, it uses Darwinian genetic reproduction principles [1], [2]. A particular problem depending only on variables is formulated first. As in GP, genetic algorithm also starts with composing an initial population of parameters. Parameters formed in an initial population are constant or variable length strings (chromosomes), which are supposed to represent the problem parameters. Chromosomes formed by this way contain binary or numerical values that are called genes. From this point all the things told about GP is applicable for GA. Fitness assignment, selection, applying genetic means, obtaining a newer populations and stopping criteria are all in the same manner as in GP. But the structures and applied problem types are different. GA is generally used to solve nonlinear, discrete and continuous optimization problems depending on variables.

2.3 Differences between Genetic Programming and Genetic Algorithm

Although GP and GA use same methods and means of genetic operations like reproduction crossover and mutation, the structure of chromosomes and their meaning in both of them are entirely different.

In genetic programming the structures that undergo change by genetic operations are tree like S-expressions that try to find a method of solution for a

particular problem. A result (this can be a value, an action or any kind of numerical or symbolic sign) is obtained for every S-expression once they are evaluated by a procedure determined by a predefined function evaluation style. This result leads to a fitness value for that S-expression. S-expressions are the basic structures in genetic programming that the entire search is based on [1].

In genetic algorithms the structures that undergo adaptation are strings that contain different genes, which correspond to different variables of a particular problem. The variables are represented as either binary or real coded strings. A fitness value is assigned to every string (chromosome). Fitness assignment is performed on every member of a population, which holds the current solution candidates of that particular problem.

GP and GA differ in the way they give a solution to a particular problem. The result of GA is a variable set while GP gives a procedure or style of solution composed of operations and terminals.

2.4 Closure and Sufficiency of Operation and Terminal Sets

S-expressions (functions) must obey some norms depending on particular problem involved. If one is trying to solve a mathematical problem, mathematical operations like addition, subtraction, multiplication and division has to be used in S-expressions. Some other specialized mathematical operations like rooting or absolute value evaluation can also be applied but using gate structures is an illogical thing since the terminals used in a mathematical problem are unsuitable to be an argument of gate structures. If one is trying to find a solution for a problem about a logical structure like the construction of a multiplexer, the associated operation set should not contain mathematical operations. It had better contain some gate structures like 'And', 'Or', 'Nand', 'Nor', 'Not' and maybe 'Xor'. Some 'if' statements may also be suitable for these kinds of problems. The operations used in a problem must be well defined and closed for any combination of arguments that it may encounter. An addition operation taking two arguments can't take Boolean arguments like true or false or an 'And' gate can't take numerical values as

arguments. This property is called the closure property of the operation and terminal sets.

The set of terminals and operations must be capable of expressing a solution to the problem encountered. Problem should be investigated and such terminal and operation sets be formed that the combination of operations and terminals should have the capability to yield a desired solution for the problem. This issue is called the sufficiency property of operation and terminal sets.

2.5 Initial Structures

In the initial population, S-expressions are randomly generated by ordered branch trees from the terminal and operation sets. Firstly, an operation from the operation set is chosen. This first operation is used as the root of the S-expression. Suitable number of branches is connected to the root. If an addition operation is selected as the root, the number of branches that must be connected to the root (first node) must be two since addition takes two arguments. After branch connecting operation, new nodes are attached to the branches appropriate to the number of branches. These new nodes may consist of either new operations or terminals. If a node contains an operation once more, the procedure mentioned above goes on recursively. If a terminal is encountered at a node, no further attachment of branches is done to that node and that node becomes an end point. As the result of this randomly selection procedure a tree structure can be obtained. Figure 1 shows a typical tree structure for the mathematical relation $\frac{a}{b}+(0.9*c)$.

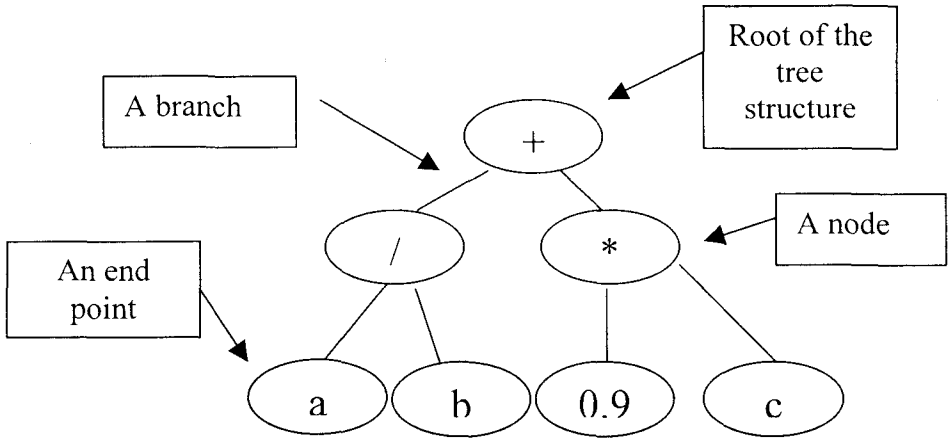

Figure 1: A typical tree structure ('a', 'b', 'c' and '0.9' are terminals, '+', '/' and '*' are operations.

This generation process can be implemented in several ways. One method is 'full' method; the other method is 'grow' method. The depth (tree length) of a tree is called the longest non-inverted path from the root to the end points [1] (the direction from root to the end points never changes. It is always from up to down). All the trees generated by 'full' method must have all the lengths from endpoints to root equal to the specified depth of the tree. As a result of 'full' method trees having equal lengths from all end points to the root are obtained. The 'grow' method depends on generation of trees with variable lengths from the root to the end points. By this method non-symmetric tree structures can be obtained. Generally for initial populations a mixture of 'full' method and 'grow' method is used. In this method half of the S-expressions are generated by 'full' method and half of the S-expressions are generated by 'grow' method. This mixed method is called 'ramped half and half' [1] generative method. For example, If an initial population consisting of 200 individual S-expressions is preferred and the maximum specified depth is 6 and 'ramped half and half' generative method is used, there will be 100 functions generated by 'full' method and 100 functions generated by 'grow' method. 20 of the functions generated by full method will be of tree length 2, 20 of them will be of tree length 3, and so forth up to length 6 [1].

Duplication of S-expressions is not a desirable thing in the initial population. This decreases the genetic diversity of the initial population. Each newly created S-expression must be checked before inserting it into the initial population [1]. Generation process continues until pure diverse functions are obtained. In later generations, creation of duplicate individuals via the genetic operation of reproduction is an inherent part of genetic programming.

2.6 Representation and Evaluation of S-expressions

After the initial population is obtained, it is time for the evaluation of S-expressions in the population. Usually the S-expressions can be represented as variable length strings. Added to this, a vector or vectors are used in order to show the content of the variables used in the string. Figure 2 shows a typical tree structure of mathematical expression.

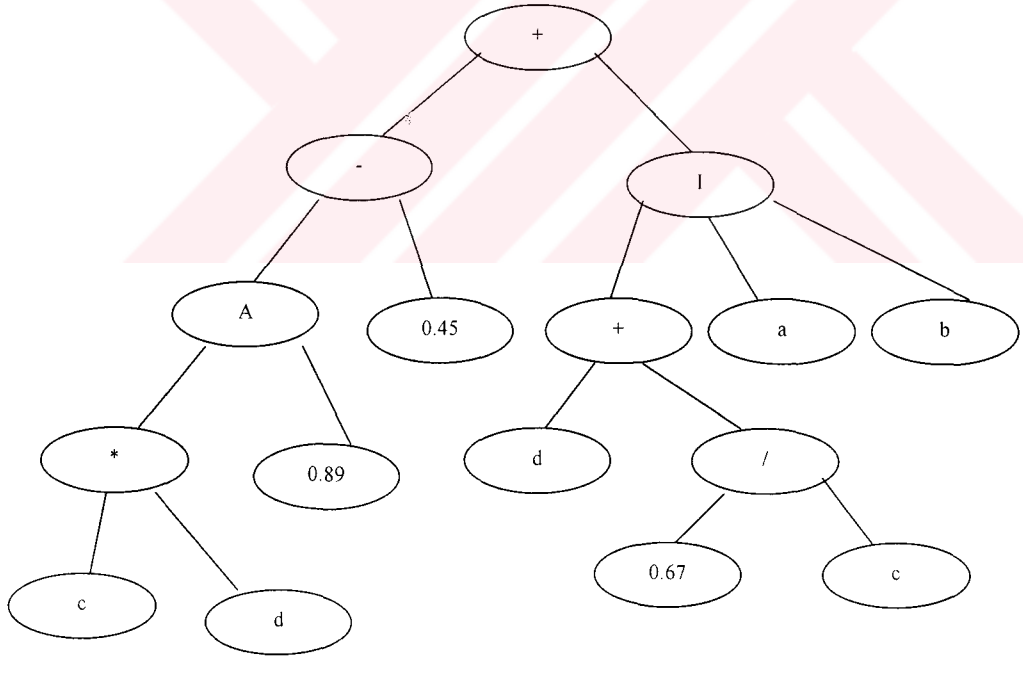

Figure 2: A typical tree structure of mathematical expression

This tree represents a basic mathematical function composing of addition, subtraction, multiplication, division, comparison with zero (represented by 'I') and arctangent operation (represented by 'A'). It also has some constants and terminals. The constants can't be changed while the terminals can take various values. This tree structure is represented as a string vector as in Example 1.

(+, -, A, *, c, d, e, e, I, +, d, /, e, c, a, b)

Example 1: A typical tree structure representation of a mathematical equation in string form (all the constants are represented by the letter 'e').

As seen the string is formed according to a hierarchy where upper and left elements have higher priority to be written first. To keep the constants and terminal values in hand, a vector of numbers must be structured. The vector for the string in Example 1 is as in Example 2.

(0, 0, 0, 0, 2, 4, 0.89, 0.45, 0, 0, 4, 0, 0.67, 2, 0, 1)

Example 2: The vector keeping the constants and terminals of the string ('0', '1', '2', '4' are the particular values used instead of 'a' 'b' 'c' and 'd' respectively. The operations '+', '–' , '*', '/', 'A' and 'I' are represented as '0' in the vector. The value of constants are put instead of 'e')

The string is evaluated and a value is obtained as the result of the string. This is done in the following manner. If the number of arguments that an operation in the string is followed is equal to or greater than the number of arguments it takes, the operation is evaluated with the necessary number of arguments and the vector and the string are shortened by the size of arguments. This procedure goes on until all the string is evaluated and a value for the string is obtained. For example consider a string and its value vector in Example 3.

String vector: (+, -, A, *, c, d, e, e, I, +, d, /, e, c, a, b)

Value vector: (0. 0. 0. 0. 2. 4. 0.89. 0.45. 0. 0. 4. 0. 0.67. 2. 0. 1)

Example 3: Representation of a string and its value vector

Consider multiplication operation. It takes two arguments and it is followed by two terminals 'c' and 'd'. So 'c' and 'd' are multiplied. String and value vector are shortened accordingly. The last argument is written in place of '*' but the content of the vector at that position takes the value obtained as the result of multiplication of these terminals. The new string and value vector obtained as the result of this shortening procedure are as in Example 4.

(+, -, A, **d**, e, e, I, +, d, /, e, c, a, b)

(0, 0, 0, **8**, 0.894, 0.45, 0, 0, 4, 0, 0.67, 2, 0, 1)

Example 4: Evaluation of an operation in a string and obtained new string and value vector as the result

As seen, the value of d is not 4. Instead it is 8. This evaluation procedure is done until all the string is evaluated and only one value is obtained for the string.

All the functions (S-expressions) in a population are evaluated in order by some evaluation criteria depending on the particular problem like above. A value, a vector of values or an action can be obtained as the result of evaluation procedure for each S-expressions depending on the problem. The results of S-expressions are considered according to their tendency to solve the particular problem. Some of the S-expressions are more liable to give better results and are more suitable to yield a desirable solution to the problem. Better S-expressions must have more chance to

be used in genetic operations of proceeding populations to provide effective continuation of the search technique. So the term fitness comes to the stage.

2.7 Fitness and Selection

Fitness is a kind of measurement criteria for S-expressions showing how much the evaluated S-expressions are close to the desired correct answer. The most common approach to measuring fitness is to create an explicit fitness measure for each S-expression in the population. Each individual is assigned a scalar fitness value by means of some well-defined explicit evaluation procedure [1]. The survival of S-expression or its counterparts depend mainly on its fitness values with respect to other S-expressions.

Mainly there are four kinds of fitness measures. Raw fitness is the fitness that is stated in the natural terminology of the problem [1]. For example, if a problem in which an ant tries to find some food in a closed region by restricting the food number to number A (A>0) is considered, our raw fitness must vary between 0 and A [1]. When raw fitness is error, the raw fitness of an S-expression is the absolute value of the difference between the returned value by the S-expression and the correct answer. The number of mismatches is an indicative of raw fitness. Standardized fitness is a fitness type, which can be described by the Equation 1 [1].

$$s(i) = R_{\max} - r(i) \qquad (1)$$

In the equation above $s(i)$ is the standardized fitness, R_{max} is the maximum row fitness and $r(i)$ is the row fitness for i'th S-expression. As it can be understood from the equation, the smaller the standardized fitness the better the S-expression suits to solve the problem. Adjusted fitness is formulated as in Equation 2 [1].

$$a(i) = \frac{1}{(1 + s(i))} \qquad (2)$$

In Equation 2, $s(i)$ is the standardized fitness and $a(i)$ is the adjusted fitness for the i'th S-expression. Adjusted fitness lies between 0 and 1 for all the

individuals. It has a bigger value for better individuals. Adjusted fitness generally selects better individuals because it exaggerates the fitter S-expressions. Normalized fitness of an individual can be derived by dividing an individual's adjusted fitness by the sum of all the individuals' adjusted fitness. It can be formalized as in Equation 3 [1].

$$n(i) = \frac{a(i)}{\sum_{k=1}^{M} a(k)} \qquad (3)$$

In Equation 3, *n(i)* is the normalized fitness *a(i)* is the adjusted fitness for the i'th S-expression and *M* is the number of individuals in the population (population size). Normalized fitness ranges between 0 and 1. It is larger for better individuals. The sum of the normalized fitness values of all individuals is 1. Usually normalized fitness is used in selection methods where exaggeration of the better individuals is not such a desired issue.

Selection procedure is a preliminary work for the primary genetic operations and it is usually made according to some issues depending on the fitness of the S-expressions. Generally fitness-proportionate selection is used in most of the problems. Fitness-proportionate selection is selection of individuals according to the normalized fitness of the S-expressions. If selection of better individuals is desired to be exaggerated in the population, adjusted fitness is better

There are some alternative selection methods used too. Rank selection depends on the rank (not the numerical value) of the S-expressions. The population is divided into ranks. Individuals are selected according to some criteria about the ranks they are found. Rank selection eliminates the dominating effects of the comparatively high-fitness individuals [1]. In tournament selection two individuals are selected randomly. Then the individual with better fitness is used for primary genetic operations.

2.8 Primary Genetic Operations

2.8.1 Reproduction

Reproduction operates on a single individual. As the result one offspring is produced having the same structure as the S-expression. There are two steps for this operation. Firstly a parental S-expression is selected from the population according to some selection method based on fitness. Then the selected S-expression is copied from the current population to the new population without any alteration in the S-expression.

There is one special kind of reproduction in which the best individual according to fitness is transferred into new generation. This operation is called 'reproduction of the best of generation'. This operation guaranties that the next generations' best individual will not have a fitness value worse than the previous generations' best individual if there are no random processes in the definition of the problem. Best of generation operation can be applied to more than one individual sometimes. The other reproduced individuals are selected by some other selection criteria. Generally reproduction is applied to % 10 of the current population. As a result % 10 of the new population is formed from the individuals of previous population [1].

2.8.2 Crossover

Crossover is the second primary operation in genetic programming. It supplies variation in the population [1]. Two S-expressions, called parents are selected according to a selection method. These two parents change their parts randomly. One point is selected randomly from each S-expression. It is usually done in a way that the selected point is sometimes a terminal but more often an operation (In the applications, the probability of the selected point to be a terminal is 0.1). After that, these points are broken from the tree like structures completely with the following nodes and branches. The broken parts are then connected to the other S-expression's broken point. As a result two different individuals are formed from two different S-expressions. % 90 of the individuals in new generation are

produced by crossover generally. Figure 3 shows two parent individuals and Figure 4 shows two newly obtained offspring from these parents by crossover operation.

