Universiteit Leiden

Master Computer Science

On the Potential of Feature-based Algorithm Selection for Pseudo-Boolean Problems

Name:	Mohamed Zehni Khairullah
Student ID:	s1902636
Date:	[dd/mm/yyyy]
Specialisation:	Data Science
1st supervisor:	Dr. Hao Wang
2nd supervisor:	Prof.dr.Thomas Bäck

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

The current methods that exist in the field of Exploratory Landscape Analysis (ELA) orbit around the continuous decision space and do not include the discrete decision space in the general sense. Given the recent interest in the field of optimization in the discrete space and the attempts to generate features that are problem specific, we believed that there is a need to find feature set(s) are that unique for a general set of objective functions. In this thesis, we attempted to finding such feature set(s) by proposing an adjustment to how a tool, $flacco$, create a $FeatureObject$ for the continuous space to make it function for the discrete decision space. We used the Pseudo-Boolean Functions as our test functions to generate the features from Cell Mapping angle features and Generalized Cell Mapping (GCM) features for different approaches: minimum, mean and near. Then we attempted to predict the Expected Run Time (ERT) and classify the best algorithm based on data obtained from IOHprofiler. We discovered that there is indeed a potential to generate general features in the discrete decision space and utilize them along with machine learning models to select the best algorithm, but not to predict the ERT. The best model that we found was Multinomial Logistic Regression (MLR) with reduced features and a tuned penalty C parameter.

Acknowledgements

I would like to express my gratitude to my first supervisor Dr. Hao Wang for the useful guidance, comments, remarks and engagement through the learning process of this master thesis. Furthermore I would like to thank my second advisor Prof.dr. Thomas Bäck for introducing me to the topic as well for the support on the way.

I like to thank my friends who I met at LIACS, who have willingly helped me through the masters degree by providing support and keeping me harmonious.

Last but not least, I like to thank my parents for always motivating me to aspire in my educational endeavors and their unconditional love.

"My Heart is in the Work"
- Andrew Carnegie

Mohamed Zehni Khairullah

Contents

Chapter 1

Introduction

In the world of science, the problem of optimization have been occurring since scientist needed to solve complex mathematical equations and problems in a way that is optimal. The methods of optimization are widely used in various of fields such as engineering, physics, operations research, bioinformatics, machine learning etc. Such hard problems are usually solved using classic heuristic optimization approaches such as local search, which moves from a solution to the others in an attempt to find the optimal solution from a set of candidate solutions. Two of the most famous hard optimization problem are the traveling salesman and the boolean satisfiability problem. Another approach to solve such problems is to use an Evolutionary Algorithm (EA), which is a subset of evolutionary computation that is inspired by biological evolution concepts such as mutation, recombination, reproduction and selection. These two approaches can be applied to different fields of optimization such as continues, disrecete, combinatorial search spaces and even mixed search spaces without/with constraints. Moreover, the research community has been showing keen interest to create and improve new heuristic optimization algorithms that can find an optimal solution to a problem or a set of problems with the least amount of time e.g. run time. Furthermore, when time is of importance, the velocity of an algorithm is determined by the best solution found given a constraint on run time. In this thesis, we attempt to explore a way to characterize the decision space of a selected set of Pseudo-Boolean functions using existing Exploratory Landscape Analysis (ELA) techniques that we adapt to the aforementioned problems' decision space. An overview of ELA is provided in section 2.1.

1.1 Summary of Notation

X	Input represented as a binary string
Y	Objective function that evaluates the input x
F	A feature set that contains many features f
f	A feature within a feature set F
N	Number of observations in a data set
D	Continuous dimension (number of variables)
L	Discrete dimension (length of a bit-string)
K	Partition dimension

M Number of cells per dimension D

1.2 Problem

The current techniques that exist in the field of Exploratory Landscape Analysis (ELA) and general feature construction revolve around the continuous decision space and do not include the discrete decision space in a general manner. Therefore, we define the problem to be general feature construction of discrete decision space or in other words, features construction to characterize the discrete decision space. A feature function takes in binary input(s) and outputs a real number that corresponds to feature of the input. The mathematical notation is formulated as follows:

$$f : \{0, 1\}^L \to \mathbb{R}$$

We are interested in finding features set(s) F that is unique per objective function Y and can be used to approximate the expected run time (ERT) and/or select the best performing algorithm by utilizing various machine learning models. Figure 1.1 exhibits a bird-eye visualization of how we approach solving the problem.

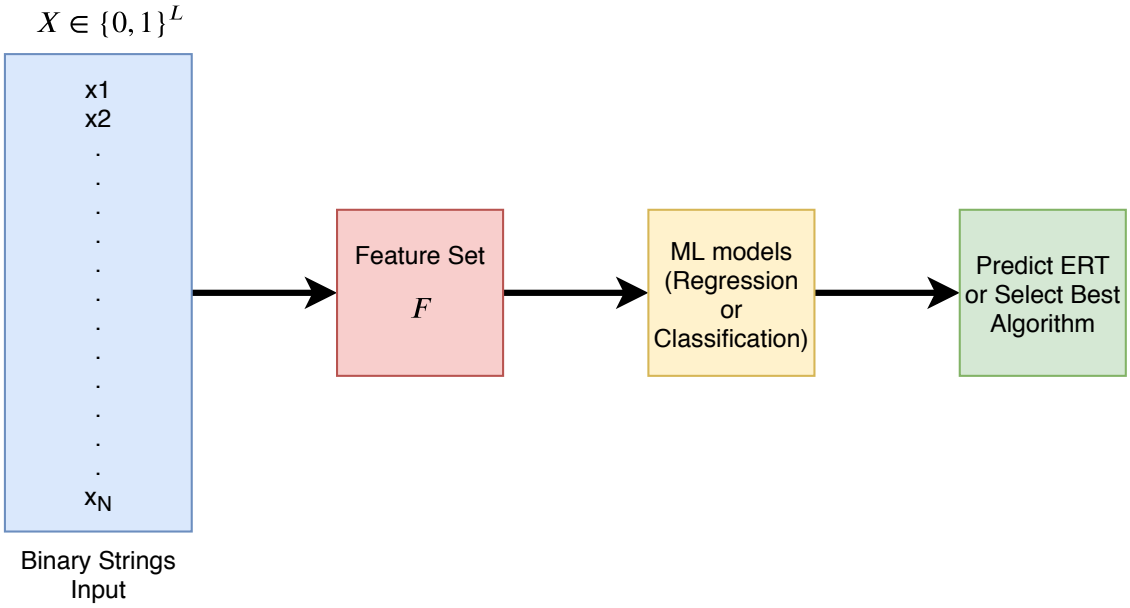

Figure 1.1: The approach to be taken to solve the problem of characterizing the discrete decision space and predict ERT or classify the best algorithm

1.3 Selected Pseudo-Boolean Functions

A Pseudo-Boolean function refers to a function of the form:

$$f : B^L \to \mathbb{R}$$

Where $B = \{0, 1\}$ corresponds to the Boolean space, which means that the values are restricted to 0 and 1. Furthermore, any Pseudo-Boolean function can be expressed uniquely as a multi-linear polynomial as follows:

$$f(x) = a + \sum_{i} a_i x_i + \sum_{i<j} a_{ij} x_i x_j + \sum_{i<j<k} a_{ijk} x_i x_j x_k + ...$$

The degree of the function is simply the length of bit string or as we denoted it as the discrete dimension L. For the purpose of this thesis, we make use of the 23 Pseudo-Boolean functions subject to maximization presented in [2] and the functions are:

1. *OneMax*
2. *LeadingOnes*
3. *Harmonic*
4. *OneMax + W([n/2],1,1,id)*
5. *OneMax + W([0.9n],1,1,id)*
6. *OneMax + W([n], $\mu = 3$,1,id)*
7. *OneMax + W([n],1, $v = 4$, id)*
8. *OneMax + W([n],1,1,r1)*
9. *OneMax + W([n],1,1,r2)*
10. *OneMax + W([n],1,1,r3)*
11. *LeadingOnes + W([n/2],1,1,id)*
12. *LeadingOnes + W([0.9n],1,1,id)*
13. *LeadingOnes + W([n], $\mu = 3$,1,id)*
14. *LeadingOnes + W([n],1, $v = 4$, id)*
15. *LeadingOnes + W([n],1,1,r1)*
16. *LeadingOnes + W([n],1,1,r2)*
17. *LeadingOnes + W([n],1,1,r3)*
18. *LABS: Low Autocorrelation Binary Sequence*
19. *Ising-Ring*
20. *Ising-Torus*
21. *Ising-Triangular*
22. *MIVS: Maximum Independent Vertex Set*
23. *N-Queens*

1.4 Selected Optimization Algorithms

The aforementioned problems were endeavored to be solved using 11 different optimization algorithms that are based on local search or evolutionary algorithms. The 11 algorithms are further defined in [2] and are as follow:

1. (1+(λ, λ)) GA
2. (1+1) EA
3. Greedy Hill Climber (gHC)
4. (1+10) EA$_{r/2,2r}$
5. (1+10) EA
6. (1+10) EA$_{LogNormal}$
7. (1+10) EA$_{Normal}$
8. (1+10) EA$_{var}$
9. Fast Genetic Algorithm (fGA)
10. 'Vanilla' Genetic Algorithm (vGA)
11. Randomized Local Search (RLS)

1.5 Research Questions

In this thesis, the main focus is to first find existing feature construction methods that are suitable for the discrete decision space, apply them to a uniform random generated binary input with different number of observations to each objective function. Second, use the generated features to approximate the ERT of each problem and/or classify the best performing algorithm. This leads to the following research questions:

- $Q1$: Are there feature construction methods in the continuous space that are suitable for the discrete space?
- $Q2$: How many sample points are needed to construct a reliable feature set?
- $Q3$: Can a selected feature set be general for any objective function?
- $Q4$: Is it more feasible to predict ERT or classify the best performing algorithm?

1.6 Outline

The structure of this thesis is as follows: Chapter 2 gives an overview about how the idea of ELA began along with detailed explanation of $flacco$ and the Cell Mapping feature sets (angle and GCM) that are used to generate features. In Chapter 3, we explain the methodology we followed to adapt the structure of $flacco$ to generate features for the discrete decision space. In addition, we explain the experiments' setup for feature generation next to feature analysis supported with plots for each feature set. In Chapter 4, we explain the data gathered, how its split and what models are utilized for the regression and classification tasks. Furthermore, we test the models used using Leave-One-Out Cross-validation (LOOCV) and we perform feature selection and hyperparameter tuning to find the best model.

Chapter 2

Related Work

2.1 Exploratory Landscape Analysis (ELA)

The history of Landscape Analysis (LA) began in the early 1990's with "tunably rugged" fitness landscape in [9], which is also referred to as *The NK model*. Two years after, the Fitness Distance Correlation (FDC) was introduced in [8]. FDC measures the search difficulty of a Genetic Algorithm (GA) and can be used to predict the performance of a GA on problems with known global maxima. Epistasis came into the scene in 1997 to explore further how to characterize GA-hardness as noted in [19]. Last but not least, before the evolution of ELA, few other researchers developed ways to characterize the decision space such as information content, dispersion metric and ruggedness, neutrality and smoothness in [21, 14, 15]. In 2011, ELA emerged as a way to employ a number of techniques to gain knowledge about the property of an unknown optimization problem with a special focus on the features that relate to the performance of optimization algorithms [16]. The approaches presented aimed to characterize

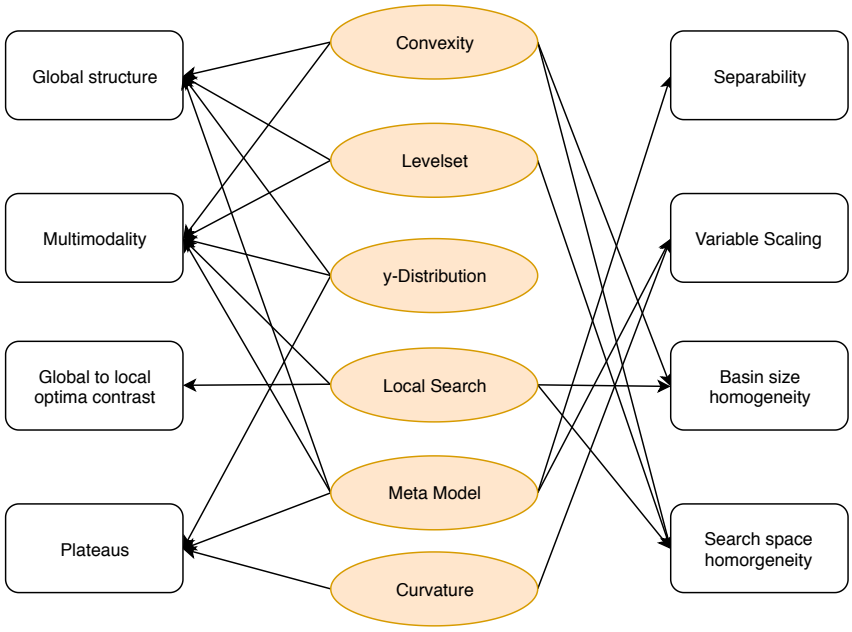

Figure 2.1: Relationship between high level features (white) and low level features (yellow) that are standard ELA

the properties of continuous optimization problems then exploit the information to give rec-

ommendations of the best suited algorithm. The Black-Box Optimization Benchmark (BBOB) functions in [4] were used in almost every preceding paper. Furthermore, given the recent development within this field of research, researchers have attempted and succeeded in developing various methodologies for ELA in the continuous dimension. The main features that were used were a set of high-level features [17] that were attained by experts i.e. (multimodality, global structure, separability, variable scaling, search space homogeneity, basin-sizes, global to local contrast and plateaus). Many researchers identified this set of high-level features to be questionable and therefore, low-level features were introduced in [16]. The aforementioned features are grouped into six classes namely, measuring convexity and linearity (Convexity), measures that are related to the distribution of the objective function values (y-Distribution), the relative position of an objective function value compared to the median of all values (Levelset), estimating meta-models such as quadratic or linear regression (Meta-Model), local searches that are conducted from the initial design points (Local Search) and represent the class of curvature features by an approximation of the Hessian or the gradient (Curvature). There is an assumed relationship between the high and low level features as exhibited in Figure 2.1. Beyond those features, other notable approaches of feature construction were developed such as the Fitness Landscape Analysis [20] and Cell Mapping [12]. In summary, all the approaches in this research domain have been focused only on the continuous decision space. However, a recent survey on automated algorithm selection noted that discrete decision space can be characterized by generating problem specific features [11].

2.2 *flacco*

Following the various developments by researchers around the world, many tools and libraries existed, but unfortunately, they were implemented using different programming languages such as $Python$, $Matlab$ and R. Therefore, the first version of the R-package *flacco* was released combining many state of the art features in a way that is easy to use and compute by others. In 2017, the official documentation of *flacco* was published [10]. The package provides a collection of the majority of the feature sets that were developed and introduced by other researchers. Particularly, *flacco* computes 300+ different numerical landscape features distributed across the following 17 feature sets: Low-level features or Initial ELA Features [17], Cell Mapping (Angle, Convexity and Gradient Homogeneity) and Generalized Cell Mapping Features [12], Barrier Tree Features [6, 3], Nearest Better Clustering Features [13], Information Content Features [18, 21], Dispersion Features [14], and Miscellaneous Approaches (Basic, Linear Models, Principal Components).

After screening and as an answer to $Q1$, two feature sets are relevant to our research namely, Simple Cell Mapping (Angle) and Generalized Cell Mapping, which will be explained later.

2.2.1 *FeatureObject*

The *FeatureObject* is used as an input for all features computation. It serves as a way to pre-process the data and the parameters associated in features computation. According to an example of $flacco$, it takes in a continuous input from the uniform distribution of size (N, D), a vector of evaluations Y of the input, the name of the objective function and vector that contains the number of blocks where the length of the vector is D. The package creates

the *FeatureObject* and outputs the following: (1) number of observations N, (2) number of variables (continues dimension D), (3) upper and lower boundaries for each dimension in D, (4) number of cells (blocks) per dimension (5) the total number of cells along with the number of empty and non-empty cells and (6) the average number of observations per cell. In addition, the *FeatureObject* contains a matrix that consists of the variables X $(x_1, x_2, ..., x_D)$, the evaluation Y of each observation and the $cell\_id$ that each observation is assigned to and another matrix that contains the cell centers with the $cell\_id$ it corresponds to. The aforementioned matrices are used to compute the feature sets within $flacco$. To elaborate, the $cell_id$s are determined independently from the evaluation of the input. To calculate the $cell_id$s, the following is needed: (1) the continues input X, (2) the *block width* or the *cell size per dimension* computed as $\frac{upper-lower}{blocks}$, (3) N and D and (4) the cumulative product cp of the vector of blocks with the value 1 being at the first position of the vector. The $cell_id$s are computed as follows:

```
for i in range(N):
    for j in range(D):
        cellID[j] = cp[j] * np.floor((X[i, j] - lower[j]) / blocksWidth[
    j])
    cellIDs[i] = int(np.sum(cellID - cp[np.arange(len(blocks))] * (X[i,:
    D] == upper)))
```

In addition, the $cell\_centers$ are computed by determining the coordinates of the center of each cell. Therefore, the $cell\_centers$ object will be of size ($blocks^D$, D) and will be used to determine the $cell_id$s using the same way mentioned before.

