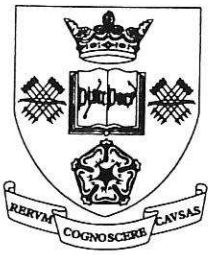
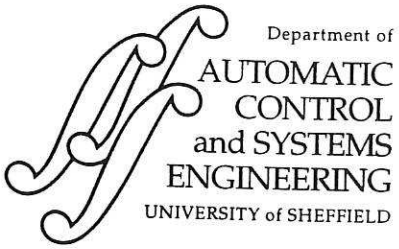
Monograph:
Swain, A.K. and Zalzala, A.M.S. (1998) An Overview of Genetic Programming:Current Trends and Applications. Research Report. ACSE Research Report 732 . Department of Automatic Control and Systems Engineering

---





AN OVERVIEW OF GENETIC PROGRAMMING: CURRENT TRENDS AND APPLICATIONS

A.K. Swain and A.M.S. Zalzala

Robotics Research Group,
Department of Automatic Control and Systems Engineering
The University of Sheffield
Mappin Street, Sheffield S1 3JD, United Kingdom

Research Report #732
November 1998

RRG
Tel : +44 (0)114 2225250
Fax : +44 (0)114 2731729
EMail : rrg@sheffield.ac.uk
Robotics Research Group

AN OVERVIEW OF GENETIC PROGRAMMING: CURRENT TRENDS AND APPLICATIONS

A.K. Swain and A.M.S. Zalzala

Robotics Research Group,
Department of Automatic Control and Systems Engineering,
University of Sheffield, Mappin Street, Sheffield S1 3JD, UK.

Tel: 0044 114 2225136; Fax: 0044 114 2731729
Email: rrg@sheffield.ac.uk

Abstract

Genetic programming (GP) as an automatic programming method has been rapidly gaining more attention of researchers in the last few years. GP techniques rooted in biological evolution started effectively in early 90's with an attempt to evolve computer programs without being explicitly programmed to do so. This paper provides an extensive survey of genetic programming technique. Initially this paper deals with the fundamental theoretical issues associated with GP and latter this emphasises on the diverse fields of its applications.

Keywords - *Genetic programming, automatic programming, genetic algorithm, evolutionary computing.*

1 Introduction

Evolutionary computing approaches, based on the theory of natural evolution, have been studied rigorously in the last two decades. Besides the three prominent evolutionary approaches: genetic algorithm (GA) [1, 2], evolutionary strategy (ES) [3, 4] and evolutionary programming (EP) [5, 6], genetic programming (GP) [7, 27] is currently a rapidly emerging field of research. GP is a stochastic adaptive search technique in the field of automatic programming (AP) to generate computer programs. Automatic generation of computer programs to solve a specific problem without the knowledge of exact solution procedure was the basic thought of Arthur Samuel and R. Friedberg in late fifties [8,9]. Early works on the evolution of computer programs [10-12] using the most acclaimed GA techniques were buried under the popularity of massive GA banner. John Koza [7] represented the complex structures of a computer program by a variable length non-linear tree structure, rather than the fixed length bit string representation of standard GA. Then, usual GA operators such as selection, crossover and mutation are used to evolve these tree structures. He only coined the name GP for this new research area. His immense effort made GP to acquire an important place in the broader field of evolutionary computation [13-20]. Essentially, a GP searches for the most correct computer program from the whole space of computer programs. Search progresses with most fit individual computer program reproducing more often and generating increasingly more fit new computer programs (offspring) using crossover and mutation operators as in conventional GA. Since the development of this new field of automatic programming, not many up-to-date survey papers have been published except very few [21-24]. In this paper a positive effort has been taken to cover the current trends and techniques along with the various applications of GP.

2 Genetic Programming

Genetic programming is a major variation of GA to automatic generation of computer programs, which are evolved to solve or approximately solve problems [7, 25-27]. The evolving individuals are themselves computer programs represented by tree structures. Tree structure is not the only way of representing the GP individuals; other types of representations are discussed in section 3.1. In tree representation for GP, the individual programs are represented as rooted trees with ordered branches. Each tree is composed of functions as internal nodes and terminals as levels of the problem. Then it is required to generate syntactically correct programs by the use of any programming language like LISP, C, and C++, which represent programs internally as parse trees. Koza selected LISP programming language which has the property that the functions can easily be visualised as trees with syntax in prefix form, and the syntax is preserved by restricting the language to suit the problem with appropriate number of constants, variables, statements, functions, and operators. A particular problem is solved by this restrictive language formed from user defined *terminal set* T and *function set* F. Each function in the function set must be able to accept gracefully as arguments the return value of any other function and any data type in the terminal set. This is known as *closure* property. The functions and terminals are selected a priori in such a way that they will be useful to get the solution for the problem at hand. For solving a particular problem a GP requires five major preparatory steps [7, 21]. These are selection of the

- terminals
- functions
- fitness function
- control parameters of the system such as population size, maximum individual size, crossover probability, mutation probability etc.
- termination conditions

GP starts with an initial population of randomly generated tree structured computer programs, as discussed above. Fitness is assigned to each individual program, which evaluates the performance of the individual on a suitable set of test cases. Individuals are selected based on their fitness and

the selected individuals are allowed to survive. Then, crossover creates offspring by exchanging subtrees between two parent trees at selected random crossover points. An example of this crossover mechanism has been illustrated in Fig. 1. Other genetic operators such as mutation, permutation, editing and a define-building block operation [29], are seldom-used [7]. The GP procedure outlined above is shown in Fig. 2. To understand the working of a GP a very simple example is given below.

2.1 A Simple Example of GP:

As an example of GP, consider the problem of symbolic regression for prediction. For this task a set of data points have been provided, and the job is to find the underlying functional relationship in symbolic terms, which then can be used to predict for a new set of input values.

Suppose given a set of 20 data points in terms of x and y co-ordinates of a simple function $y = 1.31\ x^2 + 1.50\ x$.

The preliminary preparatory steps discussed above are decided as follows:

- Terminal set - **T** = {x, R}

where variable x is set a particular value during training and R is a random floating point number between -2.0 and 2.0, i.e., whenever the terminal element R is selected as a node then at that point a random number between -2.0 and 2.0 is inserted into the tree as the terminal node value.

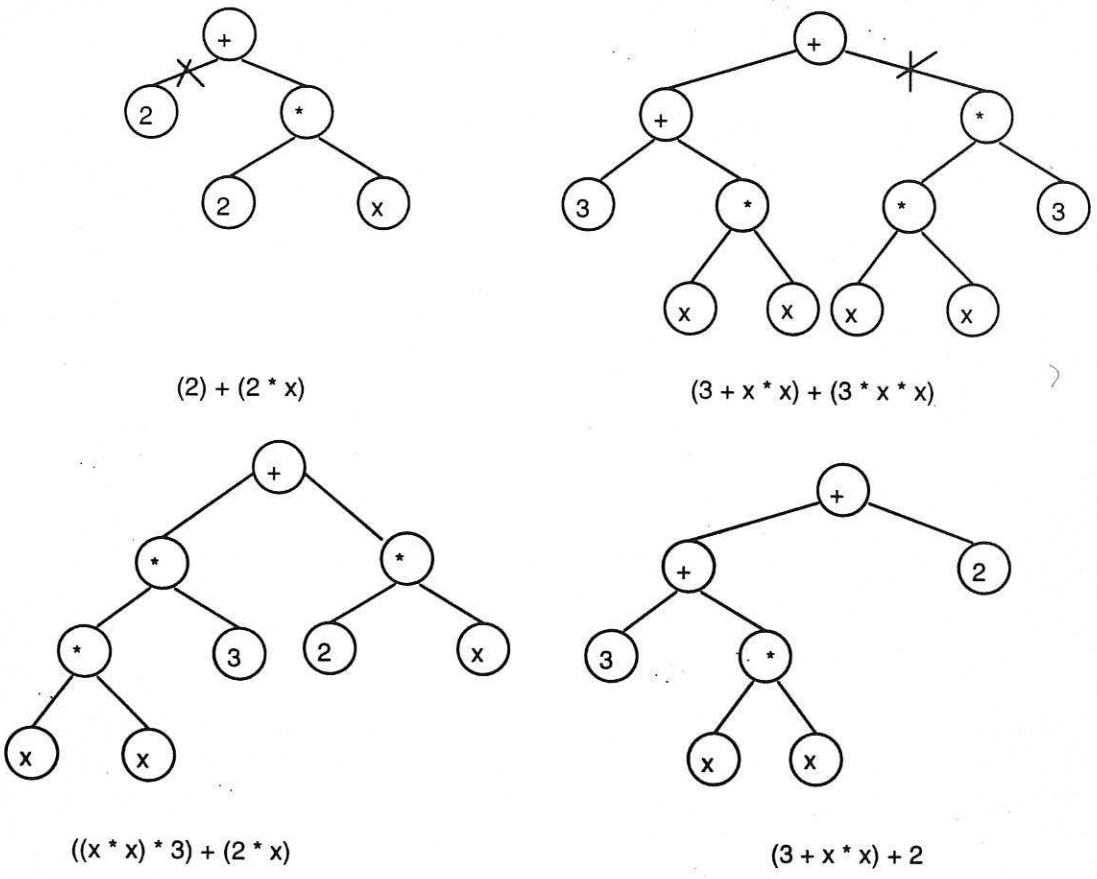


Fig. 1. An example of sub-tree crossover in GP. Crossover sites are shown with cross marks on the parent tree.

Generational GP Algorithm

Step - 1. Initialise the population

Step - 2. Evaluate the individuals using a fitness function.

Step - 3. Create a new population pool by

a) selecting individuals using any selection algorithm

b) altering the selected individuals by applying different genetic operators

Step - 4. If the termination criterion not satisfied Then

repeat Steps 2 and 3

else output the single best individual of the entire run.

Fig. 2. The structure of a generational GP algorithm.

- Function set - F = {+, -, *, %}
- Fitness function: $0.5 \sum_{i=1}^{20} e_i^2$

where e_i is the difference between the actual value of y_i predicted value of output for a given input x_i. A total of 20 fitness cases have been chosen between the interval [-1, 1].

- Control parameters: Set as per the training requirement.
- Termination criteria: Either maximum number of generations reached or mean squared error value less than a small constant

After defining the five preliminary steps the actual GP process will be progressed as per the steps outlined in Fig.2.

3 Design Issues for Genetic Programming

In spite of the remarkable growth of GP in the last few years, there is still ample of scope for research in this field. Many of the techniques used in GP research have been inspired by the works done in the field of GAs [22]. Still GP is considered to be in its infancy, and needs to be explored in more detail. In this section many important design issues required for the successful implementation of GP have been discussed.

3.1 Program Representation

Program representation issues have been dealt with rigorously by GP community [7, 30-33]. Few type of representations have been suggested by different researchers. Out of which tree structure is very common and most frequently used [7]. Iba et. al. [34, 35] has proposed a tree representation where function nodes are simple polynomials of the two input variables. Quite a many GP research involves linear representation. Cramer [11] has used JB language which is similar to machine codes with linear prefix description for program evolution. Also, he has experimented with linear postfix descriptions. Perkis [36] reports some benefits of linear postfix language over standard tree representation. Banzaf [37], Keller and Banzaf [31], and Wineberg and Oppacher [38] have used linear representations, which can be extended to tree genome for program evolution. The most surprising and successful linear representation is due to Nordin [39], and

latter extended by Nordin and Banzaf [40]. He has used small machine code instructions to describe the linear genome, which is claimed to be more than thousand times faster than LISP implementation. Relatively recently Teller and Veloso [41], and Teller [42] described a graphical structure for GP, which is known as PADO (parallel algorithm discovery and orchestration). It has been claimed that PADO's performance for signal classification is better than that of S-expressions as in LISP and automatically defined functions (ADFs) (ADFs are described in section 3.4.1).

The developmental system of Gruau [43-45], known as cellular encoding, is a graphical representation for designing neural networks and other network applications such as electrical circuits and finite state automata. Jacob [46, 47] has proposed a graph based structure evolution, commonly called L-system. Langdon [48] has used abstract data type structures for evolution of lists, ques and stacks. He has used a chromosome structure with multiple trees, where each tree represents one of the required abstract data type member functions. Whigham [49-50] has proposed a context-free grammar structure, which satisfies the syntactic closure requirement for GP. In context-free grammar GP (CFG-GP), the programs are represented by a derivation tree with a starting symbol S; whose internal nodes are rewrite rules and terminals are usual GP functions and terminals. A more general context sensitive grammar based GP has been reported by Wong and Leung [51]. Handley introduced a directed acyclic graph (DAG) which reduces, in certain cases, the amount of memory for representing the population pool [53]. A special case of DAG is binary decision diagram (BDD), which deals with only Boolean inputs and outputs [54, 55]. Modularised representation of tree structure adds many advantages to the system, which are discussed in section 3.4.

All these representations do not adhere to any specific execution or working order, which shows the great flexibility of GP towards program representation. This flexibility in structure selection makes the genetic operators (discussed in section 3.2) to be different from one structure to other. This necessitates the development of a generalised structure for GP.

3.2 Genetic Operators

There are three operators that are mainly used for the progress of GP evolution. They are:

- Crossover
- Mutation, and
- Reproduction

As tree representation is very common in the implementation of GP, so all our discussions will mostly concentrated on this type of representation. Hence, unless other wise mentioned above operators will refer to tree representations only. However, during the discussion of operators for other type of representations will explicitly be mentioned in the text.

3.2.1 Crossover

It is a moderate way of producing variation during some sort of sexual process, where it suffles parental genetic information to produce new combinations of genes. In most GP implementations, crossover serves as a predominant operator [7]. The subtree crossover, initially defined by Cramer [11] and latter on refined by Koza , swaps randomly chosen subtree between two parent trees. During the implementation of this crossover, leaf selection frequency, i.e., how frequently the terminal leaves will be selected during crossover, plays very important role. Koza has suggested the value of leaf selection frequency to be as 0.1. This standard subtree crossover does not preserve any kind of context in the chromosome, which is in contrast to the standard GA. This makes the individuals in GP to converge functionally rather than lexically. D'haeseleer [56] introduced two new context preserving crossover mechanisms such as *strong context preserving crossover* (SCPC) and *weak context preserving crossover* (WCPC). In both of these methods the subtrees chosen for crossover must be from the parents at exactly same position in terms of node

co-ordinates. The SCPC in combination with standard subtree crossover resulted in superior performance compared to that of standard subtree crossover.