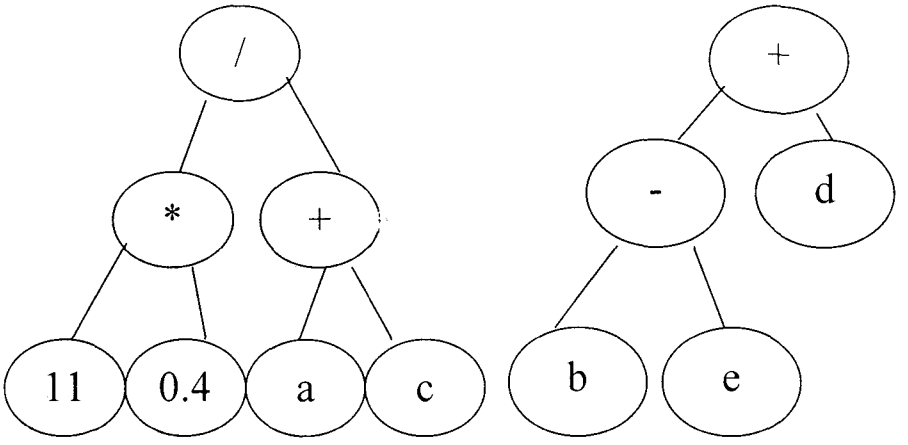

Figure 3: Two individuals before the crossover operation. The crossover point is '*' for the individual at the left side and '-' for the individual at the right side.

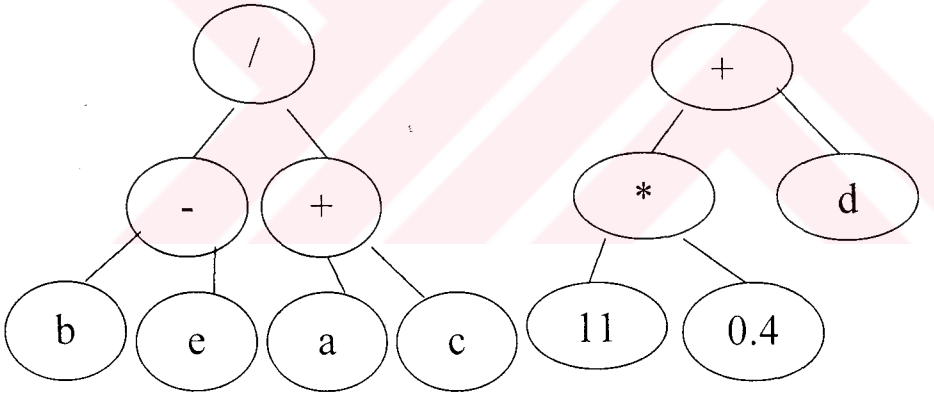

Figure 4: Two new individuals obtained as the result of crossover operation.

2.9 Secondary Genetic Operation

2.9.1 Mutation

Mutation is a secondary operation in genetic programming. It effects the variation in the population by just changing the genetic code of a chosen parental individual partially. The partial change in the parent can be on a terminal or on a branch. As a result a new individual is obtained from the parent. It is important

because it can add new genes into the population that can change the direction of the search [1]. It helps the convergence of the search by changing the direction of the search out from the localities but it is used seldom. Figure 5 represents an individual before mutation and Figure 6 shows same individual after mutation.

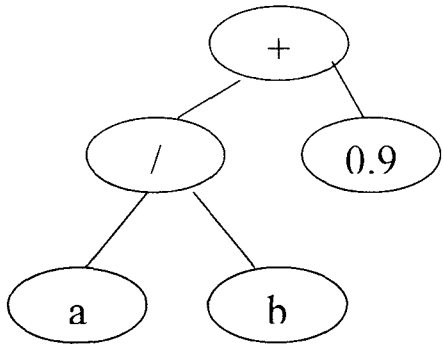

Figure 5: An individual before muatation operation. The mutation point is 0.9

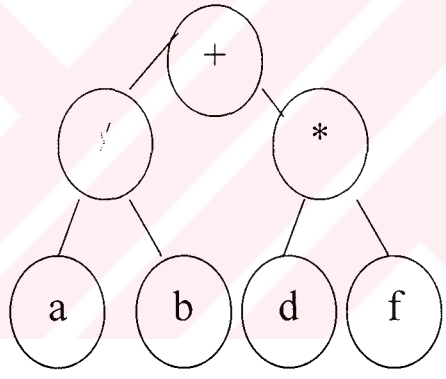

Figure 6: The same individual after mutation operation

2.9.2 Addition of New Individuals to the Population

After generations the genetic code of the population becomes insufficient to reach a desired solution for some problems. For such situations new genes must be inserted into the population. One way to overcome this difficulty is reinsertion of new individuals to the population while deleting (killing) some other individuals. Usually the deleted individuals are from the worse ones. There is no guaranty that the new individuals are better than the deleted ones, but it is sure that they have more variety in their structure than the deleted ones since genetic diversity

decreases as generations increase. This operation has the same effect with mutation. If the search converges to a locality, this operation may help to change the direction of the search and a desired solution may be obtained. It is seldomly used in problems.

2.10 General Flow of Genetic Programming

A search with genetic programming starts with composing an initial generation. It is advisable to form half of the S-expressions with 'full' method and half of the S-expressions with 'grow' method. As a result an initial population of totally different shaped and sized S-expressions are produced with varying tree lengths. These S-expressions are evaluated. They are ordered according to their fitness. If fitness criteria are enough to end the program, program is terminated, if not reproduction and crossover is made and a new generation is obtained. The same procedure is evaluated over the new generation and the program goes on until fitness criteria are satisfied. At the end of the program the most suiting function (code or chromosome) is given as output. Figure 7 shows a block diagram about the general flow of the search technique.

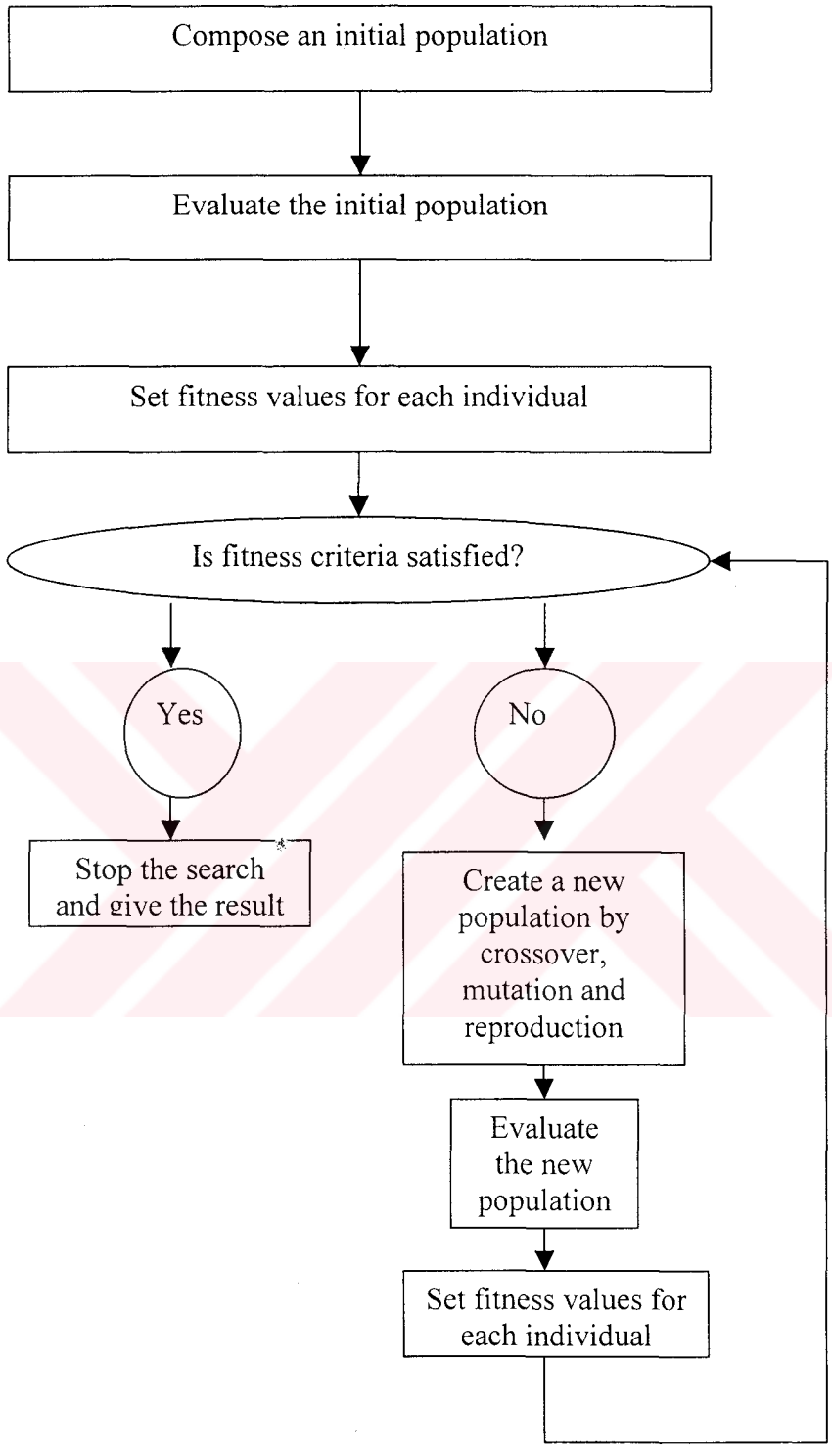

Figure 7: General Flow of Genetic Programming

CHAPTER 3

ROBOT NAVIGATION PROBLEM USING GENETIC PROGRAMMING

3.1 General

Genetic programming and its variants have been applied to multi-agent learning ([1], Ch. 12; [3-10]). Robot navigation problem can be solved in two different ways by using genetic programming via using agent systems. In the first type, all the robots are controlled by different agents. These different agents constitute S-expressions, which evolve according to the rules of genetic programming separately, which means an S-expression contain different parts for all these agents. In the second type, there is only one agent for all robots. This single agent constitutes the S-expression itself. So only one agent is responsible from the control of all the robots. The first type is called heterogeneous breeding while the second one is called homogenous breeding. Another aspect affecting the performance of the search is the communication or the way of communication between robots. First of all, the performance of heterogeneous breeding for non-communicating and communicating case is considered for different training situations encountered. Secondly heterogeneous breeding strategy with communication and homogenous breeding strategy with and without communication are used to overcome the problems encountered, but this time the training times are doubled and robots' starting positions for the doubled training

situations are changed and effect of increasing the duration for the simulations is investigated. Lastly new kinds of movements and communication ideas are tried in order to increase the robustness of the breeding strategies.

3.2 Communication Types

There are mainly three types of communicative relation between agents [3]. In communicating agents, one agent is capable of requesting data from another agent. In negotiating agents, in addition to data request, agents can negotiate with each other about their movements. In controlling agents an agent has control over other agents. In robot navigation task, communication commands such as SEND and RECEIVE are used so that the agent can signal what it sees to a remote agent and sometimes can tell it to stop or move accordingly.

3.3 Robot Navigation Problem 1 (Applying Heterogeneous Breeding Strategy with and without Communication)

The world of robot navigation consists of a rectangular grid world on which agents (A1, A2, A3, A4) and some obstacles (#) can be placed. Each object occupies a cell of the grid world. Agents can move up, down, left, right or diagonally unless they reach the world's boundaries or encounter with obstacles. The agents' goal is to find the optimal path in the grid world from given locations to their respective goals. In this problem four agents have to pass through a narrow aisle to reach their goals in training. Some problems occur when two agents try to move against in opposite directions. In some cases they can block each other. An example of a grid world is given in Figure 8.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1									A2	#
#											#
#											#
#	#	#	#	#	#	#	#	#			#
#	#	#	#	#	#	#	#	#			#
#	#	#	#	#	#	#	#	#			#
#	#	#	#	#	#	#	#	#			#
#	#	#	#	77#	#	#	#	#			#
#											#
#	A4									A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 8: A world structure with robots and obstacles.

3.3.1 Robot Navigation Problem via Using Genetic Programming by Applying Heterogeneous Breeding Strategy without Communication

3.3.1.1 Terminals, Operations and Evaluation of a Tree

Different kind of rule base 'if then else' operations are chosen to represent each agent to gain more precise information about movement type and the direction to go. The terminals are some action types capable of moving the robots. Combination of terminals and operations meaningfully constitutes an agent's tree structure. Combination of four agents constitutes an individual structure. Tree structure of an agent returns an action for the robot it is controlling. All robots move to other positions with this action type. As a result, the position and the direction of the robots change. Initially the robots' positions and directions are set. But the initial directions for robots are chosen such that they are not blocked by borders of the world in forward direction for their first movement. Below is the list of all terminals (actions) and operations ('if than else' rules) when no communication between the agents is provided.

3.3.1.1.1 Terminals (actions) for non-communicating case

1- a: go forward one unit if possible and stop, set the direction to moved direction.

2- b: go backward one unit if possible and stop, but do not change the direction.

3- c: turn clockwise 90 degrees and go one unit if possible or stop, set the direction to moved direction.

4- d: turn counterclockwise 90 degrees and go one unit if possible or stop, set the direction to moved direction.

5- e: stop and set the direction to current direction.

6- f: move in one direction randomly and set the direction to moved direction.

7- g: move in the direction of the goal if possible and set the direction to moved direction (this action can move the agent diagonally).

3.3.1.1.2 Operations (If then else statements)

1- If there is an obstacle in front (symbolized by '+' in the simulations): This operation takes two arguments. If there is an obstacle in front of the agent it returns its first argument else it returns its second argument

2- If there is another agent in front (symbolized by 'A' in the simulations): This operation takes two arguments. If there is another agent in front it returns its first argument else it returns its second argument.

3- If the agent is in the aisle (symbolized by 'B' in the simulations): This operation takes two arguments. If the agent is in the aisle the operation returns its first argument else it returns its second argument.

4- If the goal is forward (symbolized by 'F' in the simulations): This operation takes two arguments. If the agent and its goal are in the same x-component or in the same y-component, it executes its first argument else it returns its second argument.

5- If forward free (symbolized by 'R' in the simulations): This operation takes two arguments. If the cell in front of the agent is free then it returns its first argument else it returns its second argument.

3.3.1.1.3 Evaluation and Representation of a Tree Structure for an Agent

Evaluation of a tree gives us an action as a result since all the tree structures are formed from 'if then else' statements and movement actions. This action moves the agents and sets their direction. Figure 9 shows a tree structure for an agent

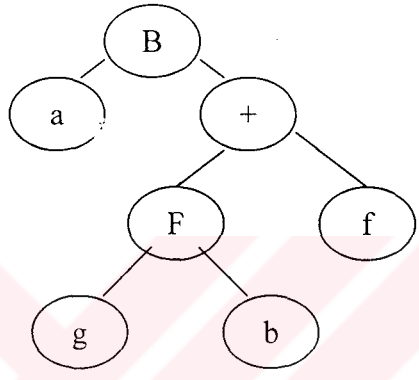

Figure 9: A tree structure for an agent.

Below is the representation of this agent in string form in the simulation

Agent: (B,a,+,F,g,b,f)

3.3.1.2 Fitness Considerations

The fitness of a genetic programming individual for a world scenario is determined by three main principles in this problem. Firstly, a high score is assigned to a GP individual, which moves a robot to its goal. Secondly, a higher score to a GP individual is assigned, which finishes the tasks quickly. Lastly, for any robots not reached its goal after the execution of GP individual, a higher score is assigned when they have been moved nearer to their goals. To fulfill the request following seven procedures are determined in computer simulation.

Step-1: Set $Time = 51$, $Fitness = 0$.

Step-2: Evaluate the tree and move agents accordingly.

Step-3: If an agent reaches its goal, then

$$Fitness = Fitness + Bonus * FT \quad (4)$$

FT is the number of agents that have reached the goals at this step.

Step4: If all of the agents are moved to their goals, then

$$Fitness = Fitness + Speed\_up * Time\ . \quad (5)$$

Return fitness.

Step-5: $Time = Time - 1$.

Step-6: If $Time = 0$ then

$$Fitness = Fitness + Cr * \sum_{ag \in AG} \{d(st(ag), gl(ag)) - d(cr(ag), gl(ag))\} \quad (6)$$

AG is the set of remained agents.

Step-7: Go to Step-2.

In Step-6 '$d(x, y)$'means distance between x and y. '$st(ag)$', '$cr(ag)$' and '$gl(ag)$' are the original, current and goal position for a robot Ag. Thus, the equation '$\{d(st(ag), gl(ag)) - d(cr(ag), gl(ag))\}$' equals to the moved distance of the robot 'ag' toward its goal [3]. The 'Bonus', 'Speed_up' and 'Cr' parameters are chosen 3000, 80 and 100 respectively. The population consists of 100 individuals. Reproduction and crossover rates for the population are % 10 and % 90 respectively. After every 10-generation 30 new individuals are inserted to the population and the worst 30 individual are deleted from the population.