2.3 Cell Mapping

The idea of Cell Mapping was first introduced in [7] and it is a useful approach to acknowledge the global behaviour of nonlinear dynamic systems. The Cell Mapping approach discretizes the decision space by dividing the state space to hypercubes. There are two Cell Mapping feature sets namely, Simple Cell Mapping (SCM) and generalized cell mapping (GCM). Both feature sets have been applied to optimal control problems [1] and SCM was used in multi-objective optimization [5].

2.3.1 Simple Cell Mapping (Angle)

Using a pre-defined number of blocks (cells), the continuous decision space is discretized. SCM calculates three different subsets of features i.e. Angle, Homogeneity and Convexity features [12]. After examining the three aforementioned subsets, only the Angle features would serve well in this research based on the expected behaviour of the objective functions mentioned in section 1.3 and their known properties. The angle features extract information based on the location of the best and worst available observation of a cell w.r.t. the corresponding cell center where the distance used is the Euclidean distance as visualized in Figure 2.2.
The resulted 8 features are as follows:

1. Distance from the center to the best points (arithmetic mean and standard deviation)
2. Distance from the center to the worst points (arithmetic mean and standard deviation)
3. Angle between the best and worst point (arithmetic mean and standard deviation)

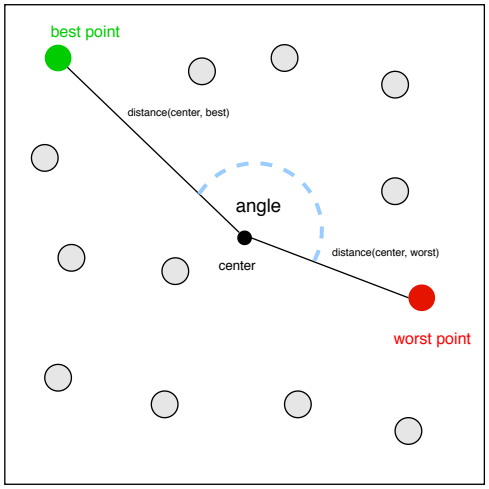

Figure 2.2: Overview of the features extracted per cell in the simple cell mapping angle features before being aggregated.

4. Best to worst point ratio (arithmetic mean and standard deviation)

Moreover, the best value of the angle is $180°$, which indicates that the best and worst points are the furthest away from each other. Lastly, the best to worst ratio is calculated by taking the difference between the best and worst objective value per cell (local best and local worst) and dividing it by the biggest difference between the best and worst point across all cells (global best and global worst).

2.3.2 Generalized Cell Mapping

Likewise SCM, GCM features are also based on the block-wise discretized decision space. According to [12] GCM can be thought of three subsets where each subset uses exactly one observation from each cell to represent it. Specifically, the minimum approach is represented by the best point w.r.t the objective function whereas the mean approach takes the mean value of the objective values in the cell. Apart from this, the near approach selects the objective function value point that is closest to the center of the cell. It is explicitly noted in [12] that in case of an empty cell, only the near approach is able to find a representative point in that cell. Intuitively, GCM contains three subsets that each corresponds to a different approach. This results in each cell being an absorbing Markov chain, which means that for each cell a transition probability for going from one cell to one of its neighbouring cells is computed. The transition probabilities can be grouped into a matrix P of order $N_c \times N_c$ where N_c is total number of cells. Building on the transition probabilities, the cells are identified into:

- Attractor: an absorbing cell within a basin attraction
- Periodic: a cell that is infinitely visited (local optima)
- Transient: a cell that is not periodic
- Uncertain: a cell that is attracted by multiple attractors

Note that the focus of [12] was on periodic cells of order 1, i.e. $P_{ii} = 1$, which corresponds to the local optima candidates according to the paper. After computing the transition probability

matrix P, the canonical form (cf) is computed as follow: the states are renumbered such as the transient states come first. If r absorbing states and t transient states are there $(N_c = r + t)$, then the transition matrix follows the canonical form:

$$P = \begin{pmatrix} I & 0 \\ R & Q \end{pmatrix}$$

Q is a $t \times t$ matrix, R is a nonzero $t \times r$ matrix, 0 is an $r \times t$ zero matrix and I is the $r \times r$ identity matrix. Q is the matrix that gathers the probabilities of transitioning from a transient state to another whereas matrix R describes the probability of moving from a transient state to an absorbing state. Next, the fundamental matrix (fm) is computed that is $N = (I - Q)^{-1}$ and it is the inverse of the absorbing Markov chain matrix $I - Q$. The (i, j)-entry n_{ij} of N is the expected number of times the chain is in state s_j, given that it begins in state s_i. Only if $i = j$ the initial state is counted. Then the fundamental matrix is defined as $fm = I + \sum_{k=1}^{\infty} Q^k$ and holds the equation $fm = N$.
GCM generates 23 features per approach as follows:

1. Total number of attractors
2. Ratio of cells that are periodic
3. Ratio of cells that are transient (non-periodic)
4. Ratio of cells that are uncertain
5. Probabilities for reaching different basins of attractions (minimum, median, arithmetic mean, maximum and standard deviation)
6. Basin sizes for certain cells (minimum, median, arithmetic mean, maximum, standard deviation and sum)
7. Basin sizes for uncertain cells (minimum, median, arithmetic mean, maximum, standard deviation and sum)
8. Number and probability of finding the attractor cell with the best objective function value.

Chapter 3

Methodology and Experiments

3.1 Feature Object Creation: Adaption to The Discrete Decision Space

Given that the design in *Flacco* [10] works only for continuous space, we introduce a few extra parameters and adjustments to adapt to the discrete space. First, we denote the variables L : the length of the bit string (Discrete Dimension), K : the partition dimension. Next, we denote that the number of dimensions D is determined by the fraction $\frac{L}{K}$ rounded to the nearest integer. The values of L and K are chosen based on the desirable design, which as mentioned controls the dimension D. Moreover, M controls that number of cells per dimension and it is used to create the blocks as M is repeated D times. Therefore, the total number of blocks is M^D. Second, the process of generating the data and preparing it for $FeatureObject$ creation is as follows: (1) a binary string matrix of size (N, L), (2) evaluate each full bit string using the objective function to be maximized, (3) split each observation to K partitions with taking in consideration the remainder bit strings to be added to the first partitions e.g. $L = 7, K = 3$ will result in 2 partitions with 4 bit strings in the first partition and 3 bit strings in the second partition, (4) intify the bit string in each $k \in K$ and (5) normalize by dividing the resulted integers by the maximum value of the integer that can be obtained as $2^{K'} - 1$ where K' is the maximum length of the bit strings across the dimension D. Following that, the data is ready to be fed into the $FeatureObject$ creation function.

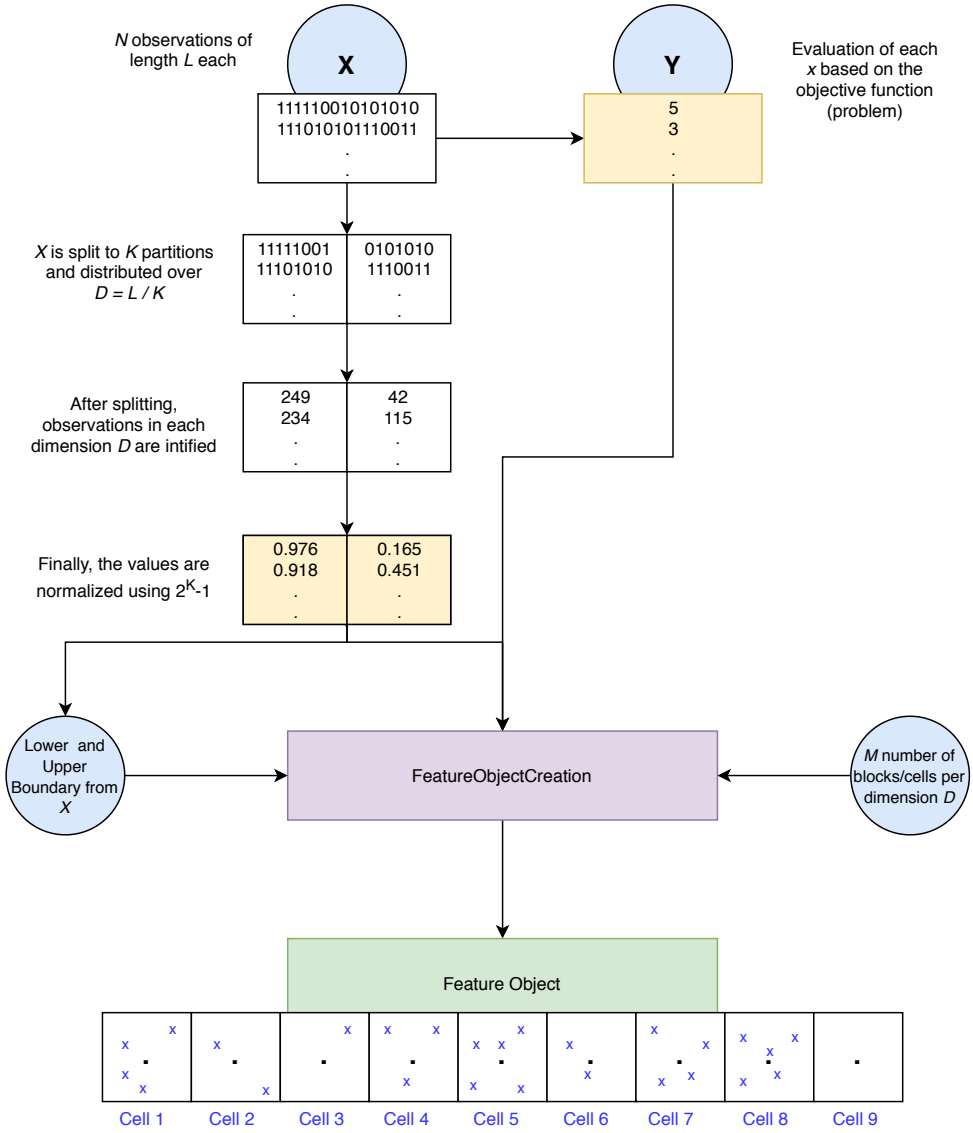

Figure 3.1: Example: $FeatureObject$ creation steps

Figure 3.1 shows an example of how the $FeatureObject$ is created. For this example, $L = 15$, $K = 7$, $M = 3$ and Y is the *LeadingOnes* function. Following the explained steps before, The continuous dimension $D = L/K$ is equal to 2 rounded to the nearest integer. The bit string of size 15 is split to two parts, 8 bit strings and 7 bit strings and then intified. Therefore, the value K' that is used in normalization becomes 8. As a result, the intified bit strings are normalized using a factor of $2^8 - 1 = 255$. From here, the lower and upper bounds are attained and fed into the *FeatureObjectCreation* function along with the number of cells per dimension D, which in this case equals to 3. Finally, the *FeatureObject* that is used to calculate the feature sets will contain a grid of 9 cells where each cell has an ID that is used to label the observations that it falls into. Note that a cell can have 1, many or no observations as illustrated in the figure.

3.2 Experiments

3.2.1 Setup and Objective

As mentioned earlier, the data set is generated from a "discrete uniform" distribution with a random seed: 123. Therefore, we generate a matrix of N observations where each observation is an array of $[0, 1]'s$ of length L. After creating the *FeatureObject*, we take into account the non-empty cells only and calculate the desired feature set. We use different values of N: 100, 500, 1000, 2000 and 5000 and fix $L = 100$, $M = 5$, $K = 50$. The aforementioned values are fixed as a result of multiple tries to insure that no empty cells are present in the *FeatureObject*. This is implemented to avoid any errors when calculating the Generalized Cell Mapping approaches as mentioned in subsection 2.3.2. Moreover, we generate Simple Cell Mapping (Angle) and Generalized Cell Mapping (all approaches). We highlight that we consider the 23 objective functions denoted by their function ids ($f1 - f23$) to be completely different and the main goal of the experiments is to conduct feature inspection that would assist in determining if a feature or more indeed helps to distinguish an objective function.

3.2.2 The *FeatureObject*

The *FeatureObject* created for the different number of observations is approximately the same. The only differences are the number of observations N, the upper and lower boundaries, cell size per dimension and the average number of observations per cell. Our goal is to create a *FeatureObject* that contains no empty cells to avoid any issues when generating the GCM features.

Feature Objects
Number of Observations 1000
Number of Variables 2
Upper Boundary [0.99989969, 0.99851245]
Lower Boundary [0.00274544, 0.00018387]
Number of Cells per Dimension [5, 5]
Cell Size per Dimension [0.19943085, 0.19966572]
Numbe of cells
..... total 25
..... non-empty 25,100.0%
..... empty 0,0.0%
Average number of Observations per cell
..... total 40.0
..... non-empty 40.0

Figure 3.2: $FeatureObject$ output example for $N = 1000$

Figure 3.2 displays an example for $N = 1000$ with the fixed variables that were mentioned previously. The number of variables correspond to the continuous dimension D that it is determined by the fraction $\frac{L}{K}$ where $L = 100$ and $K = 50$, Furthermore, the lower and upper boundaries are normalized to be between 0 and 1 and the number of cells per dimension M is 5, which makes the total number of cell 25 given that $M^D = 5^2$. Furthermore, the cell size per dimension is determined by the equation $\frac{upper-lower}{blocks}$. Last but not least, the average number of

observations per cell is 40. This means that the observations were equally distributed among the number of cells within the *FeatureObject*.

3.2.3 Simple Cell Mapping

We plot the feature values obtained per objective function with highlighting the variations of the *OneMax* and the *LeadingOnes* problems. Figures 3.3, 3.4, 3.5, 3.6 and 3.7 exhibits the plotted feature values for each of the 23 objective functions for different number of observations N.

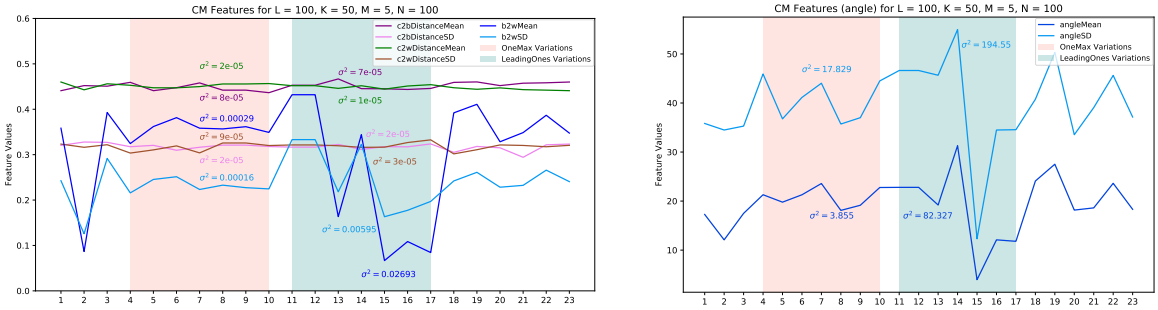

Figure 3.3: CM Angle $N = 100$

$N = 100$, it is observable that the features c2wDistance (Mean & SD) and c2bDistance (Mean & SD) do not show a high variance across the objective functions even though they look visually different. In particular, the variance of the aforementioned features for the variants of *OneMax* ($f4 - f10$) and *LeadingOnes* ($f11 - f17$) problems, exposed the lowest variance value of 0.00001 where the highest variance value was 0.00009, which is lower compared to b2w (Mean & SD) and angle (Mean & SD). However, we still consider them to expose difference between the functions. However, the pair $(f11, f12)$ had the same feature values for each feature, which makes them indistinguishable.

