Effects of Combining ALPS and SPEA2 Genetic Programming algorithms in a Portfolio Optimization Problem

Name: Khalid A. Al-Abid

Supervisor: Chris Clack

Year of Submission: 2007

Programme: MSc IS

Disclaimer: This report is submitted as part requirement for the MSc Degree in Computer Science at University College London. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.
Abstract

This Report presents the implementation of two novel techniques one is Aged-Layer Structure Population (ALPS) by Gregory S. Hornby [1] and the other is Strength Pareto Evolutionary Algorithm (SPEA2) by Eckart Zitzler [2] in a multiobjective GP-based system to help fund managers optimize financial portfolios. In the context of investors, ALPS can reduce the problem of premature convergence in evolutionary algorithms and it also can preserve the diversity in the population which can result in fitter solution(s). In the same context, SPEA2 enables multiobjective fitness evolution and can enable investors to choose a risk profile they are interested in. We have tested both techniques on historical stock data of 77 FTSE100 companies (supplied by Reuters 3000 Xtra), and a set of 22 commonly used technical and fundamental analysis factors using an investment simulator for a period of 80 months. The project investigates whether the two techniques can be combined (or whether they interfere with each other operationally). Results show that the GP-generated trading formulas consistently produce excess trading profits when both techniques were combined compared to using the SPEA2 algorithm alone.
# Table of Contents

1. Introduction  
   1.1. Project Objectives  
      1.1.1. Technical Objectives  
      1.1.2. Experimental Objectives  
   1.2. Contribution of the report  
   1.3. Report Structure  
2. Background  
   2.1. Finance  
      2.1.1. Technical Analysis  
      2.1.2. Portfolio Optimisation  
   2.2. Genetic Computation  
      2.2.1. Genetic Algorithms  
      2.2.2. Genetic Programming  
   2.3. SPEA2 & Multiobjective Optimisation  
   2.4. The ALPS Algorithm  
      2.4.1. The ALPS Paradigm  
      2.4.2. The ALPS Rules  
   2.5. Related Work  
      2.5.1. Hierarchical Fair Competition (HFC) Algorithm  
      2.5.2. Adaptive Hierarchical Fair Competition (AHFC) Algorithm  
3. Design  
   3.1. Objectives  
      3.1.1. Build a multiobjective GP system with both ALPS & SPEA2  
      3.1.2. Compare ROI when using the system with and without ALPS  
      3.1.3. Provide an easy-to-use interface for different types of investors  
      3.1.4. Live Stock Data  
   3.2. System Overview  
      3.2.1. General Structure  
         3.2.1.1. GP System Design  
         3.2.1.2. Investment Simulator Design  
   3.3. Environment Design  
      3.3.1. Description of a standard GP Cycle  
      3.3.2. Investment Simulator Cycle  
      3.3.3. Other Operations  
      3.3.4. Limitations in Design  
4. Implementation  
   4.1. Development Environment  
      4.1.1. ECJ Package  
      4.1.2. Eclipse 3.2 Package  
      4.1.3. Java 2 Standard Edition JDK 6.0  
      4.1.4. MySQL 6.0  
   4.2. Source of Historical Data  
      4.2.1. Training Data  
      4.2.2. Testing Data  

- 3 -
4.3. General Implementation Issues
  4.3.1. Classes Structure
    4.3.1.1. ECJ Classes
    4.3.1.2. ALPS Modifications
    4.3.1.3. SPEA2 Classes
  4.3.2. Investment Simulator

5. Experiments
  5.1. Experiment One
    5.1.1. Aims
    5.1.2. Experimental Design
    5.1.3. Results and Analysis
  5.2. Experiment Two
  5.3. Experiment Three
  5.4. Experiment Four
  5.5. Experiment Five
  5.6. Experiment Six
  5.7. Experiment Seven
  5.8. Experiment Eight

6. Conclusions
  6.1. Summary
  6.2. Future Work

7. References

8. APPENDICES
  Appendix A
  Appendix B
  Appendix C
  Appendix D
  Appendix E
  Appendix F
  Appendix G
  Appendix H
  Appendix I
1. Introduction

Standard Genetic Programming (GP) systems have common problems in their performance during training and validation stages such as premature convergence and lack of diversity in the population. In this project we are going to focus our experiments on the second problem (lack of diversity). In addition, we are going to implement a novel technique by Horney called ALPS in a real-world multiobjective financial portfolio optimisation problem to generate fitter non-linear multi-factor models. Then, we are going to assess these models in terms of the assessment factors (the Return on Investment ROI, the Standard Deviation, and the Sharpe Ratio) using an investment simulator (built as part of a graduation project in the Computer Science department). Finally, we are going to present these models to the investor with an option to choose the type of investments that matches their needs.

1.1 Project Objectives

The project objectives comprise both technical and experimental objectives which can be summarized as follows:

Technical Objectives

1. Install the ECJ (A Java-based Evolutionary Computation System).
2. Configure the SPEA2 option in ECJ to enable the multiobjective environment.
3. Implement the ALPS technique in a multiobjective GP system (SPEA2) that evolves non-linear multi-factor models for financial portfolio optimisation problem for FTSE100 stocks.
4. Configure our GP system (ECJ package) to work with the Investment Simulator.
5. Run five experiments of same configurations, three with different configurations which all will be described in the experiments chapter.

Experimental Objectives

1. Setup a multiobjective GP system for optimising a financial portfolio and its trading rules which will be detailed in section (3.2.1.2).
2. Evaluate the efficiency of SPEA2 technique when applied alone to our portfolio optimisation problem.
3. Evaluate the efficiency of combining ALPS and SPEA2 techniques to our portfolio optimisation problem.
4. Obtain results from the experiments and analyze these results.

1.2 Contribution of the report

To the best of our knowledge, we claim our research the first attempt to build a GP system that combines the ALPS and SPEA algorithms\(^1\). In addition, this project provides a detailed assessment of the combined system including statistical analysis of SPEA2 with and without ALPS. Therefore, we hope when we apply our system to a real-world financial portfolio optimisation problem it will help fund managers and market investors utilize a robust model that can:

- help investors make buy-and-sell decisions for FTSE100 stocks effectively,
- help investors increase the return on investment of their portfolios, and
- enable investors to choose a risk profile (high, medium, and low) that matches their needs.

\(^1\) Patel and Clack 2007 [10] investigate a single-objective implementation of ALPS ad GP for portfolio optimisation.
1.3 Report Structure

The rest of this report begins with a brief background chapter on Genetic Computation techniques such as GA and GP. Details of the SPEA2 algorithm as an example of multiobjective optimisation algorithms are included, together with a description of the ALPS algorithm including its paradigm and rules and a review of related work on the attempts in reducing the problem of premature convergence and increasing diversity in standard GP systems (this includes comparisons between the ALPS and other GP techniques). Details of the design and implementation of our GP system for certain design objectives are included in Chapter 3, along with a discussion of various design options and limitations, a GP system overview and a brief discussion of the investment simulator used and the design of the environment. Chapter 4 gives more details on the system implementation and development environment. Chapter 5 states the goals of the experimental work (eight experiments) and describes the experimental procedures for achieving these goals. Finally our concluding remarks and suggestions for further work make up Chapter 7.
2. Background

2.1 Finance

2.1.1 Technical Analysis

Technical Analysis is one of the methodologies used by investment managers to analyze stocks for buying and selling stocks. It is also widely used by financial market traders for predicting future market trends based on past performance. Many studies investigated the use of Genetic Programming (GP) techniques to learn profitable trading rules from combinations of technical indicator functions [5, 6, 7, 9]. The challenge in this project is to use a multiobjective algorithm such as SPEA2 which provides us with a range of solutions for different risk profiles. Although ALPS has succeed in solving the premature convergence and lack of diversity problems in Single-Objective GP environments, we seek, in this project, to reduce premature convergence and increase diversity in SPEA2 in order to generate better solutions and investigate whether ALPS will work effectively with SPEA2 to do this.

2.1.2 Portfolio Optimisation

In finance, portfolio is a collection of stocks, bonds, assets, or cash held by companies or individuals. If an investor’s portfolio is well selected, it will provide a high return on investment (ROI) with a minimum exposure to risk. However, this type of problem is also known as the portfolio optimisation or portfolio selection problem in which we have a fixed amount of money to invest in securities (chosen for example from FTSE100 stock market) for a given holding period. We also need to allocate a percentage of the investment money for each security in the portfolio. In order to solve this type of
problem, we need to identify the securities included in the portfolio and the associated percentages of each security in the portfolio.

2.2 Genetic Computation

In this section, we will give a brief introduction to Genetic Algorithms and Genetic Programming.

2.2.1 Genetic Algorithms

Genetic Algorithms (GA) was originally developed by John Holland in 1975 as one of the widely used algorithms in Evolutionary Computing. Figure 1 depicts a standard Genetic Algorithm Model. It runs on a population of randomly generated individuals represented by character strings. These individuals are then evaluated according to a given fitness function appropriate to the problem. Pairs of individuals, selected randomly according to their fitness, are then recombined to create new members in the population. Starting from an initial population of randomly generated candidate solutions, consecutive generations are then produced until some termination criterion is reached such as a limit on the number of generations or reaching an optimal solution(s).

![Figure 1: A standard Genetic Algorithm Model](image-url)
2.2.2 Genetic Programming

Genetic Programming is an extension to Genetic Algorithms which has a superior degree of data handling due to its tree representations compared to strings representation in Genetic Algorithms. This easily enables the genetic operators such as mutation and crossover to be applied to the population towards building complex tree structures that meet the problem in hand. Basically, GP evolves a population of individuals by repeating an evolution process over a fixed number of generations or until a termination criterion is met. In our project, Individuals are non-linear multi-factor models in a tree representation used for making investment decisions. A fitness function is needed to assess the individual’s performance in the population throughout the experiment. Our fitness function assigns a fitness value to each individual in the population based on its performance in the Investment Simulator. Based on the fitness value the individual may be removed, mutated, used for crossover to enhance the population performance, or could remain in the population as in the elitism.

The main steps of a standard GP algorithm is the same as of GA algorithm except that the representation of individuals in GA is bit strings while in GP, individuals are represented as binary trees. Although the concept of genetic operations in GA and GP is the same, GA crossover, for example, is a crossover between two bit strings while GP crossover is between two binary trees which could be more complex than the GA’s.

