MSc in Data Science and Analytics

Dissertation

Improving Semantic Diversity in Multi-Objective Genetic Programming

Author

Fergal Stapleton

Supervisor:

Dr. Edgar Galván

A thesis submitted in fulfillment of the requirements for the degree of MSc in Data Science and Analytics for the Department of Computer Science.

February 4, 2021

Abstract

Semantics in the context of Genetic Program (GP) can be understood as the behaviour of a program given a set of inputs and has been well documented in improving performance of GP for a range of complex problems. There have been a variety of different methods which have incorporated semantics into GP but a focal point for this research is the work by Galván et al. (2020) which uses a individual from a sparse region of the search space and calculates the semantic distance between this reference point and every other individual in the population. This distance is then used a an additional objective to optimize in a Multi-Objective Genetic Program (MOGP). This dissertation will serve to analyse this semantic distance as well a produce an update to this distance calculation.

Another key aspect of this dissertation is to incorporate semantics into a multi-objective evolutionary algorithm that previously has not utilized semantics before. During the last number of years the primary focus of research in semantic diversity in Multi-Objective problems has centred around frameworks which use Pareto dominance fitness-assignment where the objectives are considered separately. MOEA/D differs from these methods in that an aggregated scalar function is used to measure fitness, where the problem at hand is decomposed into a number of sub-problems. It is hoped that incorporating semantics into MOEA/D will open up a new avenue of research.

1 Acknowledgements

I like to thank Dr. Edgar Galván for guidance and support throughout this dissertation. I would also like to thank Fred Valdez for his help in the initial stages of the project regarding feature extraction from images.

Additionally I wish to acknowledge the Irish Centre for High-End Computing (ICHEC) for the provision of computational facilities and support.

Contents

2 Introduction

Genetic Programming is a sub-field of evolutionary algorithms that use genetic operations which are analogous to behavioural biology to evolve programs to solve a problem. [2] Genetic operations include crossover which combines sub-trees of parent programs and mutation which alters a small part of a program trees structure to produce new offspring. A fitness criteria is used to determine whether these changes are beneficial or not and determines if offspring should be retained into the next generation. The range of problem domains for GP are wide and GP has been found to be beneficial for problems with multiple local optima and for problems with a varying degree of complexity.

The goal of multi-objective optimization (MOO) is to find the optimal solution to a problem via the simultaneous consideration of multiple objectives. Often these objectives will work against each other when undergoing optimization, in other words if we were to consider a two class system, seeking to maximize the objective of one class may result in a minimization of another and vica versa. Therefore we need to incorporate a methodology which would promote maximization for all objectives.

A key area of study in single and multi-objective GP is diversity and can have a profound effect on an algorithms ability evolve. Diversity can be understood as a measure of how similar programs are within a given population and programs which are found to be too similar have a tendency to produce inferior offspring during genetic operations.

2.1 Project Objectives

The main cause of the loss of diversity occurs as a result of the population converging to local optima which in turn reduces the GP ability to find better solutions. Essentially the number of high ranking programs (programs that will contribute favourably under genetic operations) are reduced within the population and resultant genetic operations fail to produce offspring that diverge from the local optima.

Originally diversity measures were handled primarily by the architecture of the programs under consideration where structural changes were made to control the size, shape, primitive and terminal set in order to promote diversity. These measures are known as structural diversity. Semantic diversity deals with the behavioural aspects of a GP in that it's analyses how similar the outputs for different program are given a set of inputs. The goal and outcomes of this dissertation are therefore as such;

- To conduct an analysis on Semantic Distance when used as on Objective. The analysis showed that the semantic distance was not an objective in conflict.
- To improve semantic diversity in previously reported methods [1] by using an updated semantic distance calculation. The new calculation was found to be problem dependant on whether it made significant improvements over the original calculation but gave more insight into how semantic distance was preferenced by NSGA-II when used as an objective. The new calculation was found to preform significantly better over canonical methods.
- A new method is proposed which incorporates semantics into a multi-objective evolutionary algorithm with decomposition (MOEA/D). Results showed that incorporating semantics improved performance when compared to canonical MOEA/D.

2.2 Outline of Thesis

Section 3.2 deals with the basics of Genetic Programming focusing particular on aspects and methods used during experiments Section 3.3 first deals with Pareto dominance and Crowding distance before going into more detail on the canonical methods; NSGA-II, SPEA2 and MOEA/D. Section 3.4 first gives a formal definition of semantics before detailing specific strategies that have previously been used by other researchers. Section 3.5 details the fitness classification of our data. An importance metric for comparing performance, the hypervolume is also discussed. Section 3.6 covers the data used in the experimentation.

Section 4.1 describes some of the motivation behind updating semantic distance in previously discussed methods. Section 4.2 to Section 4.4 deals with implementation and set up of experiments.

Section 5.1 derives analysis from the semantic distance when used as an objective. Section 5.2 then compare 3 different methods that use semantics as an objective. On the back of this work further analysis is done comparing these three methods in Section 5.3. Finally the results of MOEA/D-TCH SSC are discussed in Section 5.4 before making concluding remarks and suggestions for future work.

3 Background Literature

3.1 Relevant Literature

Genetic programming (GP) was popularized in the early 1990's by Koza (1992). Since then GP has developed as field with many branching methodologies. A comprehensive book *A field guide to genetic programming* Poli et al. (2008) has been used as primary source for covering the basics of GP in this dissertation. Two popular methods in tackling Multi-objective problems are discussed in this paper; the fast and elitist non-dominated sorting algorithm (NSGA-II) by Deb K. et al. (2002) and strength Pareto evolutionary algorithm (SPEA2) by Zitzler et al. (2001). Another method, multi-objective evolutionary algorithm with decomposition (MOEA/D) was proposed by Zhang et al. (2007). The same authors (2008) noted that for certain problems with complex Pareto fronts, MOEA/D outperformed NSGA-II.

This dissertation focuses on the use of semantics in MOGP. A primary reference is the work of Galv́an et al. (2020) which uses semantic distance as an objective to optimize. Other work on semantics that is of particular interest is Uy et al. (2011) who used semantics in an expensive crossover technique to aid diversity. The topics discussed thus far have only been briefly touched upon and will discussed in greater detail in the following sections.

3.2 Genetic Programming

3.2.1 Basics

Genetic Programming (GP) takes a population of initially randomized programs which are evolved over a set number of generations via genetic operations known as crossover and mutation in order to solve a pre-defined problem. The most common representation of a GP is as a tree based structure consisting of leafs and nodes. This tree structure of leafs and nodes is defined by what are known as functions and terminals. The function set is the list of arithmetic operators used by the GP and are assigned at the non-terminal nodes, $\Re = \{+, -, *, \%\}$. The terminal set can be defined by constants and variables at the leafs of the GP tree. The set of all permissible functions and terminals is known as the primitive set. Without detailing any specific framework we can now give a very basic overview of a typical GP algorithm;

Algorithm 1 Basic GP

1: *Initialize* a population of randomized programs using a predefined primitive set.
2: **while** stopping criteria not met **do**
3: *Evaluate fitness* of each program.
4: *Select* programs to use for genetic operations
5: *Create* new offspring population via genetic operations.
6: **end while**
7: **return** Best program

When initializing the population of programs there are three popular methods in determining it's structure Full, Grow and Ramped half-and-half. If we define the depth of a tree as the depth of it's deepest leaf, then the Full method initialises the programs so that their leaves they are all the same depth. This method requires all terminals to be filled before terminating. The grow method on the other hand terminates once terminals filled for a sub tree which has reached the max depth. The Ramped half-and-half method is a compromise between these two methods where half the population is created using the Full method and the other half with Grow.

3.2.2 Tournament Selection

Tournament selection is used to determine which individuals we wish to perform genetic operations on. A set number of individuals from the population are picked at random and compared against each other to find a parent program (This process is repeated twice for crossover as two parents are required). In order to insure the selection process is not overtly biased, tournament selection only determines if a program preforms better but not by how much. This prevents loss of diversity as a particular good program will not over dominate in subsequent generations.

3.2.3 Crossover

Conceptually crossover is analogous to biological sexual reproduction where the genetic material of an offspring is a composite of it's parents genetic material. Crossover deals with the creation of child programs based on two parent programs. Copies are made of the parent programs for the crossover operation as the parent programs may still be retained for further crossover operations. A node for each copied parent program is independently selected and is referred to as the cross-over point. A child program is

created by first discarding the sub-tree of one parent at the cross-over point and retaining the root tree before the cross-over point of the other and combining the remaining the tree structures at the cross-over point. This is demonstrated below;

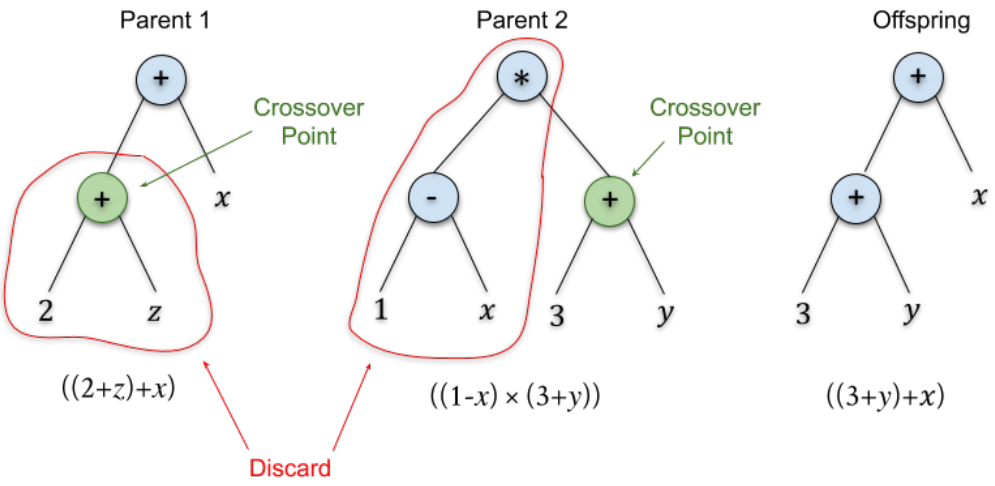

Figure 1: Diagram showing how crossover operation creates an offspring program

An extension of this method is to create a second child program where instead of discarding the unused tree structures a new tree structure is formed. As a result for every two parent programs we will have two offspring programs. This method of crossover is known as single point crossover.

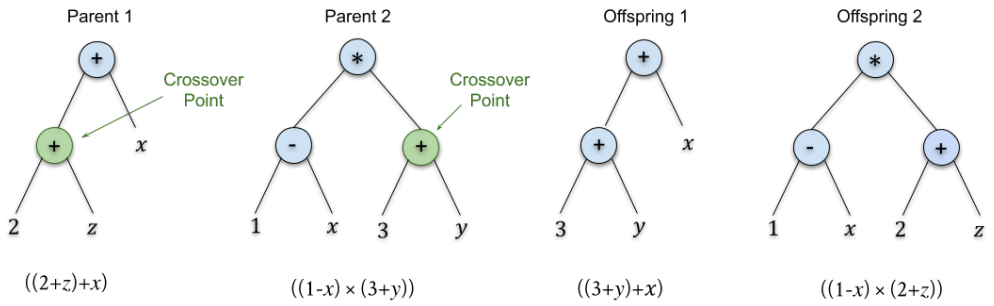

Figure 2: Diagram showing how crossover operation creates two offspring programs through single point crossover

The crossover operation can sometimes lead to predominantly leaf nodes being chosen as cross-over points which results in a low level of genetic material being passed on into future populations. To counteract this one can insure as a prerequisite that 90% of the cross-over points are functional nodes and the remaining 10% as leaf nodes.

3.2.4 Mutation

One of the most common forms of mutation is subtree mutation. Subtree mutation creates a new offspring to be retained into the next generation by randomly selecting a mutation point and replacing the sub-tree connected to this point with another randomized subtree.

3.2.5 Other considerations

Bloat can be defined as program growth with no significant increase in fitness [3], this occurs after successive crossover operations and tends to increase in occurrence as the algorithm progresses through newer generations. Programs that exhibit bloat tend to generalize solutions poorly as they move away from succinct mathematical representation (i.e parsimonious) and can result in over fitting, though this can be problem dependant as some problems may require more complexity. Another drawback of bloat is that larger programs can be more computationally expensive thus requiring more resources for the evaluation of their fitness.

Another importance consideration is that of closure. Closure can be separated into two aspects type consistency and evaluation safety. Type Consistency checks that insure the correct number of inputs for operations are chosen, e.g for multiplication we require two variables/constants, for instance $x_1 * x_2$ has two inputs (not 1 or more than 2). Evaluation Safety insures that operations are executable, for instance the protected division operator % which checks if the denominator is equal to zero and if it is sets the output to zero.

3.3 Canonical Methods

NSGA-II and SPEA2 are methods that attempt to approximate what is known as the Pareto-optimal set. Before discussing the specifics of these approaches we will define what the Pareto-optimal set is by first discussing Pareto Dominance relation. Additionally we will briefly discuss another important metric known as the Crowding Distance.

3.3.1 Pareto dominance

In general terms a multi-objective problem seeks to find a solution that either maximizes or minimizes a number of objectives. In the case of maximization this can be represented mathematically as;

$$max(f_1(x), f_2(x), ..., f_k(x)) \qquad s.t. \;\; x \in X, \tag{1}$$

Where $f_i(x)$ represents the i^{th} objective function and k $\geqslant$ 2. Typically there will not exist a unique solution that will maximize all objective functions. A candidate solution is Pareto dominant if its fitness is better or equal for all objectives and is strictly preferred by at least one in the search space. This can be formally represented mathematically by;

$$S_i \succ S_j \leftrightarrow \forall_m[(S_i)_m \geqslant (S_j)_m] \wedge \exists k[(S_i)_k > (S_j)_K] \tag{2}$$

Where $(S_i)_m$ is the i^{th} solution for objective k and $S_i \succ S_j$ denotes that solution i is non-dominated by solution j. A candidate solution is considered Pareto optimal if is

not dominated by any other candidate solution. In other words, if none of the objectives for a candidate solution can be improved without degrading at least one of the other objectives it can be considered Pareto optimal. For multi-objective problems there may exist a number of non-dominated solutions. The set of non-dominated candidate solutions for an MO problem is referred to as the first Pareto frontier when represented in objective space. In practise it is not always possible to do an exhaustive search for the true Pareto optimal set and as such this is something we seek to approximate instead.