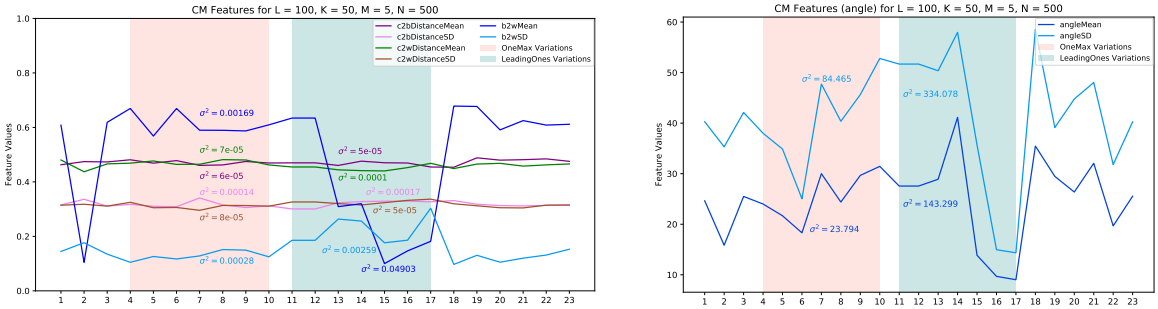

Figure 3.4: CM Angle $N = 500$

We would expect that increasing the number of samples would generate more stable results and therefore, features would be more reliable that exhibits higher variance. $N = 500$, the c2b and c2w features Mean and SD behaviour does not change as shown in Figure 3.4. However, there is a slight increases in the minimum and maximum variance yielded for the variation of the *OneMax* and *LeadingOnes* problems. The lowest variance is 0.00005 where the highest is 0.00017 for the four features of c2b and c2w. Importantly, increasing the number of samples

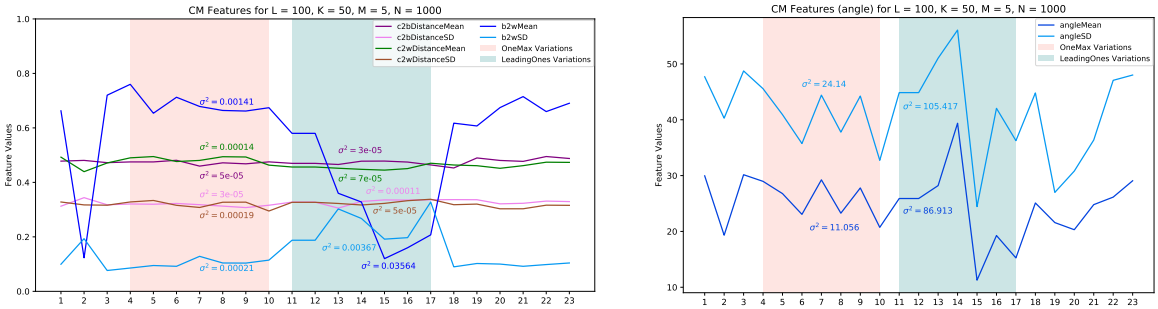

Figure 3.5: CM Angle $N = 1000$

influenced having higher variance in the features generated in oppose to a small sample such as 100. However, the two functions $(f11, f12)$ were not unique for all feature values.
$N = 1000$ shows, as plotted in Figure 3.5, the consistency observed with generating 500 observations with a slight change in variance of the features c2bDistance and c2wDistance (Mean & SD) for the groups in the shaded areas mentioned before. The minimum variance is 0.00003 and the maximum variance is 0.00019. It should be noted that the variance of the features did indeed decrease for few and increase for others. However, still the pair $(f11, f12)$ shred the same value for all feature values.

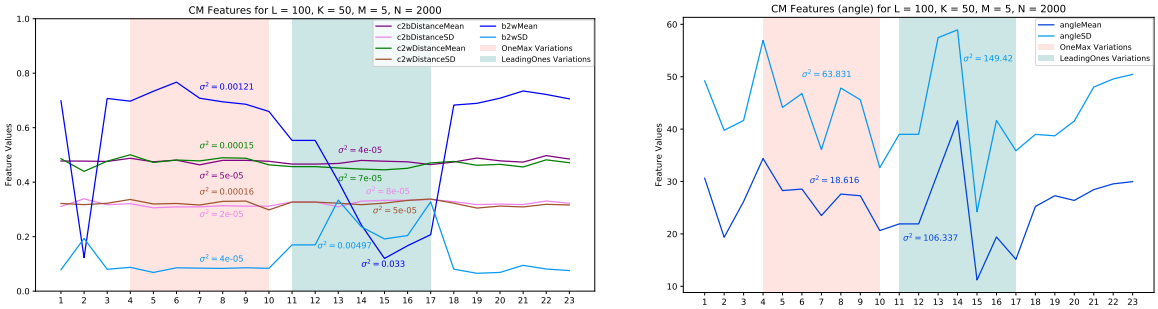

Figure 3.6: CM Angle $N = 2000$

$N = 2000$ anticipated a slightly different behaviour of the feature values. More specifically, some feature values increased where others decreased. Therefore, it was not an overall increase or decrease in values. In the group of variations, the lowest variance of c2bDistance and c2wDistance (Mean & SD) is 0.00002 and the highest is 0.00016 as shown in Figure 3.6. Moreover, we observe that $f11$ and $f12$ were not unqiue for all feature values.

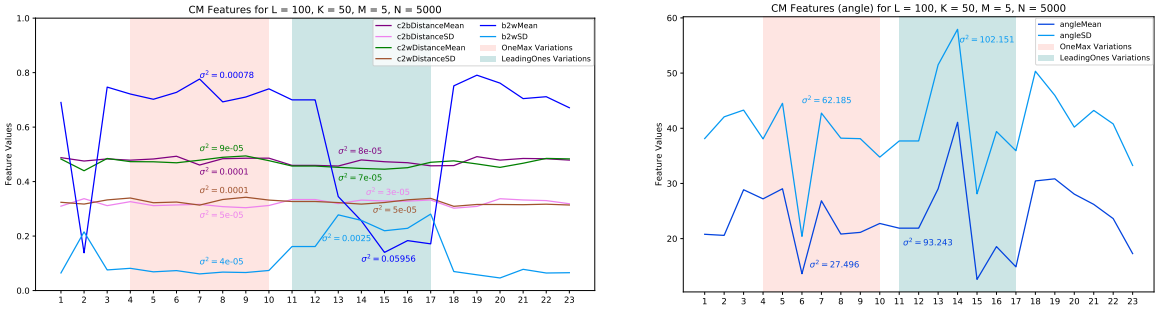

Figure 3.7: CM Angle $N = 5000$

$N = 5000$ observations, 50 times larger than the initial number of observations, the feature values relatively differ than other feature values with a lower number of observations as visible in Figure 3.7. It is worth to mention that there is a constant trend with the feature values of b2wMean and b2wSD for $f2$. Moreover, the variance yielded for the two groups of problems variations for the features c2bDistance and c2wDistance (Mean & SD) yielded the lowest variance of 0.00003 while the highest variance was 0.0001. Furthermore, $f11$ and $f12$ shared the same value for each feature.

Overall, we believe that the CM Angle feature exhibit visual and numerical difference despite the low variance for different number of observations across the 23 functions. Obviously, the angle (Mean & SD) feature values are variable for each of the problems and could be of good use. However, the highest angleMean recorded across all number of observations for all problems is $\approx 60°$, which means that the distance between the best and worst points from the center is not as ideal as we want it $(180°)$. In addition $f11$ and $f12$ were not differentiated by any feature for different number of observations.

3.2.4 Generalized Cell Mapping

As noted previously, the GCM method generates 23 features per approach: minimum, mean and near. We construct two types of plots namely, a line plot which includes only the reported mean value for features that are calculated based on different statistical measures and a scatter plot of all features per approach that shows where features values were equal w.r.t each of the functions. The goal is to again examine that the generated features are different or at least some of the features are different for each of the approaches in order to recognize different functions. It is worth to mention that *flacco* always sets the sum basin size for uncertain cells to 1, which leads to a feature that is not unique for any problem. Therefore, we remove this feature from GCM for all approaches and we end up with 22 features per approach in order to avoid any issues with predictive modeling and classification tasks later on. In the next paragraphs, we inspect the GCM features per approach for each number of observations generated.

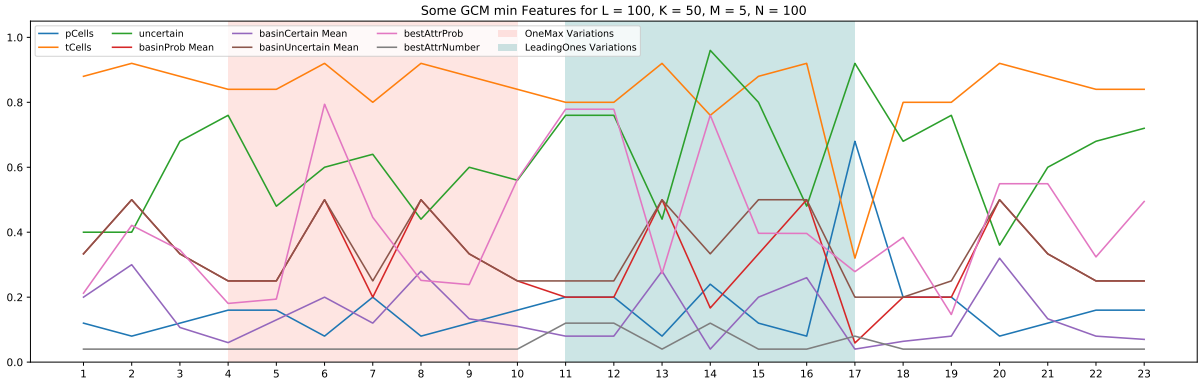

Figure 3.8: GCM Minimum Approach $N = 100$

100 observations minimum approach shows that the selected features in Figure 3.8 do show a general variance in the results for each of the objective functions. However, it is observable that there are some features that had a constant value across a range of function ids or pairs. For example, pcells shows an equal feature value for the pair $(f4, f5)$.

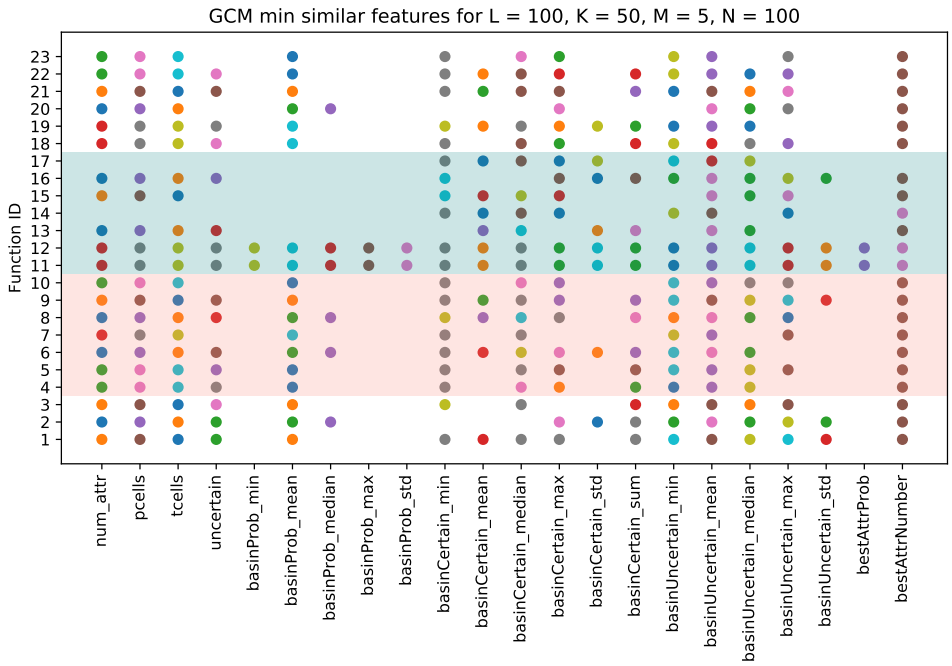

Figure 3.9: GCM Similar Features Minimum Approach $N = 100$

To analyze this further Figure 3.9, indicates where each of the features values are the same w.r.t the function id. The colors in each column resembles which function ids have the same value. Overall, most of the function values had a constant value, which makes them non-distinctive to the 23 functions. It is worthy to note that the most unique feature values were basinProb (min, median, max and standard deviation), basinUncertain_std and bestAttreProb while the worst feature were basinUncertain_mean and bestAttrNumber with no unique values or one unique value. Furthermore, the pair $(f11, f12)$ were not different across all feature values, which makes them unrecognizable whereas the most unique functions were $f14$ and $f17$ with 14 unique features (64.3%).

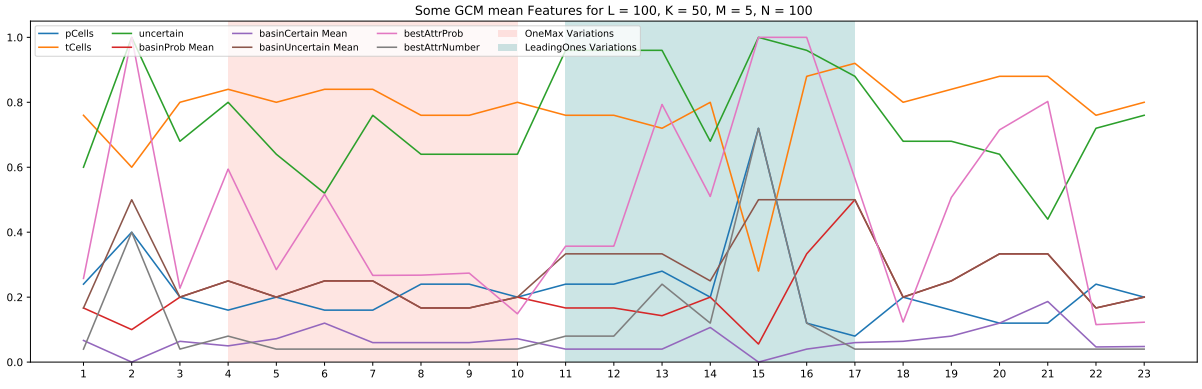

Figure 3.10: GCM Mean Approach $N = 100$.

100 observations mean approach feature values are different w.r.t the minimum approach presented previously as seen in Figure 3.10. It is noticeable that the variance of the feature values increased for few functions and decreased for others. To examine this further, Figure 3.11 stipulates that the most unique features were the min, median, max and standard deviation

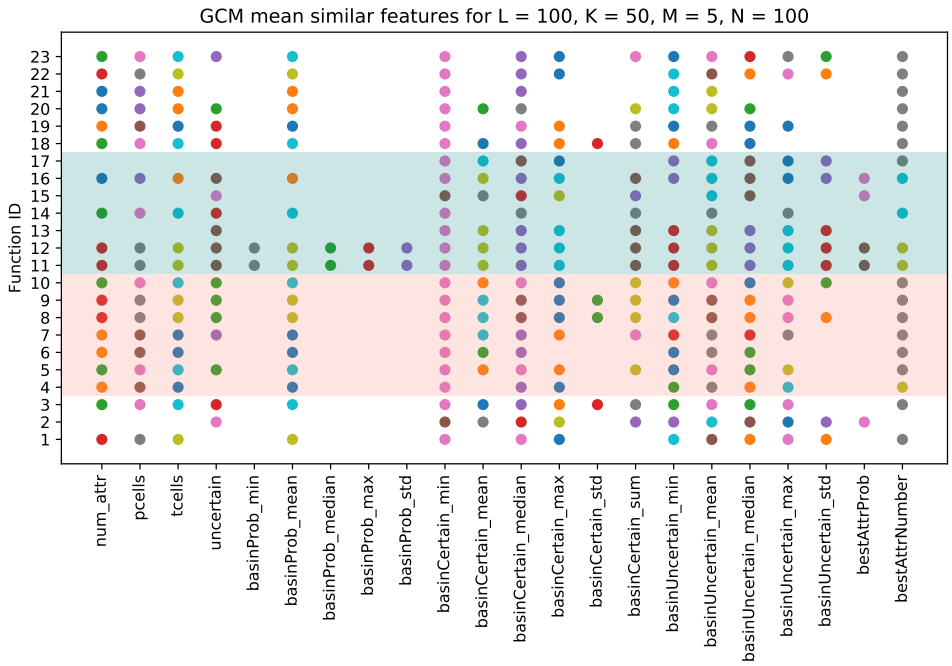

Figure 3.11: GCM Similar Features Mean Approach $N = 100$

of the basinProb feature where they were only not unique for the functions pair $(f11, f12)$. Moreover, basinCertain_std and bestAttrProb were also among the most unique features. On the other side, the worst features where no or at least one or two unique features were basinCertain (min and median), basinUncertain (min, mean and median) and bestAttrNumber. Furthermore, unlike the minimum approach the function pair $(f11, f12)$ were distinguishable by the basinCertain_std feature whereas it was not across all other functions in the mean approach. Also the most unique functions were $f15$ and $f21$ with 13 features (59%).