2.3 SPEA2 & Multiobjective Optimisation

In real-world financial problems, we may need to evaluate solutions against two or more conflicting objectives independently. This is known as multiobjective optimisation problem in which we have a set of solutions capable of optimising all the objectives in
the problem to find the best solutions. However, a single solution that can optimize all the objectives simultaneously may not exist. Hence, this type of problem uses algorithms that are different than those used for single objective optimisation problems.

In this section, we going to introduce one of the well-known algorithms used in solving multiobjective optimisation problems which is known as Strength Pareto Evolutionary Algorithm (SPEA2) by Zitzler [2]. The main idea of this algorithm is to use an external archive (buffer) for saving the nondominated (best) solutions found as a way of using elitism in the evolutionary algorithm. This leads us to take a look at the main loop of SPEA2 algorithm:

**Input:**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Population size</td>
</tr>
<tr>
<td>R</td>
<td>Archive size</td>
</tr>
<tr>
<td>T</td>
<td>Maximum number of generations</td>
</tr>
</tbody>
</table>

**Output:**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Nondominated set</td>
</tr>
</tbody>
</table>

1. **Initialization**: Generate an initial population and create an empty archive with a maximum predefined size.

2. **Fitness assignment**: Calculate the fitness values for all individuals in the population. Each individual in both the archive and the population is assigned a strength value representing the number of solutions it dominates*.

3. **Environmental Selection**: Copy all the nondominated individuals in the population to the archive, which are then tested for duplicates or dominated

---

*“Dominate”: x dominates y when x is as good as y regarding each objective, and there is at least one objective with respect to which x is better than y.
individuals, if so, they will be deleted. If the number of nondominated individuals is less than the predefined size, the remainder of the archive is filled with the fittest dominated solutions from the population. If the number of nondominated individuals exceeds the predefined size limit $R_\text{a}$, the extra individuals will be removed (truncated). Those individuals chosen for removal are the ones that have the minimum distance* to another individual.

4. **Termination:** If the current number of generations $\geq T$ or any other termination criterion, set $A$ to have those individuals in the archive. Stop.

5. **Mating Selection:** perform a binary tournament selection with replacement on the nondominated individuals in the archive only in order to fill the mating pool.

6. **Variation:** Apply crossover and mutation operators to the mating pool and replace the initial population with the current one and go to step 2.

Furthermore, there some environmental characteristics of the SPEA2 algorithm which are highlighted below:

1. **Strength Value:** A value associated with each individual in the archive which counts the number of individuals in the population dominated by (worse than) or equal to that individual, divided by the population size plus 1. It ranges between $[0, 1]$.

2. **The Fitness:** A function defined for every individual in the population as the sum of the strength values of all archive individuals that dominate (better than) or equal to that individual, plus 1. It also ranges between $[0, 1]$.

* Distance measures the Euclidian distance ($\sigma$) between two candidate solutions in the objectives search space. Therefore, we need to select the closest solution to an individual to be deleted.
There are four types of fitness values that SPEA2 can calculate:

a. Raw Fitness: is defined as the sum of the strengths of the individual dominators in both the archive and the population. Usually, raw fitness is to be minimized, where the raw fitness value for the nondominated individuals is 0.

b. Density Information: is calculated as the inverse of the distance to the K-nearest data points. K is usually taken to be the square root of the sample size (population plus archive), and the distance is in the interval [0, 1]. Below, we will illustrate the technique used to calculate the distance to the K-Nearest Neighbour:

![Figure 2: Shows the K-Nearest Neighbour technique used to calculate the Euclidian distance to in the population and archive space. This technique is used in the archive truncation.](image)

Let us assume we have three points A, B, and C in the population and the archive representing candidate solutions. Therefore, the K-Nearest Neighbour algorithm calculates the Euclidian distances between the three points to X to identify the closest point to X (in this diagram is C). Then, the algorithm will delete the solution C from the archive.
c. Total Fitness: the total fitness of an individual is defined as the sum of its raw fitness plus density information.

3. The Complexity of the algorithm: Let us assume we have $m$ objectives and a random population of size $N$, then, we can define the complexity of the algorithm to be $O(mN^2)$ [3].

Next, we will demonstrate an example of a two-objective problem and explain how the SPEA2 calculates the Pareto-optimal front in the diagram below:

![Diagram](image-3.png)

Figure 3: Pareto-optimal Front for a 2-objective optimisation problem.

The two axes of the objectives form the objective space, which represent a two-dimensional optimisation problem. The positions of solutions on the Pareto-optimal front are shown in blue, connected by dashed pink. Each Pareto-optimal solution dominates all solutions to its lower left; for example, point A dominates points C, B, D, and E, as bounded by the dashed black lines. Non-dominated solutions are shown as turquoise circles; dominated solutions are shown as olive circles. Strength is shown adjacent to each solution. Raw fitness is shown in parentheses whereas Diamonds indicate positions
in objective space of potential solutions not yet discovered by the algorithm, but that would be non-dominated once discovered [8].

2.4 The ALPS Algorithm

A common problem of evolutionary computation (EC) algorithms is that after a while the fitness of the best solution levels off at some value and no further improvements are made towards the end. This is known as the premature convergence problem which is due to the fact that the variation operators (crossover and mutation) can not produce new individuals that will move up the population into a better basin of attraction.

In this section, we introduce a novel technique of maintaining genotypic diversity of the population called the Aged-Layered Population Structure (ALPS) algorithm. The ALPS was created by Hornby [1] as a new method to reduce the problem of premature convergence in EC algorithms. In fact ALPS uses a novel measure for an individual’s age, based on the amount of time its genotypic material has been evolving in the population, and it splits the population into multiple age layers each of which contains individuals whose age lies in a particular range. Breeding is therefore restricted to individuals in adjacent age layers (the same age layer or below). By segregating individuals, ALPS hopes to give new individuals time to find a basin of attraction to learn and improve in before being pushed into competition with individuals that have been in the evolutionary process much longer. The ALPS also aims to continuously introduce new genetic material into the population by randomly introducing new individuals into the bottom age layer (layer 0) at specific points (usually called age gaps) in the evolution process.
2.4.1 The ALPS Paradigm.

In this section, we will look closely at the ALPS paradigm and describe its main components:

1. Age: The ALPS technique introduces a novel way of measuring an individual’s age and segregating individuals into age-layers. The age is measured by the number of generations an individual’s genotypic material has been evolving rather than the individual itself. This method of measuring age is in contrast to many other age-based systems, which we will discuss in next section, as new individuals created by mutation or recombination take the age of their eldest parent plus 1 rather than starting at 0. However, new randomly generated individuals start with an age of 0 as their genotypic material has not been through the evolution process yet. An individual’s age is incremented if it is used as a parent in breeding and is copied to the next generation through elitism. The age of an individual is only incremented by 1 each generation irrespective of how many children it creates in that generation. If the individual is not used as a parent its age will remain unchanged.

2. Age Gaps: Are regular intervals (milestones) used in the ALPS algorithm in which all individuals in the lower age layer (layer 0) are replaced with randomly generated ones (and initializing their ages to 0).

3. Age Layers: Individuals are held within age layers which are made up by a given aging scheme and age layer based on the age gap. The age layers restrict competition and breeding to adjacent layers only (the current age layer and the one below) and ensure younger individuals have a number of generations to move
into a good basin of attraction before they are pushed into the next layer. New layers are populated by those individuals that have reached the previous layers age limit. This will usually be the offspring of individuals from the previous layer.

4. Age Limits: Specify the maximum ages for individuals to be allowed in an age layer before moving up to a higher layer. This component is based the Aging Scheme which comes next.

5. Aging Scheme: Used to setup the age limits for each layer in the GP system. There are four schemes: Linear, Fibonacci, Polynomial (n^2), and Exponential (2^n).

Table 1 below shows an example of aging scheme for age-gap 1.

<table>
<thead>
<tr>
<th>Aging Scheme</th>
<th>Max age in layer (n)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Linear</td>
<td>1</td>
</tr>
<tr>
<td>Fibonacci</td>
<td>1</td>
</tr>
<tr>
<td>Polynomial (n^2)</td>
<td>1</td>
</tr>
<tr>
<td>Exponential (2^n)</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1: In this table, we show an example of different aging schemes along with their corresponding age limits in each layer for layers 0 to 6 for an age gap of 1.

6. Migrating to high layers: An individual is to be migrated (moved up) to the next higher layer only if its age reaches the maximum age limit in its current layer. In other words, successive movement occurs only if an individual is now too old for its layer and its fitness is better than an individual in the upper layer. Therefore, an individual is only guaranteed to stay in the population forever if it is at the
global optima; otherwise it will eventually be replaced as better individuals evolve.

Finally, we are going to show how the ALPS will interact with and work alongside SPEA2.

![Diagram](image_url)

**Figure 4:** The interaction between ALPS and SPEA2 in the multiobjective GP system.

Finally, if the maximum number of age layers is reached then all individuals of age higher than that limit are grouped in a layer which has no age limit. In the experiments chapter, we will examine the effects of small and large archive size on the ALPS performance.
2.4.2 The ALPS Rules

The ALPS imposes many restrictions around the evolution process to ensure full control over the breeding process, maintaining diversity, and reducing the problem of premature convergence. These restrictions can be summarized as follows:

1. Breeding: Individuals can only compete and breed with individuals within its own layer or the layer below. The offspring’s age is set to the elder parent plus one.

2. Age gap initialization: At each age gap generation, all individuals in the bottom age layer (layer 0) are replaced with randomly generated individuals and resetting their ages to 0. This rule preserves the diversity in the population at regular intervals (age gaps).

3. Age layer extension: A new age layer is created only when the current generation is equal to the age limit for the previous layer. For example with an age gap of 5 a new age layer will be created at generation 0, layer 2 will be created at generation 10, layer 3 at generation 15, layer 4 at generation 20, etc.

2.5 Related Work

In this section, we introduce two algorithms used to reduce the problem of premature convergence and help practitioners of evolutionary computation (EC) maintain diversity in standard GP systems and compare them to the ALPS.