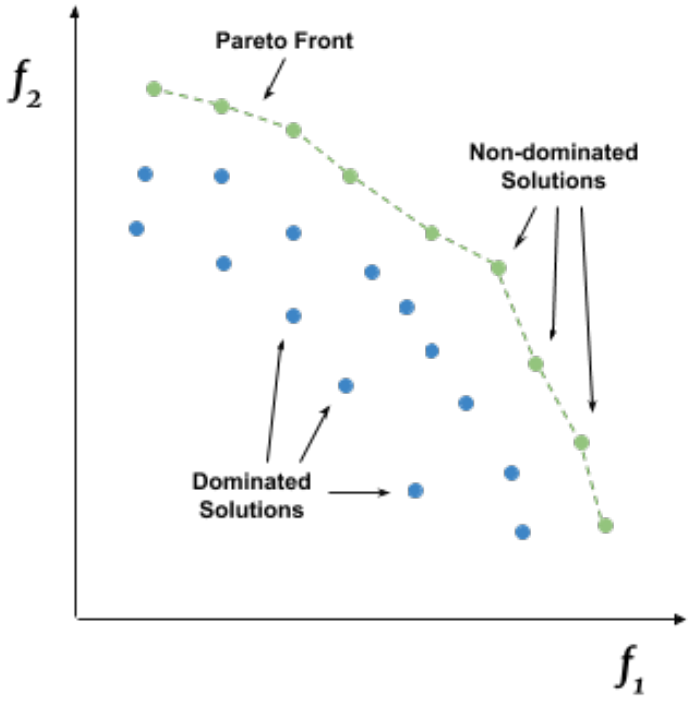

Figure 3: Diagram showing dominated and non-dominated solutions. Non-dominated solution lie along the Pareto front.

Pareto dominance relation is an integral part of MOEAs and has allowed practitioners and researchers to form important metrics in the selection process of these algorithms. Two such metrics are dominance rank and dominance count. Dominance rank is used as a fitness measure and calculates how may other solutions a candidate solution is dominated by. The lower the dominance rank the better with the lowest dominance rank of

0, i.e a solution that is not dominated by any other solution. Dominance rank can be expressed mathematically as seen in equation 3

$$S(i) = |\{j | j \in Pop \wedge S_j \succ S_i\}| \tag{3}$$

Where |.| represents the cardinality of the set. This criteria is utilized in NSGA-II. Dominance count calculates how many individuals a candidate solution dominates [1]. The higher the dominance count the better. Dominance rank and Dominance count are both used in SPEA-2.

$$S(j) = |\{j | j \in Pop \wedge S_i \succ S_j\}| \tag{4}$$

Leading on from the previous equation we can get a measure of fitness by summing the fitness of all individuals such that [4];

$$R(i) = \sum_{j \in P_t + \bar{P}_t,\ j \succ i} S(j) \tag{5}$$

3.3.2 Crowding distance

Solutions are ranked relative to each other according to a metric known as the crowding distance. The crowding distance is used to compare any pair of solutions in search space and is used in NSGA-II and SPEA2 as Pareto Dominance alone only acts as a partial order of the solutions. The crowding distance calculation is comprised of three parts;

- Initialize the distance d to zero.
- Set the boundary solutions to *inf.* These solutions are always selected due to this constraint.
- Calculate the average distance differences for an individual against it's two nearest neighbours using the Manhattan distance.

$$d = d + \frac{|f_{r+1}^{(c)} - f_{r-1}^{(c)}|}{|f_{max}^{(c)} - f_{min}^{(c)}|} \tag{6}$$

Where c denotes the class in question. Solutions with the highest crowding distance are considered better solutions, in other words the algorithm preferences localities along the Pareto front which are more sparsely populated with solutions than those which are more dense. In this manner the crowding distance resolves which solutions to retain when programs produce very similar fitness values. Figure 4 represents how the crowding distance is calculated in objective space.

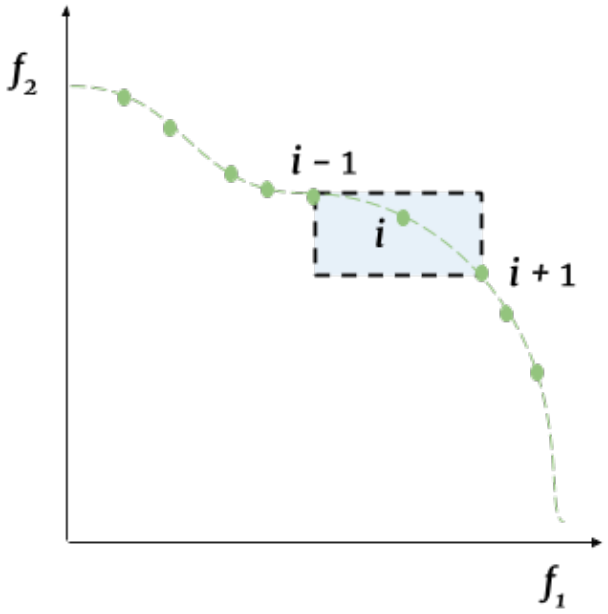

Figure 4: Showing cuboid of crowding distance

3.3.3 NSGA-II

Elitist non-dominated sorting algorithm II is a popular MOGP algorithm consisting of 3 main components; fast non-dominated sort which assigns a Pareto ranking to our solutions, crowding distance assignment and the crowding comparison operator. Elitism simply refers to the algorithms ability to retain the best performing individuals into the next generation.

Algorithm 2 Psuedocode for NSGA-II

1: $R_t = P_t \cup Q_t$
2: $\mathcal{F} = fast\text{-}non\text{-}dominated\text{-}sort(R_t)$
3: $P_{t+1} = \varnothing$
4: $i = 1$
5: **while** $|P_{t+1}| + |\mathcal{F}_i| \leqslant N$ **do**
6: $\quad crowding\text{-}distance\text{-}assignment(\mathcal{F})$
7: $\quad P_{t+1} = P_{t+1} \cup \mathcal{F}_i$
8: $\quad i = i + 1$
9: **end while**
10: $Sort(\mathcal{F}_i\ ,\ \prec_n)$
11: $P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$
12: $Q_{t+1} = make\text{-}new\text{-}pop(P_{t+1})$
13: $t = t + 1$

The algorithm can be broken into a number of steps;

- The parent and offspring population are combined into a single population R_t of size 2N.
- The combined population is sorted using the fast-non-dominated-sort function creating output of non-dominated sets $\mathcal{F}$.
- The next parent population P_{t+1} is initialized as a null set and i is set to 1. When $\mathcal{F}_i$ is smaller than the population size N all members are assigned to the population P_{t+1} with the remaining population being filled by successive non-dominated fronts based on their ranking. In other words, while the criteria $|P_{t+1}| + |\mathcal{F}_i| \leqslant N$ has not been met, P_{t+1} will be filled with population members based on their Pareto rank with $\mathcal{F}_1$ being chosen first, $\mathcal{F}_2$ chosen next on so on until no more sets can be used to fully fill the population.
- Typically the last Pareto set $\mathcal{F}_l$ will not fully fill the population and so is sorted based on crowding distance from equation 6 and the remaining population members are filled using $\mathcal{F}_i[1 : (N - |P_{t+1}|)]$.
- Now that we have our new parent population P_{t+1} we can create our offspring population Q_{t+1} using mutation and crossover and move onto the next generation t + 1.

A summary schematic of the NSGA-II procedure can be seen in figure 5. In this example candidate solutions for fronts F4 and F5 are rejected immediately after the non-dominated sort. A further crowding distance sort is done on F3 to select which solutions to retain into P_{t+1} and which to reject.

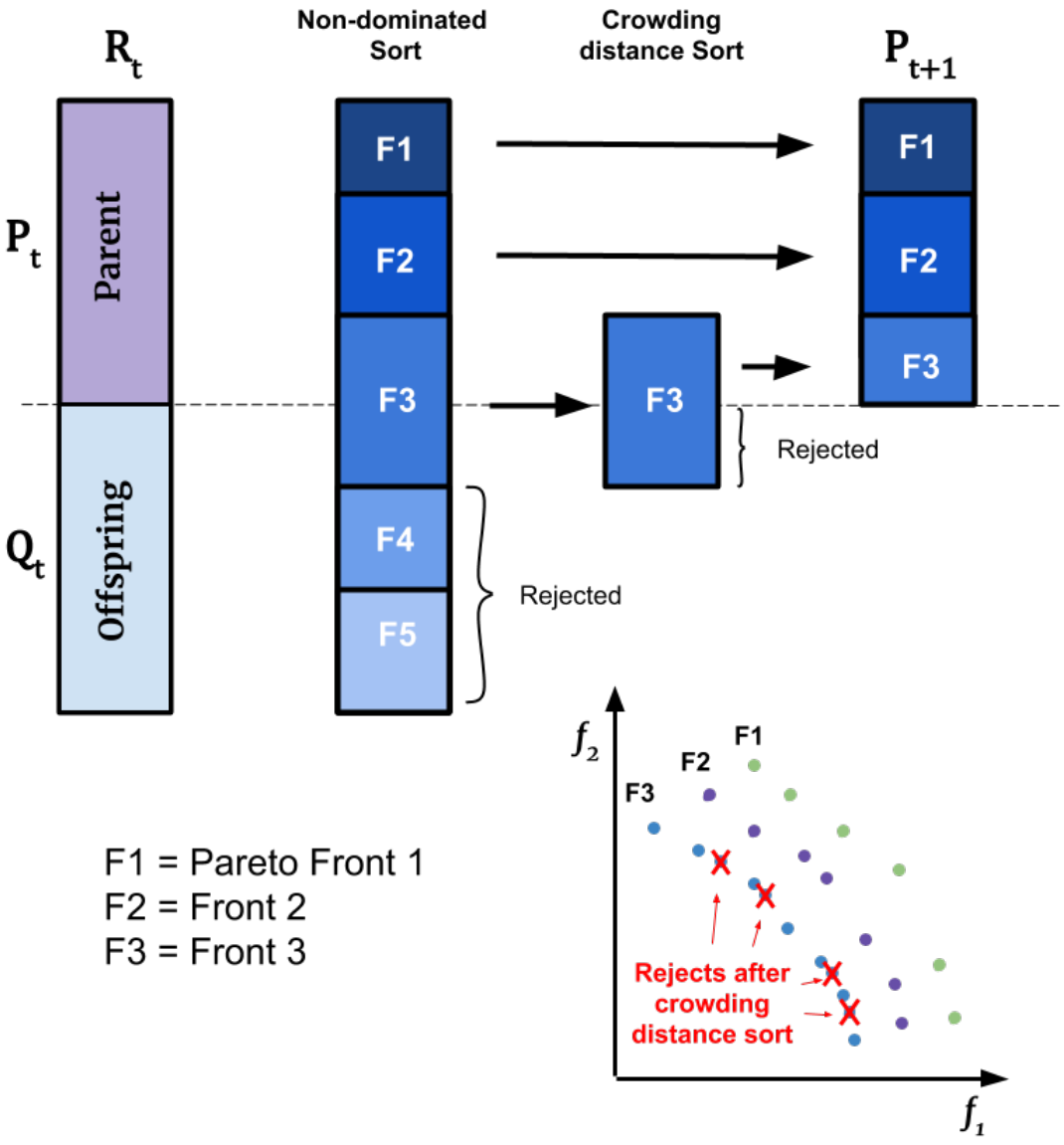

Figure 5: Diagram NSGA-II algorithm.

3.3.4 SPEA2

The Strength Pareto Evolutionary Algorithm 2 (SPEA2) proposed by Zitzler et al (2001) [6] is another method which seeks to approximate the Pareto-optimal set. SPEA2 uses an external archive population. When the number of non-dominated solutions is less than the archive size, the best dominated solutions from the parent population are used to fill the archive. Individuals may dominate the same archive members and as such have the same fitness values. To resolve this potential issue SPEA2 uses both dominance rank and dominance count. If the number of solutions is greater then a truncation measure is taken that iteratively removes individuals that have the shortest distance to their nearest neighbour. The steps and pseudocode (shown in Algorithm 3) for SPEA2 are as follows;

Step 1: Initialize population P_0 and set archive population $\bar{P}_0 = \varnothing$

Step 2: Loop over each generation and find the fitness for each individual in populations P_t and $\bar{P}_t$

Step 3: Copy all of the non-dominated individuals in P_t and $\bar{P}_t$ into $\bar{P}_{t+1}$.

Step 4: If $|\bar{P}_{t+1}| \geqslant \bar{N}$ truncate the set or if $|\bar{P}_{t+1}| \leqslant \bar{N}$ fill the set using both dominance rank and dominance count to determine retention.

Step 5: Crossover and mutation are performed on to $\bar{P}_{t+1}$ create P_{t+1} the population to be retained into the next generation.

Algorithm 3 Psuedocode for SPEA-2

1: $\bar{P}_0 = \varnothing$
2: $t = 0$
3: *stop-criteria* $= 0$
4: **while** *stop-criteria* $\neq 1$ **do**
5: $\quad \bar{P}_{t+1} = fitness\text{-}dominance(P_t\ ,\ \bar{P}_t)$
6: $\quad$ **if** $|\bar{P}_{t+1}| > \bar{N}$ **then**
7: $\quad\quad$ **while** $|\bar{P}_{t+1}| \neq \bar{N}$ **do**
8: $\quad\quad\quad \bar{P}_{t+1} = truncate(\bar{P}_{t+1})$
9: $\quad\quad$ **end while**
10: $\quad$ **else if** $|\bar{P}_{t+1}| < \bar{N}$ **then**
11: $\quad\quad$ **while** $|\bar{P}_{t+1}| \neq \bar{N}$ **do**
12: $\quad\quad\quad \bar{P}_{t+1} = fill(\bar{P}_{t+1})$
13: $\quad\quad$ **end while**
14: $\quad$ **else**
15: $\quad\quad pass$
16: $\quad$ **end if**
17: $\quad$ **if** $t \geqslant T$ **then**
18: $\quad\quad A_{t+1} = \bar{P}_{t+1}$
19: $\quad\quad$ *stop-criteria* $= 1$
20: $\quad$ **else**
21: $\quad\quad \bar{P}_{t+1} = mating\text{-}selection(\bar{P}_{t+1})$
22: $\quad\quad P_{t+1} = variation(\bar{P}_{t+1})$
23: $\quad$ **end if**
24: $\quad t = t + 1$
25: **end while**

3.3.5 MOEA/D

So far we have discussed two domination-based fitness assignment strategies; NSGA-II and SPEA2. Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D) differs in that it decomposes the MOP into a subset of scalar optimization problems [7]. It is important to note that with MOEA/D we still wish to approximate the Pareto-optimal front but instead of using dominance to determine the fitness of our solutions we use a scalar value aggregated from multiple objectives method [8]. In the original paper

by Zhang et al. (2007) proposed three scalarization methods; weighted sum, Tchebycheff and Penalty-based Boundary Intersection (PBI). As the weighted sum method was reported to poorly approximate concave fronts in their entirety and as the PBI method incorporates it's own diversity preserving measure, this dissertation will focus primarily on the Tchebycheff approach.

$$\min(g(x|\lambda)) = \max_{1 \leqslant j \leqslant m} \{\lambda \dot{|} f_j(x) - z_j\} \tag{7}$$

λ is a weight vector that is assigned to each sub problem and represents a search direction in objective space. Z_j is the ideal point and represents the ideal solution for a given problem.