Altenberg [32] suggested the brood selection in GP to select for its representations with greater evolvability under recombination. This work is then utilised by Tackett [57] for the development of a new crossover operator called brood selection recombination [58]. This brood recombination operator basically recombines two parents to produce n pairs of offspring, where n defines a brood size. Then out of these n pairs of offspring only one pair is selected by brood selection mechanism. This has been inspired by the observed fact in nature that many species reproduce more offspring but that they all do not survive. Langdon [59] has devised a crossover mechanism known as *directed crossover*, which dynamically redistributes crossover locations as population evolves. Analogous to one-point and uniform crossover operators in GA, Poli and Langdon [60,61] have implemented two similar crossover operators for GP. In *one-point crossover* for GP, a common crossover point is selected randomly between two parents by aligning both the parent trees with each other and then swapping the subtrees at that location. This has got a great similarity with SCPC discussed above, with the exception that it is more restrictive towards the selection of the crossover points. On the other hand, the *uniform crossover* in GP follows the exact steps of its implementation as in binary coded GA. Here two offspring are generated from two aligned parent strings by swapping the selected nodes. Poli and Langdon [60, 61] have also reported two variants of one point and uniform crossover operators in the context of GP. They are *strict one-point and strict uniform crossover* operators. In strict-one point crossover mechanism, the crossover point must be located on the parts of the parent trees which are exactly same, i.e., say for example, the same node functions. Similarly, in the case of strict uniform crossover the nodes swapped must be located on exactly same part of the parent trees.

Harries and Smith [62] have experimented with five different variants of tree depth based crossover operators. They compared the performance of all these operators with that of standard subtree crossover on a test suit consisting of four different problems. Iba *et* al. [63] have proposed a recombinative guidance for the standard GP crossover operator. This operator is one type of intelligent crossover operator, which is adaptive semantically. This adaptiveness is achieved through the performance values calculated for each subtrees of a GP tree. These performance values are called 'S-values'. A very impressive crossover operator, called constrained complexity crossover, has been proposed by Watson and Parmee [64]. Here, the complexity of a tree and all its nodes are calculated by assigning a weighting to each functions and terminals of this tree. This is called node complexity (NC) weightings. The NC value of a node is the product of the NC value of that node with the sum of the NC values of the nodes below that particular node. This has been claimed that this operator reduces the population size by many fold, and also it reduces the memory requirements compared to that of a standard GP. This method is also called *rapid attenuated memory* genetic programming (RAM-GP). They have extended RAM-GP by incorporating subpopulation of solutions which communicate with each other via crossover, and named it as *distributed* RAM-GP (DRAM-GP) [225, 272]. To make it effective for mixed discrete and continuous search, they have proposed another extension of DRAM-GP, which is known as *hybrid* DRAM-GP (HDRAM-GP). In HDRAM-GP in addition to standard GP operators, which search the discrete functional structure, a real numbered GA has been incorporated to search the continuous coefficient space.

Zannoi and Reynolds [65] have used cultural algorithm to select crossover points with better results. Teller [42] has used an intelligent recombination operator for his graph based PADO system. This is a bit complex but is very effective for generating better offspring. Angeline [66] has devised two self-adaptive crossover operators: *selective self-adaptive crossover* (SSAC) and *self-adaptive multi-crossover* (SAMC). Both of these crossover operators are inspired by the self-adaptive mutation for mutating finite state machines in evolutionary programming [67]. SSAC guides the crossover by using an additional number of parameter tree for each individual. These parameter trees are identical in size and shape of that of program trees and they consist of real

valued numbers in place nodes and terminals of the corresponding program tree, where these real numbers represent the probabilities of crossover of a subtree at that point. The crossover point is selected by using a Roulette wheel selection method [68, 69]. Crossover takes place for both program and parameter trees at the same crossover point. Then a Gaussian random noise is added to every parameter values of the parameter tree to get an updated parameter tree. The other crossover operator SAMC proceeds with one more tree in addition to program tree and parameter tree of SSAC. This new tree, which is of same size and shape of that of the program tree, is called a *crossover map*. Crossover map keeps track of the number of crossover operations to be performed on all these three trees to generate the offspring. In SAMC, the parameter tree consists of absolute probabilities of crossover and is updated as that in SSAC. This is a better real valued implementation of explicitly defined introns (EDI) [70, 71]. An EDI is the value that is explicitly inserted between each pair of nodes, which are indication of probability of crossover between two connected nodes. EDIs are used successfully in linear genome based GP.

The effectiveness of crossover has generated much controversy in evolutionary computation field [6, 72-75]. This issue has been discussed in the next subsection under the heading of mutation. Before concluding this section it is note worthy to say that the value of the leaf selection frequency of 0.1 as suggested by Koza is recently being opposed in the work of Angeline [66]. Angeline has shown that this value of leaf frequency may not be optimal for many problems. Recently, Angeline [76] has studied this problem more rigorously, and has drawn the conclusion that it will be more productive in GP to use a randomly generated leaf frequency value in every crossover operation. This makes the crossover operation more uniform through out the run, thereby producing better offspring.

3.2.2 Mutation

Mutation is a relatively drastic way of accomplishing variation by changing the nature of the information transmitted from parent to offspring. In natural genetics, it is treated as an occasionally occurring random operations. Analogously, the standard mutation operators in a tree based GP representation select a random node in the program tree and replace the existing subtree at that node with a randomly created subtree. In all the works of Koza, mutation operator is almost excluded altogether, and the evolution of program was totally based on the crossover operator. Recent works demonstrated that the crossover operator exhibits limited performance advantage compared to purely mutation based systems [72, 77-82]. The discussion on crossover and mutation operators in this section does not meant to emphasise the superiority of one over the other, but rather it is to strengthen the theoretical understanding of the roles of these operators in simulated evolution.

Jones [83] has shown that in the absence of clearly defined building blocks the performance of standard crossover in GA is at best comparable with macromutation. In macromutation, crossover is performed between a randomly selected chromosome from the population pool and a randomly generated chromosome outside the population pool. This type of crossover is known as *headless chicken crossover* (HCC). Angeline [78] has extended this concept to GP and implemented two different variants of the HCC. These two new operators are called *strong headless chicken crossover* (SHCC) and *weak headless chicken crossover* (WHCC) [79], respectively. In both SHCC and WHCC the offspring is generated by using standard subtree crossover between parent chromosomes and randomly generated chromosomes. Then the modified parent trees are returned to the new population pool. Moreover in WHCC, there is an even probability of receiving one out of the two offspring generated after the same crossover operation that is used in SHCC. Angeline has argued that the subtree crossover does not operate as per the building block hypothesis. This macromutation is mechanically similar to subtree crossover but semantically just a variant of subtree mutation.

Chellapilla [81, 82] has proposed six tree mutation operators for program evolution. He has verified the performance of the evolved programs based solely on these mutation operators without any crossover operation on a set of nine well-suited problems. His self-explanatory six mutation operators include OneNode, AllNodes, Swap, Grow, Trunc (Truncation), and Gaussian. The last operator, i.e., the Gaussian operator perturbs the values of terminal nodes with a Gaussian random number with zero mean and variance of 0.01.

Spencer [84] has described an operator similar to mutation operator. That operator is called constant perturbation operator, which constantly changes the values of the ephemeral random constant terminal [7].

Luke and Spector [80] have done a good comparison of the roles of crossover and mutation operators in a tree based GP. He has shown that depending on the domain, mutation is better for small population over more generations, whereas, overall crossover is more successful than mutation. Also, he has shown that the difference between these two is rather small and statistically insignificant.

This is clear from the above discussions that it necessitates more research effort to be imparted to establish the exact role of crossover and mutation on simulated evolution. However, it depicts that the subtree crossover without any strong context preservation is just a variant of mutation.

3.2.3 Selection

Selection operator heavily depends on the individual fitness values. This strong dependence of selection operator on fitness necessitates the description of the works on fitness measure, before describing the operator itself. So, now before switching over to selection operators the important issues related to the fitness evaluation have been described.

Fitness value is calculated by evaluating a problem specific fitness function assigned to GP to solve a problem. Most of the GP applications are based an absolute fitness measure using fixed test cases. Whereas, in coevolution methods [85, 86, 133] the fitness evaluation is relative rather than absolute. In coevolution case the individuals compete against each other to get evolved. Usually in classification tasks [2, 87] inclusion of program size in fitness function yields better results. Reynolds [88, 89] has reported simple and robust solutions for a task of evolving programs to steer a 2D vehicle through a corridor by introducing noise into the fitness function. Andre [90] has used a two-phase fitness evaluation for each individual to effectively coevolve two programs to control the actions of a simulated robot. Multiple objectives such as program size and execution speed can be incorporated in GP fitness function to make it a multiobjective fitness function. Fitness evaluation of individuals is the most time consuming process in a GP run. This is taken care of by various methods such as stochastic sampling, dynamic subset selection and limited error fitness [8]. In image compression task, the number of fitness cases is very large, so it makes GP difficult to handle this situation. This is overcome by chunking the entire image to subimages, and then evolving separate program for each subimage [91].

After above discussions on evaluation of individual fitness now it steers our discussion towards selection operators. Type of the selection decides the speed of the evolution, and also made responsible for the premature convergence of any evolutionary process. First of all we will discuss the most basic selection mechanism, known as Fitness proportional selection. This selection method selects an individual x_i with a probability $p_i = f_i / \sum f_i$, where f_i is the fitness function of the ith individual and $\sum f_i$ is the total fitness of all the individuals in the population. It is in use with GA since its development by Holland, but recently it is being criticised for attaching differential probabilities to the absolute values of the fitness [92]. Also, under fitness proportionate selection the existence of an individual called a *super individual* which possess an

unusually highfitness value will dominate the entire population, thereby leading to premature convergence to a possibly local optimum. Suitable *scaling* of the evaluation function, partially avoids this problem. This problem of premature convergence can be tackled by the use of selection methods that do not depend on the absolute values of the fitness function. Two such methods: *rank based selection* and *tournament selection* are discussed below.

In rank based selection methods [68, 93-95], the individuals are sorted according to their respective fitness values, and the best individual always receives a predetermined multiple of the number of copies than the worst one. Linear and exponential rankings are in common use.

Tournament selection [97] is based on competition to select the best among a randomly selected subset of individuals of the whole population. This selection is recently being increasingly used, mainly because its implementation is easy and is computationally more efficient. The tournament size, which is the size of the subset, controls the selection pressure. In reverse tournament selection [32] the worst individual in the population is replaced by a newly produced individual. This method need not necessarily be a global on, which makes its use possible in a spatially distributed population with simultaneous local tournaments.

In (μ,η) evolution strategy [4], μ parents create $\eta > \mu$ offspring, out of which the best μ offspring are deterministically selected to be used as parents in the next generation. This method is not an *elitist*. This is also known as *truncation selection* method [96]. In contrast to this, in $(\mu+\eta)$ strategy both parents and offspring take part in the selection process, out of which μ individuals will be selected as the new parents for the next generation. In evolutionary programming [6] approach, a probabilistic variant of $(\mu+\eta)$ selection strategy is followed. Where the ith individual is selected after competing with randomly selected competitors of size more than one from the same population pool. Then, a pair wise comparison is made between the ith individual and each of the competitors. If the ith individual's score is better or equal to that of its opponent, it will be assigned a 'win'. Similarly, μ such individuals with greatest number of wins are selected as parents for the next generation.

Tackett and Carmi's brood selection [58] selects two offspring out of n-pairs of offspring produced by brood selection recombination operator (section 3.2.1). The criteria for selecting best two offspring are based on a culling function, which is either equivalent or a simpler version of the normal fitness function. Increase in brood factor n enhances the performance with reduced memory requirement [98].

The selection mechanisms described above except tournament selection method all operate on the entire population of individuals, which make them unsuitable for parallel or distributed architectures. Hence, local selection methods are developed [98-100]. In migration models, the entire population is divided into sub-populations called islands or demes. Each deme is a separate breeding unit and uses local selection and reproduction rules to locally evolve the species. Diffusion models take each individual as a separate breeding unit and mate selection is restricted within a small neighbourhood.

3.3 Theoretical Foundations of GP

GP lacks a sound theoretical foundation. Very few works have been reported in this area, which are described below.

3.3.1 Schema Theorem and Building Block Hypothesis

The schema theorem [1, 2] for fixed length GAs explains their workability by means of schemata (good practical building blocks). The schema increases exponentially in a population as

generation advances, and is combined with other schemata to produce good final solution. Although, the recent GA developments [83] question the validity of schema theorem, but still it serves as the only theoretical underpinning of GA. This concept has also been extended to explain the working mechanism of GP. The schema theorems in the context of GP have been discussed at length in the next subsections.

3.3.1.1 Koza's Schema Theorem

Koza [7] defines a schema as the set of sub-trees, which contain one or more specified sub-trees. For example, consider a schema 'H' of S-expressions defined by H = {(+ 5 X), (* 5 Y)}, then all the sub-trees containing either (+ 5 X) or (* 5 Y) are instances of H. In the case of GA, the schema implicitly specifies its position with bit positions, whereas above definition of schema in GP does give only its defining components. Hence, the subtrees defining a schema can appear anywhere within the structure of the individual in different ways and even multiple times.

3.3.1.2 O'Reilly's Schema Theorem

O'Reilly and Oppacher [101-103] extended Koza's schema theorem for GP described above. This schema theorem is derived in the presence of fitness-proportional selection and sub-tree crossover. The schema here is a multi-set of sub-trees and tree fragments. Tree fragments are defined using a "don't care" symbol (#). For example, the schema H = {(- # X), (* X 2)} is the set of sub-trees with at least one occurrence of each of (- # X) and (* X 2).