2.5.1 Hierarchical Fair Competition (HFC) Algorithm

The Hierarchical Fair Competition (HFC) model is quite similar to ALPS. We can summarize it as follows:

- Has a multi-subpopulations structure.
- Splits the population into a hierarchy of fitness layers.
• Uses fitness rather than age to segregate individuals into layers.
• Introduces randomly generated individuals into the bottom layer (layer 0).

Observations:
We have found some interesting points in this algorithm which can be summarized as follows:
• Although HFC protects new individuals and their offsprings from the competition with the high-fitness individuals, there is still the problem that individuals which have converged to a local optimum near the top of a fitness layer prevent newer individuals in different basins of attraction from climbing through that fitness layer.
• For difficult problems, more levels are highly preferred. Therefore, HFC is well suited for difficult problems.
• In addition, the competition in HFC is allowed only among candidates with comparable capabilities.
• HFC avoids premature convergence by allowing emergence of new optima rather than trying to jump out of local optima.

2.5.2 Adaptive Hierarchical Fair Competition (AHFC) Algorithm
The Adaptive Hierarchical Fair Competition (HFC) model is also quite similar to ALPS.
We can summarize it as follows:
• Splits the population into layers.
• Uses fitness rather than age to segregate individuals into layers.
• Introduces randomly generated individuals into the bottom layer (layer 0) which increases the diversity in the population.
• No age limit is defined. Therefore, individuals in AHFC can climb up much higher than their peers in the ALPS.

Observations:

We are going to summarize our observations on this algorithm as follows:

• For simpler problems, AHFC can outperform the HFC.

• Both HFC and AHFC do much better than single population GP and standard multi-population GP with little additional computation effort.

• For more difficult problems, AHFC can approximate the static HFC performance, but is a little less effective.
3. Design

In this chapter we will give details on the main objectives of our design. We will then give an overview of our GP system, the Investment Simulator used, and other environmental design issues related to our project.

3.1 Objectives:

In section 1.1, we setup our project main objectives which involved installing the ECJ GP system as the working environment of our project. ECJ is developed and supported by the Evolutionary Computation Laboratory at George Mason University. ECJ is a Java-based EC capable of supporting wide range of GP and GA applications. ECJ is used here to control the evolutionary computation process in which we are going to incorporate the ALPS and SPEA2 implementations. In this section we focus on the main design objectives which can be summarized as follows:

3.1.1 Build a multiobjective GP system with both ALPS & SPEA2

In order to enable the multiobjective environment in the ECJ system we had first to configure the ECJ to turn on the SPEA2 multiobjective option. This includes enabling the SPEA2 (Subpopulation, Breeder, Evaluator, and Tournament Selection classes) in the parameter file. In order to implement the ALPS algorithm, the SPEA2 option had first to be enabled. Then the SPEA2 Tournament Selection class and other necessary classes had to be modified to apply the ALPS rules in the evolution process. Our ECJ system should evolve our non-linear multi-factor models (representing technical and fundamental analysis indicators) which are then evaluated by our fitness function using our Investment
Simulator which then ranks their fitness values according to our assessment factors we mentioned in the introduction chapter (page 1).

3.1.2 Compare ROI when using the system with and without ALPS

Our ultimate goal for the whole project is to compare the two systems (SPEA2 and ALPS system, and SPEA2 only system) in terms of the assessment factors generated by the Investment Simulator for both systems. This comparison should give an indication of the efficiency of these systems in terms of the assessment factors in general and the ROI in particular.

3.1.3 Provide an easy-to-use interface for different types of investors

For investors to take advantage of our powerful system, we developed a separate Excel sheet that shows summary statistics along with charts for the investors to choose the risk profile that matches their preference. This sheet also shows the best formulas that generated the highest ROI in the validation stage. Therefore, this easy-to-use interface is a comprehensive summary for investors to make effective buy-and-sell decisions.

3.1.4 Live stock data

Our Investment Simulator uses a stock universe of 77 FTSE100 stocks. Each of which contains 22 factors data over an investment period of 80 months (from 31.05.1999 to 31.12.2006) contained within .csv files. Since live stock data is essential for the success of our system as it will expose it to up-to-date data, this will validate the trained system on real data which can, hopefully, enable our system to be more robust. In this project, we are going to have the Reuter 3000 Xtra as our live source of data in which we can
receive up-to-date Excel sheets (.csv format) which include all FTSE100 securities of interest along with their associated 22 factors data columns for a given period.

3.2 System Overview

Here, we outline our GP system life cycle which is basically the standard GP life cycle with some modifications. These modifications are mainly in two areas:

A. Fitness Function: we extended it to have an interface between the GP system and the Investment Simulator in which the GP individuals are passed to the IS for evaluation and the assessments factors are then calculated by the IS and sent back to the GP system to complete the standard GP life cycle. The multi-fitness object in the GP individual gets updated to be ready for selection using the selection method described below.

B. ALPS Environment: we also extended the ECJ Individual class to include some necessary attributes (fields) to adopt the ALPS option such as Age, Age Layer, and many other features which will be listed later in the Implementation chapter.

3.2.1 General Structure

The following flowchart (figure 5) of our GP system walks us through the main components of the modified GP system along with the Investment Simulator system:
Figure 5: The overall system flowchart for both the GP system and the Investment Simulator.

3.2.1.1. GP System Design

Here we describe the GP system components mentioned in figure 3 in details:

1. Random Population:

   Our GP population consists of individuals represented as binary trees. These binary trees consist of two components: Terminals (operands) and Functions (operators), which are used to make up the structure of our GP individuals objects. Functions are used within the GP individual to be applied to the terminals. The functions used in our GP system are: Addition, Subtraction, Multiplication, Division, Power, and Logarithmic. Whereas, GP Individuals are the actual binary trees that use a random permutation of functions and terminals of random depths to build our assessment.
factors. These trees will be evaluated by the Investment Simulator to produce a single value. This single value is then applied on all stocks in our universe to rank these stocks according to their fitness. An example of a GP individual could be:

\[
(\frac{(-\text{CPS\_DPS} \times \text{Price\_Cash}) \times \text{MA\_30\_Day} \times \text{Price\_Mom}}{\text{MA\_30\_Day} \times \text{Price\_Mom}})
\]

which can be interpreted as the following regular expression:

\[(\text{CPS\_DPS} - \text{Price\_Cash}) / (\text{MA\_30\_Day} \times \text{Price\_Mom})\]

Note: that CPS\_DPS, Price\_Cash, MA\_30\_Day, and Price\_Mom are terminals whereas \(-, \times, /\) are functions.

2. Fitness Function:

In a multiobjective GP environment, we need a fitness function that can store more than one objective. In this project, we use SPEA2 fitness function to keep track of two objectives of interest: the ROI and the Risk (i.e Standard Deviation). This function uses an array of two elements to store the above objectives which are assigned to each GP individual by our GP system. SPEA2 uses two types of fitness which are defined below but we need first to define the term Strength Value as part of the SPEA2 components (discussed in section 4.3.1.3):

a. Strength Value: of an individual counts the number of individuals in the population this individual dominates.

b. Raw Fitness: the sum of the strength values of its dominators in both the population and the archive. The lower the raw fitness, the better, i.e. RawFitness(individual x) = 0 means this individual is a nondominated individual.
c. Adjusted Fitness: takes the Raw Fitness and normalizes it by applying the following formula: \( \text{Adj. Fitness} = \frac{1.0}{1.0 + \text{Raw Fitness}} \).

This scales the Adjusted Fitness in the range \([0, 1]\) where 0 is the worst fitness and 1 is the best.

3. Selection:

The ECJ system supports many selection methods such as tournament, fitness proportionate selection (Roulette) and many others. We have found that the SPEA2 is limited, so far, to Tournament selection method. So, we decided to implement and use this method in our GP system. Now we will describe how this method works:

Tournament Selection: in which a set of \( N \) individuals are selected randomly, \( N=7 \) by default, and compared with each other. The fittest individual is then selected to perform the genetic operation such as crossover or mutation. It has been found that for the same selection intensity, tournament selection has the smallest loss of diversity and the highest selection variance when compared with other selection methods [4].

4. Recombination:

In Genetic Programming, two parents (GP trees) are selected randomly to swap their genetic material at some random crossover point. The resulting offsprings are also sub trees contain a mixture of genetic material (terminals and functions) from both parents (see an example of single crossover in figure 6 below).
Figure 6: An example of a single point crossover between two GP individuals.

5. Replacement:

In this step, our GP system brings all the offsprings generated using the breeding step along with individuals brought over from the old population using the elitism to a new population.

3.2.1.2. Investment Simulator Design

The Investment Simulator used in our GP algorithm measures the performance of an individual (in both training and validation modes) by calculating its formula values for all stocks in the FTSE100 universe since the start of the fund. As we mentioned earlier, the Investment Simulator is called by our Fitness function in which the IS evaluates the individual’s formula values and generates a set of return values stored as a vector of values which are used to rank the population according to these values. In our project, this vector has two values: the ROI, and the Standard Deviation which are, in fact, used as our multiple objectives. This vector is also stored in each individual’s fitness object to be used into the SPEA2 Evaluator process to evolve the GP population. Next, we present
assumptions/rules that govern our Investment Simulator which we can summarize as follows:

1. The portfolio consists of one cash line (GBP).
2. The portfolio consists of a maximum of 25 stocks per month.
3. The FTSE100 stocks universe consists of 77 stocks.
4. The monthly investment value equals £1m cash regardless of previous month performance.
5. We ignore the interest received on cash holdings.
6. Our 5% trading costs to buy and sell stocks.
7. The IS runs for 60 months for training mode, and 20 months for validation.

3.3 Environment Design

In the next sub sections, we describe the two main cycles we have in our system: GP cycle and IS cycle.

3.3.1 Description of a standard GP Cycle

As we mentioned earlier in (section 3.2.1.1), the following steps summarize the standard GP cycle:

1. Initialize the GP population consists of individuals represented as binary trees. These GP Individuals use a random permutation of functions and terminals of random depths to build our assessment factors. These trees will be evaluated by the Investment Simulator to produce a single value. This single value is then applied on all stocks in our universe to rank these stocks according to their fitness.
2. In a multiobjective GP environment, we need a fitness function that can store more than one objective. In this project, we use SPEA2 fitness function to keep track of
two objectives of interest: the ROI and the Risk (i.e Standard Deviation). This function uses an array of two elements to store the above objectives which are assigned to each GP individual by our GP system.

