

# GENETIC PROGRAMMING WITHIN CIVIL ENGINEERING

David Shaw<sup>1</sup>, John Miles<sup>1</sup> and Alex Gray<sup>2</sup>

<sup>1</sup>Cardiff School of Engineering, Cardiff University, UK  
ShawD2@cardiff.ac.uk

<sup>2</sup>Department of Computer Science, Cardiff University, UK

*The application of Genetic programming to civil engineering design problems is relatively new. This paper both describes and demonstrates how by using a suitable form of representation based on graph networks, GP can be applied to structural design problems to produce solutions that offer significant improvements over traditional GA based methods. The paper concludes by presenting the direction of the work currently being undertaken at Cardiff University, which includes the conceptual design of frame structures and the visualisation of results using X3D to produce virtual reality simulations that can be used as a collaborative environment over the Internet.*

**Keywords: Civil Engineering, Genetic Programming, Structural Optimisation, X3D.**

## 1 An introduction to genetic programming.

This section provides a general background to the topic of genetic programming (GP) that is expanded in subsequent sections to demonstrate how GP can be used in structural engineering. Genetic programming (GP) is a domain-independent, problem-solving approach in which computer programs are evolved to find solutions to problems. The solution technique is based on the Darwinian principle of 'survival of the fittest' [1] and is closely related to the field of genetic algorithms (GA) [2]. However three important differences exist between GAs and GP:

- *Structure:* GP usually evolves tree structures while GA's evolve binary or real number strings.
- *Active vs Passive:* Because GP usually evolves computer programs, the solutions can be executed without post-processing i.e. active structures, while GA's typically operate on coded binary strings i.e. passive structures, which require post-processing.
- *Variable vs fixed length:* In traditional GAs, the length of the binary string is fixed before the solution procedure begins. However a GP parse tree can vary in length throughout the run. Although it is recognised that in more advanced GA work, variable length strings are used.

The ability to search the solution space and locate regions that potentially contain optimal solutions for a given problem, is one of the fundamental components of most artificial intelligence (AI) systems. There are three primary types of search; the blind search, hill climbing and beam search [3]. GP is classified as a beam search because it maintains a population of solutions that is smaller than all of the available solutions. GP is also usually implemented as a weak search algorithm as it contains no problem specific knowledge, although some research has been directed towards 'strongly typed genetic programming' [4]. However while GP can find regions containing optimal solutions, an additional local search algorithm is normally required to locate the optima. Memetic algorithms can fulfil this role, by combining an evolutionary algorithm with problem specific search algorithm to locate optimal solutions [5].

### 1.1 The primitives of genetic programming.

Every solution evolved by GP is assembled from two sets of primitive nodes; terminals and functions. The terminal set contains nodes that provide an input to the GP system while the function set contains nodes that processes values already in the system [3]. Constants can be used in GP by including them in the terminal set. Once the evolutionary process is started, the GP system randomly selects nodes from either set and thus may not utilise all of the available nodes. However increasing the size of each node set enlarges the search space. Therefore only a relatively simple node set is initially provided and nodes are usually added only if required.

### 1.2 Tree based genetic programming.

The primitives of GP, the function and terminal nodes, must be assembled into a structure before they may be executed. Three main types of structure exist: tree, linear and graph. Within this work, the input (the structure to be optimised or designed) actually forms a graph network. However by the duplication of joint data i.e. the same 'joint node' can exist in the same tree on more than one occasion, this graph network is converted into a tree structure (see Section 3).

### 1.3 Genetic operators.

There are three major evolutionary operators within a GP system:

- *Reproduction:* selects an individual from within the current population to be copied exactly into the next generation. There are several ways of selecting which individual is to be copied including 'fitness proportionate' selection, 'rank' selection and 'tournament' selection.

- *Crossover*: mimics sexual recombination in nature, where two parent solutions are chosen and parts of their sub-tree are swapped and because each function exhibits the property 'closure' (each tree member is able to process all possible argument values), every crossover operation should result in the formation of a legal structure.
- *Mutation*: causes random changes in an individual before it is introduced into the subsequent population. Unlike crossover, mutation is asexual and thus only operates on one individual. During mutation either all functions and terminals are removed beneath an arbitrarily determined node and a new branch is randomly created, or a single node is swapped for another.

