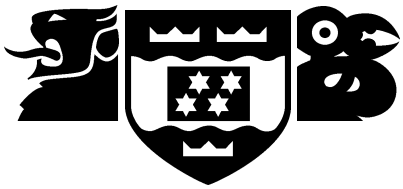
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@mcs.vuw.ac.nz

GENETIC PROGRAMMING FOR MULTI-CLASS OBJECT CLASSIFICATION PROJECT PROPOSAL

Yun ZHANG
300050446

Supervisor: Dr Mengjie ZHANG

March 22, 2004

Submitted in partial fulfilment of the requirements for
Bachelor of Science with Honours in Computer Science.

Abstract

This document gives a proposal for the project we are going to do. The main objective of this project is to investigate and refine techniques used to guide the genetic programming (GP) process, hence to extend the power of GP for multi-class (object) classification tasks, in terms of effectiveness, efficiency, and the comprehensibility of program classifiers, comparing with conventional GP approaches.

Contents

1 Abbreviations and Lingoes **2**

2 Background Introduction **2**
2.1 Classification . . . 2
2.2 Multi-class Classification . . . 2
2.3 Genetic Programming(GP) vs. Classification . . . 2
2.4 GP with respects to Multi-class Classification . . . 2
2.5 Necessity of The Project . . . 3

3 Objectives **3**

4 Detailed Specification **3**
4.1 GP-Patch-01: the ″Multi-Root Genetic Program (MRGP)″ patch . . . 3
4.2 GP-Patch-02: the ″Program Encapsulation (EP)″ patch . . . 4
4.3 GP-Patch-03: the ″Multi-genes (MG)″ patch . . . 4
4.4 GP-Patch-04: the ″Small Scope Competition with New-Blood operator (SSC-NB)″ patch 4
4.5 Evaluation Criteria . . . 6
4.5.1 Classification Performance Measure Criteria . . . 6
4.5.2 Patch Evaluation Criteria . . . 7

5 Project Timeline **8**

6 Resources Requirements **10**
6.1 Software . . . 10
6.2 Hardware . . . 10

Bibliography **11**

1 Abbreviations and Lingoes

- **GP:** Genetic Programming.
- **GP patch:** one of the research problems I am planning to consider in the course of the project. It can be thought as a patch of the GP system, which is supposed to be able to help to improve the performance of the GP system in some aspects.
- **GP population:** the set of programs the GP system will run evolutionary processes on.
- **Genetic program:** one instance/program sits in the GP population.
- **GP system:** the system which will run evolutionary processes on a set of programs.
- **NN:** Neural Network
- **OC:** Object Classification
- **OD:** Object Detection
- **SRS:** Static Range Selection.
- **STGP:** Strongly Typed Genetic Programming.
- **SVM:** Support Vector Machine

2 Background Introduction

2.1 Classification

Classification is a task of Data Mining [5] where a class value is to be sought from details of the given data, which can be a computer image for object classification tasks, a paragraph of text for information extraction tasks, or a section of acoustic signal for speech reorganization tasks. The classification problem has been considered as one of the most important and fundamental topics in the domain of artificial intelligence. It arises in a very wide range of intelligent applications, such as detecting faces from video images, diagnosing medical conditions from the output of medical tests, automatically adaptively filtering of junk emails, online indexing and searching, and so on.

2.2 Multi-class Classification

Multi-class classification is a subset of classification tasks, in which there are more than three classes of interest. Due to the nature of the technique being used on classification tasks, further difficulty is added as the number of class of interest increases.

2.3 Genetic Programming(GP) vs. Classification

Genetic Programming (GP) is a relatively young machine learning paradigm that was coined in 1990 by Hugo de Garis. It is a way of automatically constructing computer programs using a process analogous to biological evolution [10]. GP has been applied to a certain range of tasks including classification tasks such as simple image recognition in which the program must identify whether an object is present in an image or not [10]. The achievement shows that GP is a promising paradigm for classification tasks.