O'Reilly defines the *order* of a schema as the number of non-# nodes, and the *defining length* as the sum of the number of links in the tree fragments and those connect them together. The defining length of a schema is variable. The total number of links in a tree is also variable, which makes the probability of disruption of a schema H due to crossover to depend on the size and shape of the tree matching the schema. So, her schema theorem is represented in terms of maximum probability of disruption, $P_d(H, t)$:

$$E[m(H, t+1)] \geq m(H, t) \quad \frac{f(H,t)}{\bar{f}(t)} \; (1 - p_c P_d(H, t))$$

where E[m(H, t)] is the expected value of the number of instances of H at generation t, $f(H,t)$ is the mean fitness of all instances of H, $\bar{f}(t)$ is the average fitness at t-th generation, and p_c is the crossover probability.

3.3.1.3 Whigham's Schema Theorem

Whigham [104, 105] described a schema for context-free grammars. He defines a schema as a partial derivation tree rooted in some non-terminal node. By this way the schema can be represented as a single derivation tree with collection of rewrite rules. This theorem is represented as:

$$E[m(H, t+1)] \geq m(H, t) \; \frac{f(H,t)}{\bar{f}(t)} \quad \{[1 - p_m P_{dm}(H, t)] \, [1 - p_c P_{dc}(H,t)]\}$$

where $P_{dc}(H,t)$ and $P_{dm}(H, t)$ are the average disruption probabilities of the instances of a schema under crossover and mutation, and p_c and p_m are the probabilities of disruption of schemata under crossover and mutation, respectively. The probabilities of disruption of schemata under crossover and mutation depend on the size of the tree.

3.3.1.4 Rosca's Schema Theorem

Rosca's *rooted tree schema* [106], is a rooted contiguous tree fragment. This schema is defined as a subset of the set of program trees that matches a rooted tree fragment. This takes into account the positional information. For example, the schema H = (* X #) represents all the program trees with rooted node '*' and left terminal node X. The schema theorem is represented as:

$$E[m(H, t+1)] \geq m(H, t) \frac{f(H,t)}{\bar{f}(t)} [1 - (p_m + p_c) P_d(H, t)]$$

where $P_d(H, t) = \sum_{h \in H \cap Pop(t)} \frac{O(H)}{N(h)} \frac{f(h)}{\sum_{h \in H \cap Pop(t)} f(h)}$ and $N(h)$ is the size of the program, O(H) is the order of the schema, and $f(h)$ is the fitness.

3.3.1.5 Poli and Langdon's Schema Theorem

Poli and Langdon define a schema as a subset of the set of program trees composed of terminals, functions and don't care symbols. They define *length* of a schema H as the total number of nodes in the schema, and *defining length* as the number of links in the minimum tree fragment including all don't care symbols. Here, the order O(H), length N(H) and defining length L(H) of a schema are totally independent of the size and shape of the programs. This schema theorem in a generational GP with fitness-proportional selection, one point crossover and point mutation is as follows:

$$E[m(H,t+1)] \geq m(H, t) \frac{f(H,t)}{\bar{f}(t)} (1 - p_m)^{O(H)} \{ 1 - p_c [p_{diff}(t) (1 - \frac{m(G(H),t) f(G(H),t)}{M\bar{f}(t)} + \frac{L(H)}{(N(H)-1)} \frac{m(G(H),t) f(G(H),t) - m(H,t) f(H,t)}{M\bar{f}(t)}]\}$$

where $G(H)$ is the hyperspace associated with the hyperplane H, $p_{diff}(t)$ represents the fragility of the shape of the schema H in the population, and M is the population size.

None of the schema theorems described above provide a complete under standing of the propagation of good schemata.

3.3.2 Fitness Landscapes

Fitness landscape is the mapping of genotype of a population of individuals to their fitness, and subsequent visualisation of that mapping. The correlation of the fitness of parent with offspring is directly represented in a fitness landscape by the ruggedness of the landscape. Stuart Kauffman [109] introduced landscapes with tuneable ruggedness, called NK landscapes. Kinnear [110] analysed the structure of the fitness landscape for GP over a range of problems of known difficulty.

3.3.3 Dynamics

Altenberg [33] has shown the evolution of evolvability in GP. This is achieved through the differential proliferation of blocks of code within programs. He has considered a simplified model for the analysis of GP dynamics. He has suggested new selection techniques and genetic operators to control the evolution of evolvability. Further, he has analysed the effect of soft brood selection on a simple model of GP dynamics.

3.4 *Modularization and Encapsulation in GP*

Already it has been shown that the role of sub-tree crossover in GP is more destructive rather than constructive. This can be circumvented by modularization and encapsulation of GP code. Different modularization techniques suggested for GP are described below.

3.4.1 Automatically Defined Functions

Koza [111, 226] introduced the concept of automatically defined functions (ADFs). ADFs are evolvable functions within an evolving GP with a goal to improve the performance of GP to reuse already discovered functions. To implement ADFs, the individual trees in a GP are subdivided into *result producing branch* and *function defining branch*. ADFs are defined in function defining branch and are evaluated for fitness by execution of result producing branch. These can be thought of as functional building blocks to cater for better overall performance by arranging them within a program. This arrangement is done by crossover in the result-producing branch. During ADF evaluation all the genetic operators acts only within each ADF. Kinnear [112, 113] has extensively studied the properties of ADF. Jäske [114] has shown that in a code division multiple access (CDMA) communication system ADFs do not enhance code reuse but help protect against shrinking crossover. These concepts are further extended by reusing the functions evolved in one problem to solve another problem, which needs those [115]. The fixed architecture feature of the above ADF-GP has improved by evolving the complete structure of the program in the work of architecture altering operations [116].

3.4.2 Automatically Defined Macros

Analogous to Koza's ADF, Spector [117] has developed an approach to simultaneously evolve a main program and a collection of automatically defined macros (ADMs). ADMs are used to produce new control structures during program evolution. He has shown that ADMs are likely to be useful in the applications, where context sensitive or side effect operators play important roles and those use exotic control structures.

3.4.3 Module Acquisition

Module acquisition uses the concept of encapsulation [7] and selects a part of subtree up to a certain depth as a module. This has been described by Angeline and Pollack [118, 119], and Angeline [120] with two mutation operators, which are *compression* and *expansion*. Two forms of compression have been described, they are *depth compression* and *leaf compression*. This newly formed module from compressed code is added to a genetic library (GLiB). Expand is just an inverse operator to compress. Modularization protects code from destructive actions of sub-tree crossover. However, Kinnear [112] has shown that there is no obvious advantage with the use of module acquisition.

3.4.4 Adaptive Representation

Rosca and Ballard [121] have devised a modularization scheme known as *adaptive representation*. This is based on discovering good building blocks evaluated by fitness measure as stable components for future evolution. These promising blocks, serve as new functions by extending the representation for advancement of evolution. He divided the evolution process into epochs to build new individuals from newly discovered functions. Epoch is defined as the number of generations in which no new set of candidate building blocks are discovered. Once this is done, only the less fit individuals are replaced by randomly generated individuals using the extended function set. So every time it dynamically reshapes the search space. He has experimentally shown that adaptive representation gives superior performance in terms of the number of generations and solution size as compared with ADF based GP.

3.5 *Other GP Issues*

In this section some of the important issues not described above are discussed.

3.5.1 Loops and Recursion

Loops and recursions are integral part of any programming language for solving a complex problem in any computer system. The major problem associated with this is infinite loops. This problem can sometimes be overcome partially by imposing arbitrary time-out limits. Koza [7] has used GP to produce a relatively simple problem with recursive flavour, the Fibonacci sequence. To evolve sorting algorithms Kinnear has used a loop construct *dobl* [113]. Evolution of tree search programs, where the goal is to find a specific node or state in it, is an example where each program is recursive in nature. Brave [122, 123] has developed program for this tree search problem. Koza [124] describes an automated procedure for creation of a pattern-recognising computer program consisting of initially unspecified iterative calculation, and detectors. Koza and Andre [125] have developed a new architecture altering operation of restricted iteration creation, which is used for classifying program for the transmembrane segment identification. This method itself automatically finds the need for iteration and particular sequence of iterative step. Wong and Leung [126] describe a generic GP (GGP) to evolve general recursive functions for the even n-parity problem from noisy training examples. They have also studied the effect of noise in training examples on the speed of the learning recursive functions.

3.5.2 Strongly Typed GP

In standard GP the type checking of all the arguments to a function is satisfied via the closure property. In real life problems putting these type of restriction is difficult. This difficulty can be overcome by using strongly typed GP (STGP) [127]. STGP is an improved variant of GP that ensures that all the generated parse trees obey any constraints on data type. This way it increases the complexity of the program but shows great benefit in terms of decreased search time and generality of the final program. He has used generic functions and generic data types to increase the efficacy of STGP. This STGP can handle only two levels of typing, which restricts its use; such a case arises in clique detection problem. This situation can be overcome by adding more levels of typing [128]. Clack and Yu [129] have developed a faster STGP which uses expression based parse tree representation rather than statement based parse tree as in STGP, which outputs smaller trees with smaller search space. Their approach uses a type system, which supports both polymorphic and higher order types thereby providing greater reuse. They also used a type unification algorithm rather than table look up system of STGP for determining contextual equivalence of two types.

3.5.3 Coevolution

In coevolutionary systems more than one evolution takes place among different population groups. Coevolution of fitness helps to get rid of static fitness landscape. A large problem can be solved by decomposing it into smaller sub-problems each with separate interacting evolutionary process [131]. Axelrod [132] used a competitive fitness function to evolve strategies for the iterated prisoner's dilemma. Hillis [133] has evolved a strong network with two sets of population using bipartite competition. Angeline and Pollack [85] used a hierarchical tournament fitness for evolving strategy for a Tic-Tac Toe game. The pseudo-random number generator of Jannink [135] is coevolved with testers. Juille and Pollack [136] used competitive coevolution within one population, by performing a number of tournaments between individuals to calculate fitness. Rather than this competitive coevolution, co-operative coevolution takes co-operation between population to evolve a solution [134, 137, 138]. Cultural coevolution learning [139, 140], involves the evolution of both parameters of learners (genes) and their data representation (culture).

3.5.4 Code Growth

During the process of evolution, the programs generated using GP grow rapidly in size with many extraneous inert codes [120, 141-146]., which has various names such as introns, bloat, redundancy, parsimony, fluff, MDL and structural complexity. It has been reported that introns protect fit individuals from the destructive effects of the subtree crossover [70, 71, 147]. McPhee and Miller [148] showed that with some function sets longer program replicates more accurately. Rosca and Ballard [149] analysed extensively the effect of bloat using tree schemata. The exponential growth of introns towards the end of run is more deleterious. This code growth can be controlled either by removing non-functional code or penalising longer programs.

3.5.5 Hybrid System

As with other evolutionary computation methods GP has also got many avenues to be hybridised with other search methods. These hybrid methods are described below.

O'Reilly and Oppacher [77, 150, 151] combined GP and stochastic iterated hill climbing (SIHC) to get improved performance over standard GP. Their method maintains a fitter and another current individual, where the latter initially under goes crossover with a randomly generated individual and the process is repeated with new random individuals to get a fitter one for a specified number of crossover operations, otherwise the current individual is dropped out. They have extensively compared the results of the standard GP, hybrid GP, simulated annealing (SA) [152], and hill climbing. Harries and Smith [62], also report that hill climbing GP is a strong competitor to standard GP. Iba et. al. [153] proposed a method, which blends GP with a local hill climbing search procedure that does not affect the individual tree structure but periodically employs a relabeling procedure. This relabling procedure locally tunes a non-terminal node using a statistical method or a level changing mechanism. They applied this technique to two problem domains: *Boolean GP* with adaptive logic network (ALN) and *Numerical GP* with group method of data handling (GMDH), as the relabeling process. He has reported that his Boolean GP requires fifty times fewer evaluations compared to traditional GP over a number of applications, and also it effectively adapts to time-varying environment. On the other hand, his numerical GP takes ten times fewer evaluations than traditional GP for various problems.

Hooper et al. [154] have proposed a recombinative hill climbing (RHC) method to avoid trapped in local minima encountered by hill climbing alone. In this method, crossover and mutation operations are applied to produce offspring from two selected individuals from the population. Then corresponding offspring replaces the parent if found fitter after a fitness comparison. He has shown that RHC is more reliable and robust than traditional GP while testing on two problems, symbolic regression and artificial ant.

Bolte and Thonemann [155], and Thoneman [156] have used GP for finding an optimal annealing schedule for simulated annealing to solve a quadratic assignment problem. They have reported better solutions with moderate amount of computation time. Watson and Parmee [125] devised a hybrid distributed rapid attenuated memory (HDRAM) genetic programming (discussed in section 3.2.1) method, that effectively combines GA with GP to facilitate mixed search, i.e., both discrete and continuous search.

Generic genetic programming (GGP) is an approach, which combines GP with inductive logic programming (ILP) [157]. In GGP the randomly generated initial population are induced by other learning systems or by user [126]. The declarative descriptions of the valid programs are provided by logic grammars, which are more powerful than that of context free grammar (CFG). GGP is an extension of genetic logic programming system (GLPS) [158]. It can be used to evolve programs in LISP, prolog and fuzzy prolog [157]. Howard and D'Angelo [159] combined GP with GA, called GA-P system. In GA-P system each individual consists of both a string and an expression.

The coefficient string is represented as a bit string partitioned into sections, where each section corresponds to a real number value and the expression part is a parse tree. GP evolves the expression, whereas simultaneously GA evolves the coefficient of the expression. They have used this for predicting fish population distributions and finding the relationship between physical and biological parameters of a stream. Their result is slightly more capable than standard techniques. Fuzzy genetic programming is a recently developed approach, which is a generalisation of GP [160, 161]. It is a combination of a genetic algorithm over a context-free language with a context-free fuzzy logic rule. He has applied this method for learning stock management heuristics in the MIT beer distribution game. His result shows that the order heuristics produced by this method constitute a significant improvement compared to average human performance but it lags behind the best known solution. Alba et al. [162] used fuzzy rules which are represented as tree structures. He has used a restricted crossover to generate syntactically and semantically correct rules. They have used this technique to solve cart centring problem, and then, compared its performance with intuitive fuzzy logic controller (FLC) developed manually, genetic algorithm based FLC (GAFLC) and analytical solutions. The performance of GP model was found better than intuitive FLC and comparable with GAFLC model. Spector and Luke [163, 164] have described cultural GP which improves the performance of GP through non-genetic information transfer between individuals, i.e., cultural learning.