3. The ECJ system supports many selection methods such as tournament, fitness proportionate selection (Roulette) and many others. We have found that the SPEA2 is limited, so far, to Tournament selection method. So, we decided to implement and use this method in our GP system.

4. Two parents (GP trees) are selected randomly to swap their genetic material at some random crossover point. The resulting offsprings are also sub trees contain a mixture of genetic material (terminals and functions) from both parents.

5. In this step, our GP system brings all the offsprings generated using the breeding step along with individuals brought over from the old population using the elitism to a new population.

3.3.2 Investment Simulator Cycle

In this section, this walkthrough details the main steps involved in simulating an investment of all FTSE100 stocks for 60 months (for training) and 20 months (for validation) for each assessment factor of our GP population. As a result, a set of statistics is dumped on several .csv and .stat files showing vital information for investors including best overall individual(s), best individual per generation and many more which will be discussed in next section.

At the start of each month

1. Calculate the formula values for all stocks that have been in FTSE100 since the start of the fund.
2. Calculate rankings of all above stocks based on the formula values to generate two quartiles:
   a. Top quartile: the cheapest or most attractive stocks.
   b. Bottom quartile: the most expensive or least attractive stocks.
3. Calculate the adjusted top quartile by ignoring the stocks that are no longer in the FTSE100.
4. For each stock in portfolio that falls in the bottom quartile, sell the entire holding.
5. If the cash is greater than 3% of the total fund and the portfolio is holding less than 25 stocks, then perform the following:
   a. Calculate the amount of cash in excess of 3% of the total fund value.
   b. Determine the target stocks that exist in the adjusted top quartile and not already held in the portfolio.
   c. Calculate the target investment per stock which can be described as follows: \( \text{MIN}( \text{Excess Cash/Number of target stocks, 4\% of the total value of the fund}) \).
   d. Use the target investment per stock to buy each target stock in which the cash line gets reduced accordingly.
6. If the cash is greater than 3\% of the total fund and there portfolio stock holdings with value less 4\% of the total fund, then perform the following:
   a. Find the stock holding with the lowest value and buy that stock to bring its value up to 4\% of fund total (or to the limit of available cash) in which the cash line gets reduced accordingly.
b. Repeat if there are still portfolio stock holdings with value less 4% of the fund value.

7. Calculate current total value of the fund and record in accounts.

3.3.3 Other Operations

In this section, we present a list of all other activities associated with our GP system as follows:

1. The GP System:

We need to incorporate some kind of charting facility in our GP system to support our statistical results. This option enables the end user to plot the best values of the ROI and the standard deviation for both SPEA2 & ALPS and SPEA2 only for the GP run. This facility is supported by a free open source code Java add-ons called JFreeChart for Java. JFreeChart is a free Java chart library that makes it easy for developers to display professional quality charts in their applications.

2. The Investment Simulator:

The following statistics files are generated at the end of each GP run to show vital information for the investors to be convinced of our results:

a. The “FinalStat.csv” file keeps track of the best and average of Annualized Return, Standard Deviation, and Sharpe Ratio per generation in the training mode only.

b. The “BestStat.csv” file keeps track of the index of the best of Annualized Return, Standard Deviation, and Sharpe Ratio per generation in the training mode only. The best individual is based on its sharpe ratio.
c. The “PopStat.csv” file writes the Annualized Return, Standard Deviation, and Sharpe Ratio for each individual in the population for both training and validation modes.

d. The “StocksOut.csv” file shows the buying and selling actions, the final contents of the portfolio and the total fund values for each individual in population for each month for the validation mode only.

3.3.4 Limitations in Design

There are some limitations in our GP system which we will highlight in this section along with our solutions or workarounds (if any) as follows:

1. GP system:

   • There is only one selection method supported by SPEA2 algorithm so far which is the Tournament selection. This needs consulting the algorithm’s documentation to determine the algorithm requirements and compliance in case we need to incorporate another selection method.

   • There are some restrictions in the design of our portfolio which are based on the customer’s requirements mentioned in (section 3.2.1.2) such as the number of stocks available for the IS and the amount of cash for investment. These requirements are considered part of the IS design objectives.

   • For each GP individual in our population, we have implemented 22 terminals and 6 functions to formulate the structure of our GP trees. This will limit us to those 22 terminals, which correspond to the data currently available to us. In case we need to implement new factors in the future as they become available, we need to append their names and codes to the .params file and implement their
functionality in the GP system before we use them. This requires recompiling the whole project source code before the GP system can evolve the population. The same is true for the functions.

2. Investment Simulator (IS):

- The FTSE100 stock universe data is only available for 80 months. We think it should be longer than that to enable our system to be more robust.

- We are restricted to invest in FTSE100 stocks universe. In case we need invest in more stocks, we to should extend our FTSE100 to a bigger universe which has wider range of stocks to invest in.

- The fact that we need to read, in advance, all the .csv files (containing the stocks data) every time we run a simulation. It is well known that reading data from the text files is much slower than querying data from a database. The impact of this problem can dramatically increase when we apply our system to an online environment where we need faster access to historical data. This problem can be solved by importing all the .csv files to a database such as mySQL 6.0 which is very fast and easy to install RDBMS. So when the IS needs to read a set of stocks it can query the stocks’ records from the database instead of reading its data from .csv file.

- In the IS implementation, it assumes one GP subpopulation exist. This limits GP practitioners using our system who need to use large population with multiple subpopulations from adapting our system and the simulator. To overcome this problem we had to fix the simulator to consider multiple subpopulations in its entire process and output.
4 Implementation

This chapter will describe the implementation of the design and testing of the implemented system. This starts by stating all the applications used in developing the system. This will follow a highlight on the source of our historical data for both training and validation stages. Then we will discuss the implementation issues in our GP system and the IS such as ECJ classes, ALPS classes, SPEA2 classes, and the IS structure.

4.1 Development Environment

In this section, we describe all software development applications used in this project which can be summarized below:

4.1.1 ECJ Package

Evolutionary Computation for Java (ECJ) is a generic GP and GA system developed and supported by the Evolutionary Computation Laboratory at George Mason University. ECJ is a Java-based EC capable of supporting wide range of GP and GA applications. ECJ is used here to control the evolutionary computation process of our GP population and is used to interface to the Investment Simulator.

4.1.2 Eclipse 3.2 Package

Eclipse is a full fledge an Integrated Development Environment (IDE) for Java development, it provides an area for writing Java programming language and it will also auto compile the source and it will also provide a debug function to remove bugs in the system. This is used to increase the efficiency of implementation of design.
4.1.3 Java 2 Standard Edition JDK 6.0

The Java 2 Standard Edition (J2SE) system is an open-source development kit written in Java programming language and compiled using Sun Java compiler. J2SE runtime environment must be installed in order to run any Java applications. Java was chosen as the programming language for the system because of several reasons:

1. Java is an Object Orientated programming language which features high efficiency and flexibility in both design and run time.

2. Our EC environment (ECJ package including SPEA2 extension), and the Investment Simulator are java-based applications. This restricts us to use Java as the development programming language as we are limited in time instead of building the EC system from scratch otherwise.

4.1.4 MySQL 6.0

As we mentioned in section 3.3.3, one of the limitations associated with the Investment Simulator is accessing the historical data from text files rather from database tables. This problem is solved by incorporating MySQL engine into the GP system which will enable the Investment Simulator to query the database when stocks’ historical data is needed. On the other hand, when the IS finishes the simulation cycle a set of statistical files will be dumped to database tables instead of dumping these files to text files.

4.2 Source of Historical Data

This section highlights the source of our historical data for both training and validation stages. Let’s first walkthrough the structure of the main .csv file (called All_FTSE100.csv) which consists of 14 columns starting with the stock’s name, the Reuters Instrument Code (RIC), and a series of 12 years from 1995 to 2006. If a stock
exists in the FTSE100 universe for that year then the column under that year is set to 1 otherwise 0. This structure is read by the IS to build the FTSE100 list of our universe on which the rest of the simulation cycle depends. An example of such structure is shown in Table 2 below:

<table>
<thead>
<tr>
<th>Name</th>
<th>RIC</th>
<th>95</th>
<th>96</th>
<th>97</th>
<th>98</th>
<th>99</th>
<th>00</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLIANCE UNICHEM</td>
<td>AUN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>AMVESCAP</td>
<td>AVZ</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ASTRAZENECA</td>
<td>AZN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BAA</td>
<td>BAA</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BAE SYSTEMS</td>
<td>BA</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BG GROUP</td>
<td>BG</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BHP BILLITON</td>
<td>BLT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BOOTS GROUP</td>
<td>BOO</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BRITISH LAND CO</td>
<td>BLND</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BRITISH PETROLEUM CO</td>
<td>BP</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2: An example of the main structure of our FTSE100 stocks universe in the years 1995-2006 shows 1’s and 0’s under each year’s column to indicate if that stock is in the FTSE100 universe for that year.