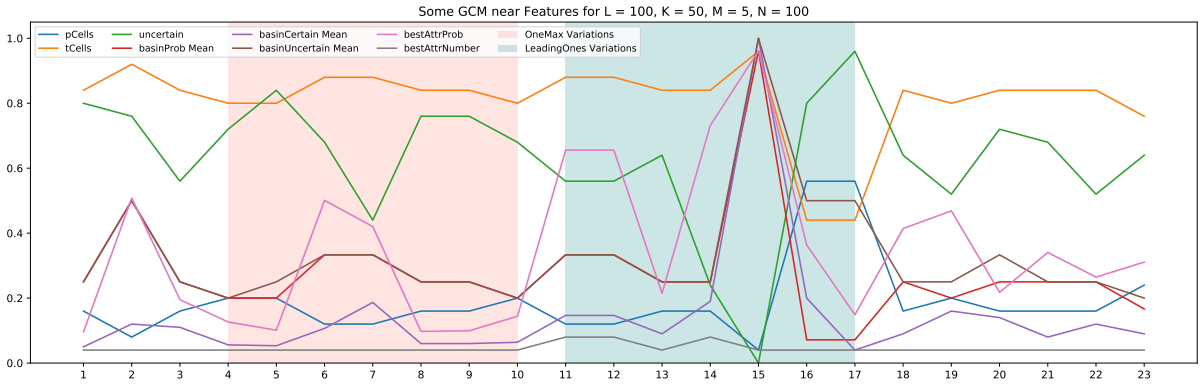

Figure 3.12: GCM Near Approach $N = 100$

100 observations near approach feature values exhibits a higher variance for the selected features than other approaches presented as presented in Figure 3.12. We notice that for $f15$ the feature values collapse to top region and to the bottom region as well, which can indicate that this function attained unique feature values. Figure 3.13 shows where the feature values were constant for all the functions. Indeed, $f15$ was unique for all feature values (95.5%) except for NumAttrNumber, which is the worst for all approaches as it does not distinguish any func-

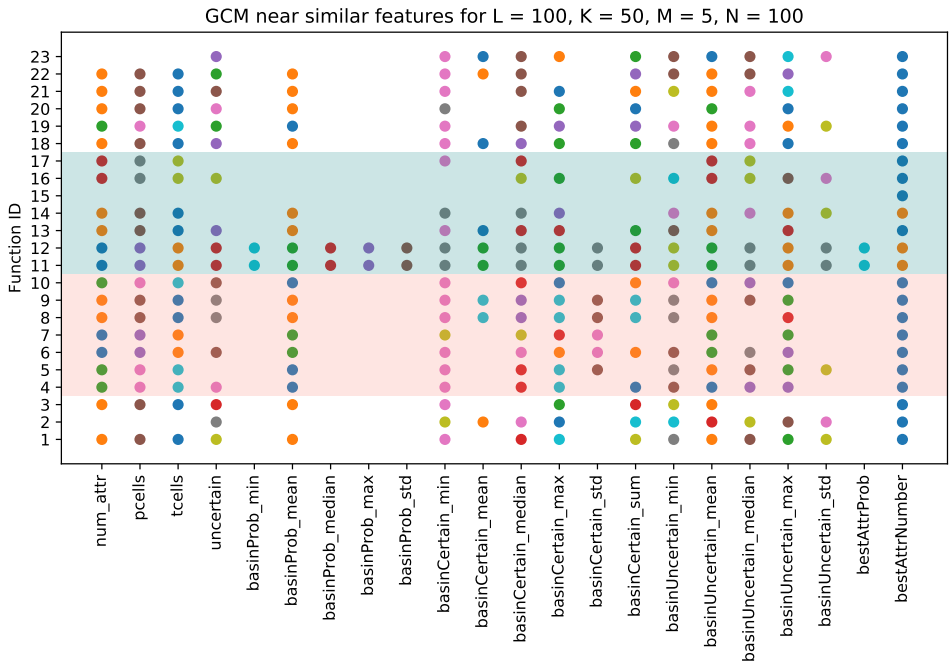

Figure 3.13: GCM Similar Features Near Approach $N = 100$

tions. The second worst feature in the near approach were basinCertain_min and basinUncertain_mean. In opposition, the best features were basinProb (min, median, max and standard deviation), basinCertain_std and bestAttrProb. Moreover, the functions pair $(f11, f12)$ had no unique values at any feature values. Therefore, based on the provided analysis of 100 observations for all approaches, we can conclude the function pair $(f11, f12)$ are not distinguishable by any of the feature values. Next, we increase the number of observations and analyze the trends.

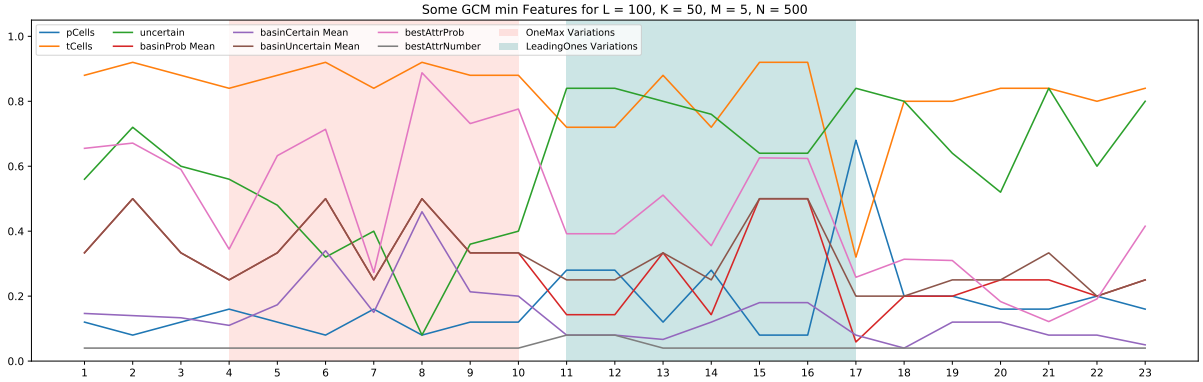

Figure 3.14: GCM Minimum Approach $N = 500$

500 observations minimum approach unveil a visual variance in the selected feature values in Figure 3.14. From the plot, $f15$ and $f16$ had a constant value for each of the feature values reported. Again, we examine where the features values were constant w.r.t the functions as shown in Figure 3.15. The most unique features were basinProb (min, median, max and standard deviation) and bestAttrProb while the worst features were bestAttrNumber, basinUncertain (mean and min), basinCertain_min, num_attr, pcells and tcells. Moreover, The most

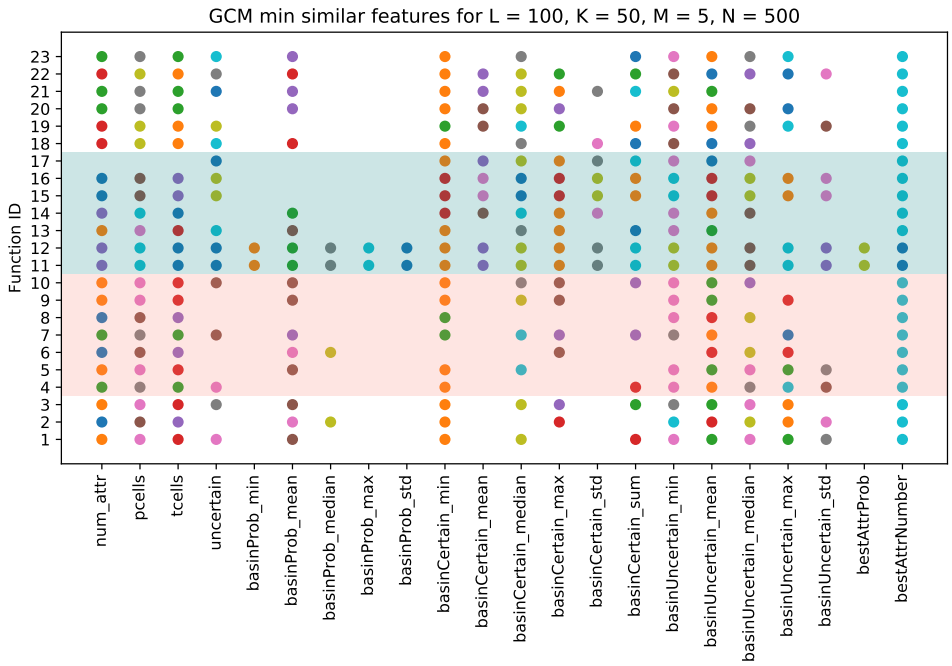

Figure 3.15: GCM Similar Features Minimum Approach $N = 500$

unique function was $f8$ with 14 features (63.3%) while the pair $(f11, f12)$ shared the same feature value for all features. Also the pair $(f15, f16)$ had the same feature values expect for the basinProb 5 features and bestAttrProb.

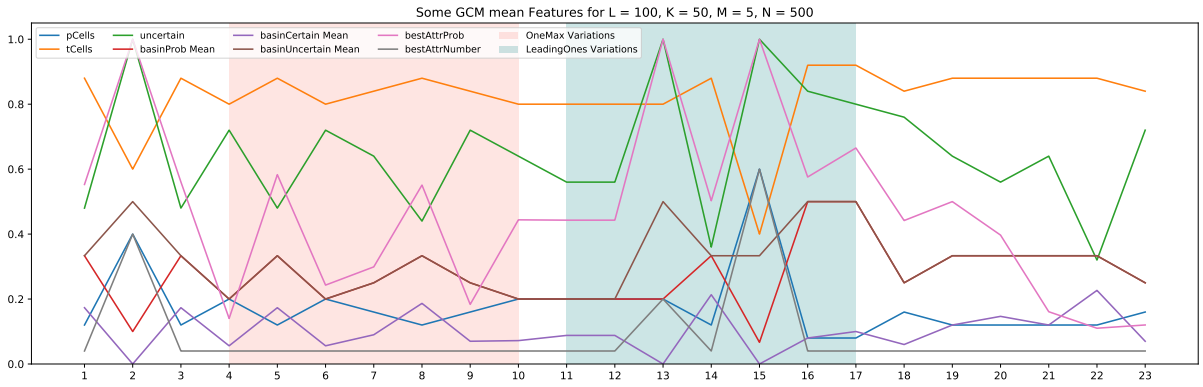

Figure 3.16: GCM Mean Approach $N = 500$

500 observations mean approach shows some feature values collapsing to specific values as displayed in Figure 3.16 in addition to a noticeable variance in the selected feature values. The most unique feature values were basinProb (min, median, max and standard deviation) and bestAttrProb as shown in Figure 3.17 while the worst features were basinCertain_min and basinUncertain_mean. Once again the pair $(f11, f12)$ had the same value for each of the feature values and the most unique functions were $f15$ and $f17$ with 11 features (50%).

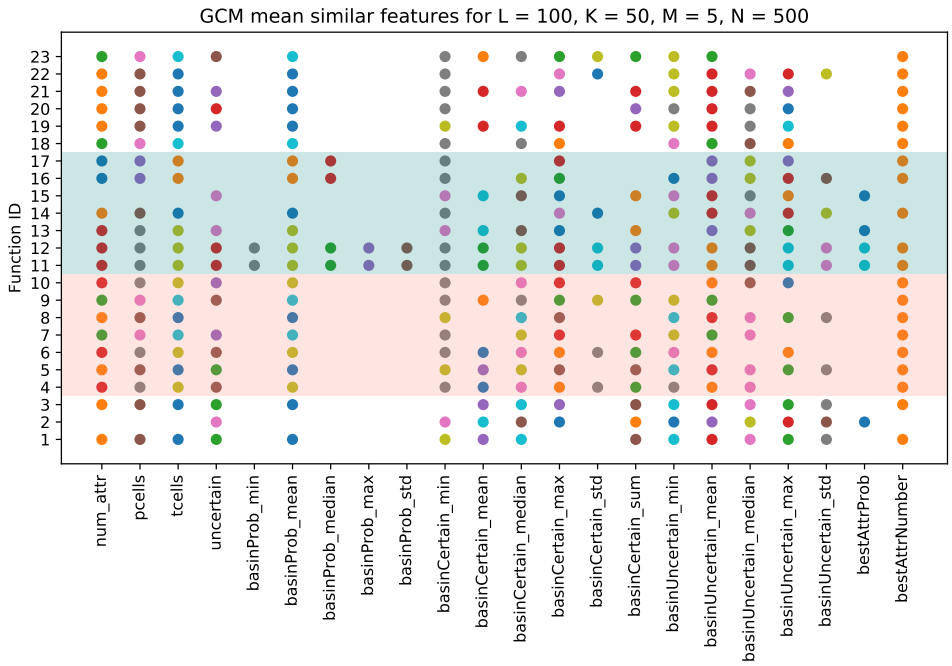

Figure 3.17: GCM Similar Features Mean Approach $N = 500$

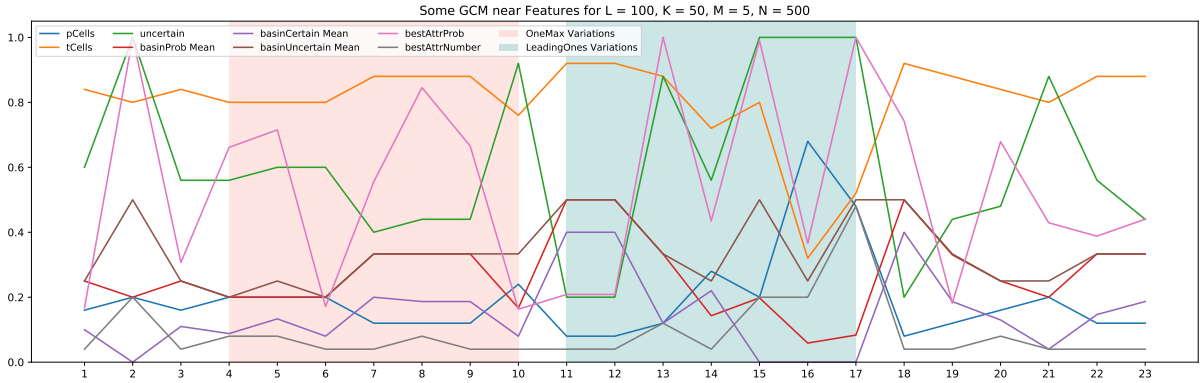

Figure 3.18: GCM Near Approach $N = 500$

500 observations near approach shows the highest variance compared to other approaches as exhibited in Figure 3.18. Moreover, the similarity trend is not as strong as the two other approaches presented. The most unique feature values as shown in Figure 3.19 were basinProb (min, median, max and standard deviation), basinUncertain_std, basinCertain_std and bestAttrProb while the worst features were basinCertain_min, basinUncertain (min and mean) and bestAttrNumber. The function with the least similar features was $f10$ with 15 features (68.2%) while the pair $(f11, f12)$ had the same value for each of the feature values.