Step 1: Initialization

Step 1.1) Initialize external population $EP = \varnothing$.

Step 1.2) Calculate the Euclidean distance between any two weight vectors and find the T closest weight vectors to each respective weight vector. For each i = 1, 2... N, set $B(i) = i_1, i_2, ..., i_T$ where $B(i)$ can be understood as a neighbourhood reference table of indices and where $\lambda^{i_1}, \lambda^{i_2}, ..., \lambda^{i_T}$ are the T closest weight vectors to λ^i.

Step 1.3) Randomly create the initial population $x^1, x^2...x^N$ and set the fitness value $FV^i = F(x^i)$.

Step 2: Update For i = 1, 2, ... , N; do the following steps

Step 2.1) Select two indices k and l randomly from the neighbour hood reference table B(i) and generate new offspring y from parents x^k and x^l by apply genetic operations.

Step 2.2) An optional problem-specific repair and improvement heuristic on y to produce y', otherwise let y = y'.

Step 2.3) Update z such that for each j = 1, 2, ... , m if z_j ¡ $f_j(y')$, then set $z_j = f_j(y')$. In the case where objective is to minimize F(x) then this inequality should be reversed.

Step 2.4) Update the neighbouring solutions for the i^th case such that for $jB(i)$, if $g(y'|\lambda^j, z) \leqslant g(x^j|\lambda^j, z)$ then let $x^j = y'$ and the fitness value $FV^j = F(y')$.

3.4 Semantics

In general terms semantics can be understood as a measure of how similar two programs are in a population relative to their behaviour. Two individuals may have different tree structures but behave very similarly, i.e return fitness values for each objective which are the same. [12] For example if we were to consider a simple symbolic regression problem where two candidate programs were of the form; $f(x) = 3 * x$ and $f(x) = x * x * x$ then the resultant fitness for these programs would invariably be the same and these programs would be considered semantically equivalent.

Pawlak et. al. gave a formal definition for program semantics in this context. [11]. Let $p \in P$ be a program from a given programming language P. The program p will produce a specific output $p(in)$ where input $in \in I$. The set of inputs I can be understood as being mapped to the set of outputs O which can be defined as $p : I \rightarrow 0$.

Def 1. Semantic mapping is a function $s : P \rightarrow S$ mapping any program p from P to the semantic space S where we can show the semantic equivalence of two programs;

$$s(p_1) = s(p_2) \iff \forall\ in \in I : p_1(in) = p_2(in) \tag{8}$$

This definition presents three important and intuitive for semantics:

1. Every program has only one semantic attributed to it.
2. Two or more programs can have the same semantics.
3. Programs which produce different outputs have different semantics.

In practice the set of inputs from which our fitness cases are derived are limited and as such def 1, is more of a generalization. For a truer representation of semantics we need to define semantics under the assumption of a finite set of fitness cases, where a fitness case is a pair comprised of a program input and it's respective program output I X 0.

Def 2. The semantics s(p) of a program p is the vector of values from the output set O obtained by running p on all inputs from input set I:

$$s(p) = [p(in_1), p(in_2), ..., p(in_l)] \tag{9}$$

Where $l = |I|$ is the size of the input set. Now that we have a formal definition in of semantics we can discuss the formal application of it in GP.

3.4.1 Semantic Similarity-based Crossover

Uy et. al. [10] developed an expensive crossover mechanism for incorporating improving semantic diversity known as semantic similarity-based crossover (SSC). SSC was the first notable example of semantics in single objective GP. SSC attempts to produce preferable offspring by insuring during the crossover operation that the two sub-trees are not semantically similar and also that they are not too semantically dissimilar. The absolute differences for every in ∈ I are calculated between parent and offspring and an average is calculated for these differences in order to obtain the semantic distance for the individual under consideration. If this semantics distance falls within the range of two threshold values (or above or below a single threshold value) then the new offspring is retained. However it may be difficult for the crossover operation to generate offspring which meet this criteria and as such crossover is applied for a predefined number of times until this criteria is meet. If the criteria is not met then crossover is applied in the normal fashion without considering semantics.

3.4.2 Semantic-based Crowding Distance

In this method the crowding distance as discussed in section 3.3 is replaced by by a semantic-based crowding distance. A pivot p is selected as the individual in the first Pareto front which is furthest away from all other individuals v in that front. This distance is calculated using the crowding distance as discussed previously. Once we have the pivot we can compute the semantic differences of the pivot against all the other individuals in the population. We can use upper and lower semantic similarity bounds to create our semantic distance as shown in equation or a single bound as like in equation

.

$$d(p_j, v) = \sum_{i=1}^{l} 1 \; if \; LBSS \leqslant |p(in_i) - v(in_i)| \leqslant UBSS \tag{10}$$

$$d(p_j, v) = \sum_{i=1}^{l} 1 \; if \; |p(in_i) - v(in_i)| \geqslant UBSS \tag{11}$$

3.4.3 Semantic Distance as an Objective

The semantic-bases crowding distance can also be used as an objective for MOGP to optimize. With the majority and minority class serving as the first two objectives to optimize semantic distance is also treated a third objective.

3.5 Fitness and Performance Measure

3.5.1 Classification Fitness

A common metric used in determining fitness for binary classification problems is to use classification accuracy; where $ACC = \frac{TP+TN}{TP+TN+FP+FN}$. However with imbalanced data sets using this accuracy measure will tend to bias towards the majority class. As such it is better to treat the minority and majority as two separate objectives where the goal is to maximize the number of correctly classified cases. This can be done using the true positive rate $TPR = \frac{TP}{TP+TN}$ and true negative $TNR = \frac{TN}{TN+FP}$ [14]

3.5.2 Hypervolume

Hypervolume (also referred to as the hyperarea) is calculated as the sum of the trapezoidal areas underneath the Pareto front and is used as a performance measure. Accessing the fit based on the hypervolume in case of objectives we wish to maximise is relatively straightforward to interpret. Typically the closer the point of inflection for the estimated Pareto front is to the top right hand corner the higher the hypervolume average will be. It is important to note however not all problems will result in Pareto fronts with a curve of this nature.

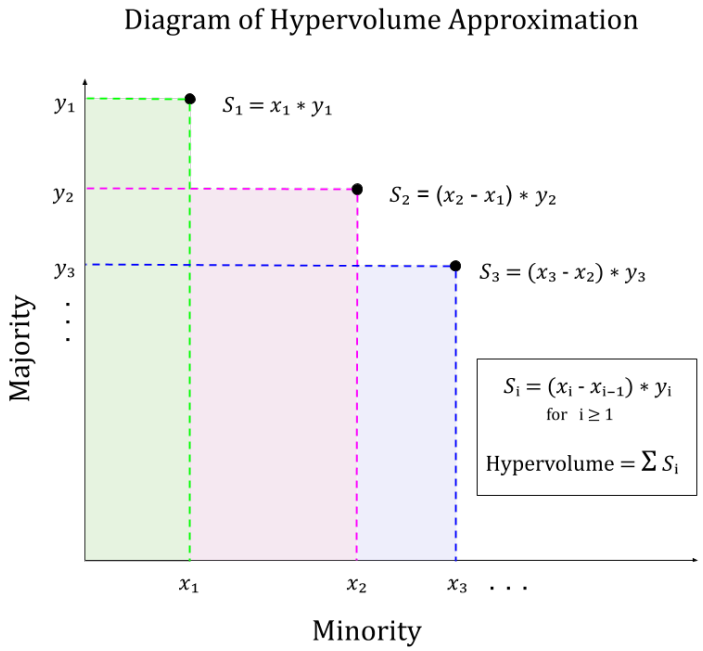

Figure 6: Diagram showing hypervolume calculation where each point represents the majority and minority fitness of a candidate solution.

3.6 Data

MNIST is a popular data set of hand written digits where the goal is to classify each digit correctly. In the following experiments we have taken a subset of the MNIST where we took a subset of the data with 6000 entries for each digit. Each digit is considered in isolation, where the digit under consideration was classified as 1 and all other digits were classified as 0. As a result the data itself is imbalanced with a ratio of 1:9. When splitting the training and test set this imbalance ratio was maintained for each set. Feature extraction was performed with by splitting each image into a series of boxes and returning the mean and standard deviation for each box.

The rest of the data set originate from the UCI Machine Learning repository [15]. The characteristics of these data sets vary greatly, with varying number of features and imbalance ratios. A brief description of each of these data sets is given below including the nature of their feature type followed by a table giving a breakdown of their imbalance ratios and number of features.

Data Sets	Cases	Positive	Negative	Imb. Ratio	#. of Features
MNIST_i	6000	600	5400	1:9	18
Ion	351	126	225	1:3	34
SPECT	267	55	212	1:4	22
Yeast_1	1482	244	1238	1:6	8
Yeast_2	1482	163	1319	1:9	8
Abal_1	731	42	689	1:17	8
Abal_2	4177	32	4145	1:130	8

Table 1: Table showing imbalance ratios and number of features for each data set

Ion Radar measurements of free electron signals in the Ionosphere. Binary response of either Good or Bad signal.

Spect Data set used to diagnose cardiac Single Proton Emission Computed Tomography images. Patients are classified into abnormal or normal cases.

Yeast_1 Deals with the cellular localisation site of protein sequence. Protein is classified as mitochondrial versus other.

Yeast_2 Also deals with the cellular localisation site of protein sequence. Protein is classified as membrane protein ME3 versus other proteins.

Abal_1 Data set used to determine the age of abalone shellfish via the number of rings on the shell after dissection. Classify shellfish with 9 rings versus 18 rings.

Abal_2 Data set used to determine the age of abalone shellfish via the number of rings on the shell after dissection. Minority class is for 19 rings versus all other ring counts.

All data sets were split 50/50 with half of the entries being attributed to the training set and the other half for the test set. The same class imbalance ratio is kept between the training and test set.

4 Description of Work undertaken

4.1 Pivot Similarity Semantic-based Distance as and Objective

In section 3.4.3 we discussed SDO which calculates the Semantic distance between each individual and a pivot. To this effect equations 10 and 11 were used to determine if the semantic difference between a pivot and the individuals fall within a predefined range.

There are two motivating reasons for why the Semantic distance from these equations ought to be reconsidered. The first reason will be explained with the aid of a simplified problem. Lets consider a problem where we only consider the semantics of two individual programs against a pivot.

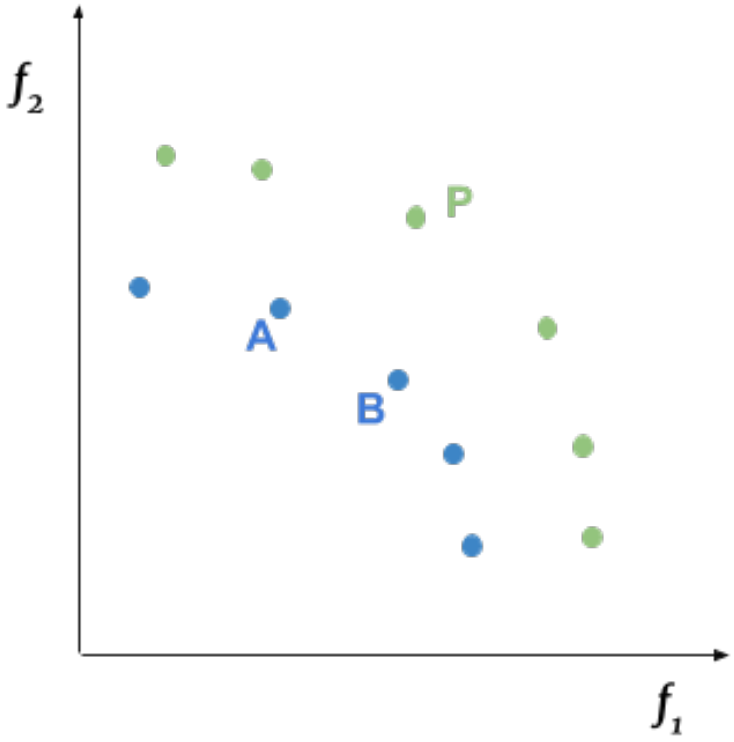

Figure 7: Diagram showing Pivot in a simplified example.

Let's now say we have only a few fitness cases we wish to use (In reality we would have typically have a much larger number of fitness cases), we chose a method that uses equation 11 with a UBSS value of 0.5 and we got an example output of;

Comparison	**Value**	**Sum**
$\lvert p(in_1) - A(in_1)\rvert$	0.6	1
$\lvert p(in_2) - A(in_2)\rvert$	0.7	1
$\lvert p(in_3) - A(in_3)\rvert$	0.8	1

Comparison	**Value**	**Sum**
$\lvert p(in_1) - B(in_1)\rvert$	0.6	1
$\lvert p(in_2) - B(in_2)\rvert$	0.0	0
$\lvert p(in_3) - B(in_3)\rvert$	0.1	0

This would give us $d(p_j, A) = 3$ and $d(p_j, B) = 1$. Now considering that in SDO we use the semantic distance as an objective.

Should we preference A whose semantics are greater for each input.

Or

Should we preference B whose semantic similarity to the pivot is closest, given that the pivot is approximated as the most diverse individual from the first front in search space.

The methods we discussed in section 3.4 would preference A. Another motivating factor for updating the semantic distance calculation is that as an objective the distance was found not to be in conflict. This is discussed in greater detail in section 5.1.