4 GP Applications

In this section a comprehensive description of the applications of GP has been described. The applications reported here do not cover all possible fields mainly due to the rapidly increasing number of such applications. Banzaf et al. [8] have described many important applications of GP. Also, Kinnear [165], Kinnear and Angeline [166], Willis et al. [167], Langdon and Poli [168], Langdon and Qureshi [22] and Koza [130] have reported many applications of GP.

4.1 Pattern Recognition, Prediction and Classification

Andre [169] has described an approach by combining GP and GA to solve three different pattern recognition problems. He has used hit-matrix to capture the two dimensional information of the patterns. These hit-miss matrices are templates used to check for a match by moving it over part of an input pattern. The hybridisation of GP and GA is made by using GA for evolving hit-miss matrices and GP to evolve algorithms controlling the template. He has also shown that GP can be used to upgrade hand written rules, and process new fonts and characters [170].

Tacket and Carmi [98] studied a pathological classification problem, called "donut" problem. They evolved the programs with different degrees of ambiguity of class membership and sparseness of data, and by removing different "bites". They have compared different breeding policies, and the two commonly used approaches of GP, the steady state and generational approach.

Masand [172] has evolved programs using GP, which outputs confidence values to automatically classified news stories. This confidence assignment allows it to classify very certain and easy texts automatically, while relatively more uncertain and difficult cases seek manual assistance of the editors. The resulted program of confidence formulae was better than that of the best human code.

Ande et al. [173] have evolved a rule for the majority classification task for one-dimensional two-state cellular automata. These rules are more accurate and qualitatively different from others. They have shown that the accuracy of 82.326 percentage obtained with this rule is better than that of any other previously known rules generated by automated methods or human expert. This evolved rule employs a large number of different domains and particles, that utilises very fine internal representation of density information along with intricate set of signals to communicate information in the cellular space.

The transmembrane segment identification is a classification problem, where it is required to classify a given protein segment as being a transmembrane domain or non-transmembrane area of the protein without using biochemical knowledge concerning hydrophobicity. Koza [111], and Koza and Andre [125, 174] have evolved programs using four different versions of GP whose performance is better than human expert has. GP is used for the automatic discovery of biologically meaningful information hidden in the database of DNA and protein sequences. Two motifs are evolved without prespecified length along with ADFs for detecting the D-E-A-D box family of proteins and manganese superoxide dismutase family [175]. Both motif detect two families either as good as, or slightly better than that of human-expert written motifs. Handley has performed a series of works on prediction, detection, classification and other related properties of α-helix and amino acid in proteins [176-179]. Robinson and McIlroy [180] have evolved GP programs to predict the faults in software file from software metrics. The selected data set consists of a total of 163 software files. They have used the measured values of five software code metrics for each file as input data and the number of faults discovered as output. Also, they have studied two other applications that are discussed in the subsequent sections. Lee et al. [181] have used GP for the long-term electric power system load forecasting. For GP modelling they have considered twelve different cases of function and terminal set combinations, where the terminal set consists of past population, gross domestic product (GDP) and electric demand. They have shown the superior performance of GP compared with that of a widely accepted regression model in energy demand.

4.2 Image and Signal Processing

Tackett [171, 182, 183] applied GP for feature discovery and image discrimination of very difficult problems. He has used GP to construct classifiers based on feature vectors and to extract own features from noisy pictures. He has shown the superior performance of GP over that of binary tree classifier system and multilayer neural networks. Robinson and McIlrey [180] have applied GP to identify eye centre locations on human face image bit map. They used one to forty faces as the learning set and one to ten faces as the testing set from a data set consisting of 85 face images. Similarly, Johnson et al. [184] used GP for the task of extracting human hand locations.

Daida et al. [185, 186] represented GP systems for image processing applications in geoscience and remote sensing. Here, they present an algorithm for extracting pressure-ridge features from European remote sensing satellite (ERS) SAR images of arctic sea ice. They describe the use of GP as a scaffold for algorithm discovery. Their results are very encouraging and well in agreement with human expert.

The PADO system [41], which uses a graph representation has been used for image and sound classification. It has been reported that the generated classification program is correct between 70 per cent and 90 per cent of the test cases. Nordin and Banzaf [187] have used GP for image and sound compression.

Poli [188] describes the use of GP to develop optimal image filters. These filters are capable of enhancing and detecting features of interest on building pixel-classification-based segmentation algorithms. In this work, he has set some criteria on terminal sets, function sets and fitness functions to produce efficient filters. His experiment on medical images has shown the great ability of GP over neural networks. Oakley [189] used GP to optimise a stack filter, which performs better than other GA based filters on noisy physiological data. He has also used GP to fit empirical equations to chaotic time series and non-linear physiological data. Harries and Buxton [190] have used GP to find optimal linear filters for edge detection in signals. For this they have convolved masks (linear FIR filters), which are built by sampling evolved GP programs with a set of known reference signals. They have reported better results than those obtained by Canny's edge detector. GP has been used to evolve the structure of adaptive digital signal processing algorithms

[191, 192]. The numerical parameters of these algorithms are optimised using simulated annealing method.

4.3 System Modelling, Identification and Control

GP has been applied successfully to a number of process engineering applications. Keane et al. [193] demonstrated the use of the GP to approximate the impulse response of a linear time invariant system. Iba et al. [33-35, 194] introduced a GP based algorithm, called STRAGONOFF for solving several system identification problems. This algorithm makes use of polynomial functions as internal nodes, thereby restricting the form of final solution. Oakley [189, 195] demonstrated the use of GP for the prediction of known chaotic series in the presence of noise. Zhang and Muhlenbein [196, 197], and Zhang [198] developed a GP based technique for evolving an optimal sigma-pi neural network system. They have studied the effectiveness of the evolved network for the prediction of nitrate levels in the watersheds of the river and a chaotic time series.

Generation of dynamic non-linear models of biotechnological batch and fed-batch fermentation has been studied by Battenhausen and Marenbach [199], Battenhausen et al. [200], Marenbach et al. [201, 202], and Marenbach and Brown [203]. Bettenhausen et al. [200] and Marenbach et al. [201] have described a GP method called structured model generator (SMOG). SMOG uses a block diagram representation to model complex dynamic processes. Use of SMOG provides data driven model generation, modification, and verification in close co-operation with process experts. Marenbach and Brown [203] have presented two algorithms; one based on one step ahead optimal inductive learning algorithm and the other uses a GP to find the model structure. They have shown that inductive learning approach outperforms the GP in terms of speed. But for highly complex process GP may prove beneficial. Hiden et al. [204] have avoided the use of time consuming recursive nodes in the method of Battenhausen and Marenbach [199]. They have used GP to calculate non-linear process gains, process variable interactions etc. for the process transfer functions and delays. Gray et al. [205-207] has used GP to build block diagram simulation model of dynamic system. They have estimated the system parameters by a combined simulated annealing and Nelder simplex minimisation method [205, 206]. South et al. [208] have compared the aspects of GP and GA as system identification tool. McKay et al. [209] have described a GP based method for generating steady-state input-output models of chemical process systems. Vazquez and Fleming [210] have described a GP based identification algorithm to evolve the structure of polynomial NARMAX models by minimising Akaike information criteria (AAIC). Hybrid GP techniques for process modelling have been proposed by McKay et al. [211] and Elsay et al. [212].

Schoenauer et al. [213] used GP in the field of macro-mechanical modelling, where the identification of both one dimensional rheological models and three dimensional hyper elastic strain energy functions have been demonstrated. Koza and Keane [214] described a GP based technique for the time optimal non-linear control of a three-dimensional broom-balancing problem. Hampo [215], and Marko and Hampo [216] have used GP for evolving control algorithms for active suspension system. Alba et al. [162] used GP to develop rule bases for FLC for cart centring problem. Wide use of GP for robot control has been discussed in the next section.

4.4 Robotics

Koza [217] experimented in the wall following robot using GP as a subsumption style robot controller [218]. Ross et al. [219] have analysed the wall following robot with and without ADFs, and have crafter this problem as a suitable benchmark for the applications of GP for solving problems involving emergent robotics behaviour. Goldish [220] used GP to evolve wall following and maze navigation algorithms. Koza and Rice [221] applied GP to generate computer programs for an autonomous mobile robot to perform a box moving tax. Lee et al. [222] have used GP to program the control architecture of a behaviour based mobile robot, and construct their behaviour

primitives and arbitrators. This approach is considered to be at an intermediate level as it avoids hand coding of all the individual behaviour controllers or evolution of a whole control system for the over all task. GP has been used to solve the motion-planning problem in a task where it is required to visually guide the hand to reach a target by avoiding obstacle [223]. Reynolds [224] used GP for the task of moving two-dimensional vehicle to avoid collisions with obstacles. This evolved controller was "brittle" due to the use of identical fitness tests. Latter on, he introduced noises in fitness test so as to evolve robust and compact solutions [88, 89]. Also, Reynolds [227] has shown that the difficulty in the noisy-sensor-based corridor following problem is only due to the representation. He used constrained roving sensors to reduce this difficulty.

Handley [228-230] has introduced genetic planner, which produces plans (i.e., programs) to control a robot to push three boxes together and then return back to a specified location in another room. Haynes and Wainwright [231] described the evolution of a program that makes an agent to correctly sense and mark the presence of items in any environment. Evolving obstacle avoiding behaviour for a real robot (Khepara) has been experimented [232-235]. Reynolds [236] studied the obstacle avoidance through a simulated two dimensional environment. Spencer [84, 237] has evolved programs for a simulated hexapod (six-legged) robot to crawl and walk. Rush et al. [238] showed the automation of behaviour based control design of a complex task for multiple co-operating robotics devices. Taylor [239] analysed the use of GP for robot juggling action. Lott [240] used GP to evolve robot control programs to transform any random 12×12 grid into a flat plane, and also studied the co-operation between robots to perform this task.

Ito et al. [241] have studied the robustness of evolved robot control programs by GP for a box-moving problem. Benett III [242] described the evolution of a single robot control program using architecture-altering operations to solve multiple problems. Distributed agent version of the predator/prey problem has been solved using GP [243, 244]. Similar to this, Qureshi [245] described the aspects of interaction and communication between agents to solve problems using GP. Ryan [245] has developed a simulation called GProbots, which is a multiagent approach to solve a single problem. Iba [247], and Iba et al. [248] have presented the emergence of co-operative behaviour for multiple agents in a dynamic environment.

Howley [249] proposed a bimodal fitness selection method to reduce the sensitivity of GP solutions to parameter changes under a test suit of minimum time control of a two-link manipulator. Gibbs [250] proposed a simple inverse kinematics for animators by using GP.

4.5 Genetic Programming and Neural Networks

Simultaneous optimisation of the architecture and the weights of a neural network can be achieved by the use of cellular encoding system [43-45, 251-254]. Cellular encoding is similar to grammar encoding [255]. A neural network starts from a single neuron with zero threshold. Its input and output is connected with all the network inputs and outputs, respectively. This single neuron then grows to a complete network with the help of graph transfer functions, which are classified into cell division and cell register modification. These registers serve as local memory of a cell. Gruau used this technique to control a six-legged insect.

Zhang and Mühleinbein [256] used breeder genetic programming (BGP) to evolve neural network with a new fitness function that quantifies the principle of Occam's razor. In another work [196], they used BGP for the synthesis of sigma-pi network. Also, Zhang [198], and Zhang and Mühlenbein [197] have evolved neural network and verified their effectiveness on the modelling and prediction of complex system.

Benigo et al. [257] described the use of GP for the search of new learning rules for neural network. They have found all the values of the rule parameters, optimal number of parameters and form of the rule. The performance of this method on a set of classification tasks was shown to be

better than backpropagation. Alcazar and Sharman [258, 259] proposed a hybrid algorithm using GP and SA to produce a recurrent neural network for adaptive filtering in a signal processing application.

4.6 Design, Optimisation and Scheduling

Nguyen and Huang [260] have used GP to evolve three dimensional jet models, with user defined interactive fitness measure. Koza et al. [261-266] has described the automatic design of different electrical circuits. Here, GP produces both the topology and the circuit component values. Also, Koza et al. [267] used ADFs and architecture altering operations for designing alnalog electrical circuits. It has been reported that Porter et al. have described the use of GP to the design of new polymeric materials [167].

Atkin and Cohen [268] discussed the potentiality of the use of GP to find optimal monitoring strategies while designing an agent. GP is used to generate programs to find when a particular section of a railway track needs maintenance [269]. GP based maintenance schedule is found to be superior to the existing one.

Graces-Perea et al. [270] used GP to a slicing tree structure that determines the facility layout for the NP-complete facility layout problem (FLP). They showed that FLP is more conducive to a GP than GA. Montana and Czerwinski [271] described the use of GP to control optimally the timing of traffic signals in a multi-junction road. This approach takes into account the congestion level and waiting time of cars at the various junctions. Watson and Parmee [272] described distributed rapid attenuated memory GP (DRAM-GP) and hybrid DRAM-GP for the engineering fluid dynamics system.

4.7 Financial Applications

Perry [273] described the use of GP for the prediction of wining horses using historical horse racing data. Warren [274] analysed the potentiality of GP for stock prices prediction. Eglit [275] described the use of GP to learn trends in financial time series. He has reported encouraging results although not astounding. Andrew and Prager [276] have used GP method to create double auction market strategies, and compared the results with that of simulated annealing method. The generated program proved to be superior to many hand-coded strategies. Robinson and McIlroy [180] predicted one-day ahead the share prices from previous data using a GP approach. They used share prices over 500 consecutive days for this work. Every time the program is inputted with last four (nine) days share price values normalised against today's share price, and the output is tomorrow's normalised share price. They obtained better results compared over many standard methods.

Chen and Yeh [277] applied GP to model the learning of economic agents in the cobweb model. They have showed that GP require less prior knowledge than GA but in long run they both perform equally well in terms of discovering rational expectations equilibrium price. Also, Chen and Yeh [278] used a GP-based computable approach to bridge the gap between non-linear test and the efficient market hypothesis (EMH), which takes into account the issues of predictability and profitability. Chen and Yeh [279] used GP to model volatility in Japanese and New York stock markets, whereas Oussaidene et al. [280] modelled the currency market.