#### 1.4 Generational genetic programming.

GP has developed two main approaches to dealing with the issue of its generations; generational and steady-state. In generational GP, there exists well-defined and distinct generations, with each generation being represented by a complete population of individuals. Therefore each new population is created from the older population, which it then replaces. Steady-state GP does not maintain these discrete generations but continuously evolves the current generation using any available genetic operators [3]. This work uses the generational approach to GP.

## 2 Applications of genetic programming in civil engineering.

Procedural programming techniques have been successfully used for the detailed design of structures where the path to the final solution is known in advance. However other design stages lack this predefined approach and thus it becomes necessary to adopt self-learning/ organising computing techniques. This work applies GP at the conceptual design stage, of the design process, with the aim of providing the designer with a series of possible solutions that can be carried forward to the later stages. The following table lists all the published applications of genetic programming in civil engineering.

Application	Author	Year	Details
Shear strength prediction of deep RC beams	Ashour et al [6]	2003	Estimation of the shear strength of deep RC beams, subjected to two point loads, from 141 published experimental results.
Modelling of wastewater treatment plants	Hong and Bhamidimarri [7]	2003	Use of genetic programming to model the dynamic performance of municipal activated sludge wastewater treatment plants.
Detection of traffic accidents.	Roberts and Howard [8]	2002	Detection of accidents on motorways in low flow, high-speed conditions i.e. late at night based on three years of traffic data whilst producing a near zero false alarm rate.
Flow through a urban basin	Dorado et al [9]	2002	Construction of sewage network model in order to calculate the risk posed by rain to the basin and thus provide prior warning of flooding or subsidence.
Prediction of journey times	Howard and Roberts [10]	2002	Investigation of GP to forecast the motorway journey times.
Estimation of design intent	Ishino and Jin [11]	2002	Using GP to automatically estimate design intent based on operational and product-specific information monitored throughout the design process.
Modelling of water supply assets	Babovic et al [12]	2002	In order to determine the risk of a pipe burst, a GP is evolved to 'data mine' a database containing information about historic pipe bursts.
Identification of crack profiles	Kojima et al. [13]	2001	Detection of cracks inside hundreds of heat exchanger tubes in a nuclear power plant's steam generator via analysis of data measured via quantitative non-destructive testing.
Modelling rainfall runoff	Whigham & Crapper [14]	2001	Discovery of rainfall-runoff relationships in two vastly different catchments.
Prediction of long-term electric power demand	Lee et al [15]	1997	Symbolic regression via genetic programming to predict the long-term electric demand of Korea based on training data from 1961 to 1980.
Evolution of traffic light control laws	Montana and Czerwinski [16]	1996	Evolution of a new type of adaptive control system for a network of traffic signals depending on variations in traffic flow.
Identification of crack profiles	Köppen and Nickolay [17]	1996	Agent generation to detect and track dark regions that could be cracks in greyscale images of textured surfaces.

**Table 1: Applications of GP in Civil Engineering**

### 3 Application of genetic programming in structural engineering.

All but one of the published applications in civil engineering to date, utilise a GP system to evolve relationships between variables as suggested by Koza [1] e.g. symbolic regression functions. This section describes how GP can be applied to structural optimisation problems by using the tree structure of GP, to represent a building or structure.

In 2000 Soh and Yang [18] published an approach that solved the key issue of how to represent the ‘node-element diagram’ of structural analysis as a ‘point-labelled tree’ (as used in GP). However because the structure (the phenotype) is now different from the GP tree (the genotype), an additional decoding step must be included in the solution procedure before any fitness evaluation can occur (Fig.3). This step was not previously required when evolving regression functions, as these solutions could be applied directly to the problem. Although this is a departure from traditional GP, by utilising this representation Soh and Yang demonstrated a system that produced better results when attempting to simultaneously optimise the geometry, sizing and topology of a truss than work using other evolutionary search techniques. It is also important to note that this tree will not degenerate into a linear search as its corresponding structure would be a mechanism. A mechanism is not a viable structure in engineering and thus will be heavily penalised by the fitness function and therefore should not be sustained within the population.