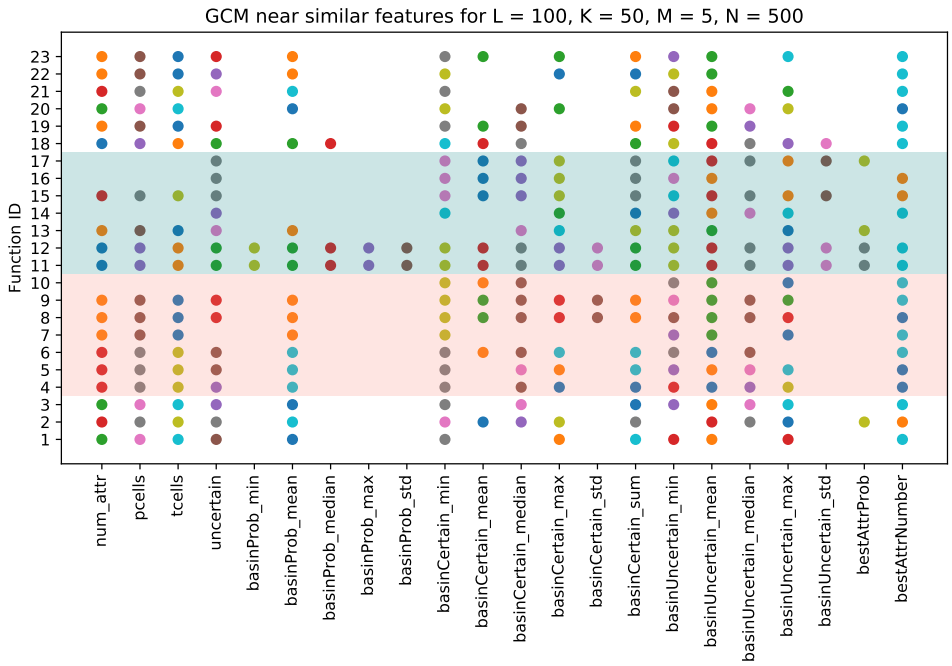

Figure 3.19: GCM Similar Features Near Approach $N = 500$

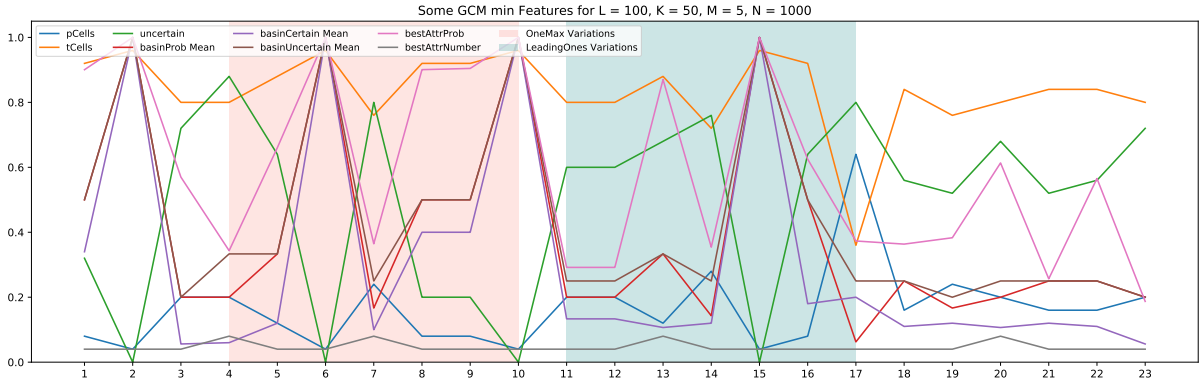

Figure 3.20: GCM Minimum Approach $N = 1000$

1000 observations minimum approach in Figure 3.20 displays a detectable change in the trend of feature values for each function especially in terms of feature value variability. Even though the variance is higher, this does not necessarily mean that the feature values are unique for all functions. According to Figure 3.21, the worst features were bestAttrNumber, basinUncertain (min and mean), basinCertain (min and max), pcells, tcells and num_attr. whereas the most unique feature values were basinProb (min, median, max and standard deviation), basinUncertain_std and bestAttrProb. Functions $f11$ and $f12$ were not assigned any unique feature values where as the most unique function was $f17$ with 15 features (68.2%).

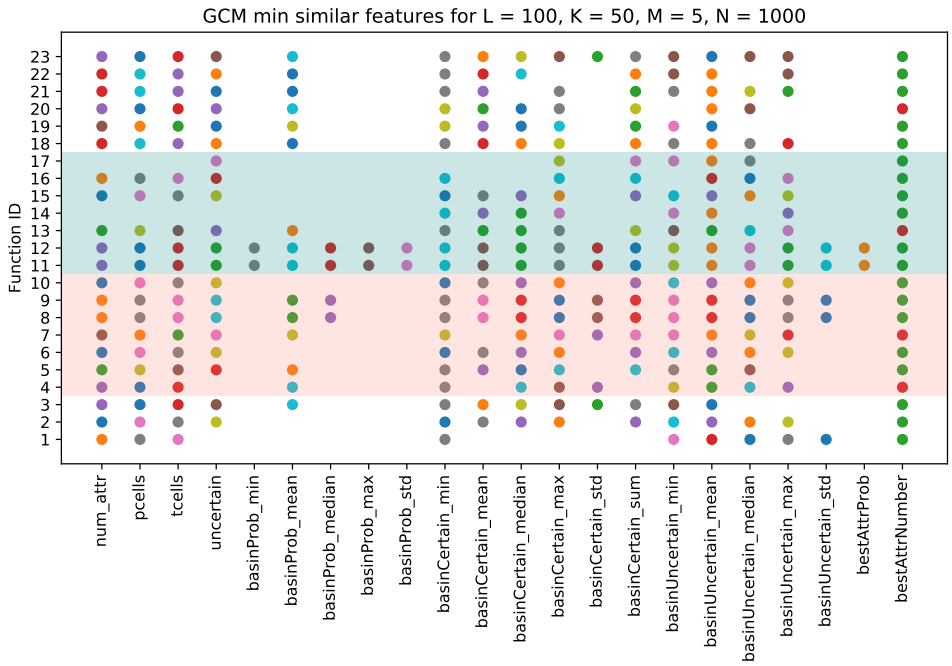

Figure 3.21: GCM Similar Features Minimum Approach $N = 1000$

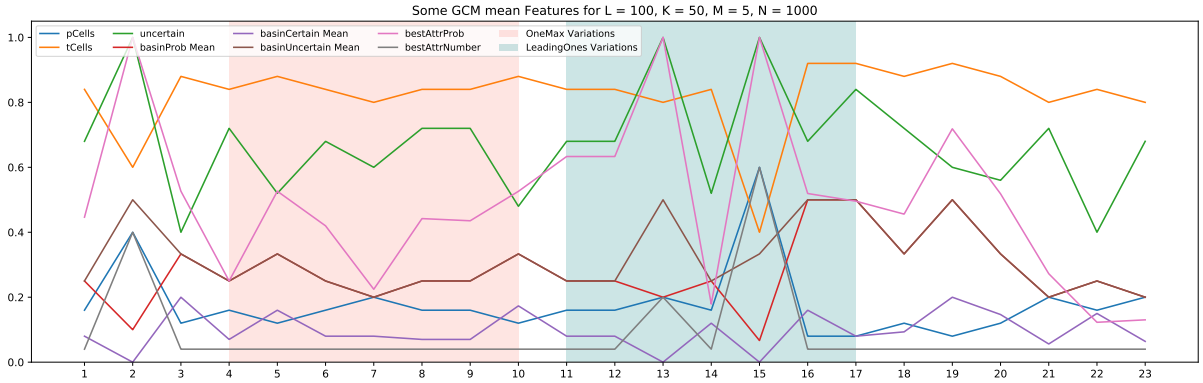

Figure 3.22: GCM Mean Approach $N = 1000$

1000 observations mean approach in Figure 3.22 has a lower variance than the minimum approach. Therefore, we expect to have more constant feature values. based on Figure 3.23, the most unique feature values were basinProb (min, median, max and standard deviation) and bestAttrProb. On the other hand, the worst features were num_attr, pcells, tcells, uncertain, basinPorb_mean, basinCertain (min, median, sum), basinUncertain (min and mean) and bestAttrNumber. The most unique function was $f20$ with 13 features (59%) while functions $f11$ and $f12$ had the same feature values for all features, which make them identical.

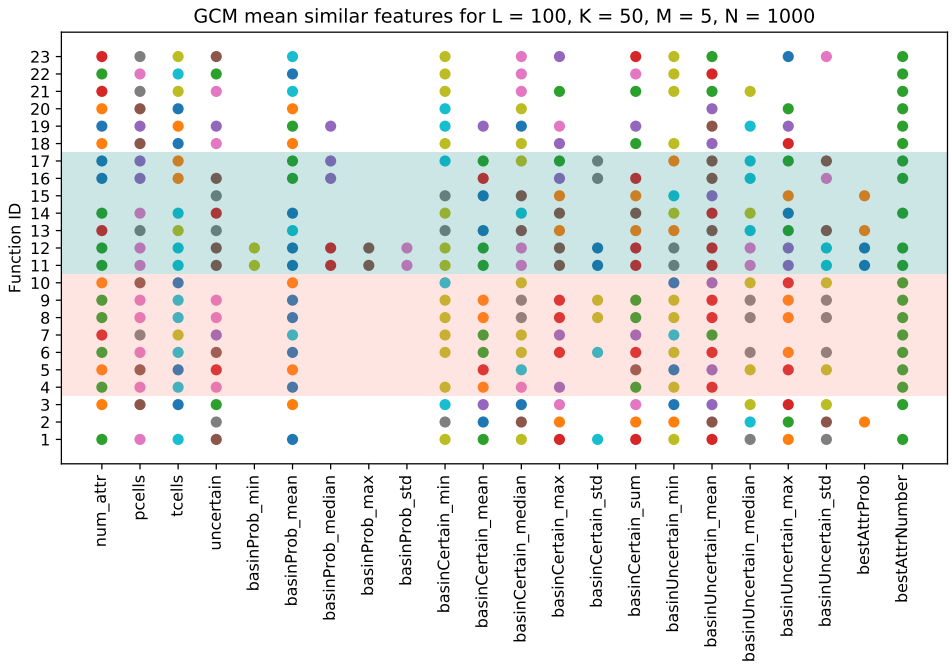

Figure 3.23: GCM Similar Features Mean Approach $N = 1000$

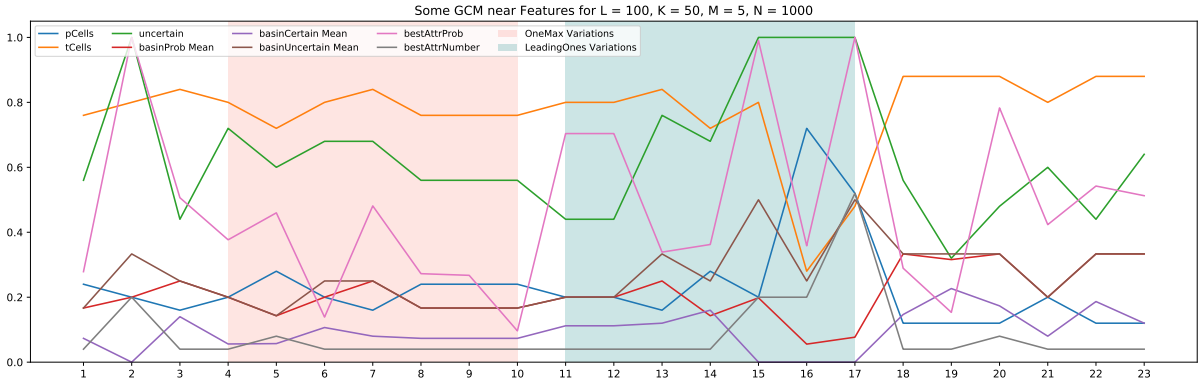

Figure 3.24: GCM Near Approach $N = 1000$

1000 observations near approach in Figure 3.24 has also a lower variance than the minimum approach, but higher than the mean approach. Looking at Figure 3.25, the most unique feature values were basinProb (min, median, max and standard deviation), basinUncertain_std and bestAttrProb. In contrast, the worst features were num_attr, pcells, tcells , basinCertain (min, median), basinUncertain_mean and bestAttrNumber. The most alike functions were $f11$ and $f12$, which had the same feature values for all features and the most unique function is $f19$ with 13 features (59%).

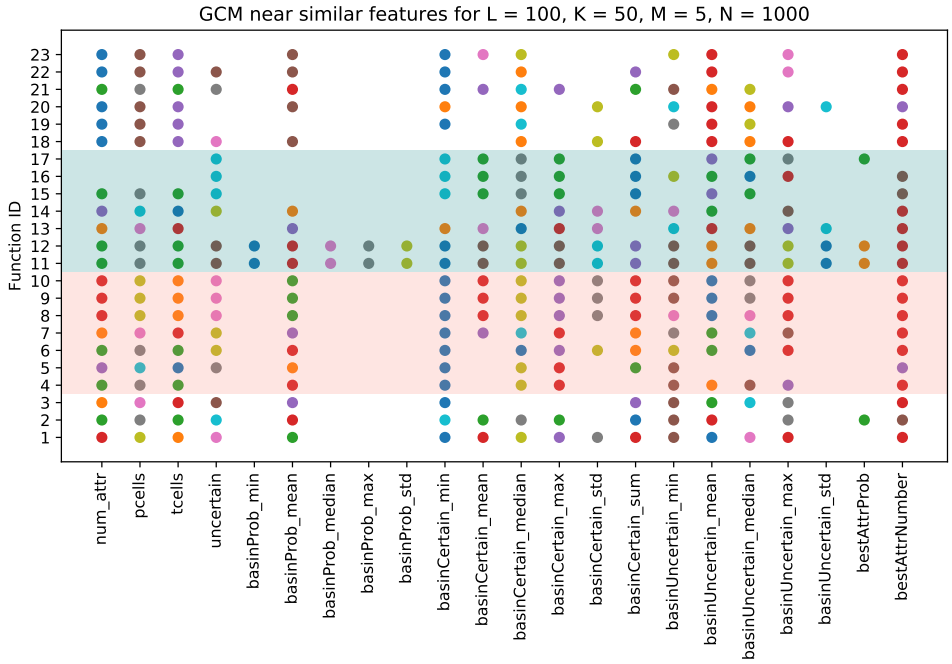

Figure 3.25: GCM Similar Features Near Approach $N = 1000$

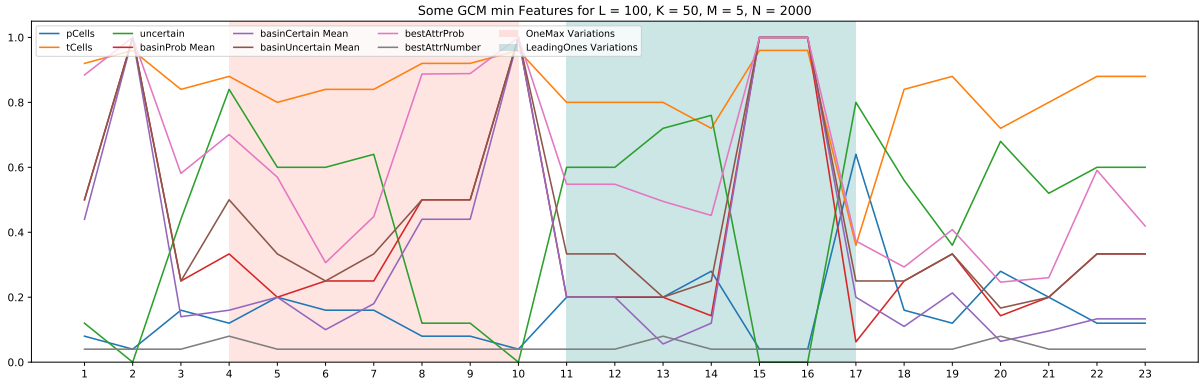

Figure 3.26: GCM Minimum Approach $N = 2000$

2000 observations minimum approach discloses a good variability among feature values as seen in Figure 3.26. Furthermore, a collapse for some feature values is obvious for functions $f2, f10, f15$ and $f16$. Examining Figure 3.27, the most unique feature values were basinProb (min, median, max and standard deviation) and bestAttrProb. Moreover, the worst features were num_attr, pcells, tcells , basinCertain (min, median, max), basinUncertain (min, mean and max) and bestAttrNumber. The most different function is $f17$ with 14 unique features (63.3%) while functions $f11$ and $f12$ had the same feature values for all features.