2.4 GP with respects to Multi-class Classification

Similarly to all other newly developed techniques, GP has a lot of shortcomings, or in another word, potential improvement areas, which are worthwhile to consider. The following is a list of some of them that are planning to be considered in this project:

1. A genetic program generated by the conventional GP system is only able to output a single floating point number. This becomes a fatal shortcoming when trying to apply it on complicated tasks such as MULTI-class classification or detection, which requires several class boundaries. Therefore, an automatically generated program (a genetic program) that is able to return several values, each corresponds to a class boundary, is highly desired. GP-Patch-01 described in section 4 will focus on coping with this problem.

2. Redundancy and unnecessary complexity have become a big problem that affects the quality and effect of the genetic program the GP system generated. GP-Patch-02 described in section 4 will focus on coping with this problem.

3. Due to the consequence of completely randomly choosing of exchanging points (more specifically the crossover and mutation points), the conventional GP system has a considerably high chance of breaking a partially correct subpart of a middle level program into useless subparts during its evolutionary processes comparing with the real species evolution. These useless subparts will be inherited by the children of this middle level program, hence the fitness of children will be lower than their parent(s). If this kind of unfortunate thing happens a lot, the Darwinian evolution theory and Natural selection, which are the hypothesis of genetic programming, will not hold any more. A visible consequence of it is the GP system will not converge. GP-Patch-03 and GP-Patch-04 will focus on coping with this problem.

2.5 Necessity of The Project

The necessity of this project is derived from the necessity of producing multi-class object classifiers with better performance, the necessity of investigating and refining the techniques used to guide the GP processes, and more specifically, the necessity of coping with the problems described above.

3 Objectives

The main objective of this project is to investigate and refine techniques used to guide the genetic programming (GP) process, hence to extend the power of GP for multi-class (object) classification tasks, in terms of effectiveness, efficiency, and the comprehensibility of program classifiers, comparing with conventional GP approaches.

The other objective of this project, time permitting, is to apply the new technique that will have been developed and shown to be effective on multi-class classification tasks, to multi-class detection tasks for further refinement and evaluation.

4 Detailed Specification

The objective of the project, namely to refine the genetic programming processes for better multi-class object classification effect, is planning to be achieved by introducing 4 GP-patches, each corresponds to one of the research questions, into an already-implemented conventional GP system. The patches will be listed below.

4.1 GP-Patch-01: the ”Multi-Root Genetic Program (MRGP)” patch

As the output of a standard genetic program will typically be a single floating point number, object classification usually turns this number into a class label by using the “Static Range Selection (SRS)” technique[5]. The SRS uses a pre-defined static set of ranges, each corresponds to a class of interest. The class an instance will be classified as is then depending on the range the returned floating point number sits in. One of the several limitations of SRS is that the performance of the system

depends too much upon the order that the classes are defined. Another one is that this method converges very slowly on difficult problems, or those with many classes [5]. Because of these, to switch from the standard GP approach to a stronger mechanism, which is supposed to be able to give a more complicated-structured output that will be more sensible on multi-class object classification, is worthwhile to consider.

The difficulty of this patch lies on to determine the structure of the GP program that can return more than one values. A multi-root tree structure (a single-direction graph with multiple starting points, each of which will give an output), a set of tree-structured programs, and decision lists are in consideration. Another issue needs to take care of is the clause on the involving functions, namely the domain of the parameters the functions can take, and the range of the value(s) the functions can return.

According to our current plan, this patch will be allowed not to be a completely application-domain-independent one purely for the refinement of genetic programming processes, but a one that can specifically focus on extending the power of GP just for determining the classification strategies (namely the class boundaries) of multi-class classification tasks.

4.2 GP-Patch-02: the "Program Encapsulation (EP)" patch

This patch will aim to simplify the programs during the evolutionary process. A GP evolutionary process will inevitably introduce redundancy and unnecessary complexity into evolved programs. Typically, the programs are simplified at the end of the evolutionary process, but not during it [10]. The redundancies and complexities have the undesirable consequences that the programs are very difficult to understand and interpret, and the search process is very inefficient because it explores unnecessarily complex parts of the search space [10]. This patch is just for coping with this kind of problems. It will examine the structure of the genetic programs during the evolutionary processes (may not be for every generation), and, if necessary, to remove certain types of redundancy of the programs, and/or to introduce (a) new structure(s) to replace a bunch of complicated ones. Hopefully it can improve the performance of the genetic operators that are applied to the program, or produce a population of more concise programs. This patch can be considered together with the usage of vectors or decision lists instead of the tree structure for structuring the program (i.e. GP-Patch-01).