3.3.1.3 World Scenarios and Evaluation Principles

There are four different world scenarios chosen in the problem for training. For all the scenarios, robot 1 have to go to the place of robot 3, robot 2 have to go to the place of robot 4, robot 3 have to go to the place of robot 1 and robot 4 have to go to the place of robot 2. The scenarios differ in the way that the

narrow aisle the agents have to cross in order to reach their goals is in different parts of the grid world for all scenarios. The four training scenarios are at Figure 10, Figure 11, Figure 12 and Figure 13.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1									A2	#
#											#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#											#
#	A4									A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 10: First training scenario. A1 and A3 will change place and A2 and A4 will change place.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1									A2	#
#											#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#											#
#	A4									A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 11: Second training scenario. A1 and A3 will change place and A2 and A4 will change place.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1									A2	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A4		#	#	#	#	#	#		A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 12: Third training scenario. A1 and A3 will change place and A2 and A4 will change place.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1		#	#	#	#	#	#		A2	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A4									A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 13: Fourth training scenario. A1 and A3 will change place and A2 and A4 will change place.

Adding the fitness of these scenarios and dividing the sum by four gives an average fitness for an individual. Fitness-proportionate selection is used to select the chromosomes according to their average fitness values. If the average fitness value is over 12000 for an individual the program terminates since the fitness value is enough to say that nearly all the tasks are completed. In case the program never terminates the best individual and its average fitness values are saved for every

generation. With non-communicating case it is seen that the program doesn't end in three days. The searching method reaches an average fitness value between 8000 and 9000 for best individuals. When the best individual structures are examined for their rate to overcome navigation tasks it is seen that none of the navigation tasks are fulfilled completely but nearly 2 or 3 robots are able to complete their tasks in their worlds. Most important problem encountered in the scenarios is that sometimes the robots go in the opposite direction in the aisle and can block themselves causing non-termination of the search (see Figure 14).

#	#	#	#	#	#	#	#	#	#	#	#
#	A3										#
#											#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#	A1	#
#	#	#	#	#	#	#	#	#	#	A4	#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#											#
#	A2										#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 14: Two robots blocking each other. Robots A1 and A4 blocked each other and they can't reach their goals.

Another important problem is that walls can block some robots; as a result the agent can't fulfill its tasks (see Figure 15).

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#	A2										#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#											#
#										A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 15: Wall blocking a robot. Robots A2 is not able to reach its goal since the wall blocked it. A2 cannot reach to its goal

The best individual trained sometimes shows different performances from one run to another but generally the results obtained are not so different. The difference is because of the existence of 'random movement' terminal in the genes. The same program sometimes gives different fitness values for different runs.

The world scenarios chosen for training the agents are among the most difficult ones. First of all, the aisles in all the training scenarios are so small that only one robot can pass through. Secondly there is only one passage to the goals for all the robots. In addition, the wall may block some robots. The performance of the agents is measured in world scenarios, which are different from the training cases. These scenarios are called validation scenarios. The world scenarios are chosen such that sometimes the shapes of the grid worlds are similar but the size of the aisle and the walls are different. Sometimes there are two or three aisles but the way that must be passed to reach the goals is narrow and sometimes the world contains different wall structures in order to obstacle the robots. So robustness of the strategies evolved by genetic programming are practiced in these validation scenarios [12]. Validation scenarios are in Figures 16-27.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1									A2	#
#											#
#											#
#	#	#	#	#	#	#	#	#			#
#	#	#	#	#	#	#	#	#			#
#	#	#	#	#	#	#	#	#			#
#	#	#	#	#	#	#	#	#			#
#	#	#	#	#	#	#	#	#			#
#											#
#	A4									A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 16: First validation scenario.A1 and A3 will change place and A2 and A4 will change place.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1									A2	#
#											#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#											#
#	A4									A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 17: Second validation scenario.A1 and A3 will change place and A2 and A4 will change place.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1		#	#	#	#	#	#		A2	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#											#
#	A4									A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 18: Third validation scenario.A1 and A3 will change place and A2 and A4 will change place.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1									A2	#
#											#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#	A4		#	#	#	#	#	#	#	A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 19: Fourth validation scenario.A1 and A3 will change place and A2 and A4 will change place.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1									A2	#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#	A4									A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 20: Fifth validation scenario.A1 and A3 will change place and A2 and A4 will change place.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1									A2	#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#											#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#	A4									A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 21: Sixth validation scenario.A1 and A3 will change place and A2 and A4 will change place.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1									A2	#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#	A4									A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 22: Seventh validation scenario.A1 and A3 will change place and A2 and A4 will change place.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1									A2	#
#											#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#											#
#	A4									A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 23: Eighth validation scenario.A1 and A3 will change place and A2 and A4 will change place.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1									A2	#
#											#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#											#
#	A4									A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 24: Ninth validation scenario.A1 and A3 will change place and A2 and A4 will change place.

#	#	#	#	#	#	#	#	#	#	#	#
#	A1									A2	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#											#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A4									A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 25: Tenth validation scenario.A1 and A3 will change place and A2 and A4 will change place.

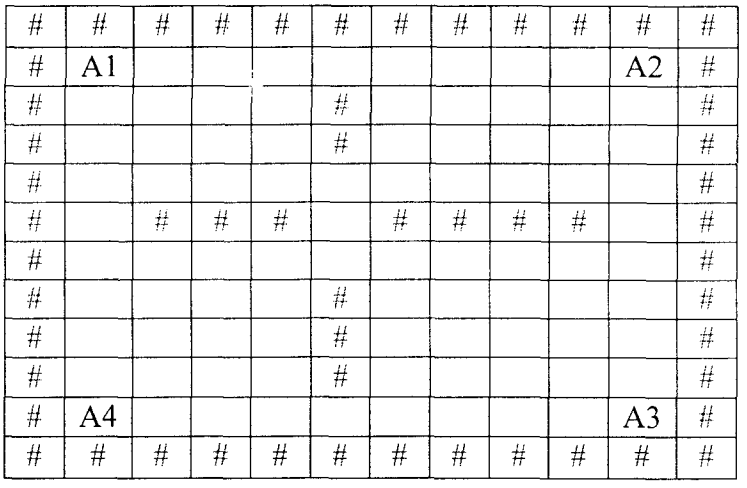

Figure 26: Eleventh validation scenario.A1 and A3 will change place and A2 and A4 will change place.

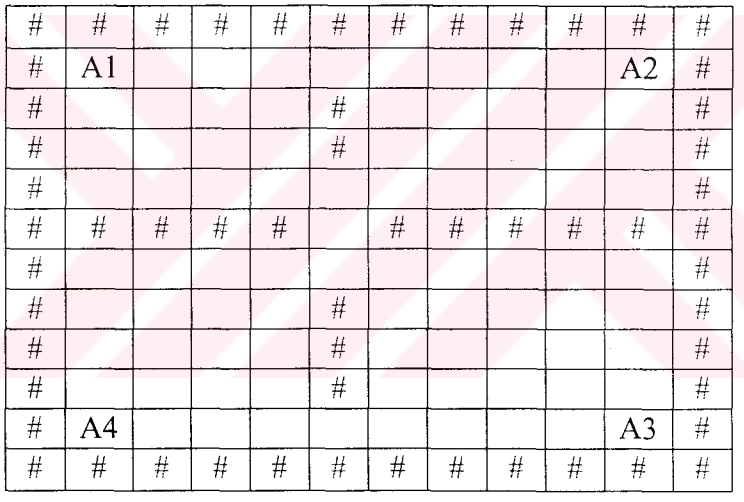

Figure 27: Twelfth validation scenario.A1 and A3 will change place and A2 and A4 will change place.

3.3.1.4 Results for Non-Communicating Case

There are several obtained results but none of the results are able to fulfill all the tasks in all the training cases. Among all the results obtained, some show very good results for training cases. The structures of four agents in one of the best

runs and average fitness value for the individual program formed by these four agents are as follows:

Agent1: '+fAgA++ffRafRaf'

Agent2:'AbAb+g+F++F+fa+AagAFfgAbggaAbAb+g+F++Fbgaa+AagFfaaa'.

Agent3: 'RRRRRaggggBBBgRRagggg'

Agent4: 'AAa+bg+Fgba'

Average fitness: 9371.8

The agents' responses in the training scenarios are shown in Figures 28-31.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#	A2										#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#											#
#										A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 28: Response of the agents to the first training scenario for non-communicating case. A2 couldn't reach its goal while other agents overcame their tasks.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3										#
#											#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#										A4	#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 29: Response of the agents to the second training scenario for non-communicating case. A4 couldn't reach its goal while other agents overcame their tasks.

#	#	#	#	#	#	#	#	#	#	#	#
#			#	#	#	#	#	#	A3	A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 30: Response of the agents to the third training scenario for non-communicating case. A3 couldn't reach its goal while other agents overcame their tasks.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3										#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2	A4	#	#	#	#	#	#		A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 31: Response of the agents to the fourth training scenario for non-communicating case. A4 couldn't reach its goal while other agents overcame their tasks.

As seen from the responses at all scenarios three of the robots are able to complete their tasks while one of the robots just hits the wall and cannot complete its task. The responses of the agents differ from one run to another because there are random movements in their control strategies.

When the trained agents are practiced for validation, good results are obtained for similar cases since the training scenarios are more complicated than the validation scenarios with similar wall structures. But this establishment is not true for non-similar world scenarios. The responses of agents to the validation data are shown in Figures 32-43.

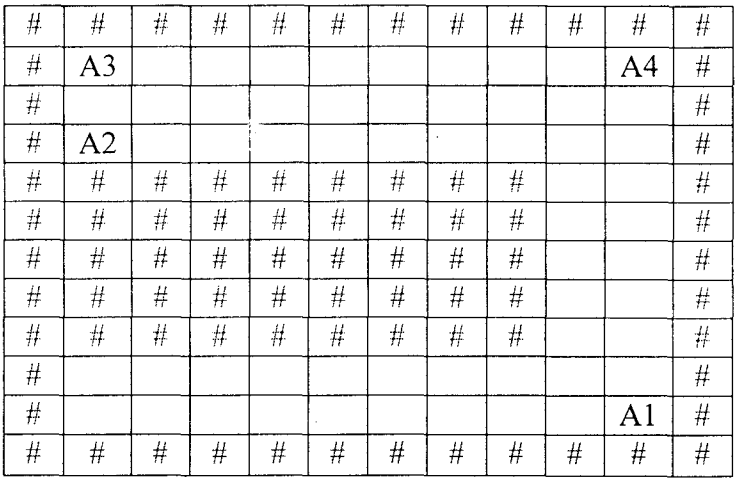

Figure 32: Response of the agents to first validation scenario for non-communicating case (Fitness=9572.8).

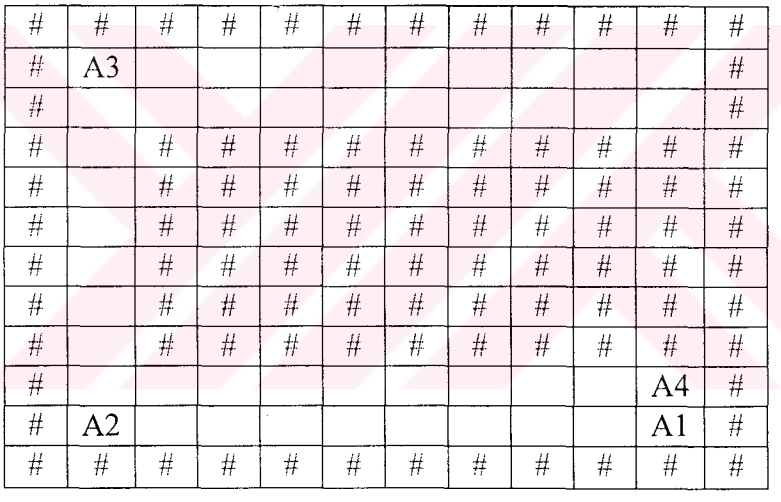

Figure 33: Response of the agents to second validation scenario for non-communicating case (Fitness= 9472.8).

#	#	#	#	#	#	#	#	#	#	#	#
#			#	#	#	#	#	#	A3	A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#											#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 34: Response of the agents to third validation scenario for non-communicating case (Fitness=9472.8).

#	#	#	#	#	#	#	#	#	#	#	#
#	A3										#
#											#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#	A2	A4	#	#	#	#	#	#	#	A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 35: Response of the agents to fourth validation scenario for non-communicating case (Fitness=9068.6).

As seen from the Figures 32-35 good results are obtained for validation scenarios, which resembles the training ones. As in the training scenarios only one robot is not able to overcome its task in first four validation scenarios while other robots overcome their tasks. For the validation scenarios, which are totally different from the training scenarios poorer results are obtained. But these scenarios are so complicated and the passageways are too narrow to allow crossing of two robots at

the same time. This is what is observed at Figures 36-38 for validation scenarios 5-7 for non-communicating case.

#	#	#	#	#	#	#	#	#	#	#	#
#									A1	A2	#
#		#	#	#	#	#		#	#	A3	#
#		#	#	#	#	#		#	#	A4	#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#											#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 36: Response of the agents to fifth validation scenario for non-communicating case (Fitness=1807.3).

#	#	#	#	#	#	#	#	#	#	#	#
#									A1	A2	#
#		#	#	#	#	#	#	#	#	A3	#
#		#	#	#	#	#	#	#	#	A4	#
#											#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#											#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 37: Response of the agents to sixth validation scenario for non-communicating case (Fitness=1807.3).

#	#	#	#	#	#	#	#	#	#	#	#
#									A1	A2	#
#		#	#	#	#	#	#	#	#	A3	#
#		#	#	#	#	#	#	#	#	A4	#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#											#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 38: Response of the agents to seventh validation scenario for non-communicating case (Fitness=1807.3).

Some improvement is observed for the validation scenarios 8-12 for non-communicating case. But still the fitness values are not at desired levels for all the scenarios remaining. Figure 39-43 shows response of the agents for scenarios validation scenarios 8-12.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#	A2		A1								#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#											#
#											#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 39: Response of the agents to eighth validation scenario for non-communicating case (Fitness=6682).

#	#	#	#	#	#	#	#	#	#	#	#
#	A3										#
#											#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#	A1	#
#		#	#	#	#	#		#	#	A4	#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#											#
#	A2										#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 40: Response of the agents to ninth validation scenario for non-communicating case (Fitness=7545).

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#											#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 41: Response of the agents to tenth validation scenario for non-communicating case (Fitness=13440).

#	#	#	#	#	#	#	#	#	#	#	#
#	A1									A2	#
#					#						#
#					#						#
#											#
#		#	#	#		#	#	#	#		#
#											#
#					#						#
#					#						#
#					#						#
#	A4									A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 42: Response of the agents to eleventh validation scenario for non-communicating case (Fitness=13440).

#	#	#	#	#	#	#	#	#	#	#	#
#										A1	#
#					#						#
#					#						#
#	A2										#
#	#	#	#	#		#	#	#	#	#	#
#	A3									A4	#
#					#						#
#					#						#
#					#						#
#											#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 43: Response of the agents to twelfth validation scenario for non-communicating case (Fitness=1800).

3.3.2 Robot Navigation Problem via Using Genetic Programming by Applying Heterogeneous Breeding Strategy with Communication

3.3.2.1 Communication Operations

Communication is an essential factor for the emergence of cooperation. As the result of evolution, simple communication protocols between agents can be obtained [11]. To provide the communication and cooperation between agent for robot navigation problem, SEND (this operation is symbolized by 'S' in the simulation) and RECEIVE (this operation is symbolized by 'P' in the simulation) operations are used in addition to the operations and terminals used for non-communicating case. An agent can tell another agent to stop or move accordingly by the help of these commands. The SEND operation takes two arguments and returns its second argument. As a side effect, it sends its first argument to the nearest agent [3]. If there is a RECEIVE operation in the tree structure of an agent, the robot controlled by this agent moves according to the message sent to it if any message is sent to it by other robots. If the robot does not take any message from other robots, the RECEIVE operation in the agent's structure returns its argument by default. If there are two or more SEND messages in a tree structure of an agent, the one that is evaluated later in tree evaluation procedure is considered as the real sent message. Let's consider two agents below:

Agent 0: Send goal forward
Agent 1: Receive Stop

Agent 0 always goes in forward direction. As side effect it sends the 'goal' message to its nearest agent. Agent 1 does not move to any direction provided that Agent 0 is away from it but if Agent 0 comes close to Agent 1, this time Agent 1 begins to move in the direction of its goal since it takes the message to go to its goal from Agent 0. This is an example of motion coordination between agents. As genetic programming progresses more complicated cooperation between agents can be observed. Tree evaluation procedure, fitness considerations and training and validation scenarios for communicating case are same as the ones with the non-communicating case.