Next, the IS identifies each stock’s file name by its RIC read previously in the main file and then it will access that .csv file to read its contents. For example, for the RIC = ‘AUN’, the IS will open the file “AUN.csv” to access its contents. The layout of each .csv file is shown in Table 3 below. The first column represents the month and year, and the next 22 columns represent the corresponding terminals values for that stock:
<table>
<thead>
<tr>
<th>Excel Column #</th>
<th>Corresponding Assessment Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Close</td>
</tr>
<tr>
<td>1</td>
<td>Price Momentum</td>
</tr>
<tr>
<td>2</td>
<td>Volume</td>
</tr>
<tr>
<td>3</td>
<td>Price Cash</td>
</tr>
<tr>
<td>4</td>
<td>Book Price</td>
</tr>
<tr>
<td>5</td>
<td>PE Ratio</td>
</tr>
<tr>
<td>6</td>
<td>Moving Average for 30 days</td>
</tr>
<tr>
<td>7</td>
<td>Moving Average Changes</td>
</tr>
<tr>
<td>8</td>
<td>Volatility</td>
</tr>
<tr>
<td>9</td>
<td>Dividend Yield</td>
</tr>
<tr>
<td>10</td>
<td>Earn Equity</td>
</tr>
<tr>
<td>11</td>
<td>BVPS</td>
</tr>
<tr>
<td>12</td>
<td>Market Capital</td>
</tr>
<tr>
<td>13</td>
<td>Change ROE</td>
</tr>
<tr>
<td>14</td>
<td>Revenue Growth</td>
</tr>
<tr>
<td>15</td>
<td>Share Yield</td>
</tr>
<tr>
<td>16</td>
<td>Adjusted Dividend Yield</td>
</tr>
<tr>
<td>17</td>
<td>EPS</td>
</tr>
<tr>
<td>18</td>
<td>Earn Growth</td>
</tr>
<tr>
<td>19</td>
<td>Equity Asset</td>
</tr>
<tr>
<td>20</td>
<td>Z Factor</td>
</tr>
<tr>
<td>21</td>
<td>CPS DPS</td>
</tr>
<tr>
<td>22</td>
<td>Simulator ERC</td>
</tr>
</tbody>
</table>

Table 3: Mapping between the CSV columns and the terminals in a stock.

4.2.1 Training Data

The IS uses a stock universe of 77 FTSE100 for training stage supplied by the Reuter 3000 Xtra system in the form of .csv files for an investment period of 60 months (from 31.05.1999 to 31.12.2003).

4.2.2 Testing Data

In the testing stage, the IS also uses the stock universe in the form of .csv files for an investment period of 20 months (from 01.01.2004 to 31.12.2006). This data is meant for testing our system on unseen data and for validating the results obtained in the training stage.
4.3 General Implementation Issues

In this section we will present the structure of our GP classes and the IS classes. First, we describe the backbone of the ECJ system, followed by a description of the ALPS and SPEA2 classes. Finally, we will describe the IS classes structure.

4.3.1 Classes Structure

Although ECJ has plenty of classes in its framework but we will give, below, a brief description of some classes of each group, the rest of the classes in each group are listed in Appendix A:

4.3.1.1 ECJ Classes

1. Evolve class is the topmost class in the evolutionary system. It provides the entry point main() which sets up the check-pointing facility, random number generators, output logging facility, parameter database, and the EvolutionState object.

2. Singleton class is a Setup which defines classes which have only a single global instance throughout the evolutionary run.

3. EvolutionState class, a Singleton object, holds the complete state of evolution at any time. EvolutionState is passed around in a lot of methods to objects which need access to these features. EvolutionState also is the entry point for the main evolutionary loop.

4.3.1.2 ALPS Modifications

We had to modify/add properties in several classes in the ECJ package to incorporate the ALPS algorithm. These classes are listed below (for a complete list, see the System Classes Appendix A):
1. Individual Class:

<table>
<thead>
<tr>
<th>Property name</th>
<th>Type - scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>Public int</td>
<td>Used to keep track of individual's age throughout the GP cycle.</td>
</tr>
<tr>
<td>AgeLayer</td>
<td>Public int</td>
<td>Used to keep track of individual's age layer it belongs at a given time.</td>
</tr>
<tr>
<td>LastGenUsedAsParent</td>
<td>Public int</td>
<td>Used to indicate in which generation this individual has been used as a parent in any breeding process, (default = -1).</td>
</tr>
</tbody>
</table>

2. Subpopulation Class:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type - Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgeLayersList</td>
<td>public LinkedList</td>
<td>A linked list of all individuals of ages ranging from the same age layer or adjacent lower layer.</td>
</tr>
<tr>
<td>AgeLayerItr</td>
<td>public LinkedListItr</td>
<td>A pointer to AgeLayerList linked list.</td>
</tr>
</tbody>
</table>

4.3.1.3 SPEA2 Classes

The SPEA2 classes are an extension to the ECJ system. SPEA2 classes were implemented in the ECJ package by Robert Hubley and can be listed below:

1. SPEA2MultiObjectiveFitness Class is a subclass of Fitness which implements basic multiobjective fitness functions along with support for the ECJ SPEA2 (Strength Pareto Evolutionary Algorithm) extension. This class contains two items: an array of floating point values representing the various multiple fitness values (ranging from 0.0 (worst) to infinity (best)), and a single SPEA2 fitness value which represents the individual's overall fitness. SPEA2Fitness is therefore, a function of the number of individuals it dominates where 0.0 is the best.

2. Tournament Selection Class does a simple tournament selection, limited to the subpopulation it's working in at the time and only within the boundary of the
SPEA2 archive (between 0-archiveSize). As we mentioned in the previous section, the ALPS algorithm is run within the SPEA2 archive file.

4.3.2 Investment Simulator

Here in this section we outline the main Investment Simulator classes used within the ECJ system were designed and implemented in our CS department as part of graduation project and shown below. Full list of the IS classes can be found in the System Classes Appendix:

1. Portfolio Class:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type - Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_STOCKS_NUM</td>
<td>private int</td>
<td>Used to set the maximum number of stocks in the portfolio. (default =25)</td>
</tr>
<tr>
<td>START_CASH</td>
<td>private float</td>
<td>Used to specify the investment amount available (default =£1M)</td>
</tr>
<tr>
<td>isEmpty</td>
<td>private boolean</td>
<td>Used to check if the portfolio is empty.</td>
</tr>
<tr>
<td>numStocks</td>
<td>private int</td>
<td>Used to keep track of the current number of stocks in the portfolio at given time.</td>
</tr>
<tr>
<td>cashHolding</td>
<td>private float</td>
<td>Used to show the current cash in the portfolio</td>
</tr>
<tr>
<td>stocksHeld</td>
<td>private ArrayList</td>
<td>Used to show the available stocks held in the portfolio.</td>
</tr>
<tr>
<td>stockAmount</td>
<td>private ArrayList</td>
<td>Used to show the available stocks held in the portfolio along with their amount of shares.</td>
</tr>
</tbody>
</table>

2. Stock Class:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type - Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FACTORS_NUM</td>
<td>private int</td>
<td>Used to set the number assessment factors to fixed number 22.</td>
</tr>
<tr>
<td>MONTHS</td>
<td>private int</td>
<td>Used to set the simulation period in months.</td>
</tr>
<tr>
<td>Id</td>
<td>private String</td>
<td>Used to set the RIC of a stock to null.</td>
</tr>
<tr>
<td>Factors</td>
<td>private float[][]</td>
<td>Used to store all the factor values for all months in a multi-dimensional array.</td>
</tr>
<tr>
<td>formulaValue</td>
<td>private float</td>
<td>Used to store the formula value in current month as a result of evaluating the GP formula on this stock.</td>
</tr>
</tbody>
</table>
5. Experiments

The purpose of the following experiments is to answer two research questions mentioned in our project objectives (section 1.1) which can be summarized as follows:

1. If ALPS can be incorporated with SPEA2, which we already done, does this:
   1.1 Reduce premature convergence in the combined system better than SPEA2 alone?
   1.2 Increase diversity in population in the combined system better than SPEA2 alone?
   1.3 Produce results that dominate results generated from SPEA2 alone?

In our project, we are going to focus on the questions 1.2 and 1.3 only. In order to answer question 1.2, we are going to measure the diversity in the combined system compared to SPEA2 alone by calculating the following:

   1.2.1 The average number of best individuals in the archive file with ROI≥10%.
   1.2.2 The distribution of the ROI of all individuals in the archive file.
   1.2.3 The distribution of the Standard Deviation of all individuals in the archive file.
   1.2.4 Measure the distribution of Sharpe Ratio of all individuals in the archive file.

For each of these experiments, we will use Ranked T-Test to compare the distributions for the combined system and SPEA2 only.

2. What are the parameters sensitivities of the combined system (ALPS & SPEA2)? In other words, do they interfere with each other? In order to answer this question, we need to measure the effect of adjusting two key parameters, chosen for their importance, in both algorithms which are the SPEA2 archive file size and the ALPS age gap:

   2.1 Measure the effect of reducing the archive file size.
   2.2 Measure the effect of increasing the age gap.
Before we proceed with the experiments, let us list our GP and IS control parameters we are going to use in our experiments:

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Parameter Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP system</td>
<td>Seed</td>
<td>Time (which gets the current system time)</td>
</tr>
<tr>
<td></td>
<td>Generations</td>
<td>499</td>
</tr>
<tr>
<td></td>
<td>Population size</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>Quit-on-run-complete</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>Stat</td>
<td>Invest.MultiObjectiveStatistics</td>
</tr>
<tr>
<td></td>
<td>Elite</td>
<td>3</td>
</tr>
<tr>
<td>SPEA2</td>
<td>Archive file size</td>
<td>10, 125</td>
</tr>
<tr>
<td></td>
<td>Multiobjective Fitness</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Subpopulation</td>
<td>SPEA2Subpopulation</td>
</tr>
<tr>
<td></td>
<td>Fitness</td>
<td>ec.app.Invest.InvestMultiFitness</td>
</tr>
<tr>
<td></td>
<td>Species</td>
<td>ec gp.GPSpecies</td>
</tr>
<tr>
<td></td>
<td>Individual Species</td>
<td>ec gp.GPIndividual</td>
</tr>
<tr>
<td></td>
<td>Evaluator</td>
<td>InvestMultiEvaluator</td>
</tr>
<tr>
<td></td>
<td>Breeder</td>
<td>SPEA2Breeder</td>
</tr>
<tr>
<td></td>
<td>Reproduction</td>
<td>SPEA2TournamentSelection</td>
</tr>
<tr>
<td></td>
<td>Crossover</td>
<td>SPEA2TournamentSelection</td>
</tr>
<tr>
<td>ALPS</td>
<td>Alpsoption</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>Agegap</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>ageScheme</td>
<td>1 (1 = Linear, 2 = Fibonacci, 3 = Poly, 4=Exp.)</td>
</tr>
<tr>
<td></td>
<td>Printstatistics</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>ALPSPredefinedSize</td>
<td>200</td>
</tr>
<tr>
<td>Investment Simulator</td>
<td>DataPath</td>
<td>C:\eclipse\workspace\ECJ\ec\app\Invest_Data</td>
</tr>
<tr>
<td></td>
<td>StatPath</td>
<td>C:\eclipse\workspace\ECJ\ec\app\Invest_Stats</td>
</tr>
<tr>
<td></td>
<td># of GP Function Sets</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>GP Function Set Size</td>
<td>29 (22 terminals, 6 functions, 1 ERC)</td>
</tr>
<tr>
<td></td>
<td>Evaluation Problem</td>
<td>Invest.InvestSim</td>
</tr>
<tr>
<td></td>
<td>Evaluation Problem Data</td>
<td>Invest.FloatData</td>
</tr>
</tbody>
</table>

Table 4: The GP and IS control parameters used in our system.