4.3 GP-Patch-03: the "Multi-genes (MG)" patch

Do the Darwinian evolution theory and Natural selection fit for the environment (consider the properties of the population set) that genetic programming is being applied? The point is: one of the main(the biggest one of my consideration) disadvantages of GP comparing with human evolution is, during the evolutionary process, GP evolution has a much bigger chance of breaking a good genome (a partially correct sub-tree) into useless parts. This is due to the randomly-choosing-separation-point approach when crossover (or mutating). One of our thinking to cope with this problem is to introduce information-cell/or gene/or genome into the structure of the program. Namely, each program may consist of several subparts, where each part may represent a part of, or exactly one, or several functionality-cells. During the evolutionary process, if a number of such sub-parts (let's call them genes) are considered as "acceptable" for achieving part of the entire task, just merging them into one. According to this, when performing crossover and/or mutation, we then be able to choose the separation point based on the boundary of the genes. The idea is quite clear, but becomes technically hard when really trying to implement it. So this patch will be the very last one we are going to try to implement.

4.4 GP-Patch-04: the "Small Scope Competition with New-Blood operator (SSC-NB)" patch

Another way of dealing with the third GP shortcoming I described in section 2 is to make the newly generated instance competes in a small scope with its relations or even its parent(s) before it can go

into the next generation. Brood Recombination Crossover (BRC), a crossover operator that allows producing more than 2 children for each pair of parents, and only selecting the best two siblings into the next generation, is exactly that kind of idea. This patch will aim to extend this sort of idea into the mutation operator by introducing a new genetic operator: new-blood. Another point of involving this operator is, do not waste time and resources on expecting two horrible parents to give good children, or expecting an extremely bad guy mutates into a good one. The algorithm/pseudo code is as the following:

```
INPUT - randomly generated programs as input.
OUTPUT - output of a generation.
G - population size (fixed);
r% -- predefined rate of reproduction (can vary);
c% -- predefined rate of crossover (can vary);
m% -- predefined rate of mutation (can vary);
b% -- predefined rate of new-blood (can vary);

Properties must hold:
r + c + m = 100;
b = m by initial;
b =< m forever;

    while(!meet the terminating criteria){

        for(all programs in the population){

            evaluate the fitness;
            store them into a fitness-linked-list IN FITNESS DECREASING ORDER;
            // it is like to do an insertion sort
        }

        R = G*r%;
        C = G*c%;
        M = G*m%;
        B = G*b%;

        // the following 3 variables are for future adjusting of r%,c%,and b%
        float RMeasure;
        float CMeasure;
        float MBMeasure;

        for (the first R many programs){
             RMeasure = the fitness of the Rth program.
             do reproduction on them.
             store them in OUTPUT.
       }
        for (programs indexed from R+1 to R+C){
             CMeasure = the fitness of the R+Cth program.
             do c on them.
             store them in OUTPUT.
       }

        // something to do with the new operator==============
        for (programs indexed from R+C+1 to R+C+M){
             MBMeasure = the fitness of the R+C+Mth program.
             do mutation on them;
             store(insertion-sort) them into TEMP(an linked-list of size M+B).
```

```
     }
     for (int I = 0; I < B; I++){
          randomly generate a new program;
          evaluate the fitness of this program;
          store(insertion-sort) it into TEMP.
     }
     drop the programs in TEMP indexed from M+1 to B.
     store TEMP into OUTPUT.
     // ====================================

     analysis RMeasure and CMeasure, hence adjust r% and c%;
     analysis MBMeasure, hence adjust b%; //note b<m holds forever.

     INPUT = OUTPUT;
     OUTPUT = null;
}
```

NOTE01: "b" (the figure denotes for the number of new programs to generate in each generation), should be decreasing in a certain speed (probably exponentially). The reason is, as the number of generation gets larger, the fitness-level of the overall population set will be much better than average (randomly generated set).