A proposed update in the distance calculation is given in equation 12, where the summed values is now taken away from the number of fitness cases. This method is referred to as Pivot Similarity Semantic-based Distance as and Objective (PSDO).

$$d(p_j, v) = l - \sum_{i=1}^{l} 1 \tag{12}$$

4.2 Implementation

Most of the coding implementation centred around updating a pre-existing MOGP framework that contained NSGA-II and SPEA2 with SDO and SSC methods readily available. In figure 8 a simplified version of the code Architecture has been presented. The classes highlighted in green represents the main code sections that have been added to from the pre-existing packages.

As part of our analysis (Sections 5.1 and 5.3) we plotted Semantic Distance versus majority and Distance Versus Minority. This required updating the code to output the distance into result files to be plotted. The distance was normalized to 1 by dividing by the number of fitness cases and was printed to the output .dat files.

For the results of tables 3 and 4 contain three methods that use three objectives. The first method SDO are results previously obtained by Galván et al. (2020) and have been reproduced with permission. The other two methods are PSDO and SSCDO. The PSDO method was incorporated into the pre-existing framework and new classes were created for running this independently. SSCDO required no additional coding work as this is essentially a combination of the SDO and SSC method. Extensive experiments were run for PSDO and SSCDO with over 10,000 independent runs. Each setting required 50 independent runs to insure statistical significance.

MOEA/D was a canonical method added to the code from scratch. Source code written Java by Wudong, L was a jumping point for integrating MOEA/D. Additionally this code was linked up with the SSC method for experiments run with the MNIST data. A list of the major files that were added or modified have been included in appendix A.

Additionally a method was created that used multiple pivots. Essentially the motivation here was to sample from various points in the search space and to compute the semantics difference against these points and all other individuals. This method was tested with the MNIST data set and compared along side NSGA-II, NSGA SDO and NSGA PSDO. The results of these can be found in appendix A and are mixed between all methods.

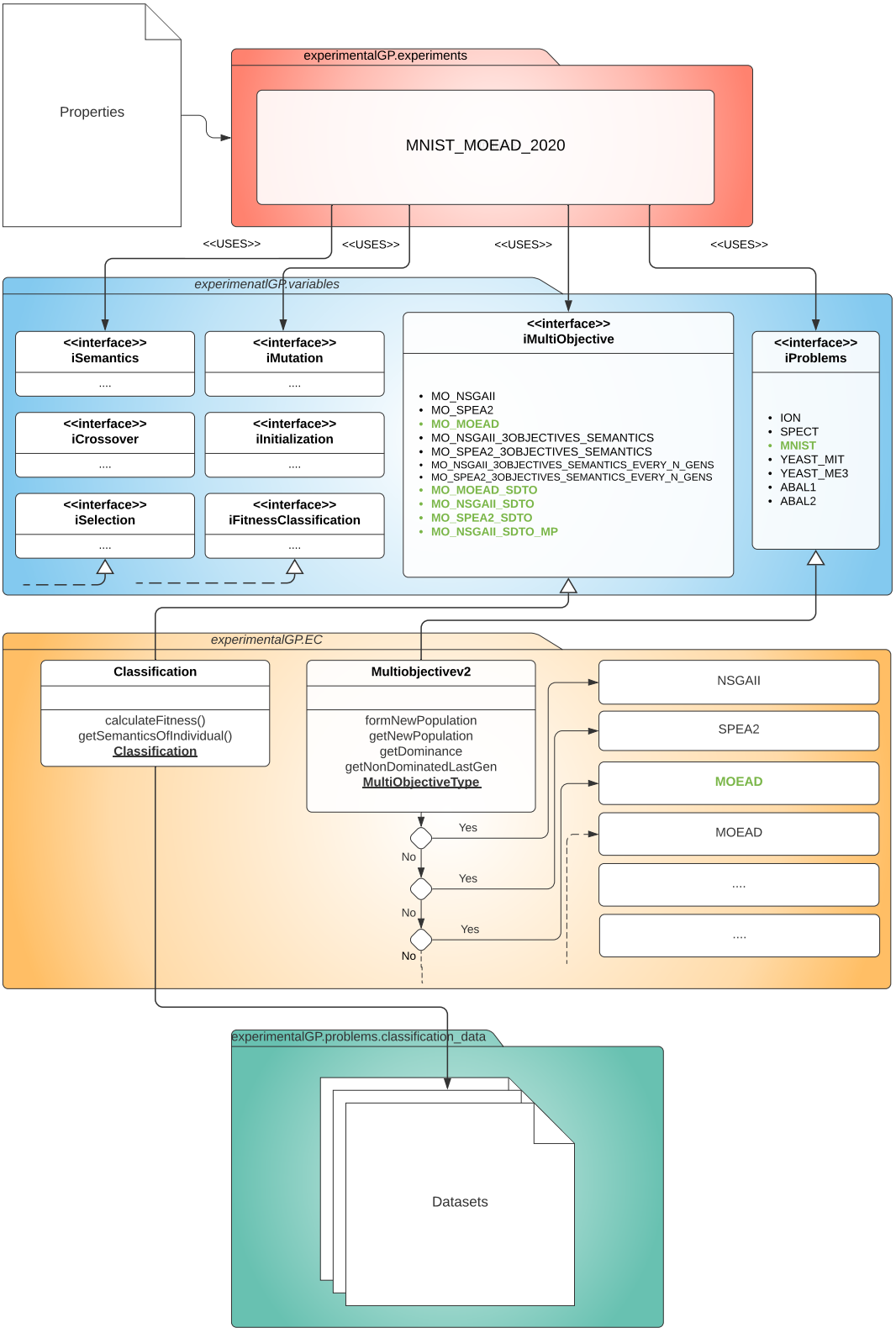

Figure 8: Diagram of code architecture

4.3 Experiments

Kay is a supercomputer operated by Irish Centre for High-End Computing and allows researchers access to high-performance computing technologies [17]. All experiments were conducted using the cluster partition of Kay and the system specifications of which are as follows;

- Architecture: Cluster of 336 nodes, with a total of 13,440 cores and 63 TiB of distributed memory.
- Processor: 2x 20-core 2.4 GHz Intel Xeon Gold 6148 (Skylake) for each node.
- Ram: 192 GiB.
- Network Adaptor: 100Gbit Omnipath.

A scheduling policy with a reference to available resources can be seen in the below table;

Scheduling Policy			
Queue type	Node Type	Max nodes per job	Max walltime
DevQ	Cluster	4	1 hour
ProdQ	Cluster	40	72 hours

DevQ was primarily used for testing while all reported experiments were done using ProdQ. 25GB of storage are available for the Users home directory which is where the experiment output files were stored [18].

To help manage the workload submitted jobs were run in parallel. Essentially each node in Kay comprises of 50 CPU's. There are a number of ways of running in parallel but one method was to create a unique property file for each run and assign one CPU to each of these property files. This would insure resources were being efficiently used and on average results for a particular data set would be produced roughly around the same time.

4.4 Running Experiments

Property files are contained at the level *ultimateExperimentalGP/*. This file specifies most of the properties we require to run our MO program. Also linked to this file is the ReadConfiguration.java file located in *experimentalGP.utils/* folder. This file needs to be updated for the Fitness Classification to work [line 250]. A description of the various properties is given below which have been covered in detail in the previous sections.

- **functionSpec** Defines our function set operators. For +, -, *, %,
- **FILE_SIMPLE_STATS** is used to specify the names of our output files. This is used with string comprehension to name of .dat outputs.
- **DEFPROBLEM** Defines which problem we are interested in. Each problem has unique code associated with it. For example MNIST dataset was given the unique code 181.
- **INITIALISATION** Defines tree growth, either Ramped, Full or Grow. Set to Ramped for all experiments.
- **POPSIZE** Population size of our individuals. Set to 500 for all experiments.
- **GENERATIONS** The number of generations we wish to run our GP for. Set to 50 for all experiments.
- **TSIZE** Tournament size. Set to 7 for all experiments.
- **TYPECROSSOVER** Defines the type of crossover we wish to use. Selected as single point crossover for all experiments.
- **TYPEMUTATION** Defines the type of mutation we wish to use. Selected as subtree mutation for all experiments.
- **INI_DEPTH** Used to define the initial tree depth upon initialization. Set to 1.
- **FIN_DEPTH** Used to define the final tree depth upon initialization. Set to 5.
- **MAXIMUM_DEPTH** Maximum depth of program trees in population
- **CROSSOVER_RATE** Determines probability of crossover being applied
- **MUTATION_RATE** Determines probability of mutation being applied

- **CONTROL_BLOAT** Do we wish to control bloat in our experiment. Defined as boolean true or false.
- **RUNS** How many times we will run our program. This is required to get accurate error rates for our experiment.
- **LEN_CONTROL_BLOAT** Controls the maximum length we allow individuals to grow. Set to 800 throughout.
- **MO** Specify if the problem is multiobjective. Defined as boolean true or false.
- **MO_TYPE** Defines which MOEA strategy to use.
- **NUMBER_OBJECTIVES** The number of objectives under consideration
- **DECOMP** Defines the decomposition method to use. 0 for Tchebychev and 1 for PBI (*source code for PBI has been implemented but no results have been gathered for this report*)
- **LBSS** Lower bound semantic similarity.
- **UBSS** Upper bound semantic similarity.

5 Analysis and Results

5.1 Semantic Distance as an Objective Analysis

A key aspect for this dissertation was to investigate semantics in the methods proposed by Galván et al. (2020). As part of the research for this dissertation the semantics distance as an objective has been further analyzed. A series of experiments were run for Ion, Spect, Yeast_1, Yeast_1, Abal_1 and Abal_2 with an UBSS value of 0.5 using just a single threshold as defined in equation 11. The semantic distance has been normalized and exported with the results along with majority and minority objectives. The full set of results can be found in appendix B. Figure 9 shows that the minority and majority class are in conflict. The last Pareto front is relatively smooth and exhibits the trade off in fitness for solutions along the x and y axis.

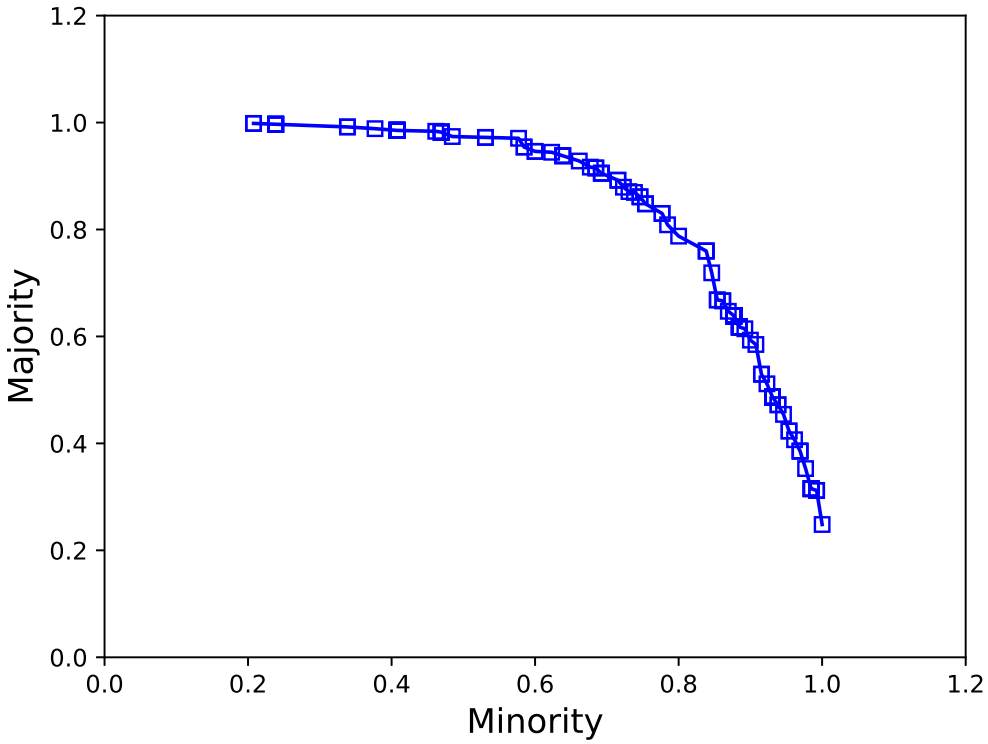

Figure 9: Plot of Majority vs Minority for Yeast_1 data using NSGA-II SDO

Figures 10 and 11 on the other hand show that semantic distance is not in conflict when compared against Minority and Majority objectives. This is observed clearly in the plots as there are solutions that ought to be dominated (and hence removed) and also by virtue of the high density of solutions close (1,1) in each plot. The high density of solutions close to this ideal point indicates that the condition of a trade off in objectives has not been met.