Next, we will describe our experimental design of our experiments by specifying the aims of these experiments, the methodology used, and concluding with the results and analysis.
5.1. Experiment One

5.1.1 Aims

In our first experiment we try to answer the research question (1.2), mentioned in the previous section, which is stated in question (1.2.1) by measuring the average number of best individuals in the archive file with ROI \( \geq 10\% \) as a way of measuring the diversity of the combined system and SPEA2 alone.

5.1.2 Experimental Design

The experiment will start by performing 20 runs of the GP system under the following conditions for training stage:

1. ALPS option = true.
2. Number of generations = 499.
4. For training stage, we will use a training data set for the IS to simulate each GP individual on every stock in the FSTE100 universe for 60 months from 31.05.1999 to 31.12.2003 per run.
5. For the validation stage, we will use a validation data set for 20 month (from 01.01.2004 to 31.12.2006) for one run only.

Then the experiment will perform another 20 runs of the GP system under the same conditions mentioned above except for the ALPS option turned off.
5.1.3 Results and Analysis

In this experiment, the chart below shows that the combined system has almost doubled the average number of best solutions with profits (ROI≥10%) per run compared to the SPEA2 alone.

![Bar Chart](chart.png)

Figure 7: The average number of best solutions per run for high ROI values for both algorithms. The combined system has outperformed the SPEA2 alone.

**Ranked T-Test:**

When we ran the Ranked T-Test on the two populations, it should tell us that there is a very small possibility that the two populations come from the same distribution with the P-Value = 0.0026989.
Other Statistical Measures:

Figure 8 below supports our findings by showing the standard deviation $\sigma$ (which measures the diversity) and the mean (which measures the performance) of both algorithms:

![Graph showing normal distribution of statistics for SPEA2 and ALPS](image)

Figure 8: The normal distribution of the statistics mentioned in figure 7 with supporting indicators such as the standard deviation and the mean.

5.2 Experiment Two

5.2.1 Aims

In our second experiment we try to answer the research question (1.2), mentioned in the chapter’s introduction which is stated in question (1.2.2) by measuring the distribution of ROI of all individuals in the archive file as another way of measuring the diversity of the combined system compared to SPEA2 alone.

5.2.2 Experimental Design
The design of this experiment is the same as in the first experiment.

5.2.3 Results and Analysis

In this experiment, our system has suggested results with remarkable improvements in terms of the ROI generated in each run. The chart below shows that the combined system (ALPS&SPEA2) has better diversity in the ROI values in validation mode compared to SPEA2 alone.

![Chart showing diversity comparison between ALPS&SPEA2 and SPEA2 in training and validation modes.]

Figure 9: The diversity in the ROI for ALPS & SPEA2 versus SPEA2 only in the validation mode.

Ranked T-Test:

When we ran the Ranked T-Test on the two populations, it should tell us that there is a very small possibility that the two populations come from the same distribution with the

\[ P-Value = 1.96119961 \times 10^{-8}. \]
**Other Statistical Measures:**

Figure 10 below supports our findings by showing the standard deviation $\sigma$ (which measures the diversity) and the mean (which measures the performance) of both algorithms:

![Graph showing standard deviation and mean](image)

Figure 10: The normal distribution of the statistics mentioned in figure 9 with supporting indicators such as the standard deviation and the mean.

### 5.3 Experiment Three

#### 5.3.1 Aims

In this experiment we will answer the research question (1.2), mentioned in the chapter’s introduction which is stated in question (1.2.3) by measuring the distribution of the standard deviation of all individuals in the archive file as another way of measuring the diversity of the combined system compared to SPEA2 alone.

#### 5.3.2 Experimental Design

The design of this experiment is the same as in the first experiment.
5.3.3 Results and Analysis

In this experiment, our system has suggested results with remarkable fluctuations in terms of the standard deviation generated in each run. The chart below shows that the combined system (ALPS&SPEA2) has better diversity in the standard deviation values in validation mode compared to SPEA2 alone.

![Graph showing Standard Deviation comparison between ALPS&SPEA2 and SPEA2 alone](image)

Figure 11: The diversity in the Standard Deviation for ALPS & SPEA2 versus SPEA2 only in the validation mode.

**Ranked T-Test:**

When we ran the Ranked T-Test on the two populations, it should tell us that there is a very small possibility that the two populations come from the same distribution with the P-Value = 4.26878X10^{-13}.

**Other Statistical Measures:**

Figure 12 below supports our findings by showing the standard deviation $\sigma$ (which measures the diversity) and the mean (which measures the performance) of both algorithms:
Figure 12: The normal distribution of the statistics mentioned in figure 11 with supporting indicators such as the standard deviation and the mean.

5.4 Experiment Four

5.4.1 Aims

In this experiment we will answer the research question (1.2), mentioned in the chapter’s introduction which is stated in question (1.2.4) by measuring the distribution of the Sharpe Ratio of all individuals in the archive file as another way of measuring the diversity of the combined system compared to SPEA2 alone.

5.4.2 Experimental Design

The design of this experiment is the same as in the first experiment.

5.4.3 Results and Analysis

In this experiment, our system has suggested results with remarkable fluctuations in terms of the Sharpe Ratio generated in each run. The chart below shows that the difference between the maximum and minimum values in the combined system is larger than that in SPEA2 alone system in validation mode.

- 50 -
Figure 13: The diversity in the Sharpe Ratio for ALPS & SPEA2 versus SPEA2 only in the validation mode.

**Ranked T-Test:**

When we ran the Ranked T-Test on the two populations, it should tell us that there is a very tiny possibility that the two populations come from the same distribution with the P-Value = 4.36461X10^-12.

**Other Statistical Measures:**

Figure 14 below supports our findings by showing that the standard deviation $\sigma_2$ which measures the diversity of the ALPS & SPEA2 has outperformed $\sigma_1$ (which measures the diversity of SPEA2 only system. In addition, there is a slight improvement in the performance of the ALPS & SPEA2 compared to that of SPEA2 only:
Figure 14: The normal distribution of the statistics mentioned in figure 13 with supporting indicators such as the standard deviation and the mean.

5.5 Experiment Five

5.5.1 Aims

In this experiment we will answer the research question (1.3), mentioned in the chapter’s introduction by comparing the generated results from the combined system with the results from SPEA2 only as another way of measuring the diversity of both systems.

5.5.2 Experimental Design

The design of this experiment is the same as in the first experiment.

5.5.3 Results and Analysis

In this experiment, our system has shown remarkable results in terms of the ROI, for example, generated in each run. The chart below shows that generated ROI by the
combined system has dramatically improved compared to slight improvements in SPEA2 alone system in validation mode.

![Generated ROI for ALPS & SPEA2 vs SPEA2 only in validation mode](image)

Figure 15: The generated ROI for ALPS & SPEA2 has dominated these for SPEA2 only in the validation mode.

**Ranked T-Test:**

When we ran the Ranked T-Test on the two populations, it should tell us that there is a very tiny possibility that the two populations come from the same distribution with the P-Value = 2.342119X10^{-8}.

**Other Statistical Measures:**

Figure 16 below supports our findings by showing that the mean of the combined system has outperformed the mean of the SPEA2 only system:
Figure 16: The normal distribution of the statistics mentioned in figure 15 with supporting indicators such as the mean which indicates that the combined system has outperformed the SPEA2 only.

5.6 Experiment Six

5.6.1 Aims

In this experiment we will answer the research question (2.1), mentioned in the chapter’s introduction which focuses on the parameters sensitivity for both systems. Therefore, we need to measure the effect of adjusting two key parameters, chosen for their importance, in both systems which are the SPEA2 archive file size and the ALPS age gap. In this experiment, we will measure the effect of reducing the archive file size to a small value (say 10).
5.6.2 Experimental Design

The design of this experiment is the same as in the first experiment except that we are going to set the archive size to 10.

5.6.3 Results and Analysis

When we used a very small archive file (size=10) compared to the population size, there were significant drawback in both the Sharpe Ratio and the ROI between the two algorithms (the ALPS & SPEA2, SPEA2 only). In fact, the chart below shows the resultant Sharpe Ratio has improved significantly in SPEA2 only system compared to the combined system (Figure 17).

![Generated Sharpe Ratio for ALPS & SPEA2 vs SPEA2 only](chart.png)

Figure 17: Significant improvements in the Sharpe Ratio of SPEA2 only system compared to the combined system in validation mode with archive file size =10.

The chart above shows that the performance of SPEA2 only algorithm has not been totally affected by the archive file size. Whereas, ALPS was dramatically affected by the size of the archive file even though that both algorithms were bounded by the SPEA2
archive space this was 10 in this experiment. This unfortunate result in the ALPS & SPEA2 algorithm shows that the GP system was not able to take advantage of the ALPS novel aging feature during the selection process because of the restriction on creating new age layers within a small archive file. Therefore, for the ALPS algorithm to work properly, the archive size should be large enough to have a mixture of different age layers to be utilized by the ALPS algorithm during the selection process. In the next experiment, we will examine the impact of any improvement in the performance when we enlarge the archive size to a larger value, say 125.

5.7 Experiment Seven

5.7.1 Aims

This experiment is an extension to the previous one in which we are going to show the effect of increasing the archive file size to a large value (say 125).

5.7.2 Experimental Design

The design of this experiment is the same as in the first experiment except that we are going to set the archive size to 125.

5.7.3 Results and Analysis

When we used a relatively large archive file (size=125) compared to the population size, there were substantial improvements in both algorithms in terms of the ROI. The chart below (Figure 18) shows the ROI for both algorithms in validation stage with archive file of size 125. As we can see from the chart below that increasing the archive file size has dramatically improved both algorithms with an advantage to the ALPS & SPEA2 system compared to the SPEA2 only. We can also see the effect of increasing the archive size
has boosted the lower bounds of the ROI for both systems to 10% and the upper bounds to 16% for SPEA2 only and above 18% to the combined system.