NOTE02: I tried to prove that introducing new-blood operator will have a better performance, but I failed. The conclusion is: If one can prove that the randomly-separated-mutation on badly-fitted-instances will be better than a completely newly randomly generated set of instances of the same size, then my new operator fails. Otherwise, (or even by common-sense-view), the operator must be useful. And because the percentages are all variables, at the very least the new operator won't make things worse. View my messy proof at `http://www.mcs.vuw.ac.nz/~yz/cgi-bin/wiki?PrOOf1`

4.5 Evaluation Criteria

4.5.1 Classification Performance Measure Criteria

All evaluations will be done on three or four data sets of increasing difficulty. The evaluation result will be compared with the evaluation result of the same test using the same dataset on the conventional GP system I will start with.

Classification Accuracy Measurement

TPF, FPF, and Standard ROC Curves Measurement-01
ROC-related measurements have been widely and successfully applied on measuring the classification accuracy of binary-class tasks. This measurement will use exactly the method that is using on binary-class classification, by evaluating on each of the class with respect to all objects that are classified as not belong to this class.

TPF, FPF, and Standard ROC Curves Measurement-02
As the evaluation output is for human analysis but not for computer read, we do want a straightforward view of the measurement result. If we do want to measure the classification accuracy of each class with respect to every other class, the following way may be able to produce a reasonable straightforward view of what we want: Roughly consider each ROC curve as a hyperbola, and use the positive gradient of the asymptote of that hyperbola as a record, let's call it ROC figure. Finally we construct a 2D n*n square-array (n is the number of classes of the multi-classification), then we'll end up with a 2D array which will record the information of the ROC curve for every pair. Print the 2D array on screen, then we will be able to focus on the extremely out of control values hence debug our system.

4.5.2 Patch Evaluation Criteria

Patch-Evaluation-Criteria-01
To evaluate each of the GP-patches I'll have implemented by evaluating the performance of applying each of the partially-refined-GPs (the conventional GP along with just one GP-patch we want to evaluate) on multi-class object classification problem.

Patch-Evaluation- Criteria-02
To evaluate the entirely-refined-GP (the conventional GP along with all GP-patches I'll have implemented and evaluated as individually-useful) on multi-class object classification with low-level domain-independent pixel statistics. This will be the figure to show the achievement of this project.

Patch-Evaluation- Criteria-03
Evaluate the entirely-refined-GP on typically object classification tasks such as human face classification, satellite image classification, and so on, with high-level handcrafted domain-dependent features introduced.

Patch-Evaluation- Criteria-04
Evaluate the entirely-refined-GP on object detection tasks.

5 Project Timeline

Week	Tasks
SEMISTER1-WEEK01 SEMISTER1-WEEK02 SEMISTER1-WEEK03	Fix project topic Produce project proposal
SEMISTER1-WEEK04 SEMISTER1-WEEK05	Do more background reading hence produce bibliography and the "background and literature review" Getting familiar with the GP package I'm going to use Getting familiar with image programming
SEMISTER1-WEEK06 SEMISTER1-BREAKWEEK01 SEMISTER1-BREAKWEEK02 SEMISTER1-WEEK07 SEMISTER1-WEEK08 SEMISTER1-WEEK09	Implement GP-Patch-01 and experiment with it. Getting more familiar with the GP package Evaluate GP-Patch-01 by Patch-Evaluation-Criteria-01.
SEMISTER1-WEEK10 SEMISTER1-WEEK11 SEMISTER1-WEEK12	Implement GP-Patch-02 and experiment with it. Evaluate GP-Patch-02 by Patch-Evaluation-Criteria-01.
MIDYEAR BREAKWEEK01 MIDYEAR BREAKWEEK02 SEMISTER2-WEEK01 SEMISTER2-WEEK02	Implement GP-Patch-04 and experiment with it. Evaluate GP-Patch-04 by Patch-Evaluation-Criteria-01. Produce Progress Report, which is due on Semister2-Week01
SEMISTER2-WEEK03 SEMISTER2-WEEK04 SEMISTER2-WEEK05 SEMISTER2-WEEK06 SEMISTER2-BREAKWEEK01 SEMISTER2-BREAKWEEK02	Implement GP-Patch-03 and experiment with it. Evaluate GP-Patch-03 by Patch-Evaluation-Criteria-01. Produce the first draft report.
SEMISTER2-WEEK07 SEMISTER2-WEEK08 SEMISTER2-WEEK09	Do Patch-Evaluation-02 Do Patch-Evaluation-03 and Patch-Evaluation-04 if time permitting Produce the second draft report.
SEMISTER2-WEEK10 SEMISTER2-WEEK11 SEMISTER2-WEEK12	Produce Final Report