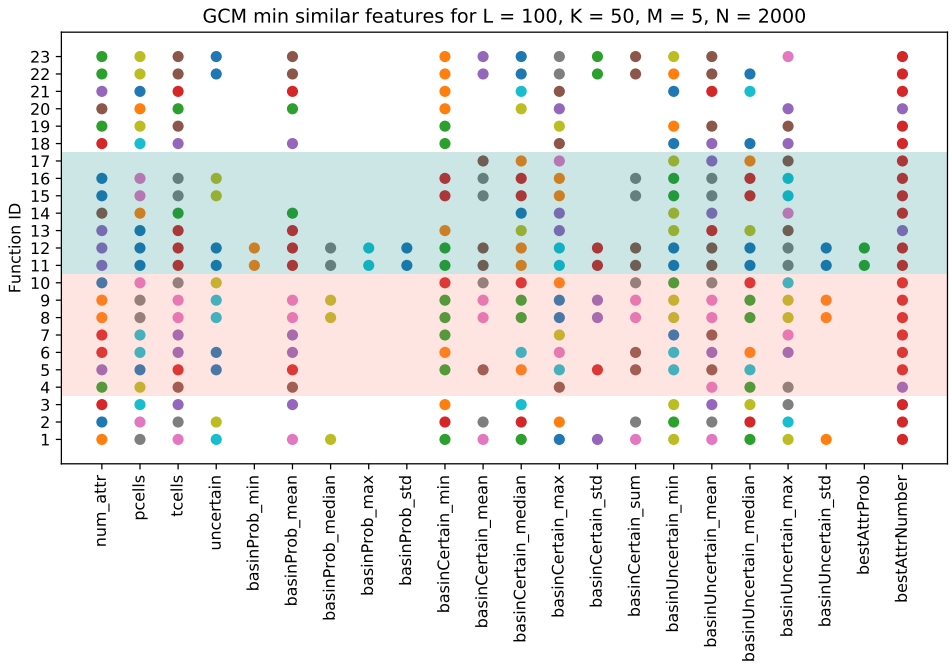

Figure 3.27: GCM Similar Features Minimum Approach $N = 2000$

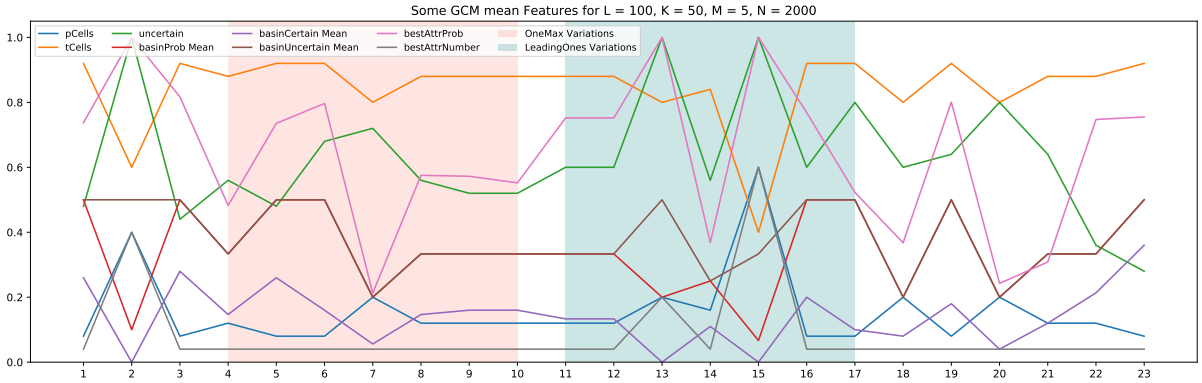

Figure 3.28: GCM Mean Approach $N = 2000$

2000 observations mean approach in Figure 3.28 shows a lower variance than the minimum approach with a collapse of feature values uncertain and bestAttrProb to the same value for functions $f2, f13$ and $f15$. Furthermore, inspecting Figure 3.29 shows that $f11$ and $f12$ are not unique for any feature values, but $f14$ and $f15$ were the most unique with 13 features (59%). In addition, the worst features are num_attr, pcells, tcells, basinProb_mean, basinCertain_min, basinUncertain (min, mean) and bestAttrNumber whereas the best features were basinProb (min, max and standard deviation) and bestAttrProb.

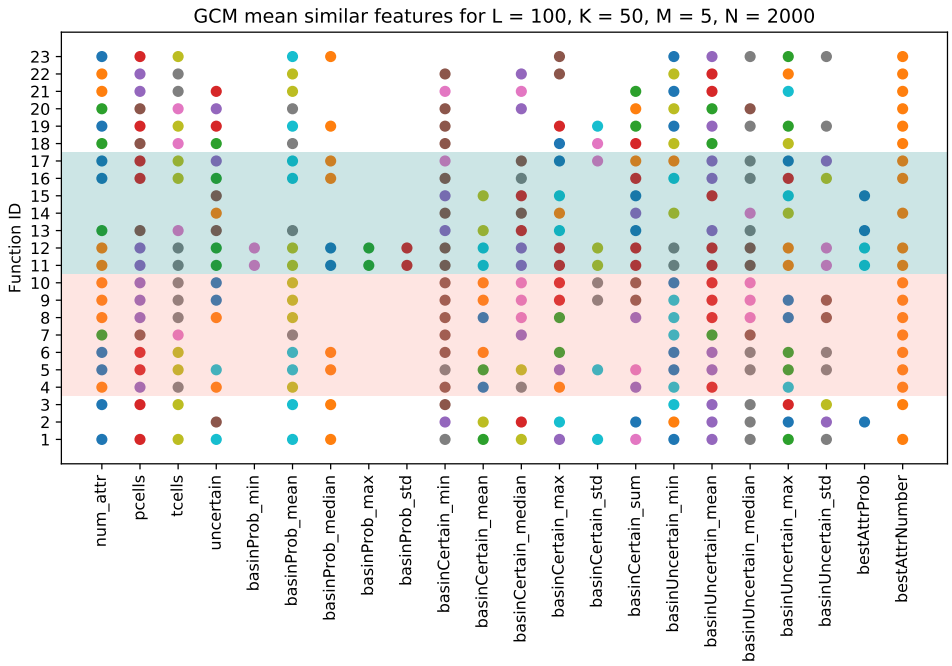

Figure 3.29: GCM Similar Features Mean Approach $N = 2000$

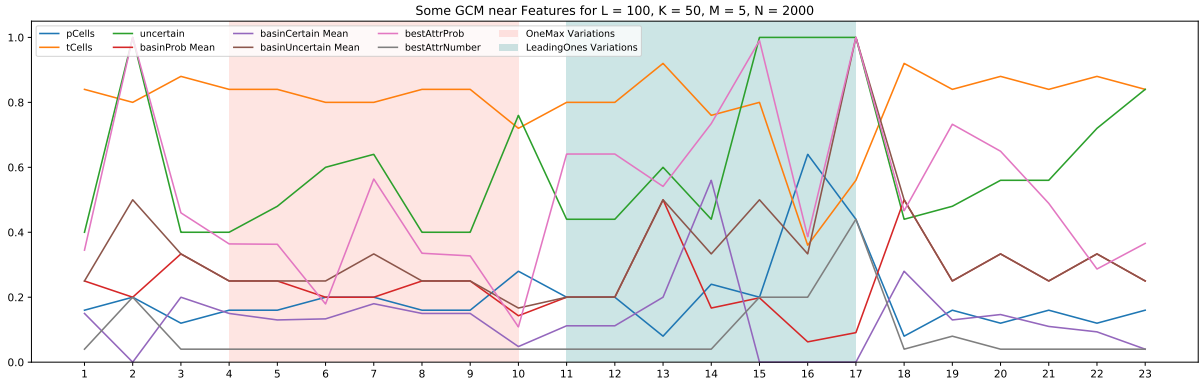

Figure 3.30: GCM Near Approach $N = 2000$

2000 observations near approach, see Figure 3.30, comes in the middle between the minimum and mean approach in terms of variability. To investigate further, Figure 3.31 displays that the most unique feature values where basinProb (min, median, max and standard deviation) and bestAttrProb. At the same time, the worst features were basinUncertain (mean and max) and bestAttrNumber. Furthermore, the function pair $(f11, f12)$ did not have any unique feature values whereas $f10$ is the most unique with 18 features (81.8%).

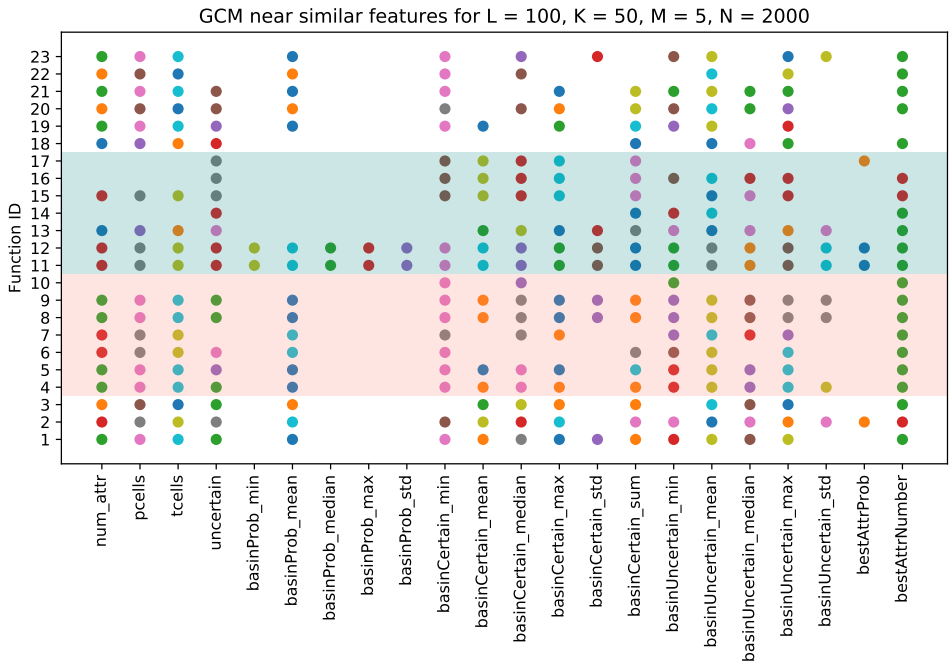

Figure 3.31: GCM Similar Features Near Approach $N = 2000$

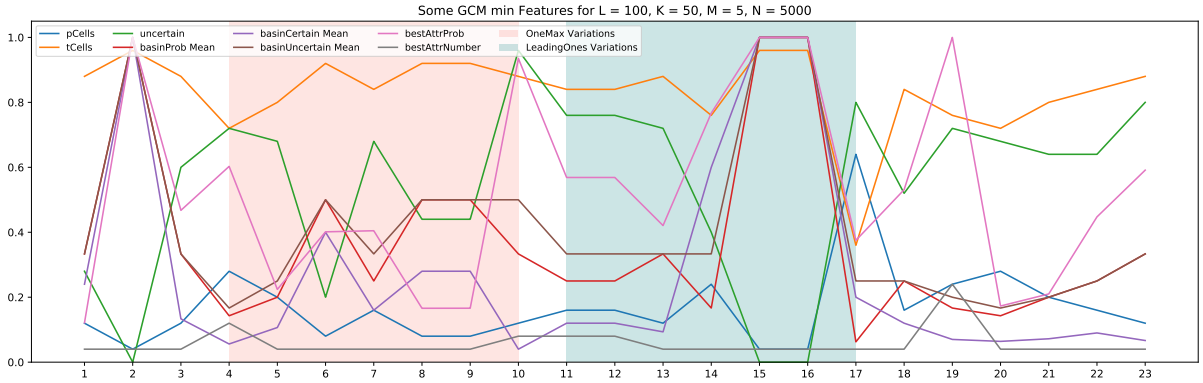

Figure 3.32: GCM Minimum Approach $N = 5000$

5000 observations minimum approach comes with the highest variance of feature values as show in Figure 3.32. There is a collapse of selected plotted values for $f2$. In addition a constant trend specially for the $LeadingOnes$ variations. To discuss this further, Figure 3.33 indicates that $f2$ had 7 unique features (31.8%) and the most unique functions were $f6$ and $f17$ with 14 features (63.6%). Nevertheless, the pair $(f11, f12)$ had no unique values for all feature values. The most unique features were basinProb (min, median, max and standard deviation), basinUncertain_std and bestAttrProb while the worst features were bestAttrNumber, num_attr, pcells, tcells and basinUncertain_mean.

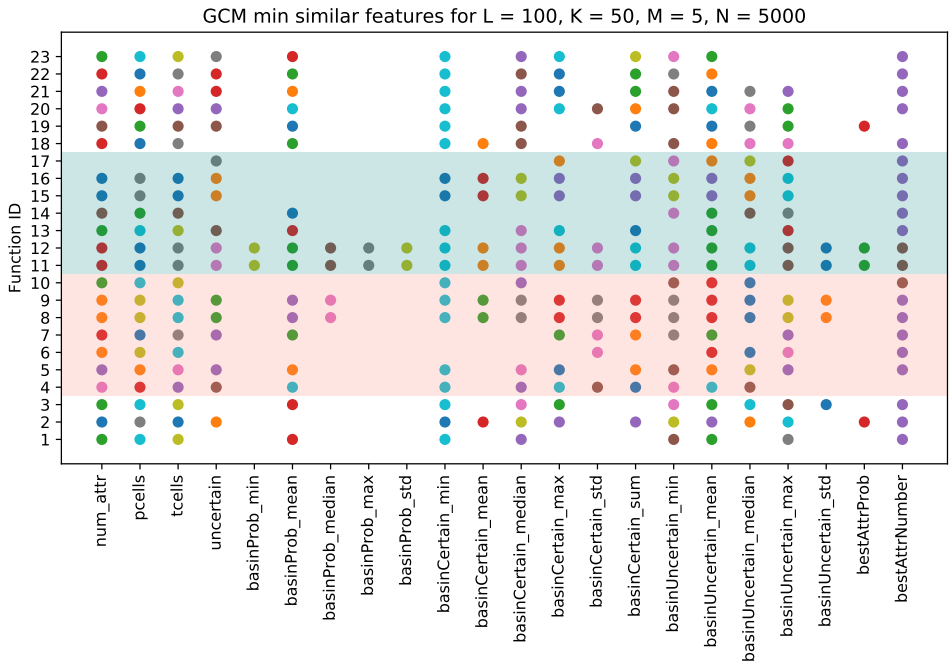

Figure 3.33: GCM Similar Features Minimum Approach $N = 5000$

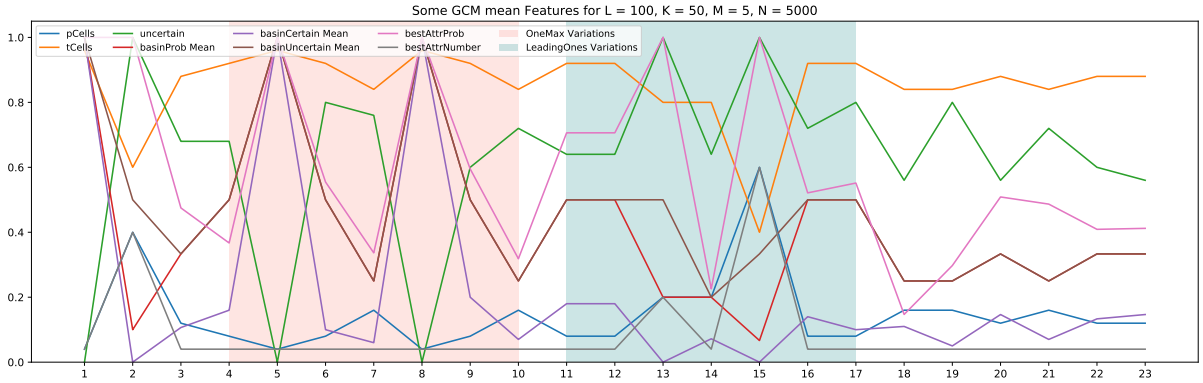

Figure 3.34: GCM Mean Approach $N = 5000$

5000 observations mean approach comes after the minimum approach in attained variance. A collapse of feature values is present for $f1, f5, f8$ and a mild collapse for $f13$ and $f15$ as shown in Figure 3.34. To confirm the uniqueness of the feature values w.r.t the functions, Figure 3.35 displays that the aforementioned functions that had the collapse were at most unique in 4 feature values (18.2%) while the most distinctive function is $f7$ with 11 features (50%). In addition the pair $(f11, f12)$ were not recognizable by any features. The worst features are num_attr, pcells, tcells, uncertain, basinCertain (min, median and max), basinUncertain (sum, min, mean, median and max) and bestAttrNumber. On the other hand, the best features are basinProb (min, max and standard deviation).