#### 3.1 Structural encoding.

Soh and Yang [18] propose that the GP tree should compose two types of node: inner nodes which are GP functions representing the cross-sectional areas of the members  $A_p$  ( $p= i,j,k,l,m,n$ ) and outer nodes which are GP terminals representing various node points  $N_i$  ( $i= 1,2,3,4$ ) (Fig.1). To create a GP parse tree, one member must be selected from the structure to be the root node. This member then has its’ corresponding start and end joints represented by children nodes to its left and right. Then from left to right the members associated with each joint node are removed from the structure and used to replace the relevant joint nodes in the GP tree (Fig.1). This procedure continues until every structural member is represented in the GP tree. However it is important that the left-right relationship of child and parent node is maintained as the GP tree is constructed. Therefore each members’ start and end joints are represented by the far left and far right children. For example function  $A_j$  connects nodes  $N_1$  and  $N_2$  (Fig.1).

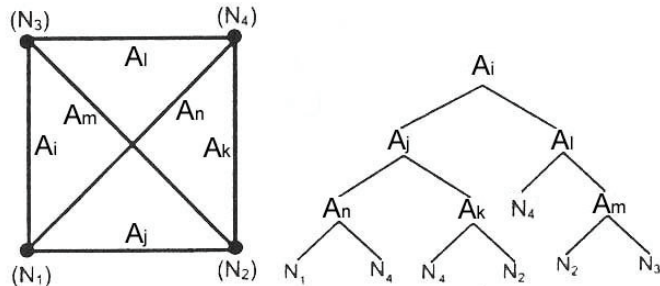


Figure 1: GP tree encoding for structural analysis

This approach to structural encoding appears very simple when compared to the complex binary string used by a GA to represent member properties. For example Soh and Yang [18] have published their results for the evolution of a truss that was capable of carrying six specified loads. Soh and Yang required a GP tree containing 29 nodes (16 joint nodes and 15 member nodes) where as the GA representation required a chromosome of 25,200 bits [18].

#### 3.2 Architecture of the current work.

The work currently being undertaken at Cardiff University uses the structural encoding approach listed above to produce a tree based representation of an engineering structure, however many important differences do exist. This work does not use a finite element package (FEM) to analyse the each possible structure as FEM is computationally very intensive which slows down the generation of results. It was considered more important at the conceptual design stage quickly to deliver to the designer, a range of possible solutions that could be analysed at a later date by FEM. This work also uses an XML interface to allow the designer to input structures to the system and the same interface also enables the 3D visualisation of results, using models created by the virtual reality modelling language (VRML). The work is delivered using Java and at all levels the principle of objects is utilised so that each tree is an object composed of separate node objects.

### 4 An Example of Structural Optimisation.

This section highlights the differences and improvements between the current work and existing GA and GP based work. The following example aims to minimise the overall weight of the structure by selecting progressively lighter members that can sustain the necessary forces and moments. Whilst it is acknowledged that other factors contribute to the overall cost of a structure e.g material cost and fabrication, structural weight does make a substantial contribution to a structure’s cost and therefore it is used here as a first level approximation to a minimum cost structure.

### 4.1 10 Member Planar Truss.

An illustrative example of a 10-member planar truss is now provided (Fig.2) that compares the current GP based work to the results generated by a previous GA and GP based systems. In this example, given the initial structure, the goal was to minimise the structure's weight by varying its' shape and topology. The solution is reported by splitting it into three sections the Model, View and Controller. These three sections relate to the 'MVC' architecture that this work is based around. MVC separates the underlying GP system from the input GUI and the visualisation output.

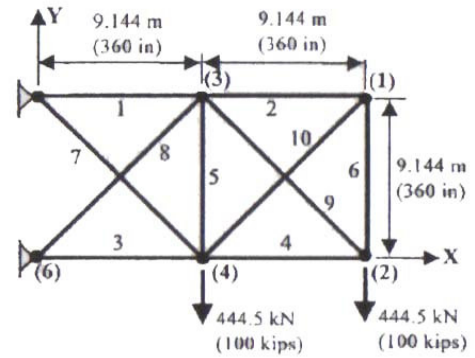


Figure 2: 10 member planar truss

### 4.2 Controller- GP Tableau.

The Controller allows the user to input data and constraints into the system. With this work, the user can alter the variables that control the evolutionary process, the population size and number of generations. As optimisation problems do not usually have an explicitly defined end point, the GP system runs for a stipulated number of generations before returning the 'best-so-far' individual as the solution. The other essential input for structural optimisation problems is an initial structure to improve.