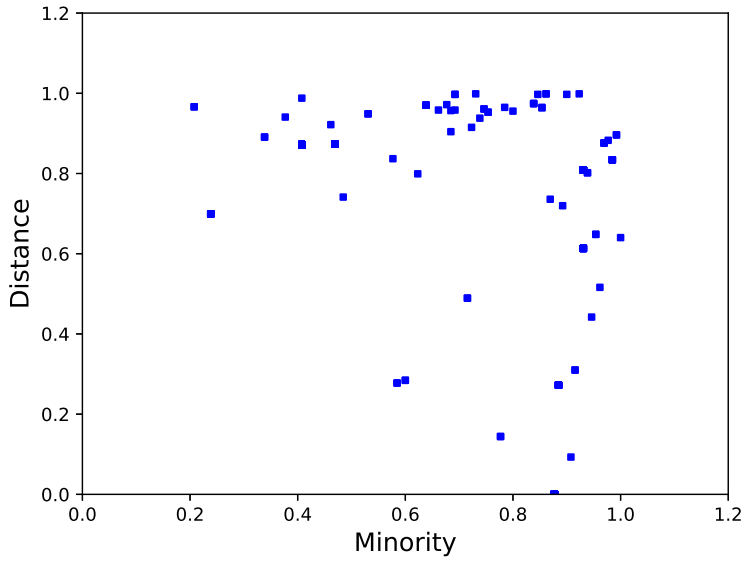

Figure 10: Plot of Distance vs Minority for Yeast_1 data using NSGA-II SDO

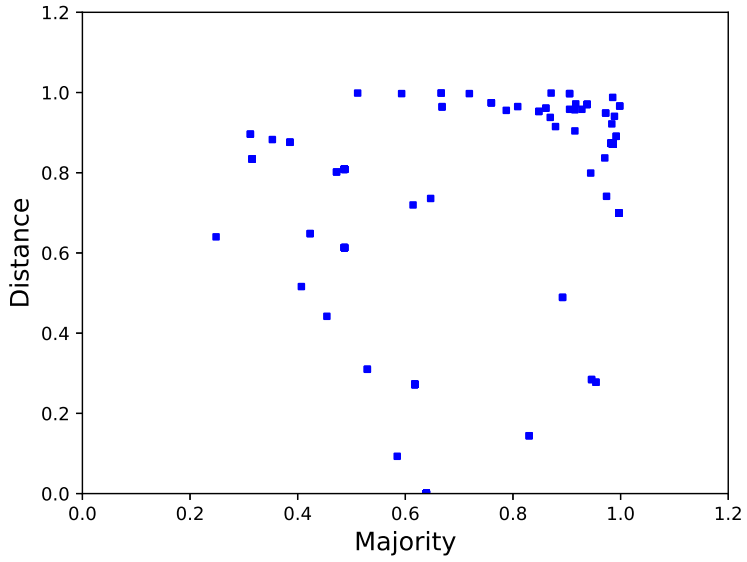

Figure 11: Plot of Distance vs Majority for Yeast_1 data using NSGA-II SDO

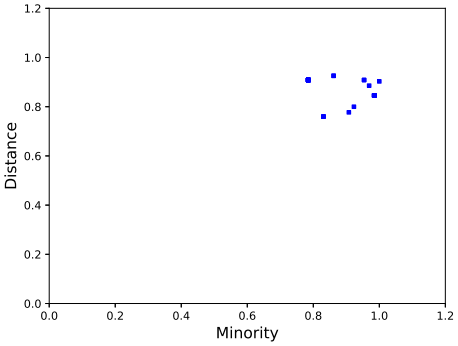

Figure 12: Plot of Distance vs Minority for Ion data using NSGA-II SDO

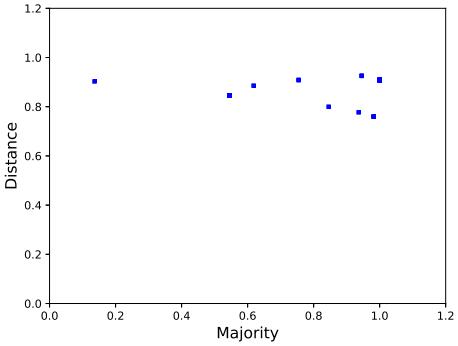

Figure 13: Plot of Distance vs Majority for Ion data using NSGA-II SDO

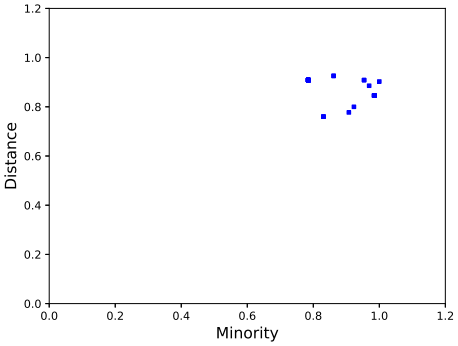

Figure 14: Plot of Distance vs Minority for Ion data using NSGA-II SDO

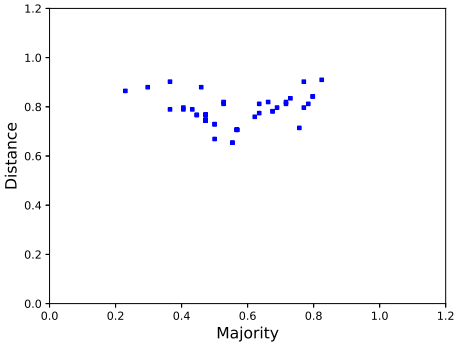

Figure 15: Plot of Distance vs Majority for spect data using NSGA-II SDO

While it is difficult to see a trend in the plot for Yeast_1, the rest of the plots show distinctly that higher semantic distances were preferred in the Last Pareto front for Ion and Spect. Plots 16 - 21 all appear to have formed two distinct clusters; one with relatively large semantic distance and the other with relatively small semantic distance.

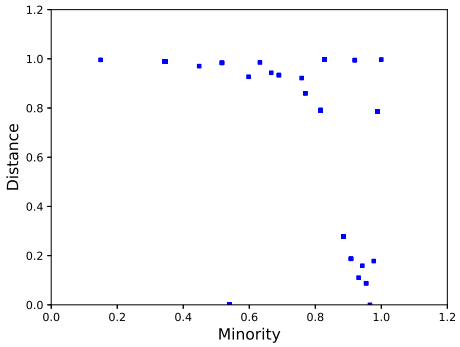

Figure 16: Plot of Distance vs Minority for Ion data using NSGA-II SDO

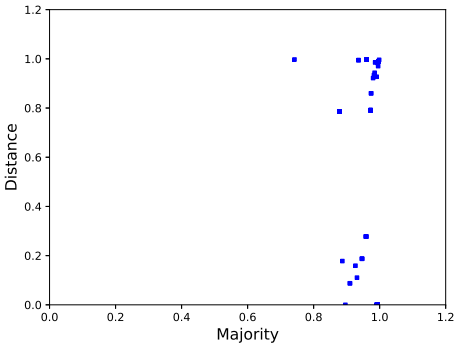

Figure 17: Plot of Distance vs Majority for spect data using NSGA-II SDO

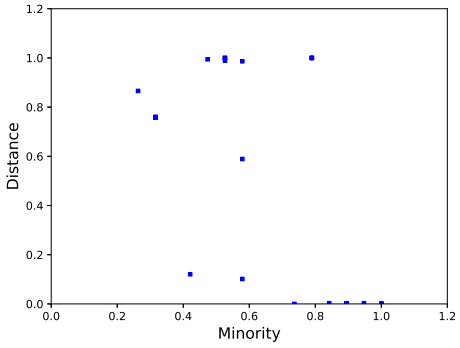

Figure 18: Plot of Distance vs Minority for Ion data using NSGA-II SDO

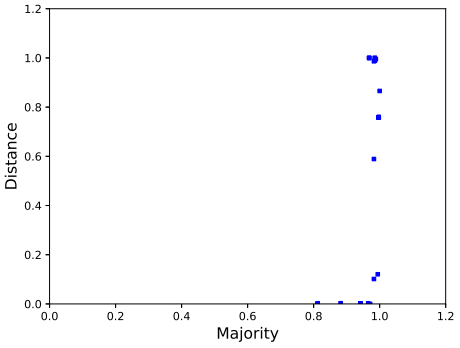

Figure 19: Plot of Distance vs Majority for spect data using NSGA-II SDO

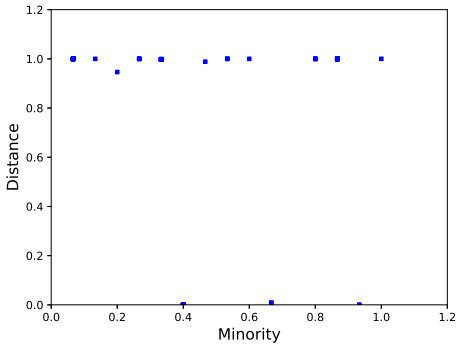

Figure 20: Plot of Distance vs Minority for Ion data using NSGA-II SDO

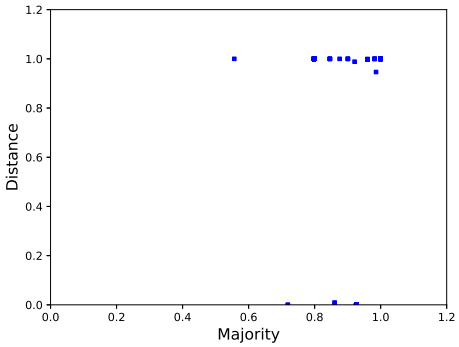

Figure 21: Plot of Distance vs Majority for spect data using NSGA-II SDO

5.2 Comparison between Various Semantic methods

Under consideration are the canonical forms of NSGA-II and SPEA2, the results of which have been reproduced in table 5 and 6 and the three methods which use a third objective; SDO which has been reproduced from [1], PSDO which used the updated Pivot distance that preferences individuals who are semantically similar to the pivot and SSCDO which utilizes SSC method of expensive crossover in addition to considering the semantic distance as an objective (original SDO).

Significance tests for the 3 Objective methods were compared against canonical NSGA-II and SPEA-II using two sample t tests. The null hypothesis $H_0 : \mu_a = \mu_b$ vs the alternative $H_0 : \mu_a \neq \mu_b$ where μ_a is the average hyperfront for the canonical method and μ_b is the average hyperfront for one of the 3 objective methods. Each method was run 50 times to insure results would be statistically significant. A significance level of $\alpha = 0.05$ was chosen and any value above 1.96 or below -1.96 was considered to be statistically significant. The t statistic can then be calculated between the two methods;

$$t = \frac{\mu_a - \mu_b}{\sqrt{S_a^2/n + S_b^2/n)}}$$

In tables 3 and 4 results that were found to be significantly better are denoted with a + and results that are found to be worse are denoted with a -. The best result for each data set has been underlined.

Table 2: Average hyperarea (± std. deviation) and last run Pareto Front for NSGA-II and SPEA2 for 50 independent runs. Results have been reproduced under permission from Galván et. al [1]

Dataset	NSGA-II Hypervolume		SPEA2 Hypervolume	
	Average	PO Front	Average	PO Front
Ion	0.766 ± 0.114	0.938	0.786 ± 0.094	0.948
Spect	0.534 ± 0.024	0.647	0.544 ± 0.032	0.659
Yeast_1	0.838 ± 0.011	0.876	0.838 ± 0.008	0.877
Yeast_2	0.950 ± 0.009	0.976	0.946 ± 0.015	0.978
Abal_1	0.847 ± 0.058	0.961	0.832 ± 0.078	0.960
Abal_2	0.576 ± 0.122	0.842	0.544 ± 0.147	0.834

Table 3: Average hyperarea (± std. deviation) and last run Pareto Front for NSGA-II SDO, NSGA-II PSDO and NSGA-II SSC methods. NSGA-II SDO Results have been reproduced under permission from Galván et. al 2020