Figure 18: Significant improvements in the ROI for SPEA2 only system compared to the combined system in validation mode with archive file size =125.

5.8 Experiment Eight

5.8.1 Aims

In this experiment we will answer the research question (2.2), mentioned in the chapter’s introduction which focuses on the parameters sensitivity for both systems. In this experiment, we will measure the effect of increasing the age gap size to a large value say age gap=15.
5.8.2 Experimental Design

The design of this experiment is the same as in the first experiment except that we are going to set the age gap size to 15 when ALPS option is turned on. Note that SPEA2 is insensitive to adjusting the age gap as this parameter belongs to the ALPS algorithm.

5.8.3 Results and Analysis

When we used a relatively large age gap (size=15) when ALPS option is turned on, there were significant improvements in the combined system rather than in the SPEA2 only in terms of the ROI. Figure 19 shows the ROI for both algorithms in validation stage with age gap of size 15. As we can see from the chart below that increasing the age gap size has dramatically improved the combined system by 25-30% compared to the SPEA2 only. We can also see the effect of increasing the age gap size has boosted the lower bounds of the ROI for the combined system from 10% to 12% and the upper bounds from 18% to 24%.

![Generated ROI for ALPS & SPEA2 vs SPEA2 only](image)

Figure 19: Significant improvements in the ROI of the combined system in validation mode with age gap size =15 compared to the SPEA2 only which has no improvement.
The chart above shows that the performance of SPEA2 only algorithm has not been affected by the age gap size as SPEA2 is not sensitive to this parameter as we mentioned above.
6 Conclusions

In chapter, we will summarise the achievements in our project in terms of the technical and experimental objectives we set earlier. Then we will take a look at any future improvements in the project towards achieving better results and robustness in our project.

6.2 Summary

We will list, again, all our project objectives, both technical and experimental, which all have been

Technical Objectives

1. The ECJ system was successfully installed and configured.
2. The SPEA2 option in ECJ was successfully configured.
3. The ALPS algorithm was successfully designed and implemented.
4. Our GP system was successfully configured with the Investment Simulator.
5. We successfully ran five experiments with the same configurations and three experiments with different configurations as described in the experiments chapter.

Experimental Objectives

1. Our multiobjective GP system was setup successfully for optimising a financial portfolio and its trading rules which was detailed in section (3.2.1.2).
2. The efficiency of the SPEA2 technique was successfully evaluated when applied alone to our portfolio optimisation problem using the archive file size as adjustable parameter. We also observed that SPEA2 algorithm was tolerant to
small archive file (size=10) while the combined system was not. In addition, we found that SPEA2 worked amazingly well with large archive file (size =125).

3. The efficiency of the combined system (ALPS & SPEA2) was profitably evaluated using a thorough set of training and testing experiments. The results have shown:
   a. How the ALPS & SPEA2 has a better diversity than SPEA2 only through several measurements mentioned in the experiments chapter.
   b. What the parameters sensitivities of the combined system are, such as the archive file and age gap as adjustable parameters, when applied to our portfolio optimisation problem.

4. We have obtained outstanding results from the experiments we have done and we have analyzed these results to conclude our findings and recommendations for future work which we will highlight next.

5. We can also conclude the following remarks:
   a. The design of our GP system mandates that the ALPS structure is bounded or controlled by the SPEA2 archive file as it is the working environment of the ALPS algorithm.
   b. As the age gap is a factor parameter in the ALPS algorithm, this factor has played a vital role in boosting it results when we used large age gaps which allows the algorithm to create more age layers which increases the diversity in the population and the resultant fitness.
c. The same thing applies to the archive file size in the SPEA2 algorithm, which has a main ingredient of profitable results when using large archive size compared to the ALPS.

d. SPEA2 is tolerant to small archive file size whereas ALPS does not work properly in small archives.

e. We can also conclude from this project that the larger the age gap in the ALPS algorithm the better the results we expect from the combined system due the effect of the age gap on the population total fitness which we discuss earlier.

6.3 Future Work

Although we have had a great success in our experiments on both the training and validation data sets, we think that there still opportunities for potential improvements. Beside the potential improvements we mentioned in section 3.3.4 future experiments, for example, can be carried out to run on different investment simulators with different features. This will improve the GP learning process and expose our system to different environments, restrictions and may be different objectives. One could also test our system on different hardware platform (other than MS-Windows) which could lead them to more promising results compared to the experimental environment we tested our system in.
7 References


APPENDIX A
System Classes
1. ECJ classes

Evolve class is the topmost class in the evolutionary system. It provides the entry point main() which sets up the check-pointing facility, random number generators, output logging facility, parameter database, and the EvolutionState object.

Singleton class is a Setup which defines classes which have only a single global instance throughout the evolutionary run.

EvolutionState class, a Singleton object, holds the complete state of evolution at any time. EvolutionState is passed around in a lot of methods to objects which need access to these features. EvolutionState also is the entry point for the main evolutionary loop.

Initializer class is responsible for creating the initial GP population. This class might do this by randomly-generating individuals or by loading them from a text file provided by the user.

Evaluator class contains a prototypical Problem which defines the problem domain individuals are being evaluated against. At evaluation time, the Evaluator clones a separate Problem for each evaluation thread.

Breeder class breeds new populations from previous ones.

Exchanger class performs inter-subpopulation exchanges or inter-process exchanges (possibly located on other machines).

Finisher class is responsible for cleaning up the GP population at the end of an evolutionary run.
Statistics class performs a variety of statistics accumulation and logging tasks throughout the evolutionary process which can be dumped to the user screen or text files.

Population class is received from the Initializer and updated during the evolutionary run. The Population holds an array of Subpopulations. Population and Subpopulation are both Groups. It's up to the particular Breeder and Evaluator used as to how the Population and its one or more Subpopulations are to be bred and evaluated. The EvolutionState class must have at least one population.

Subpopulation class is an array of Individuals. There are always one or more Subpopulations in the Population. Each Subpopulation has a Species, which governs the formation of the Individuals in that Subpopulation. Subpopulations also contain a Fitness prototype which is cloned to form Fitness objects for individuals in the subpopulation. The initial subpopulation is populated with new random individuals using the populate() method.

Individual class is the basic atomic unit of evolution that is evaluated and bred. It defines a possible solution to the given evolutionary domain problem. Each Individual holds a Fitness instance, which defines its evaluated fitness in the population.

Species class represents a group of individuals which have the same basic genetic makeup. A Species instance contains a prototypical Individual which is commonly used to generate new Individuals at the request of the Initializer when it is creating an initial population.
BreedingPipeline class is the basic mechanism for breeding individuals from a previous generation to form the next generation. BreedingPipeline takes Individuals from one or more BreedingSources, which may be either SelectionMethods or other BreedingPipelines, and produces new individuals derived from those source individuals. BreedingPipelines can be used in a multithreaded environment.

SelectionMethods class selects individuals from an old population and returns them. SelectionMethods might include Tournament Selection, Fitness Proportional Selection, etc. In addition, SelectionMethods don't have parent sources.

2. ALPS Classes

Individual class:

<table>
<thead>
<tr>
<th>Property name</th>
<th>Type - scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>Public int</td>
<td>Used to keep track of individual's age throughout the GP cycle.</td>
</tr>
<tr>
<td>AgeLayer</td>
<td>Public int</td>
<td>Used to keep track of individual's age layer it belongs at a given time.</td>
</tr>
<tr>
<td>LastGenUsedAsParent</td>
<td>Public int</td>
<td>Used to indicate in which generation this individual has been used as a parent in any breeding process, (default = -1).</td>
</tr>
</tbody>
</table>

EvolutionState class:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type - Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpsoption</td>
<td>public Boolean</td>
<td>Used to check if ALPS option is enabled. (default = false).</td>
</tr>
<tr>
<td>P_ALPSOPTION</td>
<td>public string</td>
<td>A parameter label to implement above property in the parameter file.</td>
</tr>
<tr>
<td>AgeGap</td>
<td>Public Int</td>
<td>Used to segregate individuals in a population into age layers based on their ages. The lower age layer (layer 0) gets initialized at each age gap.</td>
</tr>
<tr>
<td>P_AGEGAP</td>
<td>public string</td>
<td>A parameter label to implement above property in the parameter file.</td>
</tr>
<tr>
<td>Printstatistics</td>
<td>Public Boolean</td>
<td>Used to enable charting statistics (default=false).</td>
</tr>
<tr>
<td>P_PRINTSTATISTICS</td>
<td>public String</td>
<td>A parameter label to implement above property in the parameter file.</td>
</tr>
<tr>
<td>Property Name</td>
<td>Type - Scope</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LargestAgeLayer</td>
<td>Public Int</td>
<td>Used to store the largest/last age layer reached so far.</td>
</tr>
<tr>
<td>IndexOfPreviousAgeLayer</td>
<td>public Int</td>
<td>This flag is used to check if an individual is already selected as a parent and we need to select another one in the same age layer or lower. (default = -1).</td>
</tr>
<tr>
<td>AgeLayersListSize</td>
<td>public Int</td>
<td>Used to represent the number of individuals of the same age layer or one below for a given age layer. This eliminates the need to traverse the list to count its members.</td>
</tr>
<tr>
<td>AgeScheme</td>
<td>public Int</td>
<td>Used to setup the age maximum limits for each age-layer based on the AgeGap according to the following aging schemes: Linear, Fibonacci, Polynomial (n^2), and Exponentials (2^n). (default=1)</td>
</tr>
<tr>
<td>P_AGESCHEME</td>
<td>public string</td>
<td>A parameter label to implement above property in the parameter file.</td>
</tr>
<tr>
<td>ALPSPredefinedSize</td>
<td>public Int</td>
<td>Used to enforce a predefined number of age layers.</td>
</tr>
<tr>
<td>(default =10)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_ALPSPredefinedSize</td>
<td>public string</td>
<td>A parameter label to implement above property in the parameter file.</td>
</tr>
<tr>
<td>MaxAgeLimitArray</td>
<td>public Int</td>
<td>A list of maximum age limits which are used to determine when to initialize the age lower layer.</td>
</tr>
</tbody>
</table>

Subpopulation class:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type - Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgeLayersList</td>
<td>public LinkedList</td>
<td>A linked list of all individuals of ages ranging from the same age layer or adjacent lower layer.</td>
</tr>
<tr>
<td>AgeLayerItr</td>
<td>Public LinkedListItr</td>
<td>A pointer to AgeLayerList linked list.</td>
</tr>
</tbody>
</table>

3. SPEA2 classes

SPEARBreeder class breeds each subpopulation separately, with no inter-population exchange, and using the SPEA2 approach. Prior to breeding a subpopulation, a SPEA2Breeder will first fill part of the new subpopulation (the archive) with the individuals with an SPEA2 fitness of less than 1.0 from the old subpopulation. If there are fewer individuals with this cutoff than can fit in the archive, the free slots are filled with the lowest scoring SPEA2 fitness individuals. If there are more
individuals with SPEA2 Fitness less than 1 than can fit in the archive then the archive
will be truncated and individuals which are close to others will be removed.