4.8 Art, Computer Graphics and Computing

Gritz and Hahn [281, 282] used GP to generate controller programs for dynamically controlled robots, whose actions are determined by these programs. The animator has only to specify the metric for good motion. They have described two examples of which one is the task of moving a lamp as the animated figure to a certain place, and the other one is the articulated humanoid figure

with 28 degrees of freedom, which is taught to do a variety of simple tasks. Das et. al. [283] applied GP to virtual reality in which they populate a virtual world with both images and sounds. They have reported very interesting results. Spector and Alpern [284] described a system using GP that produces new bebop jazz melodies from a case-base of melodies. Here, the fitness function for GP is based on user provided critical criteria.

Crosbie and Spafford [285] described a decentralised detection approach for intrusion detection, which builds a computer defence system. Bruce [286] used GP to generate object oriented programs, that makes use of the concept of simultaneous program induction operating on single data structure.

5 Conclusions

This paper has described the fundamental and implementation aspects, along with various applications of genetic programming. One part of this work emphasises the importance of GP for solving variety of applications. This illustrates the inherent capability of GP to solve complex problems. Although GP has shown very impressive performance in diverse application fields, it lacks in sound theoretical foundations. Theoretical research will greatly help in the rapid progress of this area. Under the possible theoretical improvements, the major issues to be addressed are the establishment of a sound working principle of GP and enhancement of its overall performance. Many researchers are working to increase the convergence rate of GP for solving different problems by suitably modifying the operators, proposing alternative representations and fitness measures, or by hybridising with other existing methods. As with other evolutionary techniques, the parallel implementation of GP is an active area of research. GP is used in the area of machine learning and classifier system, which is well treated in GA literature [287]. It is evident from this work that GP is a very young field of research and it will remain active in the coming years.

Acknowledgement

The first author would like to thank commonwealth scholarship commission in the United Kingdom for providing the support to pursue this work.

References

[1] J. Holland, *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, 1975.

[2] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison Wesley Publishing Co., 1989.

[3] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. New York, Oxford University Press, 1996.

[4] H.-P. Schwefel, *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology Series. John Wiley & Sons, New York, 1995.

[5] L. Fogel, A. Owens and M. Walsh, *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, New York, 1966.

[6] D.B. Fogel, *Evolutionary Computation: towards a new philosophy of machine intelligence.* IEEE Press, New York, 1995.

[7] J.R. Koza, *Genetic Programming: On the Programming of Computers by Natural Selection.* MIT Press, Cambridge, MA, 1992.

[8] W. Banzaf, P. Nordin, R.E. Keller and F.D. Francone, *Genetic Programming: An Introduction.* Morgan Kaufmann Publishers, Inc., USA, 1998.

[9] J.R. Koza, "Genetic programming as a means for programming computers by natural selection," *Statistics and Computing*, 1994, 4, pp.87-112, 1994.

[10] R. Forsyth, "BEAGLE: a Darwinian approach to pattern recognition," *Kybernates*, 10, pp.159-166, 1981.

[11] N.L. Cramer, "A representation for the adaptive generation of simple sequential programs," in *Proceedings of an International Conference on Genetic Algorithms and the Applications* (J.J. Grefenstette, ed.), pp.183-187, Carnegie-Mellon University, Pittsburgh, PA, 1985.

[12] C. Fujiki and J. Dickinson, "Using the genetic algorithm to generate LISP source code to solve the prisoner's dilemma," in *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (J.J. Grefenstette, ed.), pp.234-240, MIT Cambridge, MA. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[13] M. Tomassini, "Evolutionary Algorithms," in *Towards Evolvable Hardware: the Evolutionary Engineering Approach of Evolutionary Algorithms, Proceedings*, pp.19-47, 1996.

[14] G.D. Smith, "Commercial applications of genetic algorithms," in *Proceedings of Adaptive Computing and Information Processing Conference*, pp.483-506, 1994.

[15] Z. Michalewicz, "Evolutionary Computation: An Overview," in *Proceedings of Scandinavian Conference on Artificial Intelligence* (A. Aamodt and J. komorowski, eds.), pp.322-337, IOS Press, 1995.

[16] Z. Michalewicz and M. Michalewicz, "Evolutionary Computation: Main Paradigms and Current Directions," *Appl. Math. Comp. Sc.*, vol.6, no.3, pp.393-413, 1996.

[17] P.J. Angeline, "Genetic programming's continued evolution," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), MIT Press, Cambridge, MA, 1996.

[18] T. Bäck, U. Hammel and H.-P. Schwefel, "Evolutionary computations: Comments on the history and current state," *IEEE Transactions on Evolutionary Computation*, vol.1, no.1, pp.3-17, April 1997.

[19] W.W. Gibs, "Programming with primordial ooze," *Scientific American*.

[20] D. Molnar, "Genetic programming: will bill gates become Billy appleseed?," *Computer Bits*, vol.6, no.6, June 1996.

[21] J.R. Koza, "Survey of genetic algorithms and genetic programming," *Proceedings of the WESCON Conference*, pp.589-594, 1995.

[22] W.B. Langdon and A. Qureshi, "Genetic programming-computers using "natural selection" to generate programs," Research Note RN/95/76, University College London, UK, 1995.

[23] P.J. Angeline, "Genetic programming: A current snapshot," in *Proceedings of the Third Annual Conference on Evolutionary Programming* (D.B. Fogel and W. Atmar, eds.), Evolutionary Programming Society, 1994.

[24] J.R. Koza, "Future work and practical applications of genetic programming," *HandBook of Evolutionary Computation*, 1996.

[25] J.R. Koza, "Introduction to genetic programming," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), MIT Press, Cambridge, MA, 1994.

[26] K.E. Kinnear, Jr., "A perspective on the work in this book," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.3-19, MIT Press, Cambridge, MA, 1994.

[27] J.R. Koza and J.P. Rice, *Genetic Programming: The Moovie*. Cambridge, MA, MIT Press, 1992.

[28] J.R. Koza, "Genetic evolution and co-evolution of computer programs," *Artificial Life II* (C.G. Langton, C. Taylor, J.D. Farmer and S. Rasmuseen, eds.), Addison-Wesley Publishing Co., pp.603-630, 1992.

[29] J.R. Koza, "Genetic programming: A paradigm for genetically breeding population of computer programs to solve problems," Report Number: STAN-CS-90-1314, Stanford University, 1990.

[30] W. Banzaf, "Genotype-phenotype-mapping and neutral variation - a case study in genetic programming," *Parallel Problem Solving from Nature III* (Y. Davidor and H.-P.Schwefel, eds.), 1994.

[31] R.E. Keller and W. Banzaf, "Genetic programming using genotype-phenotype mapping from linear genomes into linear phenomes," *Genetic Programming 1996: Proceedings of the First Annual Conference (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), Stanford University, CA. MIT Press, Cambridge, MA, 1996.*

[32] L. Altenberg, "Emergent phenomena in genetic programming," *Evolutionary Programming - Proceedings of the Third Annual Conference* (A.V. Sebald and L.J. Fogel, eds.), pp.233-241, World Scientific, Singapore, 1994.

[33] L. Altenberg, "The evolution of evolvability in genetic programming," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.47-74, MIT Press, Cambridge, MA, 1994.

[34] H. Iba, T. Sato and H. de Garis, "Numerical genetic programming for system identification," *Proceedings of the Workshop on Genetic Programming: From Theory to Real-world Applications* (J.P. Rosca, ed.), pp.64-75, Tahoe City, CA, 1995.

[35] H. Iba and H. de Garis and T. Sato, "A numerical approach to genetic programming for system identification," *Evolutionary Computation*, vol. 3, no. 4 , pp.417-452, 1996.

[36] T. Perkis, "Stack-based genetic programming," *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, pp.148-153, Orlando, Florida, IEEE Press, USA, June 1994.

[37] W. Banzaf, "Genetic programming for pedestrians," *Proceedings of the 5th international Conference on Genetic Algorithms, ICGA-93* (S. Forrest, ed.), pp.628, University of Illinois at Urbana-Champaign, Morgan Kauffman, San Francisco, CA, 1993.

[38] M. Wineberg and F. Oppacher, "The benefits of computing with introns," *Parallel Problem Solving from Nature III* (Y. Davidor and H.-P.Schwefel, eds.), vol.866 of Lecture notes in Computer Science, 1994.

[39] P. Nordin, "A compiling genetic programming system that directly manipulates the machine code," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.311-331, MIT Press, Cambridge, MA, 1994.

[40] P. Nordin and W. Banzaf, "Evolving Turing complete-programs for a register machine with self-modifying code," *Genetic Algorithms: Proceedings of the Sixth International Conference ICGA 95* (L. Eshelman, ed.), pp.310-317, Pittsburgh, PA. Morgan Kaufmann, San Francisco, CA, 1995.

[41] A. Teller and M. Veloso, "PADO: Learning tree structured algorithms for orchestration into an object recognition system," Technical Report CMU-CS-95-101, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1995.

[42] A. Teller, "Evolving programmers: The co-evolution of intelligent recombination operators," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.45-68, MIT Press, Cambridge, MA, 1996.

[43] F. Gruau, "Genetic synthesis of Boolean neural networks with a cell rewriting developmental process," *Proceedings of the workshop on combinations of Genetic Algorithms and Neural Networks - COGANN92* (J.D. Schaffer and D. Whitley, eds.), pp.55-74, IEEE Computer Society Press, 1992.

[44] F. Gruau, "Genetic synthesis of modular neural networks," *Proceedings of the 5th international Conference on Genetic Algorithms, ICGA-93* (S. Forrest, ed.), pp.318-325, University of Illinois at Urbana-Champaign, Morgan Kauffman, San Francisco, CA, 1993.

[45] F. Gruau, "Genetic micro programming of neural networks," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.495-518, MIT Press, Cambridge, MA, 1994.

[46] C. Jacob, "Genetic L-system programming," *Parallel Problem Solving from Nature III* (Y. Davidor and H.-P.Schwefel, eds.), vol.866 of Lecture notes in Computer Science, 1994.

[47] C. Jacob, "Evolving evolution programs: Genetic programming and L-systems," *Parallel Problem Solving from Nature IV. Proceedings of the International Conference on Evolutionary Computation* (H. Voigt, W. Ebeling, I. Rechenberg and H.-P.Schwefel, eds.), pp.42-51, Springer-Verlag, 1996.

[48] W.B. Langdon, "Evolving data structures using genetic programming," *Genetic Algorithms: Proceedings of the Sixth International Conference ICGA 95* (L. Eshelman, ed.), pp.295-302, Pittsburgh, PA. Morgan Kaufmann, San Francisco, CA, 1995.

[49] P.A. Whigham, "Grammatically-based genetic programming," *Proceedings of the Workshop on Genetic Programming: From Theory to Real-world Applications* (J.P. Rosca, ed.), pp.33-41, Tahoe City, CA, 1995.

[50] P.A. Whigham, "Inductive bias and genetic programming," *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications GALESIA* (A.M.S. Zalzala, ed.), vol. 414, pp.461-466, Sheffield, UK. IEE London, UK, 1995.

[51] M.L. Wong and K.S. Leung, "Evolving recursive functions for the even-parity problem using genetic programming," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.221-240, MIT Press, Cambridge, MA, 1996.

[52] S. Handley, "On the use of a directed acyclic graph to represent a population of computer programs," *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, Orlando, Florida, IEEE Press, USA, June 1994.

[53] M. Keijzer, "Efficiently representing populations in genetic programming," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), MIT Press, Cambridge, MA, 1996.

[54] S. Droste, "Efficient genetic programming for finding good generalizing Boolean functions," 1997.

[55] M. Yanagiya, "Efficient genetic programming based on binary decision diagrams," in *IEEE Conference on Evolutionary Computation*, vol.1, pp.234-239, Perth, Australia. IEEE Press, New York, 1995.

[56] P. D'haeseleer, "Context preserving crossover in genetic programming," in *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, pp.256-261, Orlando, Florida, IEEE Press, USA, June 1994.

[57] W.A. Tackett, *Recombination, Selection, and the Genetic Construction of Computer Programs*. Ph.D. Thesis, University of Southern California, Department of Electrical Engineering Systems, 1994.

[58] W.A. Tackett and A. Carmi, "The unique implications of brood selection for genetic programming," in *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, pp.160-165, Orlando, Florida, IEEE Press, USA, June 1994.

[59] W.B. Langdon, "Directed crossover within genetic programming," 23rd April 1996.

[60] R. Poli and W.B. Langdon, "Genetic programming with one point crossover and point mutation," Technical Report: CSRP-97-13, April 1997.

[61] R. Poli and W.B. Langdon, "On the ability to search the space of programs of standard, one-point and uniform crossover in genetic programming," Technical Report: CSRP-988-7, January 1998.

[62] K. Harries and P. Smith, "Exploring alternative operators and search strategies in genetic programming," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[63] H. Iba and H. de Garis, "Extending genetic programming with recombinative guidance," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.69-88, MIT Press, Cambridge, MA, 1996.

[64] A.H. Watson and I.C. Parmee, "Steady state genetic programming with constrained complexity crossover," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[65] E. Zannoni and R. Reynolds, "Extracting design knowledge from genetic programs using cultural algorithms," in *Proceedings of the Fifth Evolutionary Programming Conference* (L. Fogel, P. Angeline and T. Bäck, eds.), San Diego, CA, Cambridge, MA. MIT Press, Cambridge, MA, 1996.

[66] P.J. Angeline, "Two self-adaptive crossover operators for genetic programming," in *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.89-110, MIT Press, Cambridge, MA, 1996.

[67] L.J. Fogel, D.B. Fogel and P.J. Angeline, "A preliminary investigation on extending evolutionary programming to include self-adaptation on finite state machines," *Informatica*, vol.18, pp.387-398, 1994.

[68] Z. Michalewicz, *Genetic algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Berlin, 1996.

[69] L. Davis (ed.), *HandBook of Genetic Algorithms*. New York, Van Nostrand Reinhold, 1991.