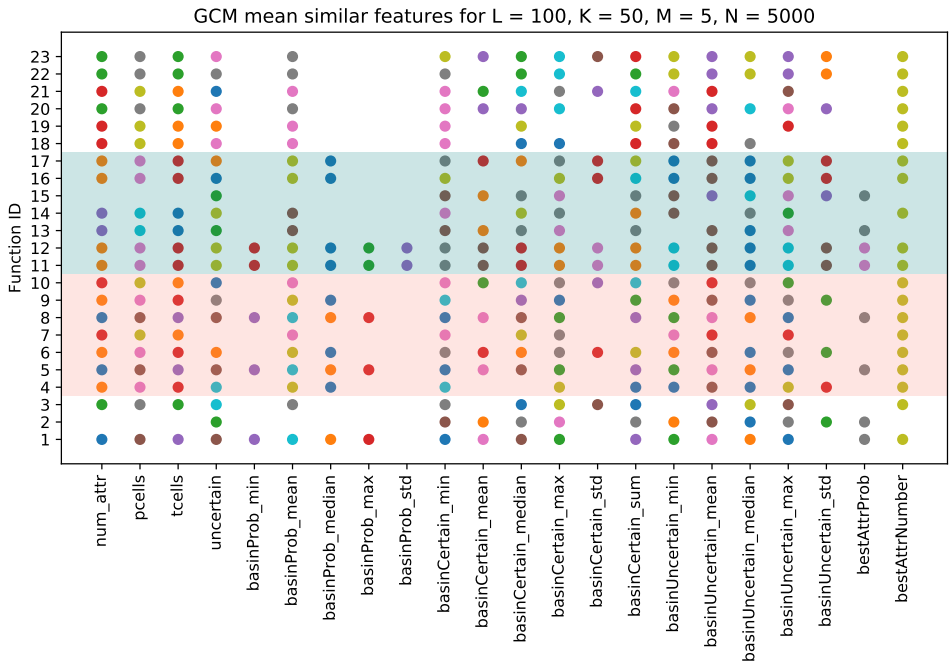

Figure 3.35: GCM Similar Features Mean Approach $N = 5000$

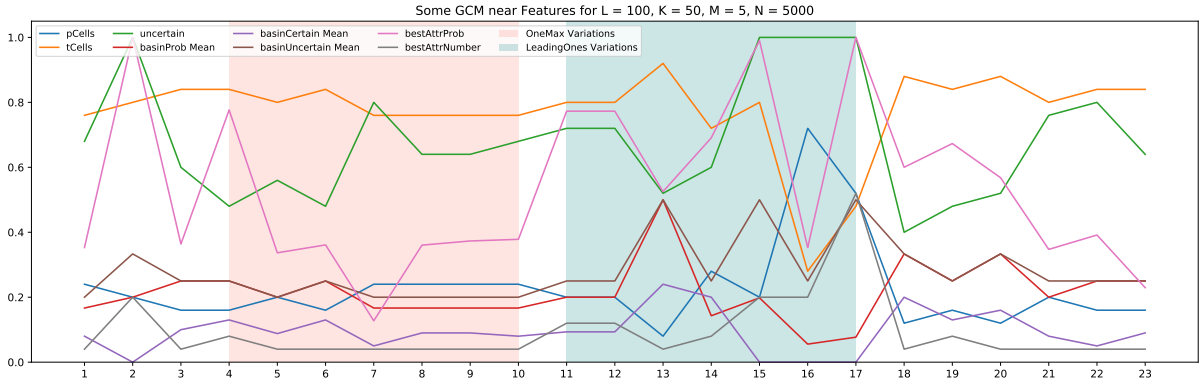

Figure 3.36: GCM Near Approach $N = 5000$

5000 observations near approach exhibits a low variance compared to other approaches and there is no strong unique trend in Figure 3.36 except for a constant trend. Therefore, we expect more constant values than other approaches. To explore this further, Figure 3.37 shows that the most unique features are basinProb (min, median, max and standard deviation) and bestAttrProb while the worst features are uncertain, basinCertain (min, mean, max and sum), basinUncertain (min, mean, median and max) and bestAttrNumber. Last, the most unique functions were $f13$ and $f14$ with 13 features (59%) whereas the pair $(f11, f12)$ had not unique values for all features.

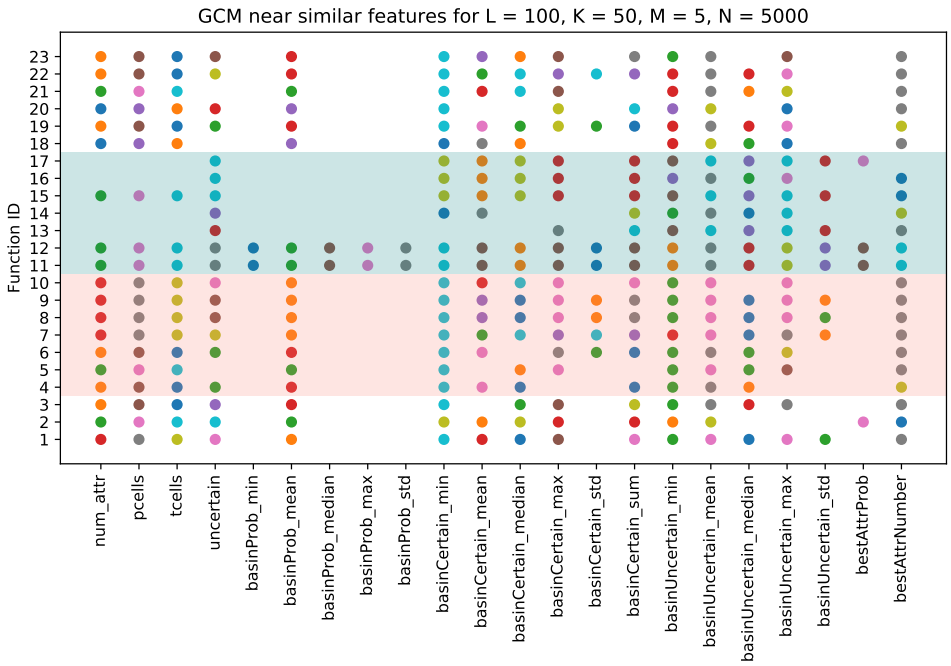

Figure 3.37: GCM Similar Features Near Approach $N = 5000$

Overall, within GCM is there no features that are distinctive for the objective functions per approach in general. However, we believe that combining features from different approaches would help to create a set of features that is unique. It is worthy to mention that the most unique features across all number of observations generated for the minimum and near approaches were basinProb (min, median, max and standard deviation), bestAttrProb and basinUncertain_std. In addition, the mean approach gave the same best features without the basinUncertain_std. On the other hand, the worst features for the min appraoch were basinUncertain (min and mean), bestAttrNumber, numAttr, pcells and tcells. Also the worst features for the mean approach were basinUncertain (min and mean), basinCertain (min and median), bestAttrNumber, numAttr, pcells and tcells. Moreover, the worst features for the near approach were only basinUncertain (min and mean) and bestAttrNumber. Therefore, We expect that least unique features are gonna be discarded when performing feature selection, while the best features will be retained.

In response to $Q2$, the CM Angle features manifest different features for all the 23 functions except for $f11, f12$. In addition, the highest number of unique feature recorded across all number of observations for the minimum approach was 68.2% ($N = 1000$), mean approach 59% ($N = 100, 1000$ and 2000) and near approach 95.5% ($N = 100$). So, when it comes to GCM features, the number of sample points needed to construct a reliable feature set varies per approach. Moreover, to answer $Q3$, while a combination of features can be used to characterize the objective functions, the pair $(f11, f12)$ were not proven to be different as a result of the feature inspection conducted. Given that, we consider the aforementioned functions to be the same.

Chapter 4

Modeling

Our main goal is to predict the Expected Run Time (ERT) or classify the best algorithm to solve an optimization problem. The ERT and the labels that indicate the best algorithm are attained from IOHprofiler [2] while the input that will be used is the generated features: SCM (angle) and GCM all approach for all number of observations. In order to achieve this, we built and experimented with various machine learning models. For the regression task, we decided to build a Linear Regression (LR), LASSO and Random Forest Regression (RFR). Moreover, for the classification task, we decided to build a Random Forest Classifier (RFC), Multinomial Logistic Regression (MLR) and Supper-Vector Classifier with RBF Kernel (SVC). In addition, we fix a random state ($random\_state = 1$) to make the models comparable if randomness is applicable. Last, we perform two feature selection methods: 1) Recursive Feature Elimination (Backward Elimination) with Leave-One-Out Cross-validation (LOOCV) using 50 Monte Carlo simulations and filter out only the features that were selected at least 50% of the time 2) All Relevant Feature Selection using Boruta algorithm. Last, we apply Hyperparamter Tunning for the classification models using MiP-EGO [22]. All training and validation split of the data is done using 5 random seeds: $\{19, 29, 52, 6, 67\}$. After fitting a model, the metrics on the training, validation and LOOCV are averaged and reported with their standard deviation. We note that the 50 runs of RFECV are conducted using 50 random splits of the data, but the model fitting of the selected features afterwards is done with the 5 random seeds.

4.1 Data

As mentioned earlier, the data set is built using a combination of the generated features, SCM (angle) and GCM (all approaches) along with the minimum ERT reported and the best algorithm it corresponds to from the *IOHprofiler*. In total, we have 115 observations given that we generated 5 different number of observations for the 23 functions; hence, $23 \cdot 5 = 115$. In addition, we have 8 features from SCM (angle) and $22 \cdot 3$, so we have 74 features in total. Then our data is a matrix of size (115x74). It is worthy to mention that we consider the ERT value and the classification label to be the same for all different number of observations for every problem. Moreover, we split the data set to a training (80%) and validation set (20%) with the *stratify* option enabled in classification data to assure that the classes in the training set are also in the validation set. Also, for some GCM features, the reported value is nan, so we replace it with the value of 0.
After attaining the results from the *IOHprofiler*, we inspect the minimum ERT (response variable) for the regression task and the distribution of the classes where each class is the

Minimum	μ	σ	Max
96.636	1249806.932	2971889.401	10281859.0

Table 4.1: Minimum, Mean, Standard Deviation and Maximum of ERT in the data set.

best algorithm that scored the least ERT w.r.t each function. Given that we have a relatively small number of observations and many features, we do not expect any regression model to be reliable. This is also because after running a statistical summary, we discovered that there is a very high standard deviation and a huge difference between the minimum and maximum ERT within that response variable as shown in Table 4.1.

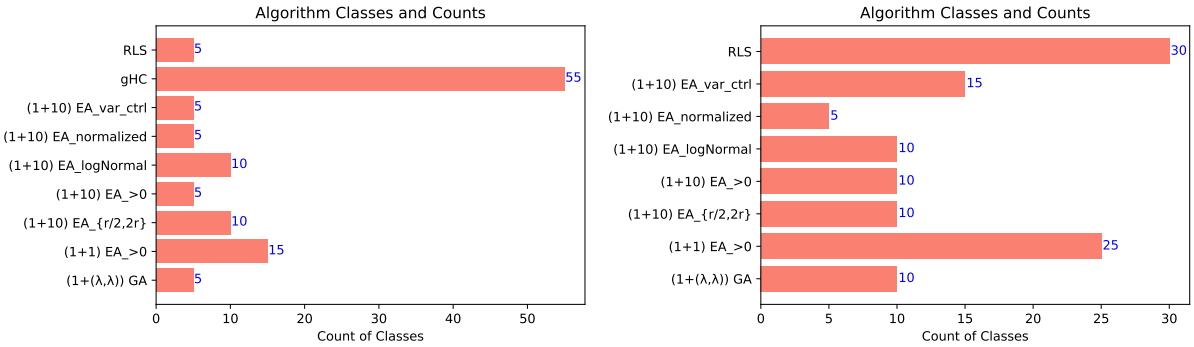

Figure 4.1: Algorithm classes counts. Left: original algorithm classes counts with gHC. Right: algorithm classes counts after removing gHC. Removing gHC results in more balanced classes.

Proceeding to the classification labels, we note that the labels (best algorithms) were not equally distributed, which can have a negative effect on classification models. Figure 4.1 left displays the original classes counts obtained. It is observable that the classes are unbalanced. *gHC* corresponds to 55 observations where the second highest *(1+1) EA_>0* is assigned to 15 observations. Therefore, we decided to remove the results of *gHC* and consider the algorithms results again. The new classes distribution exhibited in Figure 4.1 right looks slightly better and we have 8 classes in total. Therefore, we will continue with the aforementioned labels that does not include *gHC*.

4.2 Models

4.2.1 Linear Regression

Linear regression is a standard method used for numerical predictions, and describes the relationship between a response variable and a single or many explanatory ones. It is mainly based on a linear approach which tries to calculate an optimal prediction line according to the minimized mean squared error function. Y plays the role of response variable, and in this case, it reflects the ERT values. Predicted values of linear models represent single points on the regression line described by the following formula:

$$\hat{y} = \alpha + \beta_i x_i, i = 1, 2, 3, ..., p, \hat{y} \in \mathbb{R}$$

where α is the intercept of the regression line, and β_i $i = 1, 2..., p$ is the regression coefficient of regressor x_i. Intercept α represents the distance of the line from the origin in Cartesian representation. In other words, α is the value of $\hat{y}$ when all $x_i = 0$, and coefficient β_i is the angle between the regression line and the y = 0 one. Therefore, linear regression model outcome is the estimated average values of Gaussian distribution of Y given the x_i regressors. In addition, the Linear Regression model follows the assumptions that (1) Independence of observations, (2) Normality of residuals, (3) No multicollinearity, (4) No autocorrelation and (5) Homoscedasticity.

4.2.2 LASSO

LASSO is a type of Linear Regression that uses shrinkage to reduce model complexity and prevent overfitting that is usually caused by Linear Regression. The meaning of shrinkage is where the data are shrunk to a central point e.g. the mean. a LASSO model uses the $L1$ regularization method, which adds a penalty λ that is equal to the absolute value of the magnitude of the coefficients β. This particular type of regularization method results in smaller models (fewer βs) and this is because that some coefficients can become zero. Therefore, these coefficients are removed from the model. Furthermore, large penalties result in coefficients that are closer to zero, which is optimal to produce simpler models.

4.2.3 Random Forests

Random Forests is method that depends on ensemble learning. This means that it operates by creating multiple decision trees during training and outputs the class or the mean prediction for regression. Intuitively, for each decision tree constructed during training the classification or prediction is based on averaging the result of each decision tree. Therefore, it is suitable for both classification and regression tasks. Moreover, it is known that Random Forests corrects the prediction of each tree and therefore prevents overfitting the data. However, it comes with a high variability in the results.

4.2.4 Multinomial Logistic Regression

The second classification model that we will use is Multinomial Logistic Regression. Unlike Linear Regression, it does not have any particular assumptions that have to be met, and we do not assume any specific distribution of the features (predictors). The response variable Y has categories $j = 1, 2, ..., C$, where in this case, $C = 8$. Then the probability of an instance being in class j is given by

$$\pi_j = P(Y_i = j), j = 1, 2, ..., C$$

Here, for C categories one of them is chosen to be the baseline category (one-vs-all). This results in $C - 1$ regression equations that construct together the model. Furthermore, the baseline category is the one to which all other categories are constant. When choosing the first category as the baseline, the $C - 1$ regression equations are as follows:

$$log\frac{\pi_j(x_i)}{\pi_1(x_i)} = \alpha_j + x_i^T\beta_j = \alpha_j + \sum_p x_{ip}\beta_{pj}, j = 2, 3, ..., C$$

where α_j is the intercept of class j, x_i^T is the feature vector of the i^{th} instance and β_j is the vector of regression coefficients for class j. The baseline category intercept and vector of coefficients are set to $\alpha_1 = \beta_1 = 0$. As a result, the probability that instance i belongs to class j is given by:

$$\pi_j(x_i) = \frac{exp(\alpha_j + x_i^T \beta_j)}{\sum_h exp(\alpha_h = x_i^T \beta_h)}$$

Instance i is assigned to the class with the highest probability. In addition, it is important to mentioned that we use the Limited-memory BFGS (lbfgs) optimization method as it is recommended for multinomial cases and it uses the L2 regularization (LASSO) method by default. The MLR model minimize the categorical cross-entropy given by:

$$L = -\frac{1}{n} \sum_{i=1}^{n} \sum_j y_{ij} log(\pi_j(x_i))$$

where n is the number of training instances.

4.2.5 Support-vector Machine

The last classification model we are using for training is Support-vector Machine (SVM). SVM is a known to be a set of supervised learning methods that are used for classification, regression and outliers detection. SVM follows the idea of creating a hyperplane or set of hyperplanes in high dimensional space. Optimally, a sepration is achieved by the maximizing the distance between the hyperplance and the nearest training data instance of any category, which is called the functional margin (M). Therefore, the larger the margin, the lower error of the classifier. In addition, it comes with many advantages such as effectiveness in high dimensional space, high performance when it comes a higher number of dimensions that number of samples and it is memory efficient. Here, we only use the Support-vector machine classifier. The margin (M) optimization is given by:

$$\max_{\beta,\alpha,M} M$$

$$\text{subject to } \frac{y_i(x_i^T \beta_+ \alpha)}{||\beta||} \geq M, i = 1, 2, ...N$$

where N is the number of training instance and $||\beta||$ is length of β. Furthermore, we employ the Radial Basis Function (RBF) kernel trick to account for the non-linearity in the data. The kernel trick maps the features to a larger set of features using a specific transformation. The RBF transformation is given by:

$$K(x, x') = \exp -\gamma(||x - x'||^2)$$

where γ is a positive real number greater than 0. Furthermore, *sklearn* uses a penalty term C, which tells the model how much we want it to avoid missclassifying training points and it is set to 1 by default.

4.3 Evaluation Metrics

To assist the precision of the ERT prediction, we use the Mean Absolute Percentage Error (MAPE), which is defined as follows:

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{Y_i - \hat{Y}_i}{Y_i} \right|$$

Where Y is the true value and $\hat{Y}$ is the predicted value.
Moreover, to decide how good is a classifier, we employ F1 Score represented below:

$$F1 = 2 \cdot \frac{precision \cdot recall}{(precision + recall}$$

Where *precision* is defined as:

$$\frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

and *recall* is defined as:

$$\frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

and the average is method is 'micro', which corresponds to calculating the metric globally by counting the total number of true positives, false negatives and false positives.