Within this work, the structure is represented by an XML Document Type Definition (DTD) created by MIT [19] called 'StructureXML'. StructureXML not only provides a semantic, storage format for a structure but also provides an interface that allows this system to separate its' structural analysis, evolutionary and visualisation modules. However the StructureXML DTD does not provide any information about a structure's response due to loading. Therefore a new XML DTD was created called 'GPStructureXML' that includes the forces and moments experienced by a structural member and the displacement, rotation, reaction forces and moments experienced each individual joint. The following GP Tableau summaries the main choices made when applying the major preparatory steps of GP to the problem.

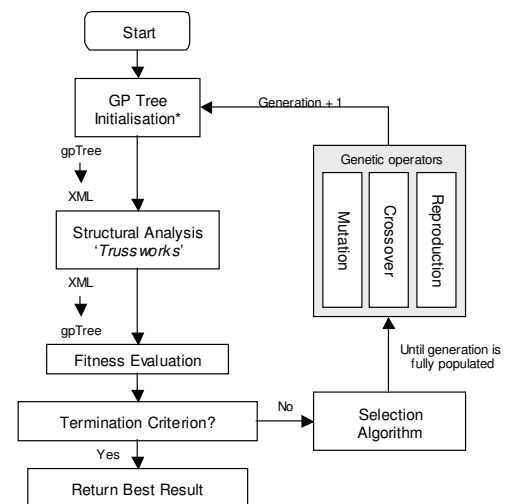
Objective	For a given 10 Member Planar Truss, subjected to two loads acting simultaneously the objective is to minimise its weight by varying shape and topology.
Terminal Set	Structural joints.
Function Set	Structural members.
Fitness Case	
Raw Fitness	Multiplying the structure's objective function (overall weight) by the structure's corresponding penalty factor.
Standardised Fitness	Equals the raw fitness for this problem.
Hits	
Wrapper	
Parameters	M=500, G=51, $p_c=0.8$ , $p_m=0.1$ ( $p_r=0.04$ , $p_f=0.04$ , $p_s=0.02$ )
Success Predicate	None.

Table 2: GP Tableau for the 10 member planar truss problem

### 4.3 Model.

The Model represents the underlying data of the object, in this instance the structural analysis and evolutionary parts of the overall system. As stated previously the each GP tree should be composed of two types of node: the inner nodes which are GP functions representing the trusses members and the outer nodes which are GP terminals representing the structural nodes. This system uses Java based objects to represent each node therefore encapsulation can be used to simplify the genetic operators as the actual data is 'hidden'. When a GP parse tree is created using a fully object orientated language, encapsulation adds to this simple approach.

The fitness function rewards a lightweight truss but penalises any structure that violates a specified constraint e.g. maximum allowable member stress. A penalty based approach was employed, rather than outright rejection because the good solutions will typically be located on the boundary between the



\* Each newly created gpTree is checked for uniqueness before it is inserted into the population to ensure the variety of the initial population is 100%.

Figure 3: The solution procedure

feasible and infeasible regions. Therefore this fitness function allows the GP search to approach the optimum solution from the direction of both the feasible and infeasible regions. However, no stochastic search method can guarantee to find the optimal solution but as this system is to be used during the conceptual design stage this is not essential as its' aim is to provide a number of options, to the designer, to be considered and carried forward to the detailed design stage.

The structural analysis module is composed of the 'Trussworks' package [19] that allows users to create and analyse 3D structures using the Direct Stiffness Method. Trussworks was selected because its source code is freely available and it is written in the same language as the all other parts of the system (Java). A finite element package was also considered but this would result in a significant increase in computational power and thus slow down the solution procedure and would also require a StructureXML interface to be created. It was decided that at the conceptual stage a faster but more approximate answer would be advantageous as FEM would be used during the design stage.