[70] P. Nordin, F. Francone and W. Banzaf, "Explicitly defined introns and destructive crossover in genetic programming," in *Proceedings of the Workshop on Genetic Programming: From Theory to Real-world Applications* (J.P. Rosca, ed.), pp.6-22, Tahoe City, CA, 1995.

[71] P. Nordin, F. Francone and W. Banzaf, "Explicitly defined introns and destructive crossover in genetic programming," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.111-134, MIT Press, Cambridge, MA, 1996.

[72] K.J. Lang, "Hill climbing beats genetic search on a Boolean circuit synthesis of Koza's," in *Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, CA. Morgan Kaufmann, San Francisco, CA, 1995.

[73] D.B. Fogel and L.C. Stayton, "On the effectiveness of crossover in simulated evolutionary optimization," *Bio-Systems*, vol.32, pp.171-182, 1994.

[74] L.J. Eshelman and J.D. Shaffer, "Crossover's niche," in *Proceedings of the 5th international Conference on Genetic Algorithms, ICGA-93* (S. Forrest, ed.), pp.9-14, University of Illinois at Urbana-Champaign, Morgan Kauffman, San Francisco, CA, 1993.

[75] D.B. Fogel and J.W. Atmar, "Comparing genetic operators with Gaussian mutation in simulated evolutionary processes using linear systems," *Biological Cybernatics*, vol.63, pp.111-114, 1990.

[76] P.J. Angeline, "An investigation into the sensitivity of genetic programming to the frequency of leaf selection during subtree crossover," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.21-29, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[77] U.-M. O'Reilly and F. Oppacher, "A comparative analysis of GP," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.23-44, MIT Press, Cambridge, MA, 1996.

[78] P.J. Angeline, "Comparing subtree crossover with macromutation," in *Proceedings of the Sixth Conference on Evolutionary Programming* (P. Angeline, R. Reynolds, J. McDonnell and R. Eberhart, eds.), Sringer-Verlag, 1997.

[79] P. Angeline, "Subtree crossover: Building block engine or macromutation," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), pp.9-17, Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[80] S. Luke and L. Spector, "A comparison of crossover and mutation in genetic programming," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), pp.240-248, Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[81] K. Chellapilla, "Evolutionary programming with tree mutations: Evolving computer programs without subtree crossover," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), pp.431-438, Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[82] K. Chellapilla, "Evolving computer programs without subtree crossover," IEEE Transactions on Evolutionary Computation, vol. 1, no. 3, pp.209-216, Sep. 1997.

[83] T. Jones, "Crossover, macromutation and population based search," in *Proceedings of the Sixth International Conference on Genetic Algorithms,* (L. Eshelman, ed.), pp.73-80, Morgan Kauffman, San Francisco, CA, 1995.

[84] G.F. Spencer, "Automatic generation of programs for crawling and walking," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.335-353, MIT Press, Cambridge, MA, 1994.

[85] P.J. Angeline and J.B. Pollack, "Competitive environments evolve better solutions for complex tasks," *Proceedings of the 5th international Conference on Genetic Algorithms, ICGA-93* (S. Forrest, ed.), pp.264-270, University of Illinois at Urbana-Champaign, Morgan Kauffman, San Francisco, CA, 1993.

[86] C.W. Reynolds, "Competition, coevolution and the game of tag," in *Proceedings of the Fourth International Workshop on Synthesis and Simulation of Living Systems* (R.A. Brooks and P. Maes, eds.), pp.59-69, MIT Press, 1994.

[87] K.E. Kinnear, Jr., "Generality and difficulty in genetic programming: Evolving a sort," *Proceedings of the 5th international Conference on Genetic Algorithms, ICGA-93* (S. Forrest, ed.), pp.287-294, University of Illinois at Urbana-Champaign, Morgan Kauffman, San Francisco, CA, 1993.

[88] C.W. Reynolds, ""Evolution of corridor following behavior in a noisy world," From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour (SAB'94) (D. Cliff, P. Husbands, J.-A. Meyer and S.W. Wilson, eds.), pp.402-410, MIT Press, 1994.

[89] C.W. Reynolds, "Evolution of corridor following behaviour: Using noise to promote robust solutions," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.221-241, MIT Press, Cambridge, MA, 1994.

[90] D. Andre, "Evolution of mapmaking ability: Strategies for the evolution of learning, planning and memory using genetic programming," *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, Orlando, Florida, IEEE Press, USA, June 1994.

[91] P. Nordin and W. Banzaf, "Programmatic compression of images and sound," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.345-350, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[92] T. Blickle and L. Thiele, "A comparison of selection schemes used in genetic algorithms," TIK-Report 11, Swiss Federal Institute of Technology, Switzerland, 1995.

[93] J.E. Baker, "Adaptive selection methods for genetic algorithms," in *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Hillsdale, NJ: Lawrence Erlbaum, pp.101-111, 1985.

[94] D. whitley, "The genetic algorithm and selection pressure:: Why rank-based allocation of reproductive trials is best," in *Proceedings of the Third International Conference on Genetic Algorithms* (J.D. Schaffer, ed.), pp.116-121, San Mateo, Morgan Kauffman, San Francisco, CA, 1989.

[95] J.J. Grefenstette and J.E. Baker, "How genetic algorithm work: A critical look at implicit parallelism," in *Proceedings of the Third International Conference on Genetic Algorithms* (J.D. Schaffer, ed.), pp.20-27, San Mateo, Morgan Kauffman, San Francisco, CA, 1989.

[96] H. Mühlenbein and D. Schlierkamp-Voosen, "The science of breeding and its application to the breeder genetic algorithm," *Evolutionary Computation*, vol.1, pp.335-360, 1994.

[97] D.E. Goldberg, B. Korb and K. Deb, "Messy genetic algorithms: Motivation analysis and first results," *Complex Systems*, vol. 3, no. 5, pp.523-530, 1989.

[98] W.A. Tackett and A. Carmi, "The donut problem: Scalability and generalization in genetic programming," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.143-176, MIT Press, Cambridge, MA, 1994.

[99] P. D'haeseleer and J. Bluming, "Effects of locality in individual and population evolution," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.177-198, MIT Press, Cambridge, MA, 1994.

[100] R. Poli, "Parallel distributed genetic programming," Technical Report CSRP-96-15, University of Bermingham, UK, 1996.

[101] U.-M. O'Reilly and F. Oppacher, "The troubling aspects of a building block hypothesis for genetic programming," Working Paper 94-02-001, Santa Fe Institute, Santa Fe, NM, 1992.

[102] U.-M. O'Reilly and F. Oppacher, "Using building block functions to investigate a building block hypothesis for genetic programming," Working Paper 94-02-029, Santa Fe Institute, Santa Fe, NM, 1994.

[103] U.-M. O'Reilly and F. Oppacher, "The troubling aspects of building block hypothesis for genetic programming," in *Foundations of Genetic Algorithms 3* (L.D. Whitley, M.D. Vose, eds.), pp.73-88, Estes Park, Morgan Kaufmann, San Francisco, CA, 1995.

[104] P.A. Whigham, "A schema theorem for context-free grammars," *IEEE Conference on Evolutionary Computation*, vol.1, pp.178-181, Perth, Australia. IEEE Press, New York, 1995.

[105] P.A. Whigham, "Search bias, language bias, and genetic programming," in *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.230-237, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[106] J.P. Rosca, "Analysis of complexity drift in genetic programming," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), pp.286-297, Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[107] R. Poli and W.B. Langdon, "An experimental analysis of schema creation, propagation and disruption in genetic programming," in *Genetic Algorithms: Proceedings of the Seventh International Conference* (E. Goodman, ed.), East Lansing, MI. Morgan Kaufmann, San Francisco, CA, 1997.

[108] R. Poli and W.B. Langdon, "A new schema theory for genetic programming with one-point crossover and point mutation," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), pp.278-285, Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[109] S.A. Kauffman, *The Origin of Orders: Self-Organisation and Selection in Evolution*. Oxford University Press, New York, NY, 1993.

[110] K.E. Kinnear, Jr., "Fitness landscapes and difficulty in genetic programming," *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, pp.142-147, Orlando, Florida, IEEE Press, USA, June 1994.

[111] J.R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, 1994.

[112] K.E. Kinnear, Jr., "Alternatives in automatic function definition: A comparison of performance," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.119-141, MIT Press, Cambridge, MA, 1994.

[113] K.E. Kinnear, Jr., "Generality and difficulty in genetic programming: Evolving a sort," *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93* (S. Forrest, ed.), pp.287-294, University of Illinois at Urbana-Champaign, Morgan Kauffman, San Francisco, CA, 1993.

[114] H. Jäske, "On code reuse in genetic programming," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[115] G. Seront, "External concepts reuse in genetic programming," in *GP papers from 1995 AAAI Fall Symposium*, pp.96-98, 1995.

[116] J.R. Koza, "Gene duplication to enable genetic programming to concurrently evolve both the architecture and work-performing steps of a computer program," in IJCAI-95 Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, vol. 1, pp.734-740, Montreal, Quebec, Canada, Morgan Kaufmann, San Francisco, CA, 1995.

[117] L. Spector, "Simultaneous evolution of programs and their control structures," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.134-154, MIT Press, Cambridge, MA, 1996.

[118] P.J. Angeline and J.B. Pollack, "Competitive environments evolve better solutions for complex tasks," *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-*

93 (S. Forrest, ed.), pp.264-270, University of Illinois at Urbana-Champaign, Morgan Kauffman, San Francisco, CA, 1993.

[119] P.J. Angeline and J.B. Pollack, "The evolutionary induction of subroutines," in *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, Bloomington, Indiana. Lawrence Erlbaum Associates, Hillsdale, NJ, 1992.

[120] P.J. Angeline, "Genetic programming and emergent intelligence," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.89-110, MIT Press, Cambridge, MA, 1994.

[121] J.P. Rosca and D.H. Ballard, "Learning by adapting representations in genetic programming," *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, Orlando, Florida, IEEE Press, USA, June 1994.

[122] S. Brave, "Using genetic programming to evolve recursive programs for tree search," in *Fourth Golden West Conference on Intelligent Systems* (S. Louis, ed.), pp.60-65, International Society for Computers and Their Applications, ISCA, 1995.

[123] S. Brave, "Evolution of planning: Using recursive techniques in genetic planning," *Artificial Life at Stanford 1994* (J.R. Koza, ed.), pp.1-10, Stanford Book store, Stanford, CA, 1994.

[124] J.R. Koza, "Automated discovery of detectors and iteration-performing calculations to recognize patterns in protein sequences using genetic programming," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp.684-689, 1994.

[125] J.R. Koza and D. Andre, "Evolution of iteration in genetic programming," .in *Proceedings of the Fifth Evolutionary Programming Conference* (L. Fogel, P. Angeline and T. Bäck, eds.), San Diego, CA, Cambridge, MA. MIT Press, Cambridge, MA, 1996.

[126] M.L.Wong and K.S. Leung, "Learning recursive functions from noisy examples using generic genetic programming," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.238-246, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[127] D.J. Montana, "Strongly typed genetic programming," *Evolutionary Computation*, 1994.

[128] T.D. Haynes, D.A. Schoenefeld and R.L. Wainwright, "Type inheritance in strongly typed genetic programming," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.359-375, MIT Press, Cambridge, MA, 1996.

[129] C. Clack and T. Yu, "Performance enhanced genetic programming," University College of London,1997.

[130] J.R. Koza, "Future work and practical applications of genetic programming," *Hand Book of Evolutionary Computation*, 1996.

[131] M. Potter and K. De Jong, "A co-operative coevolutionary approach to function optimization," George Masson University, 1994.

[132] R. Axelrod, "Evolution of strategies in the iterated prisoner's dilemma," *Genetic Algorithms and Simulated Algorithm* (L. Davis, ed.), Morgan Kaufmann, San Francisco, CA, 1987.

[133] D. Hillis, "Co-evolving parasites improves simulated evolution as an optimization procedure," in *Artificial Life II* (C. Langton, C. Taylor, J. Farmer and S. Rasmussen, eds.), Reading MA: Addison-Wesley Publishing Co. Inc., 1992.

[134] M. Porter and K. De Jong, "A cooperative coevolutionary approach to function optimization," *Parallel Problem Solving from Nature III* (Y. Davidor and H.-P.Schwefel, eds.), pp.249-257, vol.866 of Lecture notes in Computer Science, 1994.

[135] J. Jannink, "Cracking and co-evolving randomizers," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.425-443, MIT Press, Cambridge, MA, 1994.

[136] H. Juille and J.B. Pollack, "Massively parallel genetic programming," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.339-358, MIT Press, Cambridge, MA, 1996.

[137] M.A. Potter, K.A. De Jong and J.J. Grefenstette, "A co-evolutionary approach to learning sequential decision rules," in *Genetic Algorithms: Proceedings of the Sixth International Conference ICGA 95* (L. Eshelman, ed.), pp.366-372, Pittsburgh, PA. Morgan Kaufmann, San Francisco, CA, 1995.

[138] A. Teller, "Evolving programmers: The coevolution of intelligent recombination operators," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.45-68, MIT Press, Cambridge, MA, 1996.

[139] L. Hunter, "Cultural coevolution learning: Synergistic evolution of learning agents and problem representations," Technical Report, 1995.

[140] M.Z. Abramson and L. Hunter, "Classification using cultural co-evolution and genetic programming," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.249-254, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[141] W.A. Tackett, "Mining the genetic program," *IEEE Expert*, vol. 10, no. 3, pp.28-38, 1995.

[142] W.A. Tackett, "Greedy recombination and genetic search on the space of computer programs," in *Foundations of Genetic Algorithms 3* (L.D. whitley and M.D. Vose, eds.), pp.271-297, Morgan Kaufmann, San Francisco, CA, 1995.

[143] T. Soule, J.A. Foster and J. Dickinson, "Code growth in genetic programming," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.215-223, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[144] T. Soule and J.A. Foster, "Code size and depth flows in genetic programming," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), pp.313-320, Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[145] W.B. Langdon and R. Poli, "Fitness causes bloat," Technical Report CSRP-97-09, 1997.