T1-WEEK01
T1-WEEK02
T1-WEEK03
T1-WEEK04
T1-WEEK05
T1-WEEK06
BREAKWEEK-1
BREAKWEEK-2
T1-WEEK07
T1-WEEK08
T1-WEEK09
T1-WEEK10
T1-WEEK11
T1-WEEK12
BREAKWEEK-1
BREAKWEEK-2
T2-WEEK01
T2-WEEK02
T2-WEEK03
T2-WEEK04
T2-WEEK05
T2-WEEK06
BREAKWEEK-1
BREAKWEEK-2
T2-WEEK07
T2-WEEK08
T2-WEEK09
T2-WEEK10
T2-WEEK11
T2-WEEK12
STUDYWEEK-1
Produce proposal
Background reading
Produce bibliography and literature review
Familiar with starting GP software
GP-Patch-01
GP-Patch-02
GP-Patch-04
GP-Patch-03
Produce first draft
Patch-Evaluation-02
Produce second draft
Patch-Evaluation-03
Patch-Evaluation-04
Produce final report

6 Resources Requirements

6.1 Software

I'll use an existing Genetic Programming System as the starting point. It seems like Meng will only show me what the system looks like after I hand in the project proposal.

6.2 Hardware

- Computer devon [IP: 130.195.5.109] in Graceland.
- Various other MCS hardware resources.
- Probably a digital camera for taking dataset photos.

Bibliography

[1] BHOWAN, U. A domain independent approach to multi-class object detection using genetic programming. Master's thesis, Victoria University of Wellington, New Zealand, 2003.

[2] BHOWAN, U., AND ZHANG, M. Program size and pixel statistics in genetic programming for object detection.

[3] GUYON, I., AND ELISSEEFF, A. An introduction to variable and feature selection. *Journal of Machine Learning Research 3* (2003).

[4] KOZA, J. R. *Genetic Programming: on the programming of computers by means of natural selection*. MIT Press, 1993.

[5] SMART, W. Genetic programming for multi-class object classification. Master's thesis, Victoria University of Wellington, New Zealand, 2003.

[6] SMART, W., AND ZHANG, M. Genetic programming with gradient descent search for multiclass object classification.

[7] SMART, W., AND ZHANG, M. Multiclass object classification using genetic programming.

[8] XIE, H., ANDREAE, P., ZHANG, M., WARREN, P., AND OTHERS. Stress detection using decision trees and support vector machines. *Proceedings of the Australasian Workshop on Data Mining and Web Intelligence* (2004).

[9] ZHANG, M. *A Domain Independent Approach to 2D Object Detection based on the Neural and Genetic Paradigms*. PhD thesis, RMIT University, Melbourne Australia, 2000.

[10] ZHANG, M. Abstract of research proposal. *Marsden Fund Document 1c page 3* (2003).

[11] ZHANG, M., AND CIESIELSKI, V. Neural networks and genetic algorithms for domain independent multiclass object detection. *International Journal of Computational Intelligence and Applications* (November 2003).

[12] ZHANG, M., CIESIELSKI, V., AND ANDREAE, P. A domain independent window-approach to multiclass object detection using genetic programming.