3.3.2.2 Results for Communicating Case

There are several obtained results but none of the results are able to fulfill all the tasks in all the training cases. The best result was able to complete first and third training scenarios. Three robots are able to fulfill their tasks in the second and fourth scenarios. Only one robot is not able to overcome the task in both cases. The structures of four agents in one of the best runs and their average fitness value for the training scenarios are as follows:

Agent1: '++++g+f+Sff+gaa+++gSffaaa'

Agent2: 'RR+gPSgSSPg+gPSgSSSRSRgfFRgfgPSgSggRPdfgggff'

Agent3:'RRaRRaRRaRRaSRRRaRRaRRRaSRbfdRagRagRSRbfdggRaSgddRa RRaFfcRagRRaRRRSRbfRagdRSRbfdegggg'

Agent4:
'ARRBPFPSAAddPgPgSPFgSSFgPgPggFSASAAcaPgPgPAAgPgPPggPFgSFg PgPggAca+dPgFgPg'

Average fitness: 12530

The agents' responses in the training scenarios in one of the runs are shown in Figures 44-47.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#											#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#											#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 44: Response of the agents for the first training scenario for communicating case.

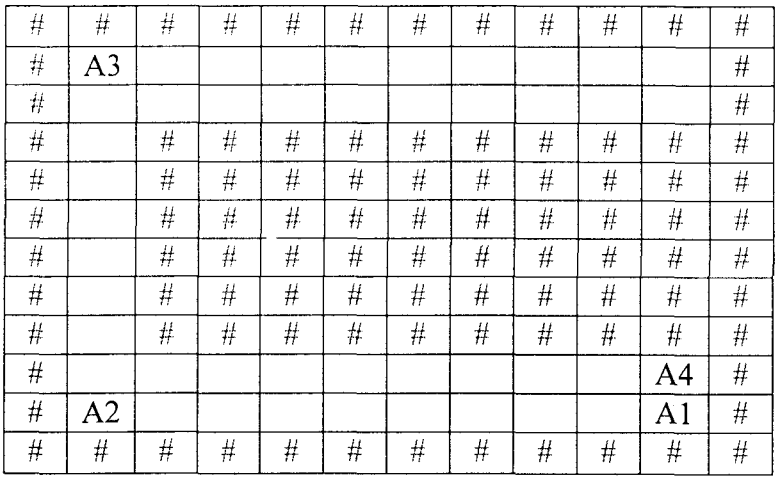

Figure 45: Response of the agents for the second training scenario scenario for communicating case.

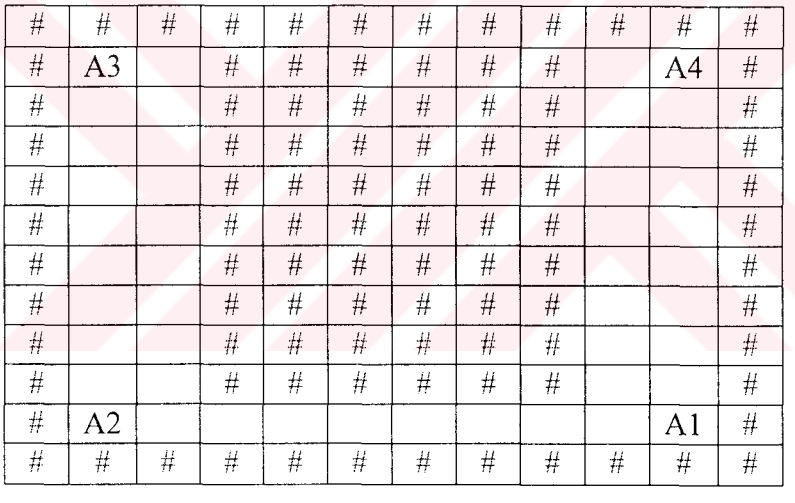

Figure 46: Response of the agents for the third training scenario scenario for communicating case.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#	A2	A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 47: Response of the agents for the fourth training scenario scenario for communicating case.

A seen from the response for all training scenarios, three of the agents are able to complete their tasks. In the first and third situations the tasks are totally overcome. This result can sometimes change because there are random types of movements in the control strategies of the agents.

When the trained agents are practiced in validation scenarios, good results are obtained for the similar validation scenarios (Figures 48-51). The training scenarios are more complicated than the validation scenarios for the similar cases.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#											#
#											#
#	#	#	#	#	#	#	#	#			#
#	#	#	#	#	#	#	#	#			#
#	#	#	#	#	#	#	#	#			#
#	#	#	#	#	#	#	#	#			#
#	#	#	#	#	#	#	#	#			#
#											#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 48: Response of the agents for the first validation scenario scenario for communicating case (Fitness=13520).

#	#	#	#	#	#	#	#	#	#	#	#
#	A3										#
#											#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#									A4		#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 49: Response of the agents for the second validation scenario scenario for communicating case (Fitness=9465).

#	#	#	#	#	#	#	#	#	#	#	#
#	A3	A4	#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#											#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 50: Response of the agents for the third validation scenario for communicating case (Fitness=9472).

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#											#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#			#	#	#	#	#	#	#		#
#	A2		#	#	#	#	#	#	#	A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 51: Response of the agents for the fourth validation scenario for communicating case (Fitness=13680).

For some of the validation scenarios, which are different from the training scenarios the fitness values, are still poor (Figures 52-54 for validation scenarios 5-7).

#	#	#	#	#	#	#	#	#	#	#	#
#											#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#	A1	#
#	A2								A4	A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 52: Response of the agents for the fifth validation scenario for communicating case (Fitness=4807.3).

#	#	#	#	#	#	#	#	#	#	#	#
#											#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#											#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#	A1	#
#		#	#	#	#	#	#	#	#	A2	#
#									A4	A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 53: Response of the agents for the sixth validation scenario for communicating case (Fitness=1807.3).

#	#	#	#	#	#	#	#	#	#	#	#
#											#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#		#
#		#	#	#	#	#	#	#	#	A1	#
#		#	#	#	#	#	#	#	#	A2	#
#									A4	A3	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 54: Response of the agents for the seventh validation scenario for communicating case (Fitness=1807.3).

Most of the validation scenarios show good results meaning the trained agents can solve most of the situations encountered. The trained agents show good behavior with validation scenarios 8-12. Figures 55-59 shows responses of the agents for these scenarios.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#											#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#	#	#	#	#	#	#		#	#		#
#											#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 55: Response of the agents for the eighth validation scenario for communicating case (Fitness=13440).

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#											#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#		#	#	#	#	#		#	#		#
#											#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 56: Response of the agents for the ninth validation scenario for communicating case (Fitness=13760).

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#											#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 57: Response of the agents for the tenth validation scenario for communicating case (Fitness=13760).

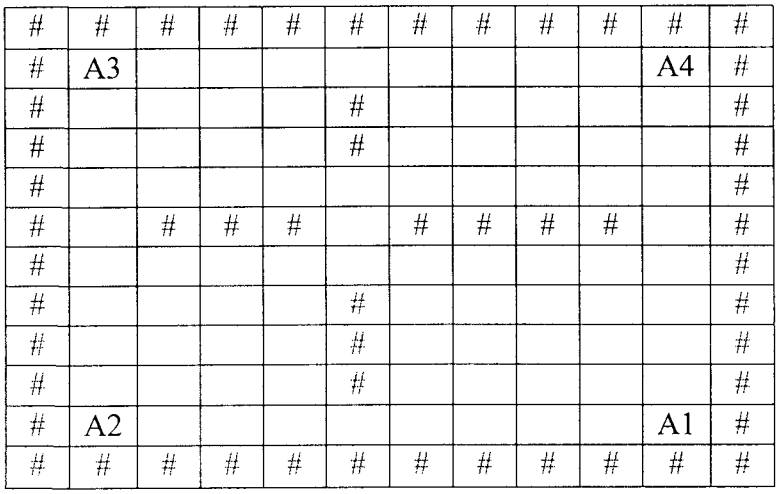

Figure 58: Response of the agents for the eleventh validation scenario for communicating case (Fitness=13280).

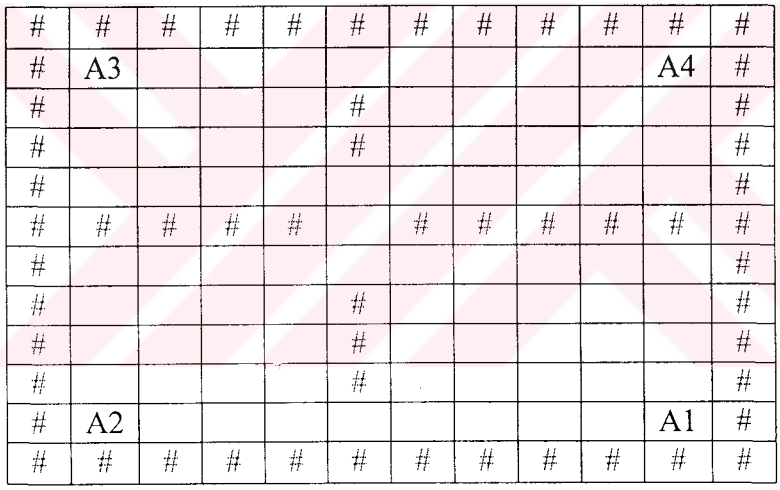

Figure 59: Response of the agents for the twelfth validation scenario for communicating case (Fitness=12960).

3.3.3 Comparison of the Results Obtained for the Robot Navigation Problem for Communicating and Non-Communicating Cases

3.3.3.1 Comparison of Agents Work for Communicating and Non-Communicating Case

3.3.3.1.1 Agents' Structures and Their Job for Non-Communicating Case

For non-communicating case the trained agents are capable of fulfilling some of the jobs while they cannot overcome some of the difficulties in some world scenarios. The huge agent structures may be minimized if some relations between the operations and the terminals of an agent are considered. In Appendix B rules for minimization of tree structures are investigated.

For non-communicating case first agent looks if there is an obstacle in front of it. If there is an obstacle, it makes a random movement. If there is no obstacle, it searches if there is another agent in front. If there is an agent in front of it, first agent goes in the direction of its goal else it goes in its forward direction. The agent's structure can be minimized to another form since some parts of its structure are useless in evaluation procedure. The agent's structure can be minimized as follows:

Agent1: '+fAgRaf'

Figure 60 shows minimal tree structure for the first agent.

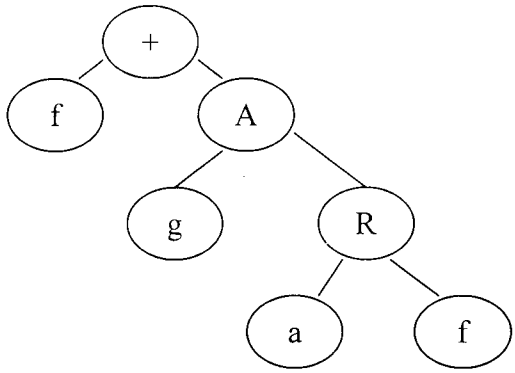

Figure 60: The minimal tree structure of Agent1 for non-communicating case.

What the second agent does is to go back in the backward direction if there is an agent in front of it. If no agent is observed in the front direction than the agent controls if there is an obstacle in front. If there is an obstacle, the agent tries to go in the goal direction. If there is no obstacle, the agent moves in its the forward direction. The huge structure of the second agent can be minimized to the agent structure shown below:

Agent2: 'Ab+ga'

The minimal tree structure of the same agent is shown in Figure 61.

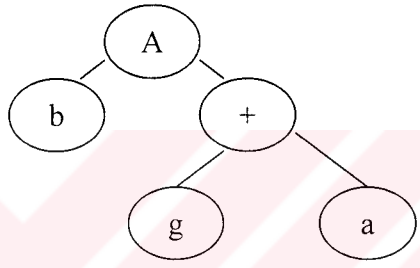

Figure 61: The minimal tree structure of Agent2 for non-communicating case.

The third agent is a simple agent. If the cell in front of the agent is free the agent goes in the forward direction, otherwise the agent goes in the direction of its goal. Minimal structure of the agent is as follows:

Agent3: 'Rag'

The minimal tree structure of the same agent is at Figure 62.

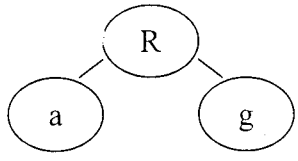

Figure 62: The minimal tree structure of Agent3 for non-communicating case

The fourth agent looks if there is another agent in front of it. If the answer is yes, it tries to go in the forward direction. If there are no agents in front of it, it searches if there is an obstacle in front direction. If there is no obstacle in front, the agent goes forward but if there is obstacle, this time the agent looks if it is in the same x-axis or in the same y- axis with its goal. If this case is true, the agent goes in the direction of its goal, else the agent moves backward without changing its direction. Minimal structure of the fourth agent is as follows:

Agent4: 'Aa+Fgba'

The minimal tree structure of the fourth agent is at Figure 63.

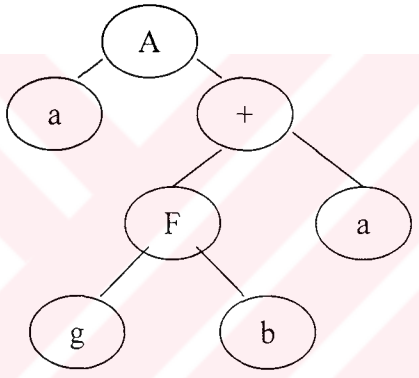

Figure 63: The minimal tree structure of Agent4 for non-communicating case

Still for non-communicating case some kind of cooperation between agents is possible. For example second agent goes backward if there is an agent in front of it. By this way it gives way to other agents to go to their goals and it does not change its direction as well although it moves backward. An interesting situation is that the agents do not consider if they are at the narrow aisle in their tasks. Their movements do not depend on the situation if they are in the aisle or not.

3.3.3.1.2 Agents Structures and Their Jobs for Communicating Case

For communicating case the trained agents are capable of fulfilling most of the jobs while they can't overcome few of the difficulties in some world scenarios.

First agent is just a simple agent moving to its goal if there is an obstacle in front of it. If there are no obstacles in front of the agent, the agent goes in forward direction one unit. While doing this, first agent always sends message to the nearest agent to it. The message tells the nearest agent to move randomly. The minimal structure of the agent is as follows:

Agent1: '+gSra' (agent sends move random message to the nearest agent to

The minimal trees structure for the first agent is shown at Figure 64.

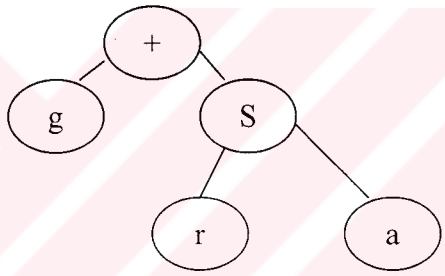

Figure 64: The minimal tree structure of the first agent for communicating case.

Although second agent seems so complicated, its minimal form counts down to a simple tree structure. First of all this agent is both a receiver and a sender. The agent sends goal message to nearest agent to it. It also obeys the message if some other agents send it any. When it doesn't take any message from other agents, the agent look whether the cell in front of it is empty. If the answer is 'yes' the agent moves in its goal direction. When the answer is 'no', the agent makes random movement. The minimal structure of the agent is as below:

Agent2: 'RgSgPf' (agent send move to goal message to the nearest agent to it, it also executes the message if any message is sent to it by other agents)

Tree minimal tree structure for the second agent is shown at Figure 65.

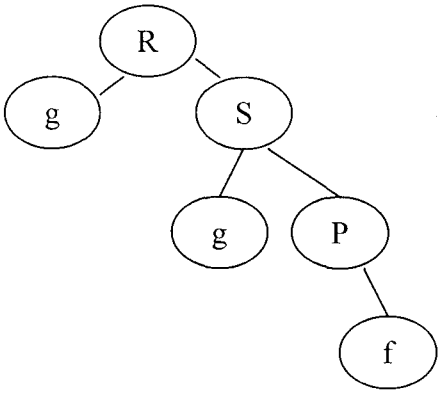

Figure 65: The minimal tree structure of the second agent for communicating case.

Third agent does an important job in order to solve the problem. Although it has a huge structure, most of the branches can be minimized to give a small tree structure. It is a sender function. Third agent wants the nearest agent to it to go backward if the cell in front of the third agent is empty and if the nearest agent to the third agent is capable of taking the message sent to it. If the cell in front of the third agent is full than the third agent wants the nearest agent to make a random movement. This intelligent send message in a way opens the way for the third agent to its goal. If an agent receives this message which is close to the third agent, the agent moves backward and opens the way for the third agent if they are side by side and trying to move in opposite directions or it makes a random movement which also helps the third agent to go to its goal more efficiently by avoiding any crowd near the third agent. What the third agent does is to go forward if the cell in front of it is empty or to go to its goal if the cell in front of it is not empty while it is sending its message. The minimal structures for this agent is as follows:

Agent3: 'RaSRbfg'

Figure 66 shows the minimal tree structures for the third agent.

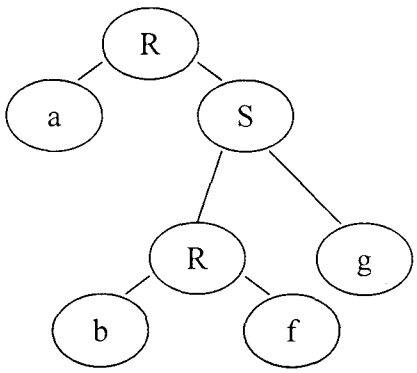

Figure 66: The minimal tree structure of the third agent for communicating case.