[146] W.B. Langdon, "The evolution of size in variable length representations," in Proceedings of the IEEE International Conference on Evolutionary Computation, 1998.

[147] T. Blickle, "Evolving compact solutions in genetic programming: A case study," in *Parallel Problem Solving from Nature IV. Proceedings of the International Conference on Evolutionary Computation* (H. Voigt, W. Ebeling, I. Rechenberg and H.-P.Schwefel, eds.), pp.564-573, Springer-Verlag, 1996.

[148] N.F. McPhee and J.D. Miller, "Accurate replication in genetic programming," in *Genetic Algorithms: Proceedings of the Sixth International Conference ICGA 95* (L. Eshelman, ed.), pp.303-309, Pittsburgh, PA. Morgan Kaufmann, San Francisco, CA, 1995.

[149] J.P. Rosca and D.H. Ballard, "Complexity drift in evolutionary computation with tree representations," Technical Report NRL 5, Rochester, NY, USA, Dec. 1996.

[150] U.-M. O'Reilly and F. Oppacher, "Hybridized crossover-based search techniques for program discovery," *IEEE World Conference on Evolutionary Computation*, vol.2, pp.573, Perth, Australia. IEEE Press, New York, 1995.

[151] U.-M. O'Reilly and F. Oppacher, "Program search with a hierarchical variable length representation: Genetic programming, simulated annealing and hill climbing," *Parallel Problem Solving from Nature III* (Y. Davidor and H.-P.Schwefel, eds.), vol.866 of Lecture notes in Computer Science, 1994.

[152] E. Aats and J. Korst, *Simulated Annealing and Boltzmann Machines*, John Wiley, 1989.

[153] H. Iba, H. de Garis and T. Sato, "Genetic programming with local hill-climbing," *Parallel Problem Solving from Nature III* (Y. Davidor and H.-P.Schwefel, eds.), vol.866 of Lecture notes in Computer Science, 1994.

[154] D.C. Hooper, N.S. Flann and S.R. Fuller, "Recombinative hill-climbing: A stronger method for genetic programming," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[155] A. Bölte and U.W. Thonemann, "Optimizing simulated annealing schedules with genetic programming," *European Journal of Operations Research*, pp.402-416, 1996.

[156] U.W. Thonemann, "Finding improved simulated annealing schedules with genetic programming," in *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, pp.391-395, Orlando, Florida, IEEE Press, USA, June 1994.

[157] M.L.Wong and K.S. Leung, "An induction system that learns programs in different programming languages using genetic programming and logic grammars," in *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence*, pp.380-387, IEEE Press.

[158] M.L.Wong and K.S. Leung, "Inductive logic programs with genetic algorithms: The genetic logic programming system," *IEEE Expert*, vol. 9, no. 5, pp.68-76, 1995.

[159] L.M. Howard and J. D'Angelo, "The GA-P: A genetic algorithm and genetic programming," *IEEE Expert*, vol. 9, no. 5, pp.11-15, 1995.

[160] A. Geyer-Schulz, "Compound derivations in fuzzy genetic programming," in *Proceedings of the 1996 Biennial Conference of North American Fuzzy Information Society*, pp.510-514, 1996.

[161] A. Geyer-Schulz, "Fuzzy genetic programming and dynamic decision making," in *Proceedings of the Eleventh International Conference on Systems Engineering*, pp.686-691, 1996.

[162] E. Alba, C. Cotta and J.M. Troya, "Type-constrained genetic programming for rule-base definition in fuzzy logic controller," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.255-260, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[163] L. Spector and S. Luke, "Cultural transmission of information in genetic programming," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.209-214, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[164] L. Spector and S. Luke, "Culture enhances the evolvability of cognition," in *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society* (G. Cottrell, ed.), pp.672-677, Lawrence Erlbaum Associates, Mahwah, NJ, 1996.

[165] K.E. Kinnear, Jr. (ed.), *Advances in Genetic Programming*. MIT Press, Cambridge, MA, 1994.

[166] P.J. Angeline and K.E. Kinnear, Jr. (eds.), *Advances in Genetic Programming 2*. MIT Press, Cambridge, MA, 1996.

[167] M.J. Willis, H.G. Hiden, P. Marenbach, B. McKay and G.A. Montague, "Genetic programming: An introduction and survey of applications," *Second International Conference on*

Genetic Algorithms in Engineering Systems: Innovations and Applications GALESIA (A.M.S. Zalzala, ed.), pp.314-319, University of Strathclyde, Glasgow, UK. 1997.

[168] W.B. Langdon and R. Poli, "Genetic programming in Europe," Report of $E_{VO}G_P$ Working Group on Genetic Programming, 1998.

[169] D. Andre, "Automatically defined features:: The simultaneous evolution of 2-dimensional feature detectors and an algorithm for using them," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.477-494, MIT Press, Cambridge, MA, 1994.

[170] D. Andre, "Learning and upgrading rules for an OCR system using genetic programming," *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, Orlando, Florida, IEEE Press, USA, June 1994.

[171] W.A. Tackett, *Recombination, Selection, and the Genetic Construction of Computer Programs*. Ph.D. thesis, University of Southern California, 1994.

[172] B. Masand, "Optimising confidence of text classification by evolution of symbolic expressions," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.445-458, MIT Press, Cambridge, MA, 1994.

[173] D. Andre, F.H. Bennett III and J.R. Koza, "Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.3-11, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[174] J.R. Koza and D. Andre, "Classifying protein segments as transmembrane domains using architecture-altering operations in genetic programming," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.155-176, MIT Press, Cambridge, MA, 1996.

[175] J.R. Koza and D. Andre, "Automatic discovery of protein motifs using genetic programming," in *Proceedings of 1996 Evolutionary Computation: Theory and Applications* (X. Yao, ed.), World Scientific Inc. Press, singapore, 1996.

[176] S. Handley, "Automatic learning of a detector for alpha-helices in protein sequences via genetic programming," *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93* (S. Forrest, ed.), pp.271-278, University of Illinois at Urbana-Champaign, Morgan Kauffman, San Francisco, CA, 1993.

[177] S. Handley, "Automated learning of a detector for the cores of α-helices in protein sequences via genetic programming," *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, pp.474-479, Orlando, Florida, IEEE Press, USA, June 1994.

[178] S. Handley, "A new class of function sets for solving sequence problems," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.301-308, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[179] S. Handley, "The prediction of the degree of exposure to solvent of amino acid residues via genetic programming," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.297-300, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[180] G. Robinson and P. McIlroy, "Exploring some commercial applications of genetic programming," *Evolutionary Computing* (T.C. Fogarty, ed.), vol. 993 of Lecture Notes in Computer Science. Springer-Verlag, 1995.

[181] D.G. Lee, B.W. Lee and S.H. Chang, "Genetic programming model for long-term forecasting of electric power demand," *Electric Power System Research*, vol. 40, pp.17-22, 1997.

[182] W.A. Tackett, "Genetic programming for feature discovery and image discrimination," *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93* (S. Forrest, ed.), pp.303-309, University of Illinois at Urbana-Champaign, Morgan Kauffman, San Francisco, CA, 1993.

[183] W.A. Tackett, "Genetic generation of dendritic trees for image classification," in *Proceedings of the World Conference on Neural Networks*, Portland, IEEE Press, 1993.

[184] M.P. Johnson, P. Maes and T. Darrell, "Evolving visual routines," in *Artificial Life IV: Proceedings of the Fourth International workshop on the Synthesis and Simulation of Living Systems* (R.A. Brooks and P. Maes, eds.), pp.198-209, MIT Press, Cambridge, MA, USA, 1994.

[185] J.M. Daida, T.F. Bersano-Begey, S.J. Ross and J.F. Vesecky, "Computer-assisted design of image classification algorithms: Dynamic and static fitness evaluations in a scaffolded genetic programming environment," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.279-284, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[186] J.M. Daida, J.D. Hommes, T.F. Bersano-Begey, S.J. Ross and J.F. Vesecky, "Algorithm discovery using the genetic programming paradigm: Extracting low-contrast curvilinear features from SAR images of arctic ice," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.417-442, MIT Press, Cambridge, MA, 1996.

[187] P. Nordin and W. Banzaf, "Programmatic compression of images and sound," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.345-350, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[188] R. Poli, "Genetic programming for feature detection and image segmentation," *Evolutionary Computing*, pp.110-125, vol. 1143, Lecture Notes in Computer Science, Springer-Verlag, 1996.

[189] E.H.N. Oakley, "Two scientific applications of genetic programming: Stack filters and non-linear equation fitting to chaotic data," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.369-389, MIT Press, Cambridge, MA, 1994.

[190] C. Harries and B. Buxton, "Evolving edge detectors," research Note RN/96/3, University College of London, UK, 1996.

[191] K.C. Sharman, A.I. Esparcia-Alcazar and Y. Li, "Evolving signal processing algorithms by genetic programming," *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications GALESIA* (A.M.S. Zalzala, ed.), vol. 414, pp.473-480, Sheffield, UK. IEE London, UK, 1995.

[192] A.I. Esparcia-Alcazar and K.C. Sharman, "Some applications of genetic programming in digital signal processing," *Late breaking papers at the Genetic Programming 1996 Conference* (J.R. Koza, ed.), Stanford University Book Store, pp.24-31, 1996.

[193] M.A. Keane, J.R. Koza and J.P. Rice, "Finding an impulse response function using genetic programming," in *Proceedings of the American Control Conference*, San Francisco, CA, pp.2345-2350, 1993.

[194] H. Iba, T. Kurita, H. Garis and T. Sato, "System identification using structured genetic algorithms, in *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93* (S. Forrest, ed.), pp.276-2866, University of Illinois at Urbana-Champaign, Morgan Kauffman, San Francisco, CA, 1993.

[195] E.H.N. Oakley, "The application of genetic programming to the investigation of short, noisy, chaotic data series," *Evolutionary Computing* (T.C. Fogaty, ed.), vol.865 of the *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[196] B.-T. Zhang and H. Müehlenbein, "Synthesis of sigma-pi neural networks by the breeder genetic programming," *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, pp.318-323, Orlando, Florida, IEEE Press, USA, June 1994.

[197] B.-T. Zhang and H. Müehlenbein, "Adaptive fitness functions for dynamic growing/pruning of program trees," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.241-256, MIT Press, Cambridge, MA, 1996.

[198] B.-T. Zhang, "Design and training of neural network models by genetic programming," *Journal of KISS B Software and Applications*, pp.1083-1092, 1996.

[199] K.D. Bettenhausen and P. Marenbach, "Self-organising modelling of biotechnological batch and fed batch fermentations," in *Proceedings of EUROSIM*, pp.445-450, 1995.

[200] K.D. Bettenhausen, P. Marenbach, S. Freyer, H. Rettenmaier and U. Nieken, "Self-organising structure modelling of a biotechnological fed batch fermentation by means of genetic programming," *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications GALESIA* (A.M.S. Zalzala, ed.), vol. 414, pp.481-486, Sheffield, UK. IEE London, UK, 1995.

[201] P. Marenbach, K.D. Bettenhausen and S. Freyer, "Signal path orientation approach to generation of dynamic process model," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.327-332, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[202]] P. Marenbach, K.D. Bettenhausen, S. Freyer, U. Nieken and H. Rettenmaier, "Data driven structured modelling of a biotechnological fed-batch fermentation by means of genetic programming," *Journal of Systems and Control Engineering, Proceedings of the Institution of Mechanical Engineers*, 1997.

[203] P. Marenbach and M. Brown, "Evolutionary versus inductive construction of neuro-fuzzy systems for bioprocess modelling," *Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications GALESIA* (A.M.S. Zalzala, ed.), pp.320-325, University of Strathclyde, Glasgow, UK. 1997.

[204] H. Hiden, M. Willis, B. McKay and G. Montague, "Non-linear and direction dependent dynamic modelling using genetic programming," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), pp.168-173, Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[205] G.J. Gray, D.J. Murray-Smith, Y. Li and K.C. Sharman, "Non-linear model structure identification using genetic programming," *Late Breaking Papers at the Genetic Programming 1996 Conference* (J.R. Koza, ed.), pp.32-37,Stanford University, CA. Stanford Book Store, Stanford, CA, 1996.

[206] G.J. Gray, D.J. Murray-Smith, Y. Li and K.C. Sharman, "Structural system identification using genetic programming and a block diagram oriented simulation tool," *Electronic Letters*, vol. 32, no. 15, pp.1422-1424, 1996.

[207] G.J. Gray, T. Weinbrenner, D.J. Murray-Smith, Y. Li and K.C. Sharman, "Issues in non-linear model structure identification using genetic programming," *Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications GALESIA* (A.M.S. Zalzala, ed.), pp.308-313, University of Strathclyde, Glasgow, UK. 1997.

[208] M. South, C. Bancroft, M.J. Willis and M.T. Tham, "System identification via genetic programming," *Proceedings of the UKACC International Conference*, pp.912-917, 1996.

[209] B. McKay, M. Willis and G. Barton, "Steady state modelling of chemical process systems using genetic programming," *Computers & Chemical Engineering*, vol. 21, no. 9, pp.981-996, 1997.

[210] K. Rodriguez-Vàzquez and P.J. Fleming, "A genetic programming/NARMAX approach to non-linear system identification," *Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications GALESIA* (A.M.S. Zalzala, ed.), pp.409-414, University of Strathclyde, Glasgow, UK. 1997.

[211] B. McKay, C. Sanderson, M.J. Willis, J. Barford and G. Barton, "Evolving a hybrid model of a batch fermentation process," *Transaction of the Institute of Measurement and Control*, 1997.

[212] J. Elsey, J. Riepenhausen, B. McKay, G.W. Barton and M.J. Willis, "Modelling and control of a food extrusion process," in *Proceedings of the Joint International Symposium on Process Systems Engineering and European Symposium on Computer Aided Design PSE97/ESCAPE 7*, Norway, 1997.

[213] M. Schoenauer, M. Sebag, F. Jouve, B. Lamy and H. Maitournam, "Evolutionary identification of macro-mechanical model," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.467-488, MIT Press, Cambridge, MA, 1996.