The results from this work indicate that for the 10 bar truss, the GP system produced a weight saving of 0.5%, when compared to a GA based method [20] however it only produced a saving of 0.01% over the results published by Soh and Yang [18]. However the more complicated the structure the greater the potential weight saving of a GP based method over a GA. For example this work found a weight saving of 0.1% when optimising a 25 bar space truss, over the existing GP system and 3.5% over the best GA work [20]. While these improvements may not seem significant it must be remembered that the solution space for these problems is relatively small and that these case studies have been repeatedly used as benchmarks.

#### 4.4 View- Visualisation.

The View accesses the results data from the Model and specifies how it is displayed. Visualisation is very important at the conceptual design stage as it allows all interested parties e.g. client, architect and engineer to collaborate more closely. Three types of visualisation are to be employed using this system. The first two relate to the visualisation of the GP parse tree: the traditional and radial while the third displays the actual structure using virtual reality.

The traditional approach to viewing GP trees starts with the root node and continues downward, with the terminal nodes at the base of the tree (Fig.4). However if the total number of nodes or the tree depth does not remain small then this method of depiction does not readily allow for the viewer to understand the tree's shape or balance as the representation becomes very large. Radial visualisation as proposed by Daida et al [21] solves these problems by mapping the GP tree nodes onto a set of concentric rings according to depth with each node being equally spaced onto the perimeter (Fig.5).

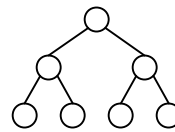


Figure 4: Traditional

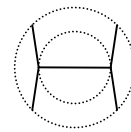


Figure 5: Radial

The third visualisation transforms any GPStructureXML document produced by the system into a X3D based, virtual reality model. The X3D specification has been developed by the Web3D consortium [22] and is currently under ISO Standard review. Touted as 'next-generation open standard for 3D on the WWW', X3D essentially extends and upgrades the capabilities of VRML97 using XML and allows the user to not only visualise the structure created but also to 'fly' around it to view it from any angle. X3D's extrusion node has been utilised in this visualisation process because given a cross-section and length, X3D can produce an object of any size. The main advantage of X3D as a file format is that the visualisations produced are very small in size typically under 100kB and thus can easily be transmitted over the internet with ease and secondly they do not require much processing power to view as they can use any Internet browser with the correct 'plug-in' installed.

#### 4.5 Conclusion.

This GP based system has been shown to provide better results to an optimisation problem when compared to an equivalent GA system while using a method of representation that is significantly more simplistic than that used by the GA. The use of XML not only provides an interface between the components of this system and other programs but also allows for the visualisation of the complete structure using a virtual reality model.

### 5 Future Aims.

Current literature has only one application of GP in structural analysis and that is the optimisation and design of trusses. However this project aims to extend GP to the conceptual stage of multi-storey office building design. Here the GP will be used to generate several novel solutions some of which the designers will take forward into subsequent design stages. This will provide many challenges as initially any GP parse tree must contain two types of member node i.e. column and beam but both of these members allow the GP tree to remain as a binary tree (Fig.6). However if a slab is also to be included because a slab must span between more than two nodes, the GP tree is no longer a binary tree (Fig.7).

The other main area of research is to allow the X3D visualisations to show the deformation of the structure under load and by using server software designed by Blaxxun create a structure that can be viewed both independently and in collaboration over the Internet. This would enable a distributed design teams to view the structure in a collaborative virtual reality environment although issues related to tethering and reattachment [23] will be beyond the scope of this project.

**6 R**  
**refer**  
**ence**  
**s.**

[1]  
 Koza  
 J.R.,  
 Gene  
 tic

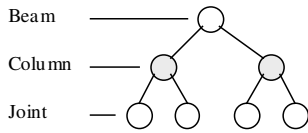


Figure 6: Column/ beam binary tree

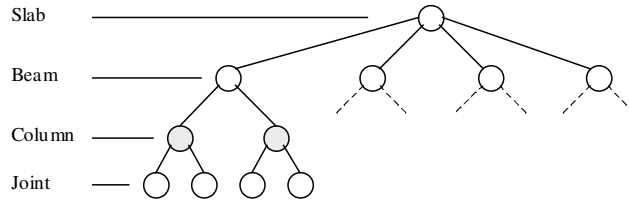


Figure 7: Slab/ column/ beam binary tree

Programming: On the programming of computers by means of natural selection, Cambridge MA: MIT Press, ISBN 0-262-11170-5, 1992.