		Hypervolume							
		Average UBSS				PO Front UBSS			
	LBSS	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
NSGA-II SDO									
Ion	–	**0.860 ± 0.033**+	**0.869 ± 0.037**+	**0.869 ± 0.033**+	**0.845 ± 0.057**+	0.948	0.958	0.962	0.950
	0.001	***0.817 ± 0.087***+	**0.819 ± 0.104**+	**0.857 ± 0.057**+	**0.861 ± 0.047**+	*0.942*	0.957	0.954	0.958
	0.01	**0.825 ± 0.084**+	**0.843 ± 0.073**+	**0.861 ± 0.045**+	**0.861 ± 0.038**+	0.946	0.956	0.957	0.944
	0.1	**0.846 ± 0.070**+	**0.848 ± 0.068**+	**0.844 ± 0.075**+	**0.864 ± 0.044**+	0.950	0.956	0.953	0.960
Spect	–	**0.591 ± 0.027**+	**0.593 ± 0.025**+	**0.594 ± 0.023**+	**0.600 ± 0.019**+	0.684	0.679	0.689	0.694
	0.001	**0.562 ± 0.021**+	***0.558 ± 0.025***+	**0.561 ± 0.019**+	**0.560 ± 0.016**+	0.668	0.653	0.660	0.644
	0.01	**0.564 ± 0.025**+	**0.560 ± 0.023**+	**0.566 ± 0.024**+	**0.559 ± 0.016**+	0.672	0.650	0.669	*0.643*
	0.1	**0.563 ± 0.022**+	**0.563 ± 0.024**+	**0.567 ± 0.018**+	**0.561 ± 0.024**+	0.664	0.658	0.655	0.658
$Yeast_1$	–	**0.850 ± 0.006**+	**0.849 ± 0.008**+	**0.849 ± 0.006**+	**0.849 ± 0.006**+	0.881	0.881	0.882	0.881
	0.001	***0.845 ± 0.007***+	**0.847 ± 0.006**+	**0.848 ± 0.004**+	**0.848 ± 0.005**+	0.879	0.882	0.879	0.880
	0.01	**0.848 ± 0.006**+	**0.849 ± 0.005**+	**0.848 ± 0.005**+	**0.850 ± 0.005**+	0.881	0.881	0.879	0.881
	0.1	**0.847 ± 0.005**+	**0.848 ± 0.005**+	**0.848 ± 0.005**+	**0.850 ± 0.005**+	*0.878*	0.879	0.879	0.883
$Yeast_2$	–	**0.961 ± 0.007**+	**0.961 ± 0.007**+	**0.960 ± 0.008**+	**0.962 ± 0.007**+	0.978	0.979	0.979	0.979
	0.001	**0.959 ± 0.008**+	**0.958 ± 0.007**+	**0.961 ± 0.006**+	**0.961 ± 0.006**+	0.981	0.978	0.979	0.978
	0.01	***0.955 ± 0.009***+	**0.959 ± 0.007**+	**0.960 ± 0.009**+	**0.961 ± 0.007**+	0.979	0.980	0.979	0.978
	0.1	**0.958 ± 0.009**+	**0.960 ± 0.007**+	**0.961 ± 0.007**+	**0.962 ± 0.006**+	*0.978*	0.978	0.981	0.979
$Abal_1$	–	0.849 ± 0.081	0.862 ± 0.087	0.847 ± 0.089	0.849 ± 0.085	*0.964*	0.970	0.966	0.967
	0.001	***0.892 ± 0.051***+	**0.905 ± 0.036**+	**0.907 ± 0.036**+	**0.906 ± 0.034**+	0.970	0.968	0.969	0.971
	0.01	**0.908 ± 0.038**+	**0.900 ± 0.056**+	**0.919 ± 0.022**+	**0.919 ± 0.026**+	0.969	0.973	0.970	0.972
	0.1	**0.910 ± 0.037**+	**0.911 ± 0.046**+	**0.912 ± 0.049**+	**0.916 ± 0.031**+	0.970	0.972	0.969	0.970
$Abal_2$	–	0.591 ± 0.102	0.623 ± 0.138	0.634 ± 0.115	0.617 ± 0.137	0.862	0.878	0.881	0.873
	0.001	**0.729 ± 0.070**+	**0.722 ± 0.063**+	***0.709 ± 0.080***+	**0.735 ± 0.074**+	0.877	*0.870*	0.879	0.885
	0.01	**0.721 ± 0.067**+	**0.725 ± 0.075**+	**0.721 ± 0.074**+	**0.723 ± 0.066**+	0.881	0.879	0.884	0.880
	0.1	**0.724 ± 0.076**+	**0.739 ± 0.065**+	**0.736 ± 0.063**+	**0.756 ± 0.065**+	0.888	0.883	0.886	0.890
Better (+) / Worse (-)		22 / 0	22 / 0	22 / 0	22 / 0				
Same (=) / NSS		0 / 2	0 / 2	0 / 2	0 / 2				
NSGA-II PSDO									
Ion	–	**0.794 ± 0.100** +	**0.811 ± 0.084** +	**0.823 ± 0.091** +	**0.795 ± 0.105** +	0.904	0.932	0.945	0.939
	0.001	**0.867 ± 0.035**+	**0.874 ± 0.029**+	**0.880 ± 0.036**+	**0.873 ± 0.045**+	0.959	0.952	0.965	0.945
	0.01	**0.852 ± 0.050**+	**0.867 ± 0.051**+	**0.880 ± 0.031**+	**0.867 ± 0.050**+	0.947	0.950	0.944	0.949
	0.1	**0.853 ± 0.062**+	**0.869 ± 0.048**+	**0.875 ± 0.051**+	**0.872 ± 0.049**+	0.941	0.951	0.956	0.938
Spect	–	**0.552 ± 0.020**+	**0.546 ± 0.022**+	**0.555 ± 0.022**+	**0.554 ± 0.017**+	0.648	0.665	0.638	0.640
	0.001	**0.550 ± 0.026**+	**0.562 ± 0.025**+	**0.561 ± 0.025**+	**0.592 ± 0.026**+	0.661	0.670	0.658	0.706
	0.01	**0.550 ± 0.025**+	**0.563 ± 0.026**+	**0.558 ± 0.025**+	**0.583 ± 0.020**+	0.649	0.675	0.667	0.669
	0.1	**0.551 ± 0.023**+	**0.560 ± 0.024**+	**0.557 ± 0.025**+	**0.593 ± 0.020**+	0.666	0.664	0.678	0.682
$Yeast_1$	–	**0.846 ± 0.006**+	**0.846 ± 0.005**+	**0.847 ± 0.005**+	**0.848 ± 0.006**+	0.864	0.868	0.871	0.869
	0.001	**0.849 ± 0.006**+	**0.848 ± 0.005**+	**0.850 ± 0.007**+	**0.850 ± 0.005**+	0.873	0.868	0.871	0.869
	0.01	**0.850 ± 0.005**+	**0.849 ± 0.007**+	**0.850 ± 0.006**+	**0.851 ± 0.006**+	0.870	0.874	0.872	0.872
	0.1	**0.850 ± 0.006**+	**0.850 ± 0.005**+	**0.850 ± 0.005**+	**0.851 ± 0.006**+	0.876	0.873	0.872	0.870
$Yeast_2$	–	**0.957 ± 0.007**+	**0.959 ± 0.007**+	**0.957 ± 0.009**+	**0.959 ± 0.007**+	0.973	0.978	0.976	0.978
	0.001	**0.960 ± 0.010**+	**0.962 ± 0.005**+	**0.964 ± 0.005**+	**0.962 ± 0.008**+	0.976	0.976	0.978	0.977
	0.01	**0.962 ± 0.006**+	**0.962 ± 0.006**+	**0.962 ± 0.005**+	**0.962 ± 0.006**+	0.977	0.975	0.974	0.975
	0.1	**0.964 ± 0.006**+	**0.960 ± 0.010**+	**0.963 ± 0.005**+	**0.961 ± 0.007**+	0.976	0.976	0.977	0.975
$Abal_1$	–	**0.890 ± 0.051**+	**0.881 ± 0.070**+	**0.885 ± 0.046**+	**0.884 ± 0.058**+	0.959	0.966	0.961	0.952
	0.001	0.861 ± 0.079	0.848 ± 0.073	**0.877 ± 0.078**+	0.864 ± 0.075	0.962	0.957	0.962	0.959
	0.01	0.864 ± 0.067	0.858 ± 0.076	0.865 ± 0.070	**0.873 ± 0.066**+	0.967	0.959	0.962	0.962
	0.1	0.858 ± 0.082	**0.887 ± 0.061**+	0.864 ± 0.074	0.860 ± 0.075	0.963	0.968	0.962	0.955
$Abal_2$	–	**0.704 ± 0.083**+	**0.699 ± 0.072**+	**0.706 ± 0.069**+	**0.711 ± 0.076**+	0.826	0.859	0.874	0.858
	0.001	**0.725 ± 0.070**+	**0.743 ± 0.079**+	**0.745 ± 0.060**+	**0.733 ± 0.075**+	0.859	0.871	0.854	0.877
	0.01	**0.741 ± 0.086**+	**0.735 ± 0.074**+	**0.724 ± 0.070**+	**0.728 ± 0.069**+	0.873	0.867	0.870	0.873
	0.1	**0.743 ± 0.061**+	**0.723 ± 0.073**+	**0.719 ± 0.088**+	**0.722 ± 0.063**+	0.877	0.847	0.846	0.851
Better (+) / Worse (-)		21 / 0	22/ 0	22 / 0	22 / 0				
Same (=) / NSS		0 / 3	0 / 2	0 / 2	0 / 0 2				
NSGA-II SSCDO									
Ion	–	–	**0.864± 0.032**+	–	–	–	0.953	–	–
Spect	–	–	**0.582± 0.026**+	–	–	–	0.679	–	–
$Yeast_1$	–	–	**0.851± 0.007**+	–	–	–	0.882	–	–
$Yeast_2$	–	–	**0.962± 0.009**+	–	–	–	0.980	–	–
$Abal_1$	–	–	**0.885± 0.059**+	–	–	–	0.971	–	–
$Abal_2$	–	–	0.644 ± 0.111	–	–	–	0.881	–	–
Better (+) / Worse (-)		- / -	5 / 0	- / -	- / -				
Same (=) / NSS		- / -	0/ 1	- / -	- / -				

Table 4: Average hyperarea (± std. deviation) and last run Pareto Front for SPEA2 SDO, SPEA2 PSDO and SPEA2 SSCDO methods. SPEA2 SDO Results have been reproduced under permission from Galván et. al 2020

		Hypervolume							
		Average UBSS				PO Front UBSS			
	LBSS	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
SPEA2 SDO									
Ion	–	**0.859 ± 0.031**+	**0.869 ± 0.029**+	**0.862 ± 0.034**+	**0.865 ± 0.047**+	0.951	0.952	0.950	0.961
	0.001	**0.858 ± 0.041**+	**0.852 ± 0.075**+	**0.870 ± 0.055**+	**0.874 ± 0.055**+	*0.946-*	0.955	0.952	0.956
	0.01	***0.837 ± 0.097***+	**0.851 ± 0.077**+	**0.875 ± 0.032**+	**0.863 ± 0.049**+	0.956	0.951	0.953	0.959
	0.1	**0.852 ± 0.071**+	**0.856 ± 0.053**+	**0.873 ± 0.035**+	**0.862 ± 0.038**+	0.947	0.949	0.952	0.950
Spect	–	**0.591 ± 0.020**+	**0.599 ± 0.021**+	**0.597 ± 0.018**+	**0.595 ± 0.022**+	0.678	0.688	0.686	0.695
	0.001	**0.569 ± 0.021**+	**0.565 ± 0.024**+	**0.566 ± 0.023**+	**0.563 ± 0.023**+	0.668	0.666	0.672	0.658
	0.01	**0.568 ± 0.023**+	**0.567 ± 0.024**+	**0.564 ± 0.025**+	**0.563 ± 0.023**+	0.666	0.674	0.664	0.658
	0.1	**0.566 ± 0.023**+	***0.560 ± 0.020***+	**0.567 ± 0.027**+	**0.561 ± 0.022**+	0.666	*0.654*	0.673	0.658
$Yeast_1$	–	**0.850 ± 0.007**+	**0.850 ± 0.006**+	**0.849 ± 0.008**+	**0.849 ± 0.004**+	0.882	0.881	0.881	0.881
	0.001	**0.848 ± 0.006**+	**0.847 ± 0.007**+	**0.848 ± 0.004**+	**0.850 ± 0.006**+	0.880	0.883	0.880	0.883
	0.01	**0.848 ± 0.006**+	**0.847 ± 0.006**+	**0.850 ± 0.005**+	**0.850 ± 0.005**+	0.881	0.880	0.882	0.879
	0.1	***0.847 ± 0.005***+	**0.849 ± 0.006**+	**0.848 ± 0.005**+	**0.849 ± 0.006**+	*0.879*	0.882	0.880	0.882
$Yeast_2$	–	**0.962 ± 0.007**+	**0.962 ± 0.006**+	**0.962 ± 0.006**+	**0.963 ± 0.008**+	0.979	0.979	0.979	0.977
	0.001	***0.958 ± 0.008***+	**0.960 ± 0.007**+	**0.960 ± 0.005**+	**0.960 ± 0.005**+	0.980	0.979	0.979	*0.977*
	0.01	**0.959 ± 0.008**+	**0.961 ± 0.007**+	**0.961 ± 0.005**+	**0.962 ± 0.007**+	0.979	0.980	0.978	0.978
	0.1	**0.961 ± 0.007**+	**0.961 ± 0.007**+	**0.960 ± 0.007**+	**0.964 ± 0.007**+	0.980	0.979	0.979	0.980
$Abal_1$	–	**0.875 ± 0.059**+	***0.868 ± 0.081***+	**0.875 ± 0.059**+	**0.873 ± 0.069**+	*0.965*	0.974	0.968	0.972
	0.001	**0.895 ± 0.061**+	**0.911 ± 0.031**+	**0.905 ± 0.044**+	**0.903 ± 0.036**+	0.974	0.973	0.972	0.972
	0.01	**0.903 ± 0.038**+	**0.906 ± 0.042**+	**0.901 ± 0.048**+	**0.910 ± 0.039**+	0.966	0.969	0.972	0.974
	0.1	**0.888 ± 0.067**+	**0.918 ± 0.032**+	**0.910 ± 0.046**+	**0.916 ± 0.027**+	0.974	0.970	0.968	0.967
$Abal_2$	–	***0.620 ± 0.148***+	**0.633 ± 0.124**+	**0.651 ± 0.146**+	**0.630 ± 0.138**+	0.874	*0.861*	0.879	0.876
	0.001	**0.717 ± 0.069**+	**0.709 ± 0.079**+	**0.722 ± 0.083**+	**0.733 ± 0.075**+	0.868	0.883	0.886	0.891
	0.01	**0.706 ± 0.084**+	**0.720 ± 0.067**+	**0.726 ± 0.067**+	**0.747 ± 0.070**+	0.884	0.880	0.877	0.887
	0.1	**0.732 ± 0.064**+	**0.733 ± 0.066**+	**0.749 ± 0.063**+	**0.737 ± 0.081**+	0.880	0.876	0.883	0.877
Better (+) / Worse (-)		24 / 0	24 / 0	24 / 0	24 / 0	22 / 2	23 / 1	23 / 0	18 / 5
Eq. (≡) / NS (x)		0 / 0	0/ 0	0 / 0	0 / 0	0 / -	0 / -	1 / -	1 / -
SPEA2 PSDO									
Ion	–	**0.864 ± 0.038**+	**0.861 ± 0.032**+	**0.865 ± 0.031**+	**0.868 ± 0.034**+	0.926	0.932	0.921	0.922
	0.001	**0.852 ± 0.076**+	**0.857 ± 0.055**+	**0.872 ± 0.033**+	**0.869 ± 0.036**+	0.872	0.895	0.846	0.893
	0.01	**0.880 ± 0.036**+	**0.864 ± 0.041**+	**0.857 ± 0.068**+	**0.871 ± 0.036**+	0.955	0.922	0.944	0.905
	0.1	**0.856 ± 0.056**+	**0.874 ± 0.034**+	**0.866 ± 0.03**+	**0.874 ± 0.034**+	0.931	0.921	0.830	0.900
Spect	–	**0.595 ± 0.028**+	**0.599 ± 0.020**+	**0.595 ± 0.020**+	**0.602 ± 0.025**+	0.669	0.670	0.664	0.657
	0.001	**0.563 ± 0.028**+	**0.568 ± 0.020**+	**0.570 ± 0.022**+	**0.565 ± 0.021**+	0.670	0.621	0.623	0.607
	0.01	**0.569 ± 0.027**+	**0.565 ± 0.021**+	**0.566 ± 0.023**+	**0.562 ± 0.022**+	0.628	0.620	0.599	0.610
	0.1	**0.567 ± 0.025**+	**0.563 ± 0.022**+	**0.564 ± 0.022**+	**0.562 ± 0.022**+	0.632	0.633	0.606	0.620
$Yeast_1$	–	**0.849 ± 0.006**	**0.849 ± 0.005**	**0.850 ± 0.006**	0.849 ± 0.006	0.874	0.875	0.867	0.872
	0.001	**0.847 ± 0.005**	**0.847 ± 0.007**	**0.848 ± 0.006**	**0.847 ± 0.005**-	0.867	0.868	0.865	0.865
	0.01	**0.847 ± 0.005**	**0.848 ± 0.006**	**0.849 ± 0.006**	**0.849 ± 0.006**	0.865	0.864	0.864	0.865
	0.1	**0.848 ± 0.006**	**0.849 ± 0.006**	**0.849 ± 0.007**	**0.849 ± 0.006**	0.863	0.866	0.867	0.871
$Yeast_2$	–	**0.961 ± 0.008**+	**0.951 ± 0.007**+	**0.962 ± 0.008**+	**0.963 ± 0.007**+	0.979	0.977	0.978	0.975
	0.001	**0.956 ± 0.010**+	**0.959 ± 0.005**+	**0.958 ± 0.005**+	**0.960 ± 0.008**+	0.977	0.975	0.977	0.975
	0.01	**0.959 ± 0.006**+	**0.960 ± 0.006**+	**0.960 ± 0.005**+	**0.963 ± 0.006**+	0.973	0.976	0.973	0.976
	0.1	**0.960 ± 0.006**+	**0.958 ± 0.010**+	**0.963 ± 0.005**+	**0.964 ± 0.007**+	0.975	0.972	0.975	0.974
$Abal_1$	–	0.858 ± 0.067	0.840 ± 0.097	0.855 ± 0.082	**0.869 ± 0.073**+	0.962	0.954	0.962	0.958
	0.001	**0.888 ± 0.048**+	**0.890 ± 0.059**+	**0.902 ± 0.045**+	**0.905 ± 0.037**+	0.964	0.958	0.958	0.965
	0.01	**0.893 ± 0.049**+	**0.908 ± 0.043**+	**0.913 ± 0.046**+	**0.917 ± 0.026**+	0.957	0.959	0.967	0.949
	0.1	**0.912 ± 0.031**+	**0.904 ± 0.053**+	**0.916 ± 0.033**+	**0.919 ± 0.027**+	0.953	0.963	0.962	0.958
$Abal_2$	–	**0.655 ± 0.104**+	**0.621 ± 0.121**+	**0.662 ± 0.107**+	**0.647 ± 0.117**+	0.863	0.864	0.866	0.862
	0.001	**0.720 ± 0.061**+	**0.717 ± 0.066**	**0.715 ± 0.067**+	**0.726 ± 0.059**+	0.861	0.818	0.840	0.850
	0.01	**0.698 ± 0.080**+	**0.716 ± 0.070**+	**0.723 ± 0.077**+	**0.728 ± 0.091**+	0.862	0.869	0.837	0.839
	0.1	**0.727 ± 0.064**+	**0.723 ± 0.063**+	**0.752 ± 0.061**+	**0.741 ± 0.086**+	0.834	0.791	0.834	0.837
Better (+) / Worse (-)		23 / 0	23 / 0	23 / 0	24 / 0				
Same (=) / NSS		0 / 1	0 / 1	0 / 1	0 / 0				
SPEA2 SSCDO									
Ion	–	–	**0.876± 0.029**+	–	–	–	0.961	–	–
Spect	–	–	**0.587± 0.022**+	–	–	–	0.672	–	–
$Yeast_1$	–	–	**0.851± 0.007**+	–	–	–	0.881	–	–
$Yeast_2$	–	–	**0.963± 0.007**+	–	–	–	0.979	–	–
$Abal_1$	–	–	**0.879± 0.078**+	–	–	–	0.974	–	–
$Abal_2$	–	–	**0.650 ± 0.116**+	–	–	–	0.874	–	–
Better (+) / Worse (-)			6 / 0						