[214] J.R. Koza and M.A. Keane, "Genetic breeding of non-linear optimal control strategies for broom balancing," in *Proceedings of the Ninth International Conference on Analysis and Optimization of systems*, pp.47-56, France, 1990.

[215] R.J. Hampo, "Genetic programming: A new paradigm for control and analysis," in *Proceedings of the Third ASME Symposium on Transportation Systems*, USA, pp.155-163, 1992.

[216] Marko, K.A. and Hampo, R.J., "Application of genetic programming to control of vehicle systems," in *Proceedings of the Intelligent Vehicles '92 Symposium*, Detroit, MI. pp.191-195, 1992.

[217] J.R. Koza, "Evolving emergent wall following robotic behavior using the genetic programming paradigm," in *Proceedings of ECAL*, Paris, 1991.

[218] J.R. Koza, "Evolution of subsumption using genetic programming," in *Proceedings of the First European Conference on Artificial Life. Towards a Practice of Autonomous Systems*, pp.110-119, Paris, France. MIT Press, Cambridge, MA, 1992.

[219] S.J. Ross, J.M. Daida, C.M. Doan, T.F. Bersano-Begey and J.J. McClain, "Variation in the evolution of subsumption architectures using genetic programming: The wall following robot revisited," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.191-199, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[220] A. Goldish, "Noisy wall following and maze navigation through genetic programming," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.423, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[221] J.R. Koza and J.P. Rice, "Automatic programming of robots using genetic programming," in *Proceedings of AAAI'92*, pp.194-201, MIT Press, 1992.

[222] W.-P. Lee, J. Hallam and H.H. Lund, "Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp.501-506, 1997.

[223] P. Chongstitvatana and J. Polvichai, "Learning a visual task by genetic programming," in *Proceedings of the IEEE /RSJ International Conference on Robotics*, pp.534-540, 1996.

[224] C.W. Reynolds, "An evolved, vision-based model of obstacle avoidance behavior," in *Artificial Life III* (C. Langton, ed.), Addison Wesley, 1993.

[225] A.H. Watson and I.C. Parmee, "Steady state genetic programming with constrained complexity crossover using species sub-populations," in Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA 97), Morgan Kaufmann, San Francisco, CA, 1998.

[226] J.R. Koza, *Genetic Programming II Video Tape: The next Generation*. Cambridge, MA, MIT Press, 1994.

[227] C.W. Reynolds, "The difficulty of roving eyes," *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, pp.262-267, Orlando, Florida, IEEE Press, USA, June 1994.

[228] S. Handley, "The genetic planner: the automatic generation of plans for a mobile robot via genetic programming," *in Proceedings of the Eighth IEEE International Symposium on Intelligent Control*, Chicago, IL. pp.190-195, 1993.

[229] S. Handley, "The automatic generation of plans for a mobile robot via genetic programming with automatically defined functions," in *Proceedings of the Fifth Workshop on Neural Networks: Neural Networks, Fuzzy Systems, Evolutionary Programming and Virtual Reality*, pp.73-78, 1993.

[230] S. Handley, "The automatic generation of plans for a mobile robot via genetic programming with automatically defined functions," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.391-407, MIT Press, Cambridge, MA, 1994.

[231] T.D. Haynes and R.L. Wainwright, "A simulation of adaptive agents in a hostile environment," in *Proceedings of the 1995 ACM Symposium on Applied Computing*, pp.318-323, ACM Press, 1995.

[232] P. Nordin and W. Banzaf, "Genetic programming controlling a miniature robot," *Working Notes for the AAAI Symposium on Genetic Programming*, pp.61-67, MIT, Cambridge, MA. AAAI, Menlo Park, CA, 1995.

[233] P. Nordin and W. Banzaf, "An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming," *Adaptive Behavior*, 5, pp. 107-140, 1997.

[234] P. Nordin and W. Banzaf, "Real time control of a khepera robot using genetic programming," *Control and Cybernatics*, 26 (3), 1997.

[235] W. Banzaf, P. Nordin and M. Olmer, "Generating adaptive behavior for a real robot using function regression with genetic programming," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), pp.35-43, Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[236] C.W. Reynolds, "An evolved vision-based behavioral model of co-ordinated group motion," *From Animals to Animats 2: Proceedings of the Second International Conference on the Simulation of Adaptive Behavior*, MIT Press, 1993.

[237] G.F. Spencer, "Automatic generation of programs for crawling and walking," *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93* (S. Forrest, ed.), pp.654, University of Illinois at Urbana-Champaign, Morgan Kauffman, San Francisco, CA, 1993.

[238] J.R. Rush, A.P. Fraser and D.P. Barnes, "Evolving co-operation in autonomous robotic systems," in *Proceedings of the IEE International Conference on Control*, IEE London, UK, 1994.

[239] S.N. Taylor, "Evolution by genetic programming of a spatial robot juggling control algorithm," *Proceedings of the Workshop on Genetic Programming: From Theory to Real-world Applications* (J.P. Rosca, ed.), pp.104-110, Tahoe City, CA, 1995.

[240] C.G. Lott, "Terrain Flattening by autonomous robot: A genetic programming application," *Genetic Algorithms at Stanford 1994* (J.R. Koza, ed.), pp.99-109, Stanford Book Store, Stanford, CA, 1994.

[241] T, Ito, H. Iba and M. Kimura, "Robustness of robot programs generated by genetic programming," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.321-326, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[242] F.H. Bennett III, "A multi-skilled robot that recognizes and responds to different problem environments," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), pp.45-53, Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[243] S. Luke and L. Spector, "Evolving teamwork and coordination with genetic programming," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.150-156, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[244] T.D. Haynes, R.L. Wainwright and S. Sen, "Evolving cooperating strategies," in *Proceedings of the First International Conference on Multiple Agent Systems* (V. Lesser, ed.), pp.450, San Francisco, CA, AAAI, CA/MIT Press, Cambridge, MA, 1995.

[245] A. Qureshi, "Evolving agents," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.369-374, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[246] C. Ryan, "GProbots and GPteams-competition, co-evolution and co-operation in genetic programming," in *Proceedings of the GP Papers from 1995 AAAI Fall Symposium*, pp.86-93, 1995.

[247] H. Iba, "Multiple-agent learning for a robot navigation task by genetic programming," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), pp.195-200, Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[248] H. Iba, T. Nozoe and K. Ueda, "Evolving communicating agents based on genetic programming," *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp.297-302, 1997.

[249] B. Howley, "Genetic programming and parametric sensitivity: a case study in dynamic control of a two link manipulator," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), pp.180-185, Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[250] J. Gibbs, "Easy inverse kinematics using genetic programming," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.422, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[251] F. Gruau, "Cellular encoding of genetic neural networks," Technical Report 92-21, France, 1992.

[252] F. Gruau and D. Whitley, "Adding learning to the cellular development process: A comparative study," *Evolutionary Computation*, 1 (3), pp.213-233, 1993.

[253] F. Gruau, D. Whitley and L. Pyeatt, "A comparison between cellular encoding and direct encoding for genetic neural networks," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.81-89, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[254] I. Kusçu and C. Thornton, "Design of artificial neural networks using genetic algorithms: Review and prospect," Technical Report 1994.

[255] H. Kitano, "Designing neural networks using genetic algorithms with graph generation system," *Complex Systems*, 4, pp.461-476, 1990.

[256] B.-T. Zhang and H. Mühlenbein, "Genetic programming of minimal neural neural nets using Occam's razor," *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93* (S. Forrest, ed.), pp.342-349, University of Illinois at Urbana-Champaign, Morgan Kauffman, San Francisco, CA, 1993.

[257] S. Benigo, Y. Benigo and J. Cloutier, "Use of genetic programming for the search of a new learning rule for neural networks," *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, pp.324-327, Orlando, Florida, IEEE Press, USA, June 1994.

[258] A.I. Esparcia-Alcazar and K. Sharman, "Some applications of genetic programming in digital signal processing," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.24-31, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[259] A.I. Esparcia-Alcazar and K. Sharman, "Genetic programming techniques that evolve recurrent neural network architectures for signal processing," in *Neural Networks for Signal Processing - Proceedings of IEEE Work Shop*, pp.139-148, 1996.

[260] T. Nguyen and T. Huang, "Evolvable 3D modelling for model-based object recognition systems," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.459-475, MIT Press, Cambridge, MA, 1994.

[261] J.R. Koza, F.H. Bennett III, D. Andre and M.A. Keane, "Automated WYWIWYG design of both the topology and component values of electrical circuits using genetic programming," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.123-131, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[262] J.R. Koza, F.H. Bennett III, D. Andre and M.A. Keane, "Four problems for which a computer program evolved by genetic programming is competitive with human performance," in *Proceedings of the 1966 IEEE International Conference on Evolutionary Computation*, vol. 1, pp.1-10, IEEE Press, New York, 1996.

[263] J.R. Koza, D. Andre, F.H. Bennett III, D. Andre and M.A. Keane, "Evolution of a low-distortion, low bias 60 decibel OpAmp with good frequency generalization using genetic programming," *Late breaking papers at Genetic Programming 1996 Conference*, pp.94-100, Stanford, 1996.

[264] J.R. Koza, F.H. Bennett III, D. Andre and M.A. Keane, "Toward evolution of electronic animals using genetic programming," 1997.

[265] J.R. Koza, F.H. Bennett III, D. Andre, M.A. Keane and F. Dunlap, "Automated synthesis of analog electrical circuits be means of genetic programming" *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 2, pp.109-128, 1997.

[266] J.R. Koza, F.H. Bennett III, D. Andre and M.A. Keane, "Evolutionary design of analog electrical circuits using genetic programming," in *Adaptive Computing in Design and Manufacturing* (I.C. Parmee, ed.) pp.177-192, Springer-Verlag London Limited,1998.

[267] J.R. Koza, D. Andre, F.H. Bennett III, D. Andre and M.A. Keane, "Use of automatically defined functions and architecture-altering operations in automated circuit synthesis using genetic programming," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.132-149, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[268] M.S. Atkin and P.R. Cohen, "Learning monitoring strategies: A difficult genetic programming application," *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, pp.328-332, Orlando, Florida, IEEE Press, USA, June 1994.

[269] C.A. Grimes, "Application of genetic techniques to the planning of railway track maintenance work," *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications GALESIA* (A.M.S. Zalzala, ed.), vol. 414, pp.467-472, Sheffield, UK. IEE London, UK, 1995.

[270] J. Garces-Perez, D.A. Schoenefeld and R.L. Wainwright, "Solving facility layout problems using genetic programming," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.182-190, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[271] D.J. Montana and S. Czerwinski, "Evolving control laws for a network of traffic signals," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.333-338, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[272] H.A. Watson and I.C. Parmee, "Improving engineering design models using an alternative genetic programming approach," in *Adaptive Computing in Design and Manufacturing* (I.C. Parmee, ed.) pp.193-206, Springer-Verlag London Limited,1998.

[273] J.E. Perry, "The effect of population enrichment in genetic programming," *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, pp.456-461, Orlando, Florida, IEEE Press, USA, June 1994.

[274] M.A. Warren, "Stock price time series prediction using genetic programming," *Genetic Algorithms at Stanford 1994* (J.R. Koza, ed.), pp.180-182, Stanford Book Store, Stanford, CA, 1994.

[275] J.T. Eglit, "Trend prediction in financial time series," *Genetic Algorithms at Stanford 1994* (J.R. Koza, ed.), pp.31-39, Stanford Book Store, Stanford, CA, 1994.

[276] M. Andrews and R. Prager, "Genetic programming for the acquisition of double auction market strategies," *Advances in Genetic Programming* (K.E. Kinnear, Jr., ed.), pp.355-368, MIT Press, Cambridge, MA, 1994.

[277] S.-H. Chen and C.-H. Yeh, "Genetic programming learning and cobweb model," *Advances in Genetic Programming 2* (P.J. Angeline and K.E. Kinnear, Jr., eds.), pp.443-465, MIT Press, Cambridge, MA, 1996.

[278] S.-H. Chen and C.-H. Yeh, "Bridging the gap between non-linearity tests and the efficient market hypothesis by genetic programming," *in Proceedings of the IEEE/IAFE Conference on Computation Intelligence for France*, pp.34-40, 1996.

[279] S.-H. Chen and C.-H. Yeh, "Using genetic programming to model volatility in financial time series," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), pp.58-63, Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[280] M. Oussaidene, B. Chopard, O.V. Pictet and M. Tomassini, "Parallel genetic programming: An application to trading models evolution," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.357-380, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[281] L. Gritz and J.K. Hahn, "Genetic programming for articulated figure motion," *Journal of Visualization and Computer Animation*, 6, pp.129-142, 1995.

[282] L. Gritz and J.K. Hahn, "Genetic programming evolution of controllers for 3-D character animation," *Genetic Programming 1997: Proceedings of the Second Annual Conference* (J.R.

Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo, eds.), pp.139-146, Stanford University, CA. Morgan Kaufmann, San Francisco, CA, 1997.

[283] S. Das, T. Franguidakis, M. Papka, T.A. DeFanti and D.J. Sandin, "A genetic programming application in virtual reality," *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol.1, pp.480-484, Orlando, Florida, IEEE Press, USA, June 1994.

[284] L. Spector and A. Alpern, "Criticism, culture, and the automatic generation of artworks," in *Proceedings of Twelfth National Conference on Artificial Intelligence*, pp.3-8, Seattle, WA. AAAI, CA/MIT Press, Cambridge, MA, 1994.

[285] M. Crosbie and E.H. Spafford, "Applying genetic programming to intrusion detection," Working Notes for the AAAI Symposium on Genetic Programming (E.V. Siegel and J.R. Koza, eds.), pp.1-8, MIT, Cambridge, MA, 1995.

[286] W.S. Bruce, "Automatic generation of object oriented programs using genetic programming," *Genetic Programming 1996: Proceedings of the First Annual Conference* (J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L. Riolo, eds.), pp.267-272, Stanford University, CA. MIT Press, Cambridge, MA, 1996.

[287] P. Tufts, "Dynamic classifiers: Genetic programming and classifier systems," *GP papers from 1995 AAAI fall Symposium*, pp.114-119, 1995.