Although the fourth agent seems so complicated, as a result it turns a very simple tree structure. The agent is both a receiver and a sender. It wants the nearest agent to it to move in the direction of its goal. Its tree structure can also be minimized to yield a goal operation. When it doesn't take any message from the other agents the fourth agent goes in the direction of its goal. The minimal structure of the agent is shown as below:

Agent4: 'SgPg'

The minimal tree structure for the fourth agent is shown on Figure 67.

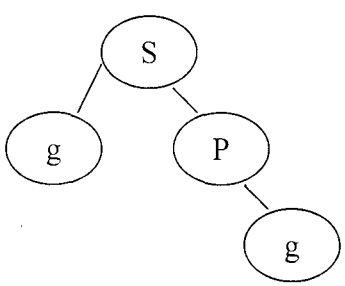

Figure 67: The minimal tree structure of the fourth agent for communicating case.

3.3.3.1.3 Comparison of Communicating and Non-Communicating Agents

As seen from the minimal structures of the agents, there is not much difference between the agents for both communicating and non-communicating case. The only difference is that in communicating case, the possibility of coordination between agents is stronger. Coordination provides the efficiency in the solution. As the result, communicating agents can more effectively complete the tasks. The last validation scenario is an important criterion. This scenario has completely different wall structure than the other validation scenarios. The results obtained for this validation scenario shows that communicating agents can overcome different structured world scenarios with respect to non-communicating agents more productively after the training step. This result is the consequence of cooperation and coordination between agents, which is provided by the communication among them by communication operations.

3.4 Robot Navigation Problem 2 (Comparison of homogenous breeding strategy (with and without communication) with heterogeneous breeding strategy (with communication) when the number of training data is increased or the step size for the robots are changed)

In this part eight training scenarios are designed. The first four training scenarios are the ones shown in Figure 10-13. For the remaining four training scenario places of robots are changed. Figures 68-71 show the new training scenarios.

#	#	#	#	#	#	#	#	#	#	#	#
#	A2									A1	#
#											#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#											#
#	A3									A4	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 68: New fifth training scenario. A1 and A4 will change place and A2 and A3 will change place.

#	#	#	#	#	#	#	#	#	#	#	#
#	A2									A1	#
#											#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#											#
#	A3									A4	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 69: New sixth training scenario. A1 and A4 will change place and A2 and A3 will change place.

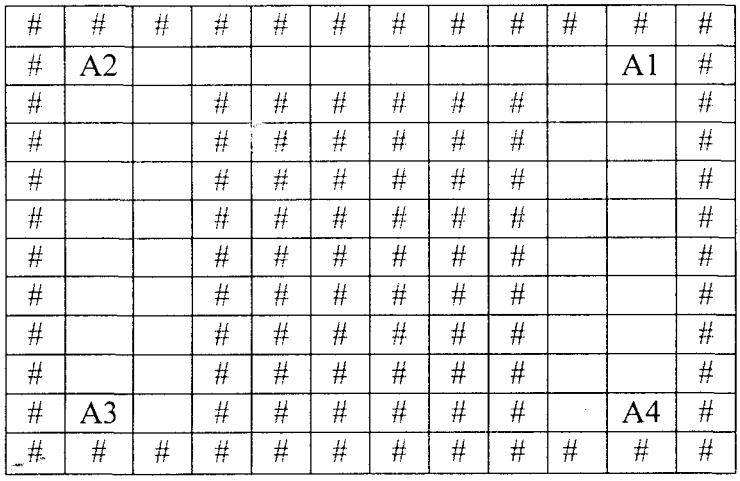

Figure 70: New seventh training scenario. A1 and A4 will change place and A2 and A3 will change place.

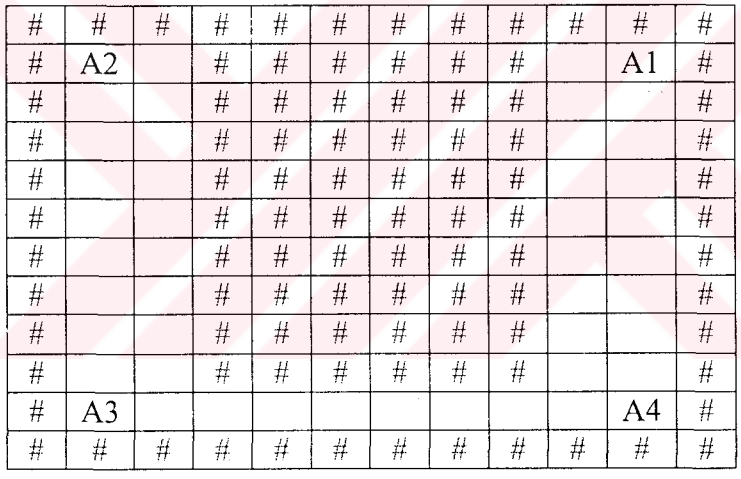

Figure 71: New eighth training scenario. A1 and A4 will change place and A2 and A3 will change place.

As seen from the Figures 68-71 the initial positions of the robots are changed while their goal positions remain the same. Fitness for all world scenarios are calculated as in part 3.3.1.2 but the average fitness for an individual is calculated by dividing the sum of fitness of training scenarios by eight since there are eight training scenarios. The operations and terminals used are chosen from the operations and terminals used in part 3.3.1.1. Same evaluation and communication

principles are used as in part 3.3.2.1. For each simulation the population is composed of 20 individuals. Crossover rate, reproduction rate and mutation rate are chosen as %80, %10 and %10 respectively. Elitist strategy is used in reproduction for the best individual of the generation. No new individuals are inserted into the population as generations pass. The search is stopped after 100 generations. The best results obtained are saved during the simulations.

3.4.1 Applying Homogenous Breeding Strategy without Communication

As the result of this case a single agent controlling all the robots is obtained. The structure of the agent is as below:

Agent='RRBBRRaffaaRRBBRRaffaaRaRRafaRRBRRaffaafRRRBBRRBBRB
BRRaffaaRRBBRRaffaaRaRRafaRRBRRaffaafRRaffaRaRRafffaaRaaRaff'

Average fitness=6472

When the agent structure is minimized following structure is obtained.

Agent='Raf'

What the agent does is to look if the grid in front of it is empty. If the answer is yes the agent goes in front direction, if the answer is no the agent makes a random movement

The obtained results for world scenarios are as in Figures 72-79.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#											#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#										A2	#
#										A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 72: Result for first training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#											#
#											#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#	A3	#	#	#	#	#	#	#	#	#	#
#					A4						#
#	A2			A1							#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 73: Result for second training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#			A1							A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#	A3		#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#		A2	#
#			#	#	#	#	#	#			#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 74: Result for third training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#			#	#	#	#	#	#		A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#		A3	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#	A2		#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#							A1				#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 75: Result for fourth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#											#
#			A2								#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#							A3				#
#							A4			A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 76: Result for fifth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3										#
#											#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#							A1				#
#	A2								A4		#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 77: Result for sixth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2		#	#	#	#	#	#		A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 78: Result for seventh training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3		#	#	#	#	#	#		A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 79: Result for eighth training scenario.

3.4.2 Applying Homogenous Breeding Strategy with Communication

As the result for this case a single agent controlling all the robots is obtained. In its structure the agent contains 'RECEIVE' and 'SEND' communication operations. Since this agent is responsible from the movements of all robots all the robots receive and send commands to each other. Structure of the agent is as below:

Agent='FFPPSRfggFPPSRfFeSRbeAb+bfgFbgSRfgg'

Average fitness=7131

The minimal structure of the agent is as below:

Agent='FPgSRfgg'

What this agent does is to make the robots go in their goal direction when there is no message send to them. The robot sends random movement message to the nearest robot if its forward is an empty cell. If the cell in front of the robot is not empty the robot wants nearest agent to move in its goal direction.

The obtained results are as in Figures 80-87.

#	#	#	#	#	#	#	#	#	#	#	#
#										A4	#
#							A1	A2			#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	A3										#
#											#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 80: Result for first training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3										#
#										A1	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#										A4	#
#	A2										#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 81: Result for second training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#		A1	#	#	#	#	#	#	A2		#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 82: Result for third training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#			#	#	#	#	#	#	A3		#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#		A1	#	#	#	#	#	#			#
#		A4	#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2										#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 83: Result for fourth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#										A4	#
#	A2										#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	A3										#
#										A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 84 Result for fifth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3										#
#										A1	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#										A4	#
#	A2										#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 85: Result for sixth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2		#	#	#	#	#	#		A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 86: Result for seventh training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3		#	#	#	#	#	#		A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 87: Result for eighth training scenario.

3.4.3 Applying Heterogeneous Breeding Strategy with Communication

As the result, four different agents controlling different robots are obtained. The structures of the agents are as in the next page:

Agent1='++g+A++Ag+++AaAa+ggaaaa+g++++FefAgfaA+g+gegaS ffA+++gaFebaaa'

Agent2='A+SgPfPPSgPgR+PgPSgSggR+gPSgSSSPgfRPgfgf'

Agent3= 'RRaRaRRRRRagdRRRaRRagRagSRbfdeRRaaRRaRRRaRagRRRa RRagRbfSRbfdegRagRRaRSRbfdRRRaRRagRbfSRbfdeRagg'

Agent4='SFgSAFgASAAddPgPggggPAPgPg'

Average fitness=9555

The minimal structures of the agents are as below

Agent1='+Sfga'

Agent2='ApgSgf'

Agent3='SRbfRag'

Agent4='SPgg'

First robot sends random movement message to nearest robot, it moves in goal direction if there is an obstacle in front of it and it moves in front direction if there is no obstacle in front of it. Second robot moves in goal direction if there is another robot in front of it and it makes a random movement when there is no agent in front of it provided that there is no message sent to it by other agents. Second agent also sends move to goal message to the nearest agent to itself. Third robot moves in front direction and sends move backward message to nearest robot to it when the cell in front of it is empty, it moves to goal direction and sends move random message to the nearest robot to it when the cell in front of it is full. Fourth robot moves in its goal direction as long as it doesn't take any message from other robots. It sends move to goal message to the nearest robot to itself.

The obtained results for world scenarios are as in Figures (88-95).

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#											#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#											#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 88: Result for first training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3										#
#											#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#										A4	#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 89: Result for second training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#	A2		#
#			#	#	#	#	#	#		A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 90: Result for third training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3	A4	#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 91: Result for fourth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#		A2								A4	#
#											#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	A3										#
#										A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 92: Result for fifth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A2	A1									#
#	A3										#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#								A4			#
#											#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 93: Result for sixth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2		#	#	#	#	#	#		A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 94: Result for seventh training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3		#	#	#	#	#	#		A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 95: Result for eighth training scenario.

3.4.4 Effect of Changing the Step Size for Robots (from 50 to 100)

Small differences are observed when the step sizes in the simulations are changed while the agents trained in part 3.4.1; 3.4.2 and 3.4.3 are used for simulations. When the agents trained by heterogeneous breeding strategy with communication are used, no variation is observed in the responses of the robots except for the fifth and sixth scenarios. The differences in these scenarios are not so complicated and they are the consequence of 'random movement" terminals in structure of the agents. Increasing the step size does not improve responses of the robots in reality for this application.

When the agent trained by homogenous breeding without communication is used, small changes in the responses are observed. No difference is observed when responses of the robots to the third and fourth scenarios are compared with the ones when the step size is 50. However, other responses obtained for this situation are different. These variations come from the nature of the obtained single agent for this case. When the structure of the agent is minimized a simple if statement is obtained having a random movement terminal in its structure (Raf). Changing the step size doesn't give an improvement or disimprovement for this case too much.

For the case with homogenous breeding and communication the obtained results for all the cases are same.

In the simulations with 100-step size, after about 35-40 steps nearly all the robots ended their walk. They are not able to move much further unless the agents controlling them return a random movement action. Increasing the step size to 100 doesn't change the responses of the robots so much. If there are 'random movement' terminals in structure of the agents the robots try to move further but this type of action doesn't change the responses to the scenarios so much.

3.4.5 Comparison of the Results for Homogenous Breeding and Heterogeneous Breeding

When the number of training cases is increased and the starting positions of the robots are changed, the individual obtained by heterogeneous breeding with communicating case show better performance with respect to the agents trained by homogenous breeding strategy no matter communication is provided between robots or not. When homogenous strategy is considered, the agent obtained with communicating case is more talented than the one obtained with non-communicating case. But 'random movement' terminal plays an important role for the agents obtained by homogenous breeding strategy. So making an argument like 'communication increased the robustness of the agent evolved by homogenous breeding strategy" is not valid.

3.5 Robot Navigation Problem 3 (trails to increase the performance of the individuals evolved)

In this part effect of new methods of communication and new movement types are considered in order to increase the performances of the individuals evolved. The applied techniques show variations from homogenous breeding to heterogeneous breeding. In the sub sections of this section what is done for different evolution types will be investigated.

3.5.1 Trials for Heterogeneous Breeding Strategy with Communication

A kind of 'move random' action, which is not obtained as the result of evaluation of the tree structure of an agent is performed in the simulations. This action is performed for the robots, which aren't able to move for about two steps provided that the robots do not reach their goals. The obtained individual in part 3.4.3, which is consisting of for different agents is used for this application. The scenarios used are the scenarios used in part 3.4. The step size for the robots is 50. An improvement in the average fitness is obtained in different runs of the simulations. The average fitness has shown variations between 10255 and 10800. These results obtained for some scenarios are same and an improvement is observed for the remaining scenarios with respect to the results obtained for part 3.4.3. Figures 96-103 show the obtained results.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#											#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#											#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 96: Result for first training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3										#
#											#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#										A4	#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 97: Result for second training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#	A2		#
#			#	#	#	#	#	#		A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 98: Result for third training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3	A4	#	#	#	#	#	#		A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 99: Result for fourth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#										A4	#
#			A2								#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#								A3			#
#										A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 100: Result for fifth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3										#
#						A1					#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#								A4			#
#	A2										#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 101: Result for sixth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2		#	#	#	#	#	#		A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 102: Result for seventh training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3		#	#	#	#	#	#		A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 103: Result for eighth training scenario.

3.5.2 Trials for Homogenous Breeding Strategy without Communication

The same action type told in part 3.5.1 is used in the simulations for this case. A new GP search is done in order to obtain a single agent controlling all the robots. The 'random movement' and 'stop' terminals are excluded from the terminal set. Instead a new terminal reversing the direction of the robot and making it move one unit in reversed is designed which is symbolized by the letter 'e' 'in the simulations. The training scenarios and genetic operation rates used in part 3.4 are used for in the GP search. The obtain agent for the simulations is below:

Agent: '++B+dRdgRdgAdaAdAd+AdaAda'

Average fitness=9800

The minimal structure of the obtained agent is as below:

Agent: '+BdgAda'

This agent turns the robot in counterclockwise direction 90 degrees and moves then ahead if there is an obstacle in their forward direction and if the robot is in the aisle. If there is an obstacle in front but the robot is not in the aisle, this time robot goes in its goal direction. If there is no obstacle but another robot in front direction, this time again the agent turns counterclockwise and moves ahead one

unit but if the front direction is totally free than the robot just goes straight ahead one unit.

Obtained responses for the scenarios are in Figures 104-111

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#	A2										#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#											#
#										A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 104: Result for first training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#										A1	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#											#
#	A2										#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 105: Result for second training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2	A1	#	#	#	#	#	#			#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 106: Result for third training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#			#	#	#	#	#	#	A3	A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 107: Result for fourth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#										A4	#
#								A2			#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#								A3			#
#										A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 108: Result for fifth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A1	#
#											#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#							A4				#
#	A2										#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 109: Result for sixth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2		#	#	#	#	#	#		A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 110: Result for seventh training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3		#	#	#	#	#	#		A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 111: Result for eighth training scenario.