A series of payoff matrices can be used to show which strategies have significant better values or 'Wins' versus the other strategies. The table can be read as follows; the strategies of the column index are compared against the strategies of the row index and for each LBSS and UBSS setting which is significantly better for the column strategy counts as one 'win' towards the count. For example SDO vs NSGA-II for the Ion data set is significantly better for all settings of LBSS and UBSS and as such has 16 'Wins' overall. As the SSCDO strategy from table 3 and 4 uses only one setting this strategy is only compared against SDO and PSDO with a LBSS value undefined and UBSS value 0.5. As such SSCDO can only have a single win in their respective cells when compared against the other strategies (As seen in the bottom row and right-most column of each data set table). The tables have been colour coded as such with solid Red denoting that the strategy is the best overall in terms of the number of wins and Yellow being the worst overall.

All of the 3 Objective strategies outperformed their respective canonical counterparts with the exception of Abal_1 which had a number of settings that produced no significant difference in the hyperarea averages for canonical NSGA-II. Additionally for Abal_1 NSGA-II SDO had the greatest number of wins over other strategies.

NSGA-II PSDO produced more wins for certain data sets like Ion, Yeast_1 and Yeast_2 when compared with NSGA-II SDO but under performed for Spect and Abal. Typically when NSGA-II PSDO out performed against NSGA-II SDO it was for settings that were held constant. For instance 3 of the wins associated with Ion were a result of keeping UBSS constant at 0.75 with LBSS values of 0.001, 0.01, 0.1 and for Spect with a UBSS value of 1.0 and LBSS values of (0.001, 0.01, 0.1). NSGA-II PSDO performed significantly worse most often when LBSS was undefined except for Abal_1 where the results were reversed, i.e when LBSS was undefined PSDO performed significantly better but was significantly worse for all 12 of the other LBSS settings. There was little or no significant difference when comparing SPEA2 SDO and SPEA2 PSDO stratgies with only SPEA2 PSDO vs SPEA2 SDO in Ion producing 2 'Wins'.

NSGA-II SSCDO preforms better than SDO for Ion but loses against SPECT. NSGA-II SSCDO performs better than PSDO for Spect and Yeast but PSDO 'wins'. over SSCDO for ABal_2. Again SPEA2 showed less significant results but SSCDO showed improvements over PSDO for Ion and Spect.

Ion

	NSGA-II	SDO	PSDO	SSCDO
NSGA-II	-	0	0	0
SDO	16	-	3	0
PSDO	16	5	-	0
SSCDO	1	1	0	-

Spect

	NSGA-II	SDO	PSDO	SSCDO
NSGA-II	-	0	0	0
SDO	16	-	7	1
PSDO	16	3	-	0
SSCDO	1	0	1	-

Yeast$_1$

	NSGA-II	SDO	PSDO	SSCDO
NSGA-II	-	0	0	0
SDO	16	-	2	0
PSDO	16	5	-	0
SSCDO	1	0	1	-

Yeast$_2$

	NSGA-II	SDO	PSDO	SSCDO
NSGA-II	-	0	0	0
SDO	16	-	2	0
PSDO	16	5	-	0
SSCDO	1	0	0	-

Abal$_1$

	NSGA-II	SDO	PSDO	SSCDO
NSGA-II	-	0	0	0
SDO	12	-	12	0
PSDO	7	4	-	0
SSCDO	1	0	0	-

Abal$_2$

	NSGA-II	SDO	PSDO	SSCDO
NSGA-II	-	0	0	0
SDO	16	-	1	0
PSDO	16	5	-	1
SSCDO	1	0	0	-

Table 5: Payoff tables for NSGA-II, NSGA-II SDO, NSGA-II PSDO and NSGA-II SSCDO for each of the 6 data sets

Ion

	SPEA2	SDO	PSDO	SSCDO
SPEA2	-	0	0	0
SDO	16	-	0	0
PSDO	16	2	-	0
SSCDO	1	0	1	-

Spect

	SPEA2	SDO	PSDO	SSCDO
SPEA2	-	0	0	0
SDO	16	-	0	0
PSDO	16	0	-	0
SSCDO	1	0	1	-

Yeast$_1$

	SPEA2	SDO	PSDO	SSCDO
SPEA2	-	0	0	0
SDO	16	-	0	0
PSDO	16	0	-	0
SSCDO	1	0	0	-

Yeast$_2$

	SPEA2	SDO	PSDO	SSCDO
SPEA2	-	0	0	0
SDO	16	-	0	0
PSDO	16	0	-	0
SSCDO	1	0	0	-

Abal$_1$

	SPEA2	SDO	PSDO	SSCDO
SPEA2	-	0	0	0
SDO	16	-	1	0
PSDO	13	1	-	0
SSCDO	1	0	0	-

Abal$_2$

	SPEA2	SDO	PSDO	SSCDO
SPEA2	-	0	0	0
SDO	16	-	0	0
PSDO	16	0	-	0
SSCDO	1	0	0	-

Table 6: Payoff tables for SPEA2, SPEA2 SDO, SPEA2 PSDO and SPEA2 SSCDO for each of the 6 data sets.

5.3 Further semantic plot analysis

In the previous section a series of experiments were done on three methods SDO, PSDO and SSCDO. In this section we will focus primarily on the results for NSGA-II with just a UBSS value of 0.5. The results for SPEA2 can be seen in the Appendix. Also, to aid our interpretation of the plots these values have broken out into their own table

	SDO	**PSDO**	**SSCDO**
Ion	0.869 ±0.037	0.811 ±0.084	0.864 ±0.032
Spect	0.593 ±0.025	0.546 ±0.022	0.582 ±0.026
Yeast_1	0.849 ±0.008	0.846 ±0.005	0.851 ±0.007
Yeast_2	0.961 ±0.007	0.959 ±0.007	0.962 ±0.009
abal_1	0.862 ±0.087	0.881 ±0.070	0.885 ±0.059
abal_2	0.623 ±0.138	0.699 ±0.072	0.644 ±0.111

Table 7: Table of hypervolume averages with ± std deviation for NSGA-II for SDO, PSDO and SSCDO with UBSS 0.5

From the Ion and Spect data sets we can see that SDO and SSCDO better approximate the Pareto front (denoted in green and blue). These methods have solution clusters which preference larger distance. If we look at table 5.3 we can see that these methods produce significantly better hypervolume averages. For yeast_1 the distance versus majority and distance versus minority the points appear to be randomly spread for all three methods. PSDO on the other hand appears tends to preference the higher semantic distance in the Yeast_2 data set.

Another interesting result of these plots is the case of Abal_2. While SDO and SSCDO appear as expected PSDO exhibits a more unusual behaviour. We know that Abal_2 has quite a large imbalance ratio between the majority and minority class (Table). This is reflected in the distance versus majority plot where the point are flushed to the right and in the distance versus minority plot where point are flushed to left. This indicates that a large portion of the candidate solutions for PSDO are biased towards the majority.

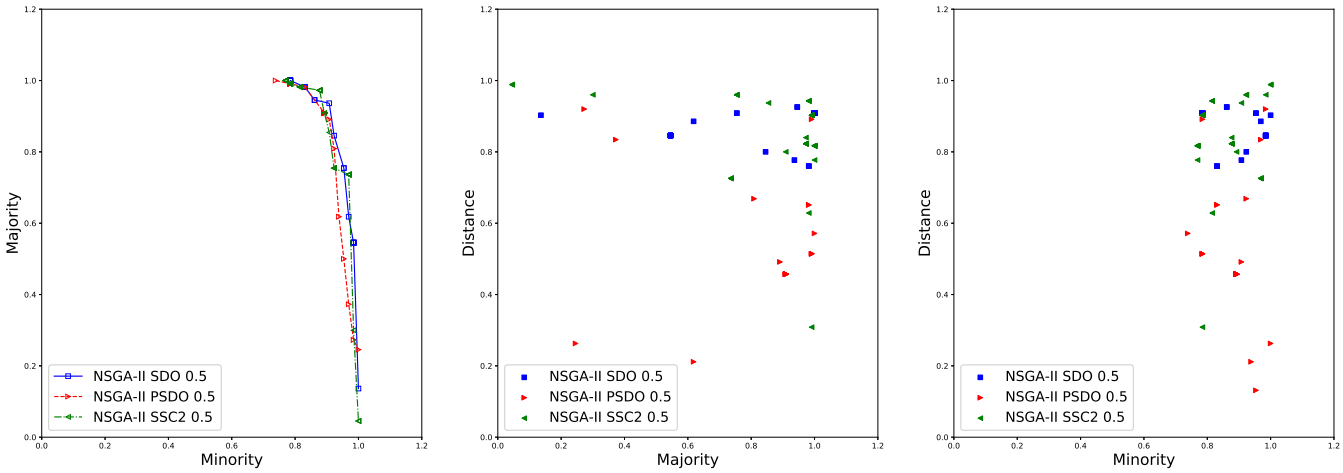

Figure 22: Plot of Majority vs Minority, Distance vs Minority and Distance vs Majority (Left to Right respectively) for Ion data using NSGA-II SDO, SDO2 and SSC2

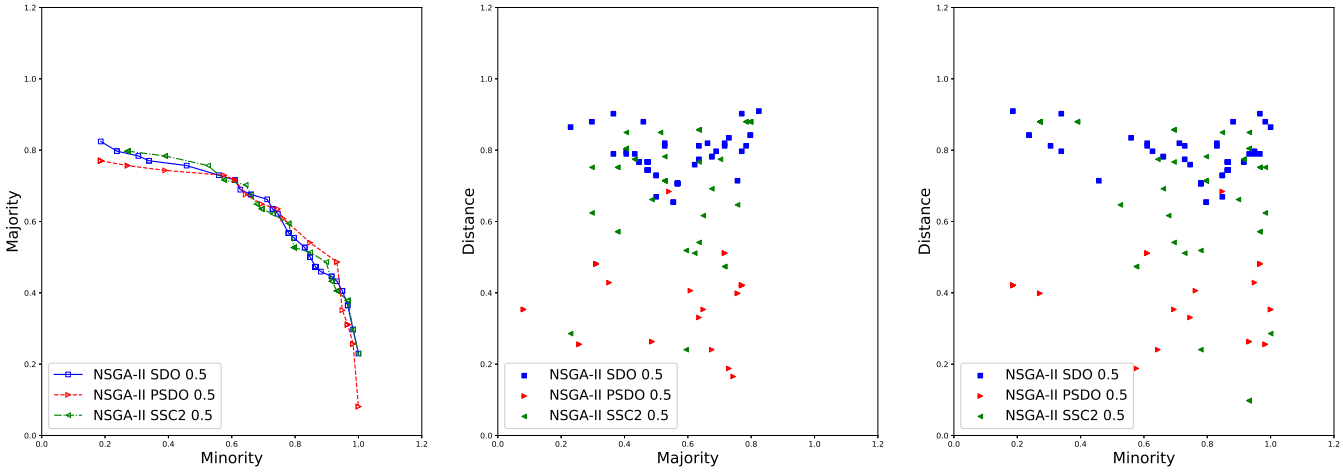

Figure 23: Plot of Majority vs Minority, Distance vs Minority and Distance vs Majority (Left to Right respectively) for Spect data using NSGA-II SDO, SDO2 and SSC2

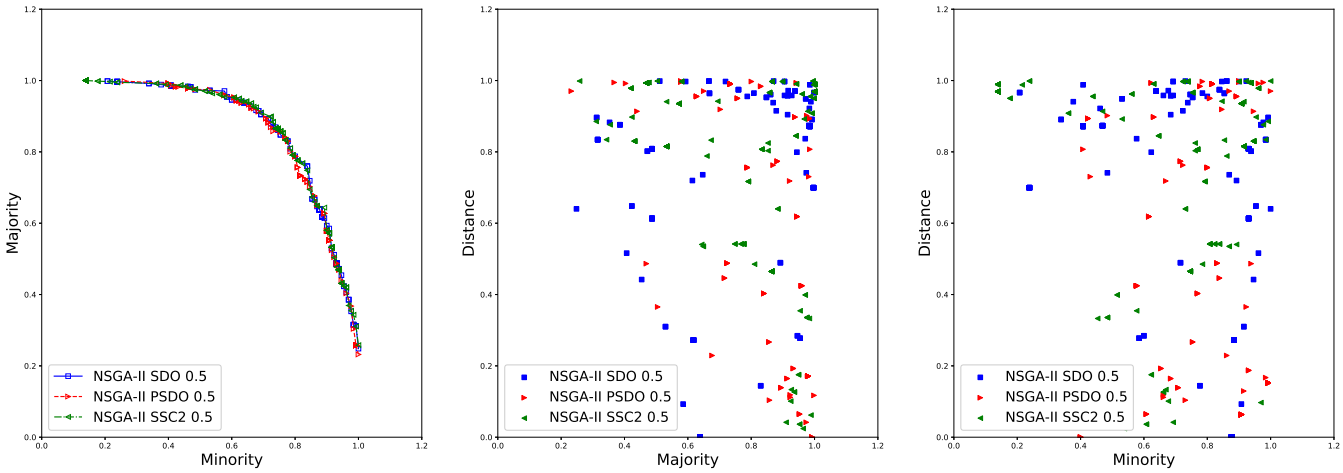

Figure 24: Plot of Majority vs Minority, Distance vs Minority and Distance vs Majority (Left to Right respectively) for Yeast_1 data using NSGA-II SDO, SDO2 and SSC2