SPEA2Evaluator class is a simple, non-coevolved generational evaluator which
evaluates every single member of every subpopulation individually in its own
problem space. The evaluator is also responsible for calculating the SPEA2Fitness
function. This function depends on the entire population and so cannot be calculated
in the Problem class.

SPEA2MultiObjectiveFitness class is a subclass of Fitness which implements basic
multiobjective fitness functions along with support for the ECJ SPEA2 (Strength
Pareto Evolutionary Algorithm) extension. This class contains two items: an array of
floating point values representing the various multiple fitness values (ranging from
0.0 (worst) to infinity (best)), and a single SPEA2 fitness value which represents the
individual's overall fitness. SPEA2Fitness is therefore, a function of the number of
individuals it dominates where 0.0 is the best.

SPEA2Subpopulation class is a simple subclass of Subpopulation with the
archiveSize field added. The archive is portion of the subpopulation so archive size
ranges from 1 to the population size -1. ArchiveSize dictates he total number of
individuals from the population which can be in the archive. In next chapter, we will
highlight the impact of selecting the ArchiveSize which is a constant integer on our
system robustness.

Tournament Selection class does a simple tournament selection, limited to the
subpopulation it's working in at the time and only within the boundary of the SPEA2
archive (between 0-archivoSize). As we mentioned in the previous section, the ALPS algorithm is run within the SPEA2 archive file.

1. Investment Simulator Classes

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type/Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_STOCKS_NUM</td>
<td>private int</td>
<td>Used to set the maximum number of stocks in the portfolio. (default =25)</td>
</tr>
<tr>
<td>START_CASH</td>
<td>private float</td>
<td>Used to specify the investment amount available (default =£1M)</td>
</tr>
<tr>
<td>isEmpty</td>
<td>private boolean</td>
<td>Used to check if the portfolio is empty.</td>
</tr>
<tr>
<td>numStocks</td>
<td>private int</td>
<td>Used to keep track of the current number of stocks in the portfolio.</td>
</tr>
<tr>
<td>cashHolding</td>
<td>private float</td>
<td>Used to show the current cash in the portfolio.</td>
</tr>
<tr>
<td>stocksHeld</td>
<td>private ArrayList</td>
<td>Used to show the available stocks held in the portfolio.</td>
</tr>
<tr>
<td>stockAmount</td>
<td>private ArrayList</td>
<td>Used to show the available stocks held in the portfolio along with their amount of shares.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type/Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FACTORS_NUM</td>
<td>private int</td>
<td>Used to set the number assessment factors to fixed number 22.</td>
</tr>
<tr>
<td>MONTHS</td>
<td>private int</td>
<td>Used to set the simulation period in months.</td>
</tr>
<tr>
<td>Id</td>
<td>private String</td>
<td>Used to set the RIC of a stock to null.</td>
</tr>
<tr>
<td>Factors</td>
<td>private float[][]</td>
<td>Used to store all the factor values for all months in a multi-dimensional array.</td>
</tr>
<tr>
<td>formulaValue</td>
<td>private float</td>
<td>Used to store the formula value in current month as a result of evaluating the GP formula on this stock.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type/Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_DATA</td>
<td>public String</td>
<td>Used to point to the path of all our FTSE100 historical data.</td>
</tr>
<tr>
<td>STOCKS</td>
<td>private int</td>
<td>Used to set the number of stocks in the FTSE100 universe</td>
</tr>
<tr>
<td>FACTORS_NUM</td>
<td>private int</td>
<td>Used to set the number assessment factors to fixed number 22.</td>
</tr>
<tr>
<td>Fitness_Mode</td>
<td>private char</td>
<td>Used to indicate the fitness mode: S for single objective and M for multiobjective.</td>
</tr>
<tr>
<td>Mode</td>
<td>private char</td>
<td>Used to indicate the GP cycle: T for training and V for validation cycle.</td>
</tr>
</tbody>
</table>
APPENDIX B

List of Figures:

Figure 1  A standard Genetic Algorithm Model.
Figure 2  Shows the K-Nearest Neighbour technique used to calculate the Euclidian distance to in the population and archive space. This technique is used in the archive truncation.
Figure 3  Pareto-optimal Front for a 2-objective optimisation problem.
Figure 4  The interaction between ALPS and SPEA2 in the multiobjective GP system.
Figure 5  The overall system flowchart for both the GP system and the Investment Simulator.
Figure 6  An example of a single point crossover between two GP individuals.
Figure 7  The average number of best solutions per run for high ROI values for both algorithms. The combined system has outperformed the SPEA2 alone.
Figure 8  The normal distribution of the statistics mentioned in figure 7 with supporting indicators such as the standard deviation and the mean.
Figure 9  The diversity in the ROI for ALPS & SPEA2 versus SPEA2 only in the validation mode.
Figure 10 The normal distribution of the statistics mentioned in figure 9 with supporting indicators such as the standard deviation and the mean.
Figure 11 The diversity in the Standard Deviation for ALPS & SPEA2 versus SPEA2 only in the validation mode.
Figure 12 The normal distribution of the statistics mentioned in figure 11 with supporting indicators such as the standard deviation and the mean.
Figure 13 The diversity in the Sharpe Ratio for ALPS & SPEA2 versus SPEA2 only in the validation mode.
Figure 14 The normal distribution of the statistics mentioned in figure 13 with supporting indicators such as the standard deviation and the mean.
Figure 15 The generated ROI for ALPS & SPEA2 has dominated these for SPEA2 only in the validation mode.
Figure 16 The normal distribution of the statistics mentioned in figure 15 with supporting indicators such as the mean which indicates that the combined system has outperformed the SPEA2 only.
Figure 17 Significant improvements in the Sharpe Ratio of SPEA2 only system compared to the combined system in validation mode with archive file size =10
Figure 18 Significant improvements in the ROI for SPEA2 only system compared to the combined system in validation mode with archive file size =125
Figure 19 Significant improvements in the ROI of the combined system in validation mode with age gap size =15 compared to the SPEA2 only which has no improvement.
APPENDIX C

List of Tables:

Table 1       In this table, we show an example of different aging schemes along with their corresponding age limits in each layer for layers 0 to 6 for an age gap of 1.
Table 2       An example of the main structure of our FTSE100 stocks universe in the years 1995-2006 shows 1’s and 0’s under each year’s column to indicate if that stock is in the FTSE100 universe for that year.
Table 3       Mapping between the CSV columns and the terminals in a stock.
Table 4       The GP and IS control parameters used in our system.
APPENDIX D
System Manual

1. On the CD-ROM attached with this report, the folder ALPS&SPEA2 contains the following systems:

A. SPEA2: The SPEA2 implementation is a built-in extension within the standard ECJ package which located in the path ALPS&SPEA2/ecj/ec/multiobjective/spea2 with extensive comments to allow developers for extending the class to their needs.

B. ALPS: The ALPS implementation is located within the standard SPEA2 folder which is located in the path ALPS&SPEA2/ecj/ec/multiobjective/spea2 with extensive comments to allow developers for extending the class to their needs. In order to implement the ALPS code within ones own system, this folder and the parameter files listed below are all needed:

1. For training mode: ALPS&SPEA2/ecj/ec/app/Invest/train_multi.params

2. For validation mode: ALPS&SPEA2/ecj/ec/app/Invest/validate_multi.params

C. The Investment Simulator used for the both system is represented by the Invest Class and path ALPS&SPEA2/ecj/ec/app/Invest with extensive comments to allow developers for extending the class to their needs. The CSV files of all the 77 FTS100 stock data can be accessed by following these instructions:

1. Access the ALPS&SPEA2 directory on the CD.

2. Click on the ALPS&SPEA2/ecj/ec/app/Invest_Data/CompanyData folder.

3. All CSV files used in this project are held within this folder where the main file “All FTSE100.csv” must be accessed first to enable the IS to read each
stock’s contents afterward. An example of stock’s contents is listed in the Appendix E.

D. The JFreeChart implementation is located within the standard ECJ folder which is located in the path ALPS&SPEA2/ECJ/EC/JFreeChart. In order to implement the JFreeChart code within ones own system, this folder is needed.

2. Installing and running our system:

The following configurations are required for our system to work:

1. Install the Sun Java 2 JDK 6.0 on the target computer with default settings.
2. Install the Eclipse SDK version 3.2 on the target computer with default settings.
3. Set the default workspace as the ALPS&SPEA2 folder on the CD.
4. Create a new project and type ECJ as the name of the project.
5. Run the Eclipse program and import ALPS&SPEA2 from the CD.
6. Import the CSV manager library from the mylibs folder on the CD by right-clicking on the name of the project in the project explorer, select properties, and click Java Build Path. Select Add External Jars and add the csvman.jar file located in the ALPS&SPEA2 folder.
7. The path to our CSV files is found in the train_multi.params represented by the field DataPath which is located in the path ALPS&SPEA2/ecj/ec/app/Invest.
8. Once the above steps are completed, Select Run on the Eclipse Toolbar (or the main menu) and choose train_multi.params file for training mode and validate_multi.params for validation mode.
9. Our system will generate 9 statistical files in training mode and 3 files in validation mode. A complete list of this these files are described in Appendix G.