3.5.3 Trials for Homogenous Breeding with Communication

The same action type told in part 3.5.1 is used in the simulations for this case. A new GP search is done in order to obtain a single agent controlling all the robots. The 'random movement' and 'stop' terminals are excluded from the terminal set. Instead a new terminal reversing the direction of the robot and making it move one unit in reversed direction is designed which is symbolized by the letter 'e' 'in the simulations. The way of communication is reconsider in this search. A robot is given the ability to send message to another robot if they are both in the small aisle provided that the single agent controlling all the robots have both

'SEND' and 'RECEIVE' operations in its structure. So communication between robots is provided only for the situations where conflict may arise between them. This idea is not used for Heterogeneous strategy with communication since this type of communication will be provided only if all the agents evolved by the GP search have 'SEND' and 'RECEIVE' commands in their structure. But in order to increase the probability that an agent evolved by Homogenous strategy will have both 'SEND' and 'RECEIVE' operations in its structure (absence of one of the communication operations means that that agent is just a non-communicating agent) the probability appearance of these operations in agent structures is increased. The training scenarios and genetic operation rates used in part 3.4 are used for in the GP search. The obtain agent for the simulations is below:

Agent: 'PSPRgaRS+PcaaASPccSRaaBcd'

Average fitness=8215

The minimal structure of the obtained agent is as below:

Agent: 'RSPRgaaAcBca'

If two robots are in a situation to negotiate with each other (they are both at the aisle), they both send go to goal message to each other if the cell in front of them is empty other wise they send go in front direction message to each other. Provided that there is no negotiation for a robot considered the robot goes straight ahead if the cell in front of the robot is free. However, if this case is not true, the robot looks if there is any other robot in front of it. If this argument is true the robot turns clockwise 90 degrees and moves a unit. If there is no other robot in front of the robot controlled by the agent (this means there is an obstacle in reality in front of the robot since it is known that the cell in front of the robot is not free) this time the robot looks if it is in the aisle or not. If it is in the aisle, the robot turns clockwise 90 degrees and makes a movement in front direction (since the robot is in the aisle and since its direction is due to the wall, this type of movement takes the robot to another cell inside or outside the aisle but the action of the agent can both make it closer to its goal or take it away from its goal). If the robot is not in the aisle

(this means the robot is due to a wall but it is not in the aisle), this time it turns counterclockwise 90 degrees and goes ahead one unit. This is how the agent moves the robots according to its tree structure. This individual gives an average fitness between 7000 and 9300. In one run the individual gives an average fitness of 8215. The obtained results for this case are as in Figures 112-119.

#	#	#	#	#	#	#	#	#	#	#	#
#										A4	#
#											#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	A2										#
#	A3									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 112: Result for first training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3		A1								#
#											#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#	A4	#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#											#
#	A2										#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 113: Result for second training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3							A4			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#	A2		#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#		A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 114: Result for third training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A4		#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#		A3	#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 115: Result for fourth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#										A4	#
#											#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#	#	#	#	#	#	#	#	#	#		#
#		A3		A2							#
#										A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 116: Result for fifth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#						A1					#
#											#
#	A3	#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#		#	#	#	#	#	#	#	#	#	#
#									A4		#
#	A2										#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 117: Result for sixth training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3									A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2		#	#	#	#	#	#		A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 118: Result for seventh training scenario.

#	#	#	#	#	#	#	#	#	#	#	#
#	A3		#	#	#	#	#	#		A4	#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#			#	#	#	#	#	#			#
#	A2									A1	#
#	#	#	#	#	#	#	#	#	#	#	#

Figure 119: Result for eighth training scenario.

3.5.4 Comparison of the Trails

When the trails used for increasing the performance agents evolved by different strategies are considered, the results obtained in section 3.5.1 are better than the other results obtained. Evolving agents using heterogeneous breeding with communication is more efficient than other techniques. A huge improvement is observed when homogenous breeding without communication is applied with the ideas used in part 3.5.2. But the results obtained in part 3.5.3 are disappointing, since the results show that applied techniques to increase the efficiency and

solvability of the problem cannot even reach the performance of the agent obtained in part 3.5.2 although a new style of communication between the robots is applied. From this point of view, the ideas used for homogenous breeding without communication is the best since it improved the robustness and applicability of agent the most. But best result is obtained when heterogeneous breeding is used with communication. The improvement in the results for this application is not so big but it is satisfactory. Least improvement is obtained when the ideas for homogenous breeding with communication are implemented. The improvement for the results in this application is limited.

CHAPTER 4

SYSTEM IDENTIFICATION WITH GENETIC PROGRAMMING

4.1 Introduction

Identification of systems is a complex task. The knowledge about the system must be sufficient in order to have a priori idea about the system. But is it possible to identify a system by only the limited data it has? A new system can be found, which suits the data given. This new system may not be the one exactly looked for. But if the data fits well, it is sure that the system is a good approximation to the desired one. Genetic programming is a powerful tool in optimization. It can also be used for system identification [13]. System can be identified by using genetic programming. A system can be described by tree structure. By using the abilities of genetic programming, systems that suit the desired response can be obtained.

4.2 Using Genetic Programming for the Problem

4.2.1 General

Given a data about a system, an approximate representation for that system had to be found. An input is given to the system, which is to be identified or to be represented by other systems. So an output is obtained. The same input is used as the terminal of the genetic programming search. The initial structures in the

population contain basic operations like addition and subtraction (both take two arguments). Three kinds of linear s-domain system blocks are used too (all take one terminal). One of the blocks is used for representing the first order systems as in Equation 8

$$F(s) = \frac{1}{s + a} \tag{8}$$

At Equation 8, 'a' is the pole of the first order system. Another block is for representing second order systems without a zero as in Equation 9.

$$N(s) = \frac{1}{\left(s^2 + 2\partial ws + w^2\right)} \tag{9}$$

At Equation 9, 'w' is the natural frequency and '∂' is the damping ratio. The last block is for the second order systems with a zero as in Equation 10.

$$T(s) = \frac{s + k}{\left(s^2 + 2\partial ws + w^2\right)} \tag{10}$$

At Equation 10, 'k' is the zero of the second order system, '∂' is the damping ratio and 'w' is the natural frequency of the second order system. Some other operations like square root taking, gain and delay, all acting on one argument are also used. Combination of operations and terminals meaningfully, is the main step for the construction of an individual in genetic programming. The operations and terminals are combined in a way to give diversity between the constructed individuals. So different sized and different shaped individuals appear in initial population. Each individual resembles a tree like structure and can be represented as a block diagram having linear and non-linear structures. Each individual can be thought to define a system. The output of each individual is the response of the system to the input. The given responses of the all individuals are compared with the data obtained from the exact system. An error is calculated and this error is the structure that we want to minimize because as error minimizes, the individuals and the exact system begin to coincide with each other meaning the individual can be used instead of the exact system. To minimize the error function we have to change

the individuals. But the change in the individuals must be in a way to improve the performance. This means minimization of error is possible only if we change the structures of the individuals in an efficient way.

4.2.2 Some Important Aspects of the Problem

Different types of operations are used to construct the individuals in the population. Since some of the operations can take parameters (natural frequency and damping ratio) a vector of parameter is also kept in hand for all individuals. There may be problems in convergence because of the nature of the problem. As time goes on some genes disappears and the gene blocks remaining in the population may become poor to converge to a desired solution. So mutation and some new individual addition to the population may become important for convergence potential of the system. Another problem is the situations with the parameters. They must be tuned by some other techniques in order to have better solutions. For the sake of reducing the time consumption of very long individuals, can be omitted by giving negligible fitness values to these structures.

4.2.3 The Operations and Terminals

4.2.3.1 Operations

The operations used in the search and the parameters they take are given below. The examples for the operations are in Appendix C.

4.2.3.1.1 Square root

The letter 'S' in the simulation symbolizes square root taking. Operation takes one input and gives one output. The input is a vector and the output is another vector having the same length. It takes the square root of the absolute value of the vector's each components. Input output relation of this operation is shown at Equation 11.

$$S(u(t)) = \sqrt{u(t)} \qquad (11)$$

4.2.3.1.2 Gain

The letter 'G' in the simulations symbolizes gain operation. This operation takes one input and gives one output. The input and output are vectors having the same length. Gain operation takes one parameter. This parameter is the gain factor of the operation (the gain-factor is taken as a random number between 0 and 3 in simulations). All the components of the input vector are multiplied by the parameter and output vector is obtained. Time-dependant input-output relation of this operation is shown at Equation 12.

$$G(u(t)) = K * u(t) \text{ (K is the gain factor)} \qquad (12)$$

4.2.3.1.3 Addition

Addition is symbolized by '+' in the simulations. This operation takes two inputs and gives one output as the result. Both of the inputs are vectors. It doesn't take any parameters. It basically adds two vectors and gives the addition vector as its output. Time-dependant inputs-output relation of this operation is shown at Equation 13.

$$+(u(t), g(t)) = u(t) + g(t) \qquad (13)$$

4.2.3.1.4 Subtraction

Subtraction is symbolized by '-' in the simulations. This operation takes two inputs and gives one output as the result. Both of the inputs are vectors. It doesn't take any parameters. It basically subtracts the second vector from the first one and gives the result as its output. Time-dependant inputs-output relation of this operation is shown at Equation 14.

$$-(u(t), g(t)) = u(t) - g(t) \qquad (14)$$

4.2.3.1.5 Delay

This operation takes one input, which is a vector. It is symbolized by the letter 'D' in the simulations. It doesn't take any parameters. A vector of predefined size is delayed one unit as the result of the operation. The first term of the output

vector becomes 0. If there are n terms in the vector, the n'th term in the output vector becomes the n-1'th term of the input vector.

4.2.3.1.6 Saturation

Saturation is symbolized by 'M' in the simulation. This operation takes one input and gives one output as result. It has one parameter. The parameter determines the saturation limit (The parameter is a random number between 0 and 10). The terms of the input vector, which are smaller than the saturation limit remains the same for the output, but the terms greater than this limit takes the value of the limit.

4.2.3.1.7 First order transfer function

First order transfer function is symbolized by 'F' in the simulations. First order transfer function contains a simple first order linear system taking one parameter, which is the pole of the system. Usually the pole is a random number between 0 and –10 in order to make the transfer function stable. The system is represented as in Equation 8. As an input it takes a vector. The response of the vector to the first order system is the output of the system.

4.2.3.1.8 Second order transfer function without zero

Second order transfer function without zero is symbolized by the symbol 'N' in the simulations. Second order transfer function contains a simple second order system having two poles (two parameters of the operation). The poles of the system may be imaginary but they must be at the left part of the imaginary axis. As an input, the system takes a vector. The response of the vector to the second order system is the output of the system. System is represented as in Equation 9.

4.2.3.1.9 Second order transfer function with zero

This operation is symbolized by the symbol 'T' in the simulations. Second order transfer function is a second order system having two poles and a zero (three parameters totally). The poles of the system may be imaginary but they must be at the left part of the imaginary axis. As an input, the system takes a vector. The

response of the vector to the second order system is the output of the system. The system is represented in Equation 10.

4.2.3.2 Terminals

The terminals of an individual generally depend on the problem encountered. The letter 'a' in the computer simulation represents the input given to the system. Generally the input vector is taken as unit step of predefined size (in the simulations 101 data points during a 10 second simulation is taken with 0.1 second time elapses) or some other waveforms like sinusoidal or unit ramp can be used too. In non-linear system identification problem in section 4.2.8.3 unit ramp is used for system identification.

4.2.4 Representation of an Individual

An individual having a tree like structure can be represented by two ways. First representation is a block diagram having inputs and an output. Figure 120 shows the tree structure of an individual. Figure 121 shows the block diagram for the same individual

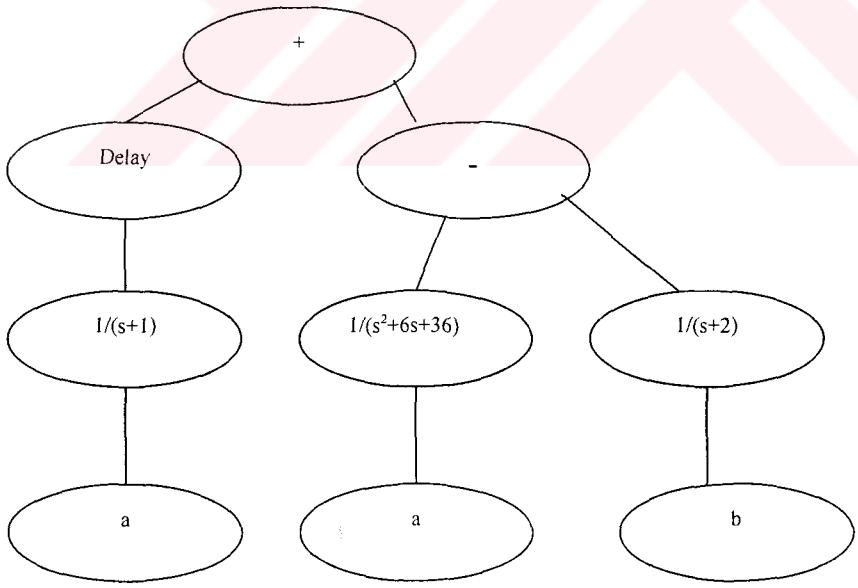

Figure 120: The tree structure of an individual.

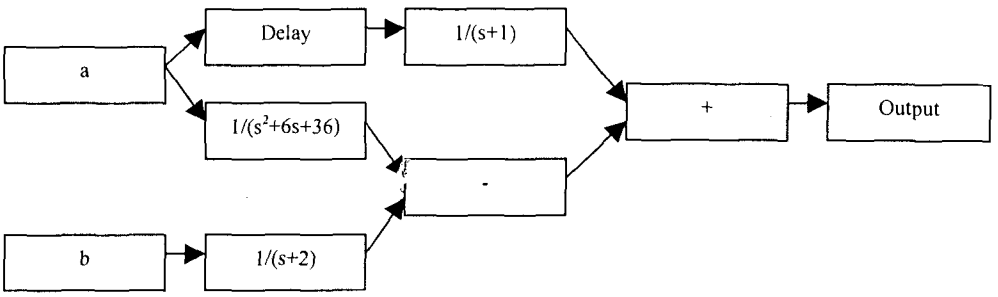

Figure 121: The block diagram of the individual represented as tree structure in Figure 120

Second way to represent a tree structure is by the string and parameter vectors (as in the simulations). The vector representation for the tree structure in Figure 120 is as below.

The String vector: (+, D, F, a, -, S, a, F, b)

Parameter vector-1: (0, 0, 1, 0, 0, 6, 0, 2, 0)

Parameter vector-2: (0, 0, 0, 0, 0, 0.5, 0, 0, 0)

Parameter vector-3: (0, 0, 0, 0, 0, 0, 0, 0, 0)

In parameter vector-1, the pole parameters of first order systems and the natural frequency parameters of second order systems are found. Gain constant and saturation level are also contained in the parameter vector-1. In parameter vector-2, damping ratio of the second order system is kept. The parameter vector-3 is used only for to show zero of the second order system with a zero.

4.2.5 Evaluation of an Individual

Evaluation of an individual in the simulation is done by performing block diagram operations to the terminals. The terminals are represented as vectors with a predefined length. 101 data points for 10-second simulation duration are taken. The data are taken at 0.1 second equal time elapses. If the individual in Figure 120 is considered the evaluation steps for the individual becomes as follows:

1- The string at first is (+, D, F, a, -, S, a, F, b)

2- First of all, the operation 'F' react with terminal 'a' and a new vector is formed which is represented as 'x1'. The string becomes (+, D, x1, -, S, A, F, B)

3- 'D' reacts with 'x1' and another vector is obtained which is called 'x2'. The string turns to (+, x2, -, S, a, F, b).

4- 'S' reacts with 'a'. The obtained vector is called 'x3'. The new string becomes (+, x2, -, x3, F, b).

5- 'F' reacts with 'b'. The obtained vector is called 'x4'. The new string becomes (+, x2, -, x3, x4).

6- The operation '-' reacts with 'x3' and 'x4'. The vector called 'x5' is obtained as the result. The new string becomes (+, x2, x5).

7- The operation '+' reacts with 'x2' and 'x5' and the output is obtained.

4.2.6 Fitness Calculation

Fitness of an individual shows closeness of that individual to the desired answer. In a system identification problem there are several ways to measure this phenomenon. A way is to take discrete values from the response and the desired output and than calculate the mean square root error of these discretely taken points. The first data is taken for the start of the simulation (when time=0). The last data is taken at the end of the simulation (when time=10). If there are n data points, the error for each data point is found, these errors are squared, the squared errors are summed up and square root of the sum is taken to give the mean square root error. The mean square root error of every individual is calculated. The individuals having low square error values are better than the ones having high square error values for genetic operations. So low mean square error for an individual means that the data taken for that individual suits the data of system to be identified better. The fitness value of an individual is calculated by the formula in Equation 15.

$$Fitness = \sqrt{\frac{1000}{(1+se)}} \qquad (15)$$

In Equation 15 'se' is the calculated square error of the individual concerned. All the individuals' fitness values are calculated by this procedure. The individuals with higher fitness values are more suitable for the problem. An individual's probability to be selected for genetic operations is proportional to its fitness value. If the fitness value for an individual is high, the probability of that individual to be selected for genetic operations is high too. In order to select an individual, the fitness values of all individuals are summed up. The result is divided into regions at the number of individuals according to the fitness values. A random number is taken between 0 and the resultant sum. The individual corresponding to the partition, the random number is taken, is selected for the genetic operations. Selection procedure between four different individuals is investigated below:

Table 1: The table for error and fitness values for four individuals

Individual number	1	2	3	4
Individual error value (se)	0.5	1	3	5
Individuals fitness value $\sqrt{\frac{1000}{1+se}}$	25.8	22.3	15.8	12.9

Sum of all the fitness values are taken first.