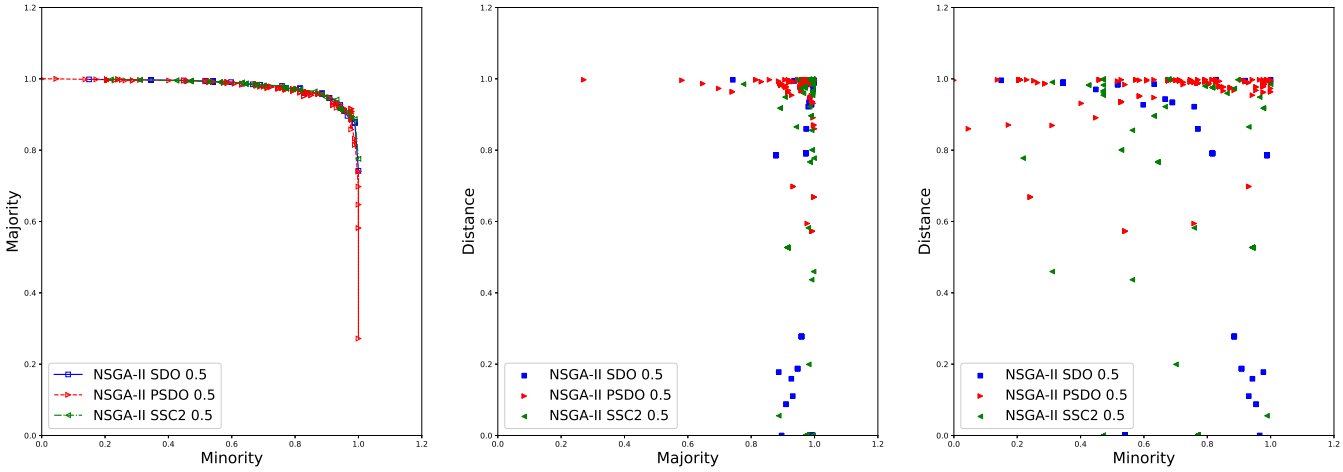

Figure 25: Plot of Majority vs Minority, Distance vs Minority and Distance vs Majority (Left to Right respectively) for Yeast_2 data using NSGA-II SDO, SDO2 and SSC2

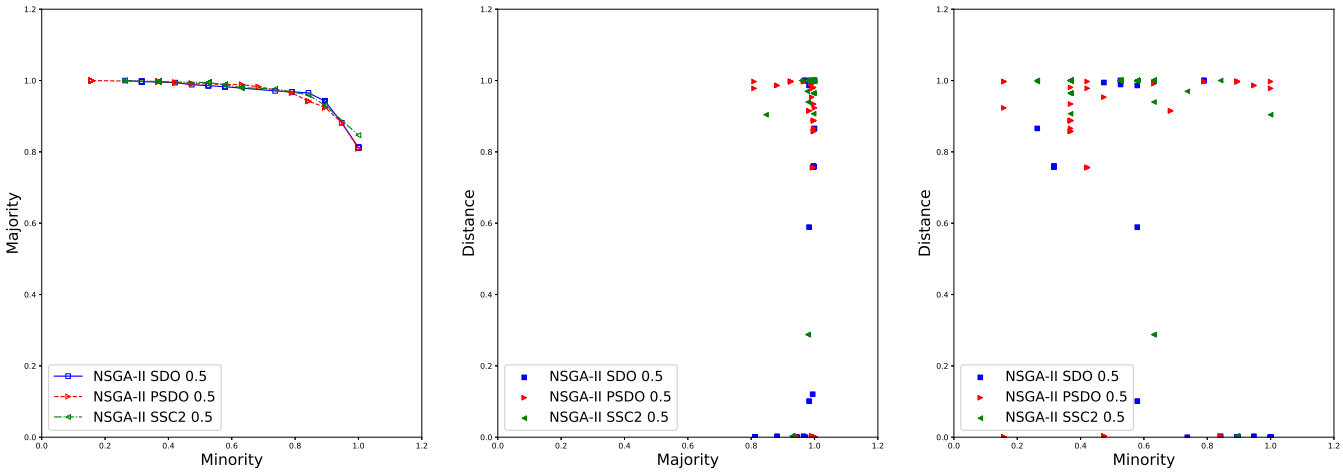

Figure 26: Plot of Majority vs Minority, Distance vs Minority and Distance vs Majority (Left to Right respectively) for Abal_1 data using NSGA-II SDO, SDO2 and SSC2

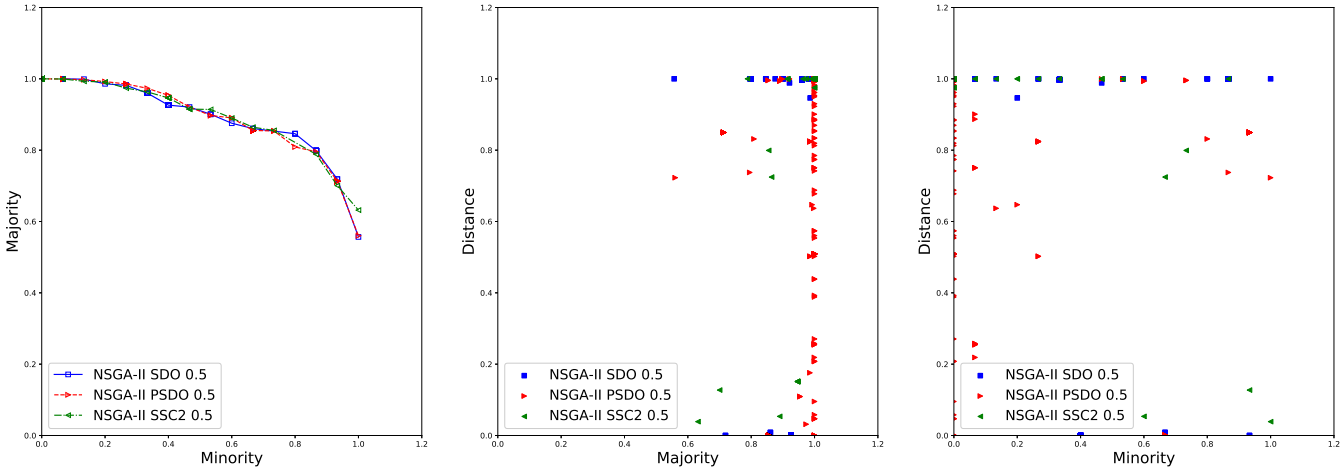

Figure 27: Plot of Majority vs Minority, Distance vs Minority and Distance vs Majority (Left to Right respectively) for Abal_2 data using NSGA-II SDO, SDO2 and SSC2

5.4 MOEA/D-TCH Results

Experimental results were gathered for a canonical MOEAD/D framework and by combining MOEA/D with the SSC method as outlined by Uy et al (2011). Both methods used the Tchebycheff approach and only consider the majority and minority as objectives. The results of these methods can be seen in table 8. By using SSC with MOEA/D-TCH we find that the hypervolume results are significantly better for for every digit.

Table 8: Average (± standard deviation) hypervolume of evolved Pareto-approximated fronts and PO fronts for MOEA/D-TCH and MOEA/D-TCH SSC for over 30 independent runs for MNIST data set.

Dataset	MOEA/D-TCH Hypervolume		MOEA/D-TCH SSC Hypervolume	
	Average	PO Front	Average	PO Front
Mnist 0	0.908 ± 0.009	0.918	**0.925 ± 0.008**+	0.928
Mnist 1	0.945 ± 0.011	0.949	**0.961 ± 0.005**+	0.957
Mnist 2	0.911 ± 0.012	0.911	**0.925 ± 0.010**+	0.928
Mnist 3	0.869 ± 0.020	0.860	**0.890 ± 0.014**+	0.894
Mnist 4	0.867 ± 0.021	0.863	**0.893 ± 0.012**+	0.886
Mnist 5	0.822 ± 0.018	0.799	**0.850 ± 0.013**+	0.845
Mnist 6	0.914 ± 0.012	0.903	**0.929 ± 0.009**+	0.928
Mnist 7	0.920 ± 0.012	0.921	**0.931 ± 0.008**+	0.925
Mnist 8	0.786 ± 0.021	0.795	**0.806 ± 0.022**+	0.803
Mnist 9	0.783 ± 0.018	0.783	**0.818 ± 0.018**+	0.813

6 Conclusion

The analysis and experimentation conducted for this dissertation clearly show the benefits of incorporating semantics in MOGP. A key finding was that regardless of which method was used in calculating semantic distance as objective, each method proved to significantly better than canonical methods.

When the various semantic-as-objective methods were analysed it was found that in spite of the distance serving as an objective to maximise, these methods had a tendency to preference programs that were semantically very similar and also semantically very dissimilar relative to the pivot. This was observed to be somewhat problem dependant occurring for Yeast_1, Yeast_2, Abal_1 and Abal_2 data sets and can be seen graphically as two clusters of points at the top and bottom of the distance versus majority and distance versus minority plots, while Spect and Ion only had a single cluster at the top. Whether these clustering patterns are related to the relatively higher class imbalance of these data sets would require more extensive research. It is of interest that Abal_2 appear to be biased toward the majority class considering the fitness classification using TPR and TNR should compensate for this bias. This is a further indication that the distance is not an objective in conflict.

It was found that integrating semantics into MOEA/D using the SSC method produced significantly better results when used with canonical MOEA/D on the MNIST data set, further showing the benefit of using semantics in MOGP.

6.1 Future work

An interesting outcome of this work is the discovery that semantic distance is not an objective in conflict. The reason the Pareto dominance relation is integrated in MOGP algorithms is that it works for objectives in conflict. New strategies could be developed using semantic distance without a dependence on dominance-based fitness assignment for this objective. It would be interesting to benchmark the MOEA/D-TCH SSC method against the UCI data sets outlined in this paper and also to compare it with MOEA/D-PBI method which has also been integrated into this framework but due to time constraints experimental results have not yet been gathered. There are also a number of extensions to MOEA/D and other decomposition methods which may find performance benefits from using semantics [8] [9].

References

[1] Galván-López, E., Mezura-Montes, E. and Trujillo, L. (2020) *Semantics in Multi-objective Genetic Programming* Pending Publication

[2] Koza, J.R., (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA.

[3] Poli, R., Langdon, W.B, and McPhee, N.F., (2008) *A field guide to genetic programming.* Published via http://lulu.com. Available at: http://www.gp-field-guide.org.uk, (With contributions by Koza, J.R.)

[4] Chiandussia G., Codegoneb, M., Ferreroc, S. and Varesioa, F.E., (2012) *Comparison of multi-objective optimization methodologies for engineering applications* Computers Mathematics with Applications Vol. 63, No. 5, PP 912-942, doi: 10.1016/j.camwa.2011.11.057

[5] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002) *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II* IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, doi: 10.1109/4235.996017.

[6] Zitzler, E., Laumanns, M., and Thiele, L., (2001) *SPEA2: Improving the Strength Pareto Evolutionary Algorithm* Conference: Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems. Proceedings of the EUROGEN 2001' Athens. Greece, September 19-21

[7] Zhang, Q. and Li, H. (2007) *MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition* in IEEE Transactions on Evolutionary Computation, vol. 11, no. 6, pp. 712-731, doi: 10.1109/TEVC.2007.892759.

[8] Zhang, Q. and Li, H. (2008) *Comparison Between NSGA-II and MOEA/D on a Set of Multiobjective Optimization Problems with Complicated Pareto Sets* in IEEE Transactions on Evolutionary Computation, vol. 13, no. 2, pp. 284-302, doi: 10.1109/TEVC.2008.925798

[9] Sato, H. (2014) *Inverted PBI in MOEA/D and its Impact on the Search Performance on Multi and Many-Objective Optimization* GECCO 14': Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, Association for Computing Machinery, pp 645–652, doi: 10.1145/2576768.2598297

[10] Uy, N.Q., Hoai, N.X., O'Neill, M. McKay, R.I. and Galván-López, E. (2011) *Semantically-based crossover in genetic programming: application to real-valued symbolic regression.* Genetic Programming and Evolvable Machines vol. 12, no. 2, pp. 91–119, doi: 10.1007/s10710-010-9121-2

[11] Pawlak, T.P., Wieloch, B., Krawiec, K., 'Semantic Backpropagation for Designing Search Operators in Genetic Programming' doi: 10.1109/TEVC.2014.2321259

[12] Galván-López, E. and Schoenauer, M *Promoting Semantic Diversity in Multi-objective Genetic Programming* doi: 10.1145/3321707.3321854

[13] Galván-López, E., Mezura-Montes, E., Ait, ElHara O., Schoenauer M. (2016) *On the Use of Semantics in Multi-objective Genetic Programming.* Parallel Problem Solving from Nature – PPSN XIV. PPSN 2016. Lecture Notes in Computer Science, vol 9921. Springer, Cham, doi: /10.1007/978-3-319-45823-6_33

[14] Bhowan, U., Johnston, M., Zhang, M. and Yao X. (2013) *Evolving Diverse Ensembles Using Genetic Programming for Classification With Unbalanced Data* in IEEE Transactions on Evolutionary Computation, vol. 17, no. 3, pp. 368-386, doi: 10.1109/TEVC.2012.2199119.

[15] 'UCU Machine learning Repository' (Accessed 10/08/2020) https://archive.ics.uci.edu/ml/index.php

[16] Wudong, L (Accessed 10/08/2020) https://github.com/ZhenkunWang/MOEADCODES/blob/master/MOEADEGO_V2009.zip

[17] *Irish Centre for High-End Computing* (Accessed 10/08/2020) https://www.ichec.ie/

[18] *Irish Centre for High-End Computing* (Accessed 10/08/2020) https://www.ichec.ie/about/infrastructure/kay