[2] Holland J.H, *Adaptation in natural and artificial systems*, Ann Arbor: The University of Michigan Press, 1975.

[3] Banzhaf W et al, *Genetic Programming- An introduction (On the automatic evolution of computer programs and its applications)*, Morgan Kaufmann Publishers, ISBN 1-55860-510-X, 1998.

[4] Montana D.J, "Strongly typed genetic programming", *Evolutionary computation*, 3(2), 1995, pp199-230.

[5] Radcliffe N.J and Surry P.D, "Formal memetic algorithms", *Lecture Notes in Computer Science 865*, 1994.

[6] Ashour A.F et al, "Empirical modelling of shear strength of RC deep beams by genetic programming", *Computers and Structures*, Pergamon, 81 (2003), pp331-338.

[7] Hong YS and Bhamidimarri R, "Evolutionary self-organising modelling of a municipal wastewater treatment plant", *Water Research*, 37(2003), pp1199-1212.

[8] Roberts S.C. and Howard D, "Detection of incidents on motorways in low flow high speed conditions by genetic programming", *Cagnoni S et al (eds): EvoWorkshops 2002, LNCS 2279*, Springer-Verlag, 2002, pp245-254.

[9] Dorado J et al, "Prediction and modelling of the flow of a typical urban basin through genetic programming", *Cagnoni S et al (eds): EvoWorkshops 2002, LNCS 2279*, Springer-Verlag, 2002, pp190-201.

[10] Howard D and Roberts SC, "The prediction of journey times on motorways using genetic programming", *Cagnoni S et al (eds): EvoWorkshops 2002, LNCS 2279*, Springer-Verlag, 2002, pp210-221.

[11] Ishino Y and Jin Y, "Estimate design intent: a multiple genetic programming and multivariate analysis based approach", *Advanced Engineering Informatics*, 16(2002), pp107-125.

[12] Babovic V et al, "A data mining approach to modelling of water supply assets", *Urban Water*, 4(2002), pp401-414.

[13] Kojima F. et al, "Identification of crack profiles using genetic programming and fuzzy inference", *Journal of Materials Processing Technology*, Elsevier, 108 (2001), pp263-267.

[14] Whigham P.A. and Crapper P.F, "Modelling rainfall-runoff using genetic programming", *Mathematical and Computer Modelling*, 33(2001), pp707-721.

[15] Lee D.G et al, "Genetic programming model for long-term forecasting of electric power demand", *Electric power systems research*, Elsevier, 40, 1997, pp17-22.

[16] Montana D.J. and Czerwinski S, "Evolving control laws for a network of traffic signals", *Proceedings of the First Annual Conference: Genetic Programming, July 28-3, 1996. Stanford University*, pp333-338.

[17] Köppen M and Nickolay B, "Design of image exploring agent using genetic programming", *Proceedings of IIZUKA '96 Japan*, 1996, pp549-552.

[18] Yang Y. and Soh C.K, "Automated optimum design of structures using genetic programming", *Computers and Structures*, Pergamon, 80 (2002), pp1537-1546.

[19] <http://web.mit.edu/emech/dontindex-build/java/trussworks/index.html> [Accessed 20/10/03]

[20] Yang J and Soh C.K, "Structural optimization by genetic algorithms with tournament selection", *Journal of Computing in Civil Engineering*, July 1997, pp195-200.

[21] Diada J.M et al, "Visualizing tree structures in genetic programming", *Lecture Notes in Computer Science 2724*, 2003, pp1652-1664.

[22] <http://www.web3d.org/x3d.html> [Accessed 13/10/03]

[23] Wernert E.A and Hanson A.J, "Tethering and reattachment in collaborative virtual environments", *Proceedings of IEEE Virtual Reality 2000*, IEEE Computer Society Press, 2000, pp292.