Resultant sum= 25.8+22.3+15.8+12.9=76.8

The partition table for the individuals is made as below:

Table 2: The partition table for four individuals

Individual number	1	2	3	4
Individuals partition	0-25.8	25.8-48.1	48.1-63.9	63.9-76.8

A random number is taken between 0 and the total sum. Assume that the random number is 40 then the selected individual must be the second individual since 40 is between 25.8 and 48.1 (40 is in the partition of the second individual).

4.2.7 Genetic Operations Used for the Search

The genetic operations used for the system are reproduction, crossover and mutation. The population size is 100. 10 % of the individuals are selected in order to be inserted in the new generation. Elitist strategy (reproduction of the best individual) is applied too. % 80 of the new individuals are obtained by crossing over operation. The probability of the crossover point of an individual to be a terminal is 17 %. So the probability of a crossover point of an individual to be an operation is 83 %. 10 % of the individuals in a new generation are obtained from mutation of the individuals of old population. For every ten generations 30 new individuals are inserted into the population while 30 worst individual at that generation is deleted.

4.2.8 Simulations

4.2.8.1 Second Order System

The system to be identified is a second order system with transfer function $T(s) = \frac{1}{s^2 + 2s + 5}$. A unit step input for ten seconds is given to the system. An output is obtained. The input for the genetic programming was a unit step of ten seconds. Optimization of parameters is not used since the system is not so complicated but the individuals' lengths are limited to avoid huge structures. At the end of the search the best result obtained has the properties below:

The String vector: (S, S, N, M, M, S, F, D, -, a, G, D, F, a)

Parameter vector-1: (0, 0, 5.55, 1.66, 1.90, 0, 5.16, 1, 0, 0, 2.71, 1, 2.80, 0)

Parameter vector-2: (0, 0, 0.8616, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Square error of the first individual = 0.35

Figure 122 shows the block diagram for the system whose response is similar to the second order system. Figure 123 shows the responses of the identified and the identifier systems for 10-second duration.

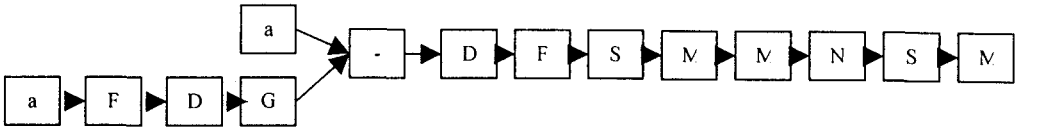

Figure 122: The block diagram for the identifier system

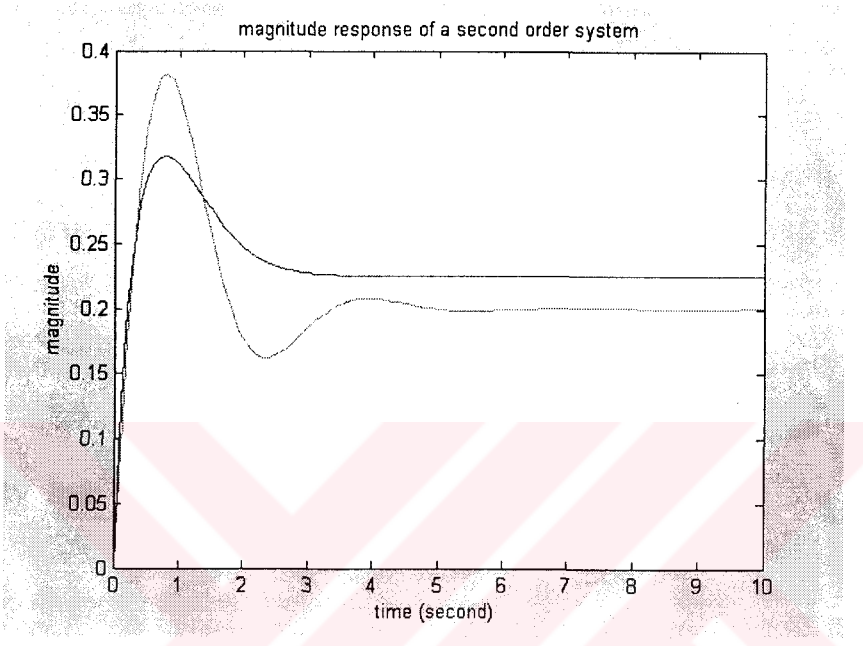

Figure 123: Response for two systems for second order case. Light line is the response of the identified system. The dark line shows the response of the identifier system.

4.2.8.2 Fourth Order System

The fourth order system is a linear system related with planes. The information about the system is in Appendix C. Transfer function for the system is defined as $T(s) = \frac{-4.553s^3 - 47.21s^2 - 8.875s + 5.65}{s^4 + 9.419s^3 + 14.01s^2 + 48.07s + 0.4271}$. A unit step input for ten seconds is given to the system. An output is obtained. The input for the genetic programming was again a unit step of ten seconds.

There are two simulations made for this fourth order system. In the first application only rules of genetic programming is used. No further optimization for the parameters is made. So the parameters of the operations remain unchanged

throughout the simulation. No limitation for the string lengths is used so huge string results are obtained. The parameter vectors are not shown since they are very complicated. The obtained results for the best individual are as follows:

The string vector:

'GGTMMM-TN-aDGTMM-+a-aMa+NFT-aMa-GMS+NNSFGM+GaG-NS-aMN-MaaDM+aTMMM--Da+a-MaGG-aM+M-MaaT--aaa+DNFT-aMDM-aa-GaNGSGT-aMFMNSaMF-aMN-MM-Da+-aMGTMa-GaDaaNGTMFa+D-MSTFFMM-aaa-GaNG-ST--aMDMDTN-aDMGSGTMM-DT-Da+a-aMa+DNFNMMGa-GTMFaNGTMG-aMDMMFGMDMaaNGMDMT-aMFMGNSa'

Square error of the first individual = 2.73

Figure 124 shows the responses of the identified and the identifier system for 10-second duration.

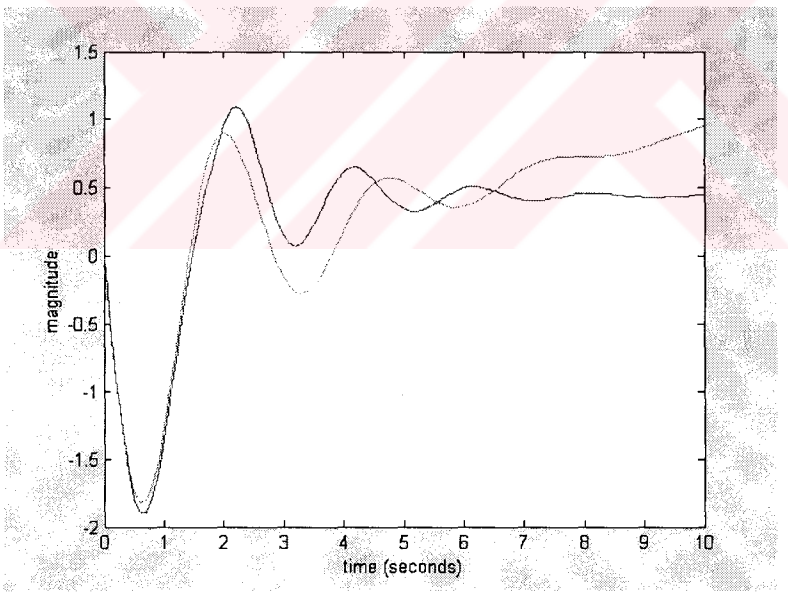

Figure 124: Response of two systems for the fourth order case. Light line is the response of the identified system. The dark line shows the response of the identifier system.

In the second simulation after every five generations steepest descent algorithm is used for all the individuals. By this way parameter optimization becomes possible. But parameters are tuned in such a way that instability in the system is not allowed. The steepest descent algorithm is evaluated for ten steps for all of the individuals but the step size for each step is not optimized by using one dimensional search algorithms since it slows down the simulation drastically. Lengths of the individuals are limited so that occurrence of huge individuals is not allowed. The obtained results for the best individual are as follows:

The string vector: (G, -, T, D, G, T, M, a, M, M, T, +, a, M, T, +, a, a)

Parameter vector-1:

(2.4676, 0, 3.0106, 1, 0.9130, 2.2626, 0.8255, 0, 0.6112, 0.6112, 2.6654, 0, 0, 2.5338, 5.9159, 0, 0, 0)

Parameter vector-2: (0, 0, 3.4828, 0, 0, 0.2795, 0, 0, 0, 0, 0.1670, 0, 0, 0, 5.6575, 0, 0, 0)

Parameter vector-3: (0, 0, 9.5019, 0, 0, 6.0383, 0, 0, 0, 0, 4.6307, 0, 0, 0, -0.7085, 0, 0, 0)

Square error of the first individual = 2.69

Figure 125 shows the responses of the identified and the identifier system for 10-second duration.

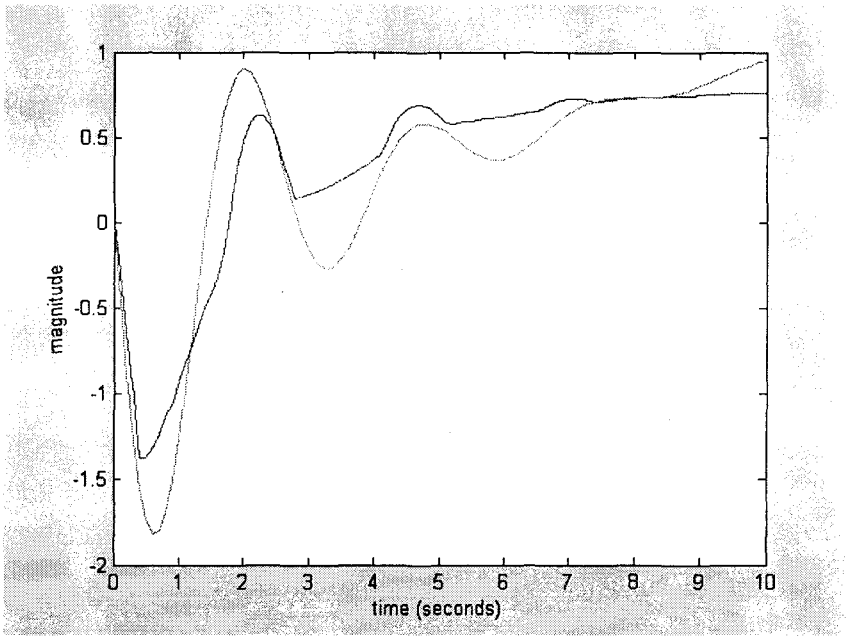

Figure 125: Response of two systems for the fourth order case with optimization of the parameters. Light line is the response of the identified system. The dark line shows the response of the identifier system.

When the responses of two different simulation techniques are compared, there is not so much difference between the obtained results when the error and fitness values are concerned. More realistic results are obtained for the second case since optimization is applied to the parameters and lengths of the individuals are limited avoiding huge string occurrences in the second case. The individual obtained in the first case is so big to be set by practical considerations but the individual for the second case can be formed. Optimization of parameters and limitation of individual structure are main driving forces for the simulation of the fourth order system.

4.2.8.3 Non-Linear System

The non-linear system is a non-linear dynamical relation related about planes. The information about the system is given in Appendix D. The system takes three inputs. Two of the inputs are not changed throughout the simulation. A unit ramp is given for the third input for 10 seconds. Initial value of the unit ramp is 0 and final value is 2. The slope of the unit ramp is 0.2 sec^{-1}. The input for the genetic

programming is the same unit ramp input. Optimization of parameters is not used for the search but the individual lengths are limited. The obtained results for the best individual is as follows:

The string vector: (+, S, +, D, D, D, D, a, S, S, M, -, +, T, F, M, S, M, D, D, D, D, D, D, a, a, M, D, M, D, D, D, D, a, -, M, S, +, D, a, S, S, M, a, a)

Parameter vector-1: (0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1.0103, 0, 0, 0.0095, 8.8088, 0.4269, 0.9317, 1, 1, 1, 1, 1, 1, 0, 0, 0.9317, 1, 0.9317, 1, 1, 1, 1, 0, 0, 0.4269, 0, 0, 1, 0, 0, 0, 1.0103, 0, 0)

Parameter vector-2:(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -0.0678, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Parameter vector-3: :(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3.7460, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Square error of the first individual = 1.09

Figure 126 shows the responses of the identified and the identifier system for 10-second duration.

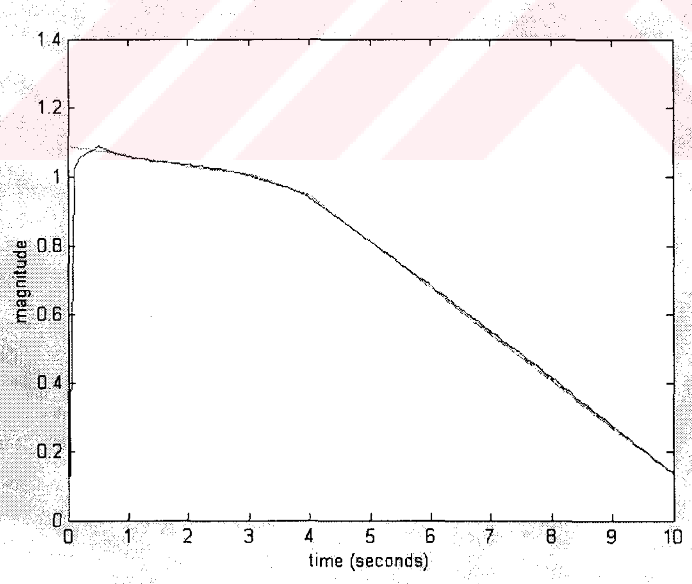

Figure 126: Response of two systems for non-linear case. Light line is the response of the identified system. The dark line shows the response of the identifier system.

As seen the identifier system suits the identified system well. The error is big in the first few seconds of the responses but then systems show nearly the same behavior as time passes.

4.2.9 Conditions and Assumptions for the Validity of the Simulations

All the systems investigated in the simulation are dynamical systems. This means the input-output relations for the systems are all time dependent. The inputs given to the system are functions of time and the outputs obtained are functions of time as a result. Modeling dynamical system blocks by genetic programming may have some drawbacks without knowing initial conditions or the states of the system. The initial conditions are sure to influence the response of the system and the states may not be as desired at the end of the simulation since only the output of the system is considered in order to minimize the error. But while simulating the systems, all the initial conditions and the states are taken to be at the origin and long time stability condition is assumed for the non-linear simulation in section 4.2.8.3 in order to diminish the effect of these situations. The obtained results are satisfactory for all the simulation types as long as these conditions are satisfied.

CHAPTER 5

CONCLUSION

Nature, with its complex structure, lets some organisms to live and some organisms to die and extinct. There is a huge competition between all the living organisms in the nature. The stronger and adaptive organisms are more likely to continue their kind while the weak ones have less chance to survive. Still in the same kind of organisms, competition among its kind may be encountered. In some animals, two offspring of the same mother may even fight because existence of another baby reduces the chance for living for both of them. So there can be competition between same types of animals too. The stronger animal will have more chance to live while the other one will not be able to continue. What determines who to survive is the nature and its means. If a kind is talented to transfer its good facilities to its future generations and if the specialities of that kind are suitable to survive in the nature, that kind will be able to continue its existence. This is how genetic programming works. Compose a group of initial programs to solve the stated problem. Give better chance for the programs that solve the problem more effectively. Use the means of nature and obtain a new generation. This is how all the search goes on. All need to be done is to investigate the problem carefully and to choose the necessary and sufficient tools to solve the investigated problem.

Genetic programming is applied to two problems in this study. The first problem is a strategy optimization. It is about to find the optimal strategy to do

some predefined tasks. Agent type structures, which are controlling the robots in a closed region, are evolved to overcome the difficulties they encounter while they do the tasks assigned to them. Means of communication is possible between these agents, which increased the performance of the programs. Generally, genetic programming is able to solve the problem more efficiently when communication is provided between the computer programs. One of the drawbacks in this kind of simulation is the huge time consumption of the simulations. Because of the structure of genetic programming, one has to deal with many programs since search starts with a population of programs. Evaluation of each program increases time consumption and practically slows down the search. Parameters of the search have important effects for convergence. Mutation, crossover and reproduction rates, maximum allowed size of tree structures are important since they directly influence the convergence of the search. Population size is chosen maximum 100 for the robot navigation problem since bigger values will cause too much time consumption. As the result of the search, expected answers are obtained towards the solution of the problem.

Second problem is about system identification. There are several ways to identify a system. One way is to compare the data taken from the system with the identifier system. If the data is close to each other the identifier system can be used instead of the identified system. When genetic programming is used, satisfactory results are obtained for second order and non-linear systems. But an extra parameter optimization is needed for a forth order system since it has a more complex structure. Drawbacks encountered in the simulations are similar to the ones encountered in the first problem. Still the time consumption is high and genetic programming parameters like crossover rate and maximum allowed size for tree structures must be chosen suitably. But the main issue of this problem is that since the operations may take the system to instability, the simulation may terminate by an error message. So the parameters for the system blocks must be chosen such that the response stays in the stability range. In the case where steepest descent

algorithm is used, this issue becomes more important since the parameters of block diagrams may make the system unstable so some care has to be taken.

Another important necessity while applying genetic programming to system identification is that the system to be identified isn't known actually. Only if we have structural knowledge about the system, we can use parameter-based methods like genetic algorithm and other classical optimization methods. There would be no need for genetic programming if the system structure were known. In that case the only problem would be tuning of the parameters of the system, which is the issue of other search algorithms. When complex non-linear system identification is considered the situation becomes more complicated. Generally (but not for our non-linear system) non-linear systems are more complicated than the linear ones and their structure are so detailed to be discovered by parameter optimization. Identification and modeling of these systems needs expert knowledge and only tuning of some parameters may be insufficient to identify these systems. So simultaneous usage of parameter tuning and genetic programming is an effective way. If only genetic programming is used, the result may become a desirable one but usually a fine-tuning of some parameters are required. Optimization of parameters only can't solve the problem since there is no priori information about the structure of the system to be identified. General conclusion from this issue is that genetic programming is a more powerful tool than genetic algorithm when a new structure is being constituted from its sub elements but genetic algorithm is more suitable with respect to genetic programming when parameter optimization of a known system structure is needed. Although they use nearly the same procedures, genetic algorithm and genetic programming are efficient for different problem types.

Genetic programming is more effective when parallel processing is applied. Long function evaluation durations can be reduced by parallel processing which is a necessity for this time consuming machine learning technique.

When two problems are compared, the first one is more suitable for using genetic programming. Firstly the problem is a strategy optimization problem and

genetic programming is a powerful technique for these kinds of problems. Rule-based 'If then else' type statements evolved easily to accomplish the task given in the problem by the methods of genetic programming. Secondly, for the second problem system parameters have an enormous effect on the performance. The parameters must be adjusted in order to reach a desired solution. The tuning of parameters can't be made easily by means of genetic programming since it is not suitable on parameter optimization problems. Another important thing is that, system identification is not just comparing the response of two systems. The initial states of the system are important too. If system identification is made, the initial positions of the states must also be considered. In assumptions, the initial states are taken to be zero and the system is assumed to be in long-term stability condition, which are not the case always. For these reasons genetic programming is more suitable for the first problem.

For the future work the robot navigation problem will be investigated in a different way. The agent structures will be divided into two separate parts where one part will be responsible from controlling the robot and the other part will be a message-sending branch. This new idea will probably increase the robustness of structures since actions of the robots will be controlled by two different strategies for each agent and time consumption will probably be diminished since work division may become distinguishable by this manner. Another idea is to form a master agent, which has a control over the other agents controlling the robots so conflicts between the slave agents can efficiently be solved.

In this thesis only system identification depending on the response is discussed. But systems are too complex to be identified by only their responses to a known input. What have to be done is to turn the discussion to the states of the system. Identification of a system must depend on the states of that system in reality not only responses to inputs. This issue is one of the most important aspects of future works for system identification by genetic programming.

REFERENCES

[1] J. Koza, Genetic Programming, On the Programming of Computers by means of Natural Selection, MIT Press, 1992.

[2] P. J. Angelina, K. E. Kinnear, Advances in Genetic Programming II, MIT Press, 1996.

[3] H. Iba, Evolutionary learning of communicating agents, in: Journal of Information Sciences, vol. 108, 1998.

[4] T. Fogarty, L. Bulls, B. Carse, Evolving Multi Agent Systems, in: G. winter, J. Periaux, M. Galan, P. Cueta (Eds.), Genetic Algorithms in Engineering and Computer Sciences, Wiley, Chichester, 1995.

[5] T. Haynes, R. Wainwright, S. Sen, Evolving a team, in: Working Notes of the AAAI-95 Fall Symposium on Genetic Programming, AAAI Press, 1995.

[6] S. Luke, L. Spector, Evolving Teamwork and Coordination with Genetic Programming, in: Genetic Programming, MIT Press, 1996.

[7] H. Iba, Multiple-agent learning by genetic programming, in: ICML96 Workshop on Evolutionary Computation and Machine Learning, 1996.

[8] H. Iba, Emergent cooperation for multiple agents using genetic programming, in: Parallel Problem Solving from Nature IV (PPSN96), 1996.

[9] H. Iba, T. Nozoe, K. Ueda, Evolving communicating agents based on genetic programming, in Proc. of the IEEE International Conference on Evolutionary computation (ICEC97), 1997.

[10] H. Iba, Multiple-agent learning for robot navigation task by genetic programming, in: Proceedings of the genetic Programming Conference (GP97), 1997.

[11] G.M. Werner, M.G. Dyer, Evolution of communication in artificial organisms, in: C.G. Langton, C. Taylor, J.D. Farmer, S. Rasmussen (Eds.), Artificial Life, vol. II, Addison-Wesley, New York, 1991.

[12] T. Ito, H. Iba, M. Kimura, Robot programs generated by genetic programming, Japan Advanced Institute of Science and Technology, IS-RR-96-0001I, in: Genetic Programming, 1996.

[13] G.J. Gray, D.J.M. Smith, Y. Li, K.C. Sharman, T. Weinbrenner, Non-linear model structure identification using genetic programming, in: Control Engineering Practice, vol. 6, 1998.

[14] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.

[15] D. Jong, A. Kenneth, Genetic algorithms: A 10 year perspective. In Grefenstette, J. John (editor), Proceedings of an international Conference on Genetic Algorithms and their Applications, 1985

[16] D. Jong, A Kenneth, On using genetic algorithms to search program spaces. In Grefenstette, J. John (editor), Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms 1987.

[17] D. Jong, A Kenneth, Learning with genetic algorithms: an overview. Machine Learning 3, vol 2, pages 121-138, 1988.

[18] H.J. Antonisse, K.S Keller, Genetic operators for high-level knowledge representation. In Greffenstette, J John (editor), Genetic Algorithms and their Applications: Proceedings of the second International Conference on genetic Algorithms, 1987.

[19] H.J. Antonisse, A grammar-based genetic algorithm, Gregory (editor), Foundations of genetic algorithms. Morgan Kaufmann, 1991.

[20] A.S. Bickel, R.W. Bickel, Tree structured rules in genetic algorithms. In Davis, Lawrance (editor), Genetic Algorithms and Simulated Annealing, Pitmann 1987.

[21] J. Koza, Genetic Programming II, Automatic discovery of reusable Programs, MIT Press, Cambridge, MA, 1994.

[22] G.J. Gray, T. Weinbrenner, D.J. Murray-Smith, Y. Li and K.C. Sherman. Issue in non-linear model structure identification using Genetic Programming. In Proceedings of the Second International Conference on Genetic Algorithms in engineering Systems: Innovations and Applications (GALESIA 97), Conference Series, 446, pages 308-313, Institution of Electrical Engineers. 1997.

[23] G.J. Gray, D.J. Murray-Smith, Y. Li and K.C. Sherman. Structural system identification using genetic programming and block diagram oriented simulation tool. Electronic Letters, 32 (15), pages 1422-1424, 1996.

[24] K. Godfrey, Perturbation Signals for System Identification. Prentice Hall International, U.K. (1993).

[25] L. Ljung, System Identification – Theory for the User, Prentice-Hall, Englewood cliffs, New Jersey, USA, 1987.

[26] B. Mckay, M. Willis, H. Hiden, G. Montagua and G. Barton. Identification of industrial processes using Genetic Programming. In Proceedings of Identification in Engineering Systems, Swansea, U.K 1996.

[27] D. Murray-Smith. Advances in simulation model validation: theory, software and applications. In F. Breitenecker and I. Husinsky, EUROISM'95 Simulation Congress, Pages (75-84), Amsterdam, The Netherlands. Elsevier 1995.

[28] D. Murray-Smith, Continuous System Simulation. Chapman and Hall, London, 1995.

[29] http://liinwww.ira.uka.de/bibliography/Ai/genetic.programming.html

[30] ftp://ftp.cs.bham.ac.uk/pub/authors/W.B.Langdon/biblio

[31] Majordomo@lists.Stanford.EDU (send a message 'subscribe genetic-programming' in order to subscribe to the mailing list)

APPENDIX A

MINIMIZATION OF TREE STRUCTURES FOR AGENTS

The big agent tree structures can be minimized. Some building blocks of the agents are useless because some of the 'if than else' blocks in agent's tree structure takes two identical terminals, which in a way annihilate the effect of that operation. If the tree structures at Figures 127 and 128 are considered, it can be told that both tree structures do the same thing.

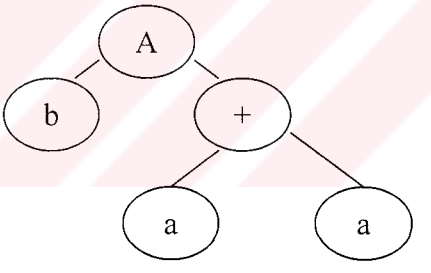

Figure 127: A tree structure for an agent

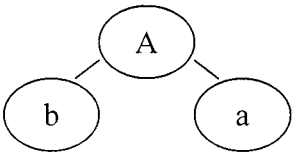

Figure 128: Minimal tree structure doing the same job as the tree structure in Figure 127

Both tree structures do the same job since the operation '+' (if there is an obstacle in front of the agent operation) takes two terminals which will yield the terminal 'a' whether there is an obstacle in front of the agent or not. Another way where minimization of the tree structure is possible can be seen for the tree shown in Figure 129.

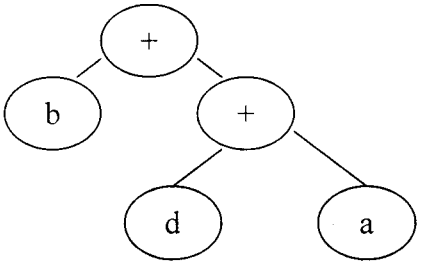

Figure 129: A tree structure for an agent

The tree in Figure 129 can be in another tree structure as in Figure 130. There is no need to use two '+' (if there is an obstacle in front of the agent operation) twice. If ones answer is 'no' others will be no too. So one of the '+' operation and the terminal 'd' can be killed to give the minimal structure at Figure 130.

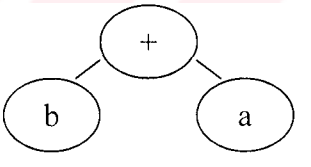

Figure 130: The minimal tree structure of the agent in Figure 129

APPENDIX B

OPERATIONS OF SYSTEM IDENTIFICATION PROBLEM

Square Root

Example 5 below shows input vector and obtained output vector as the result of square root taking operation.

U= [1, -4, 9, -16, 25] (the data starts at when t=0 seconds ends when t=0.4 seconds. Time elapse between each data is 0.1 seconds)

O= S (U)= [1, 2, 3, 4, 5] (the output obtained as the result of the input for square root taking operation

Example 5: Square root operation. 'U' is the input vector, 'O' is the output vector obtained as the result of square root taking operation to input vector.

Gain

Example 6 shows input vector and obtained output vector as the result gain operation.

U= [1, -4, 9, -16, 25] (the data starts at when t=0 seconds ends when t=0.4 seconds. Time elapse between each data is 0.1 seconds)

Gain-factor K =2

O= G (U)= [2, -4, 18, -32, 50] (the output obtained as the result of the input for gain operation

Example 6: Gain operation. 'U' is the input vector, 'O' is the output vector obtained as the result of gain operation to input vector.

Addition

Example 7 shows input vectors and obtained output vector as the result of addition operation.

U= [1, 2, 3, 4, 5] (the data starts at when t=0 seconds ends when t=0.4 seconds. Time elapse between each data is 0.1 seconds)

G= [0, 1, 0, 1, 0] (the data starts at when t=0 seconds ends when t=0.4 seconds. Time elapse between each data is 0.1 seconds)

O= + (U, G)= [1, 3, 3, 5, 5] (the output obtained as the result of the inputs for addition operation

Example 7: Addition operation. 'U' and 'G' are the input vectors, 'O' is the output vector obtained as the result of addition operation to input vectors.

Subtraction

Example 8 shows input vectors and obtained output vector as the result of subtraction operation.

U=[1, 2, 3, 4, 5] (the data starts at when t=0 seconds ends when t=0.4 seconds. Time elapse between each data is 0.1 seconds)

G=[0, 1, 0, 1, 0] (the data starts at when t=0 seconds ends when t=0.4 seconds. Time elapse between each data is 0.1 seconds)

O=-(U, G)= [1, 1, 3, 3, 5] (the output obtained as the result of the inputs for addition operation

Example 8: Subtruction operation. 'U' and 'G' are the input vectors, 'O' is the output vector obtained as the result of subtraction operation to input vectors.

Delay

Example 9 shows input vector and obtained output vector as the result of delay operation.

U= [1, 2, 3, 4, 5] (the data starts at when t=0 seconds ends when t=0.4 seconds. Time elapse between each data is 0.1 seconds)

O= D (U)= [0, 1, 2, 3, 4] (the output obtained as the result of the input for delay operation)

Example 9: Delay operation.'U' is the input vector, 'O' is the output vector obtained as the result of delat operation to input vectors.

Saturation

Example 10 shows input vector and obtained output vector as the result of delay operation.

U= [1, 3, 5, 5, 7, 8] (the data starts at when t=0 seconds ends when t=0.5 seconds. Time elapse between each data is 0.1 seconds)

Saturation parameter=6

O= M (U)= [0, 3, 5, 5, 6, 6] (the output obtained as the result of the input for saturation operation)

Example 10: Saturation operation. 'U' is the input vector, 'O' is the output vector obtained as the result of saturation operation to input vectors.

First Order Operation

Example 11 shows input vector and obtained output vector as the result of this operation.

U= [1, 1, 1, 1, 1] (the data starts at when t=0 seconds ends when t=0.4 seconds. Time elapse between each data is 0.1 seconds)

Pole parameter a=1

First order system= $\frac{1}{s+1}$

O= F (U)= [0, 0.0952, 0.1813, 0.2592, 0.3297] (the output obtained as the result of the input for the first order transfer function operation)

Example 11: First oerder operation. 'U' is the input vector, 'O' is the output vector obtained as the result of first order operation to input vector.

Second Order Operation without Zero

Example 12 shows input vector and obtained output vector as the result of this operation.

U=[1, 1, 1, 1, 1] (the data starts at when t=0 seconds ends when t=0.4 seconds, Time elapse between each data is 0.1 seconds

Natural frequency w=6

Damping ratio ∂=0.5

Second order system= $\frac{1}{(s^2+6s+36)}$

O= N (U) =[0, 0.0040, 0.0125, 0.0211, 0.0276]

Example 12: Second order operation without zero. 'U' is the input vector, 'O' is the output vector obtained as the result of second order operation to input vector.

Second Order Operation with Zero

Example 13 shows input vector and obtained output vector as the result of this operation.

U=[1, 1, 1, 1, 1] (the data starts at when t=0 seconds ends when t=0.4 seconds, Time elapse between each data is 0.1 seconds)

Natural frequency w=6

Damping ratio ∂=0.5

Zero of the system k=1

Second order system= $\frac{s+k}{(s^2+6s+36)}$

O= N (U) =[0, 0.0748, 0.1035, 0.0994, 0.0783]

Example 13: Second order operation with zero.'U' is the input vector, 'O' is the output vector obtained as the result of second order operation to input vector.

APPENDIX C

LINEAR PLANE SYSTEM

The state space representation of the system is as follows:

$$\begin{bmatrix} x_1'(t) \\ x_2'(t) \\ x_3'(t) \\ x_4'(t) \end{bmatrix} = \begin{bmatrix} -0.259 & 0 & -1 & 0.182 \\ -16.02 & -8.4 & 2.19 & 0 \\ 4.488 & -0.35 & -0.760 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} + \begin{bmatrix} 0 & 0.0723 \\ -0.7577 & 23.169 \\ *0.2213 & -4.553 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a(t) \\ u(t) \end{bmatrix}$$

$$y(t) = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} \qquad (16)$$

Here $a(t)$ is taken as a zero input. In s-domain the input output relation of the system becomes as follows:

$$\frac{Y(s)}{U(s)} = T(s) = \frac{-4.553s^3 - 47.21s^2 - 8.875s + 5.65}{s^4 + 9.419s^3 + 14.01s^2 + 48.07s + 0.4271} \qquad (17)$$

In the simulation unit step input is given to the system for ten seconds and an output is obtained as a result.

APPENDIX D

NON-LINEAR PLANE SYSTEM

The non-linear system encountered is a three input one output dynamic system. Two inputs of the system are assumed to be in long time stability condition. The constant inputs are the power level and the height of the plane. Power level is taken as 40 and the height is taken as 1000 meters. The velocity of the plane is increased from 0 to twice the speed of sound in air in ten seconds and an output is obtained. The given input in the simulation is a ramp input starting from 0 and reaching 2 in ten seconds. The response of the system to the input is obtained as the result.