4.4 Regression

In this section we train three regression models: Linear Regression (LR), LASSO and Random Forest Regression (RFR). $sklearn$ from $Python$ is used and the models are fitted using the default parameters. LR does not have any parameters while for LASSO $\alpha = 1$ where it corresponds to the penalty term and RFR parameters are: *n_estimator = 10, criterion='mse', max_depth=None, min_samples_split=2, min_samples_leaf=1, bootstrap=True*. We set the parameter *random_state = 1* to fix the randomness for LASSO and RFR.

4.4.1 Initial Results

The initial results of the regression models on the training and validation set along with the LOOCV$_{MAPE}$ are shown in Table 4.2. The results indeed confirm our early assumption that regression models are not going to be able to predict the ERT given the high variance in the response variable and the high average MAPE and its standard deviation. Despite the models' inability to predict ERT, the best three models on average on the training set are RFR$_{All}$, RFR$_{CM}$ and RFR$_{GCMNear}$ consecutively while the worst model is LR$_{CM}$.
Moreover, the best three models on average on the LOOCV$_{MAPE}$ are RFR$_{All}$ followed by RFR$_{CM}$ and LR$_{GCMNear}$ whereas the worst model is LR$_{All}$. Last, the best three models on average on the validation set are RFR$_{GCMNear}$, RFR$_{All}$ and RFR$_{CM}$ while the worst model is LR$_{All}$. Even though the models are considered to be undesirable given the high MAPE, we try to improve the results by selecting only the important features with the help of RFE with LOOCV.

Model	Features	Training	LOOCV$_{MAPE}$	Validation
LR	*All*	181948.696 (± 20152.535)	753193.163 (± 119769.211)	1238065.634 (± 670461.188)
	CM	351531.654 (± 29336.857)	383026.188 (± 32282.795)	460300.836 (± 70854.919)
	GCM Min	315619.16 (± 35372.085)	408703.461 (± 53561.807)	567884.448 (± 130505.336)
	GCM Mean	249040.518 (± 19655.667)	314633.983 (± 24432.279)	420487.194 (± 98723.155)
	GCM Near	317394.2 (± 57095.725)	399686.464 (± 72557.713)	476835.39 (± 117448.522)
LASSO	*All*	241723.818 (± 41946.745)	707587.403 (± 136959.007)	1129156.168 (± 381637.456)
	CM	351530.448 (± 29337.312)	383021.225 (± 32284.919)	460308.1 (± 70875.558)
	GCM Min	304157.2 (± 39052.484)	376848.056 (± 54347.784)	594791.184 (± 150648.773)
	GCM Mean	273771.054 (± 21999.491)	340161.139 (± 27885.073)	440324.036 (± 105329.615)
	GCM Near	316707.702 (± 58610.734)	394635.368 (± 73646.52)	471729.502 (± 115251.923)
RFR	*All*	71712.444 (± 22390.347)	211332.591 (± 87499.35)	259737.146 (± 178651.908)
	CM	86871.82 (± 30121.319)	265565.671 (± 54350.783)	308366.278 (± 201855.961)
	GCM Min	109383.77 (± 35945.67)	311347.58 (± 59768.879)	379959.056 (± 261644.28)
	GCM Mean	133154.536 (± 29876.665)	373071.715 (± 69514.567)	461900.47 (± 115465.968)
	GCM Near	94939.916 (± 27825.076)	267367.257 (± 30142.615)	247846.666 (± 89039.101)

Table 4.2: Initial regression models results averaged over 5 runs. The reported values in the brackets represent the standard deviation. Red color indicates the worst results in a column overall, while green color indicates the best results in a column overall.

4.4.2 Feature Selection

After performing RFE with LOOCV for 50 runs, 50 random splits of the data set, only the features that were selected at least 50% are chosen to fit the models. We notice that LR$_{All}$ did not have any features that passed the threshold. Furthermore, selecting less features did not assist in reducing the MAPE in a way that makes any model reliable. Nevertheless, our conclusions from features inspection in section 3.2 are confirmed that non-unique features were unlikely to be selected during the process of feature selection. For example, Figure 4.2 exhibits the number of times features were selected during RFE for RFR$_{All}$. None of the features that we identify the be common were selected. In addition, the worst features, bestAttrNumber for all GCM approaches, were selected the least. Therefore, only the "unique" features that

minimizes the error were selected, so we optimally expect the MAPE to decrease with fewer features.

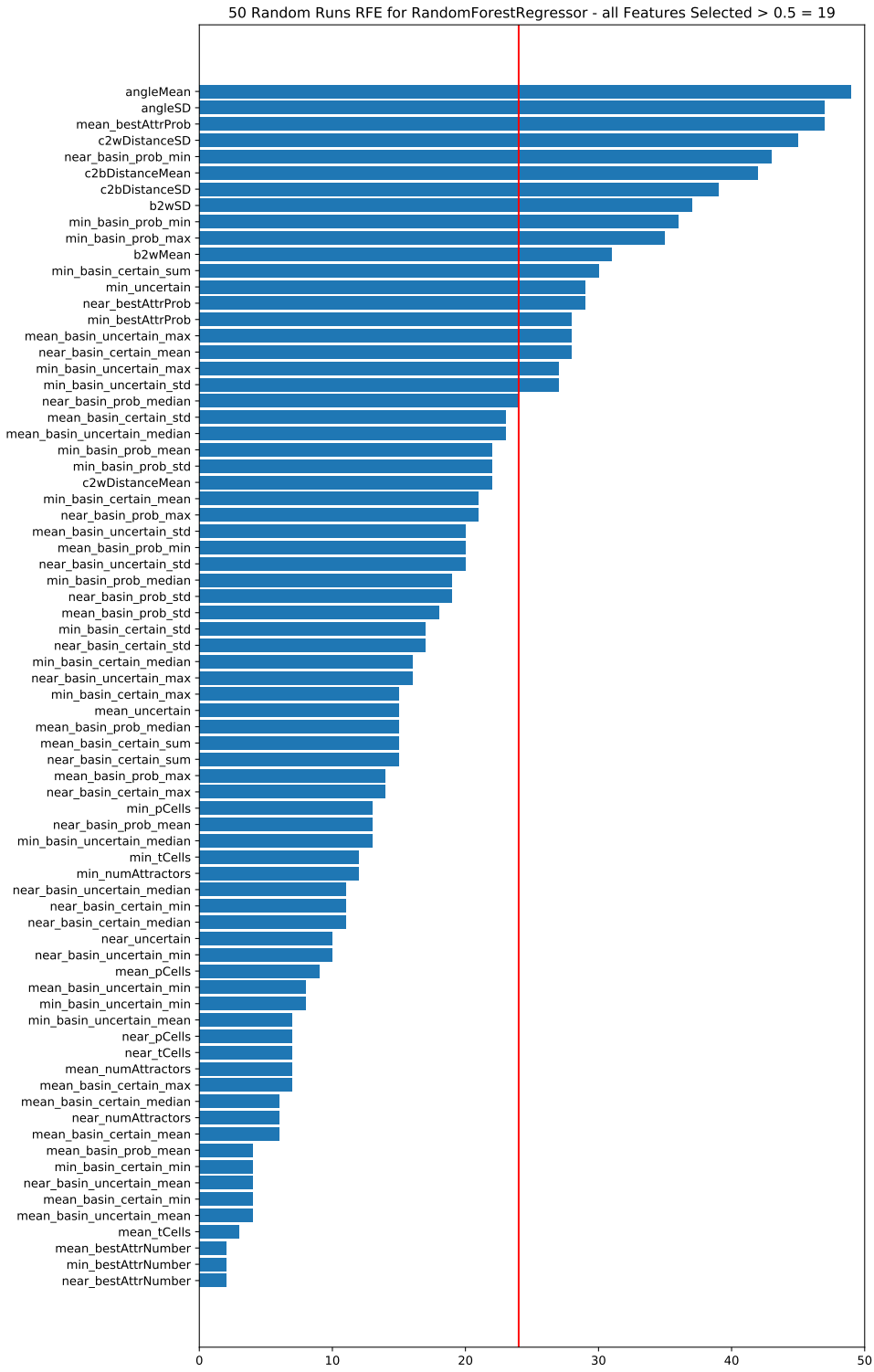

Figure 4.2: The number of times a feature was selected after 50 random runs of RFE-LOOCV for RFR_{All}. The red line indicates the threshold, which is 50% and the number features selected is 19.

The results after feature selection along with the number of features selected are reported in Table 4.3. The best three performing models on average on the training set are RFR$_{All}$ 19 features, RFR$_{GCMNear}$ 5 features and RFR$_{CM}$ 5 features. We notice that RFR$_{Near}$ approach with 5 features out of 22 managed to reduce MAPE on the training set by 34.75% and RFR_{All} 19 features out of 74 also decreased the MAPE on the training by 11.2%. In addition, reducing the features of RFR$_{CM}$ from 8 to 5 features, decreased MAPE on the training set by 21.57%. In addition, the worst performing model on the training set is LR$_{GCMMin}$ 2 features and increased the MAPE by 4.15% on the training set.

Model	**Features**	**Features Selected**	**Training**	**LOOCV$_{MAPE}$**	**Validation**
LR	*All*	0	-	-	-
	CM	2	274558.646 (± 31894.672)	281837.252 (± 32881.076)	367106.354 (± 81369.547)
	GCM Min	2	328988.828 (± 34719.288)	338642.171 (± 35760.297)	410764.606 (± 97506.439)
	GCM Mean	7	301176.486 (± 40985.792)	325039.691 (± 43453.034)	557920.712 (± 86409.371)
	GCM Near	3	261955.74 (± 30048.502)	279555.882 (± 32053.22)	300443.874 (± 40195.822)
LASSO	*All*	1	276795.28 (± 27911.836)	282599.822 (± 28577.329)	365159.966 (± 108641.684)
	CM	2	274573.284 (± 31895.544)	281852.161 (± 32882.01)	367119.13 (± 81363.792)
	GCM Min	1	319984.694 (± 34800.187)	326139.305 (± 35636.265)	384952.75 (± 100035.311)
	GCM Mean	3	315916.968 (± 43651.664)	333783.471 (± 46053.184)	478075.734 (± 98121.519)
	GCM Near	3	261964.298 (± 30048.862)	279564.155 (± 32053.532)	300460.372 (± 40198.615)
RFR	*All*	19	64107.596 (± 27065.059)	186360.078 (± 59335.248)	224892.94 (± 175759.484)
	CM	5	69957.778 (± 28839.721)	208117.129 (± 59416.292)	264433.408 (± 208005.24)
	GCM Min	9	128841.216 (± 56438.838)	307068.814 (± 41879.399)	394172.726 (± 215108.423)
	GCM Mean	4	124125.93 (± 31883.823)	349327.364 (± 82942.648)	417657.68 (± 148736.529)
	GCM Near	5	66833.732 (± 31898.19)	212238.211 (± 25484.527)	205009.628 (± 51283.695)

Table 4.3: Average results over 5 runs of RFE-LOOCV for regression models along with the number of features selected at least 50%. The values in the brackets corresponds to the standard deviation of MAPE. Red color indicates the worst results in a column, while green color indicates the best results in a column overall. The number of features in *All* is 74, *CM* is 8 and 22 features for each *GCM* approach.

Additionally, the best three models on average w.r.t the LOOCV$_{MAPE}$ are RFR$_{All}$ 19 features,

RFR$_{CM}$ 5 features and RFR$_{GCMNear}$ 5 features. The LOOCV$_{MAPE}$ decreased by 12.56% for RFR$_{All}$ as well as by 24.26% and 22.99% for RFR$_{CM}$ and RFR$_{GCMNear}$ respectively. On the contrary, the LOOCV$_{MAPE}$ for the worst model, RFR$_{GCMMean}$ 4 features, decreased by 6.57%. Last, the best three models on average on the validation set are RFR$_{GCMNear}$ 5 features, RFR$_{All}$ 19 features and RFR$_{CM}$ 5 features. The aforementioned models reduced LOOCV$_{MAPE}$ by 18.92%, 14.38% and 15.34% correspondingly whereas the worst model LR$_{GCMMean}$ 3 features increased LOOCV$_{MAPE}$ by 28.09%.
To conclude, none of the regression models are considered to be trustworthy to predict the ERT given the high variance in the response variable and the high MAPE values even after feature selection. Given that, we do not perform any further experiments or modeling for regression tasks and come to an end that ERT cannot be predict with the data generated. Next, we attempt to classify the best performing algorithm.

4.5 Classification

In this section we fit three classification models: Random Forest Classifier (RFC), Multinomial Logistic Regression (MLR) and Supprt-vector Classifier with RBF Kernel (SVC). $sklearn$ from $Python$ is used and the models are fitted using default parameters. The default parameters for RFR are: *n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, bootstrap=True*. In addition, the default parameters for MLR are: *penalty='l2', C=1.0, solver='lbfgs', multi_class='multinomial'* and for SVC *C=1.0, γ='auto'*, where auto means that the value of γ is $1/num\_features$. We fix the randomness of all the aforementioned models by setting *random_state = 1* to make them comparable.

4.5.1 Initial Results

The initial results of the classification models on the training and validation set along with the LOOCV$_{F1}$ are present in Table 4.4. From the table, RFC$_{All}$ came in the first place followed by RFC$_{GCMNear}$ and RFC$_{CM}$ on average. We expected the RFC to be the best model on the training set since we do not limit the max_depth of the tree. The reason is that we are going for a flexible model to allow the decision trees within the model to explore more. However, we observe an overfitting problem with RFC given the high F1 score compared to other models. Also. the worst performing model on the training set on average is MLR$_{GCMNear}$. If we to ignore RFC models, the best model on the training set overall would be MLR$_{All}$. Proceeding to the LOOCV$_{F1}$, the top three models on average are MLR$_{All}$, SVC$_{All}$ and RFC$_{GCMMin}$ while the worst model is MLR$_{GCMNear}$. Last, the highest scoring models on the validation set on average are RFC$_{GCMMin}$ and SVC$_{GCMMean}$ followed by RFC$_{All}$ and MLR$_{All}$. In addition, the worst model on the validation set on average is MLR$_{GCMNear}$. After inspecting the results, the top three desirable models are RFC$_{GCMMin}$, MLR$_{All}$ and SVC$_{All}$ given the scores obtained on LOOCV$_{F1}$ and validation set. In the next subsection, we perform feature selection using RFE with LOOCV and Boruta.

Models	Features	Training	LOOCV$_{F1}$	Validation
RFC	*All*	0.952 (± 0.009)	0.321 (± 0.041)	0.374 (± 0.066)
	CM	0.937 (± 0.02)	0.313 (± 0.054)	0.287 (± 0.073)
	GCM Min	0.935 (± 0.021)	0.346 (± 0.034)	0.383 (± 0.071)
	GCM Mean	0.931 (± 0.023)	0.317 (± 0.045)	0.261 (± 0.069)
	GCM Near	0.944 (± 0.021)	0.291 (± 0.026)	0.269 (± 0.099)
MLR	*All*	0.522 (± 0.047)	0.352 (± 0.024)	0.348 (± 0.031)
	CM	0.341 (± 0.023)	0.265 (± 0.025)	0.261 (± 0.044)
	GCM Min	0.328 (± 0.025)	0.272 (± 0.032)	0.278 (± 0.024)
	GCM Mean	0.358 (± 0.03)	0.315 (± 0.031)	0.261 (± 0.075)
	GCM Near	0.302 (± 0.036)	0.211 (± 0.012)	0.208 (± 0.064)
SVC	*All*	0.474 (± 0.03)	0.35 (± 0.026)	0.348 (± 0.061)
	CM	0.55 (± 0.033)	0.239 (± 0.063)	0.278 (± 0.1)
	GCM Min	0.357 (± 0.014)	0.315 (± 0.062)	0.304 (± 0.031)
	GCM Mean	0.348 (± 0.023)	0.304 (± 0.067)	0.383 (± 0.048)
	GCM Near	0.333 (± 0.017)	0.289 (± 0.022)	0.278 (± 0.05)

Table 4.4: Initial classification models results (F1 score micro) averaged over 5 runs. The values in the brackets refer to the standard deviation. Red color indicates the worst results in a column overall, while green color indicates the best results in a column overall.

4.5.2 Feature Selection

RFE-LOOCV

We run RFE with LOOCV for 50 runs and only choose the features that were selected at least 50%. We fit the models again over five runs with the selected features on the training and we perform CV to estimate its predictive ability and compare it to the validation set evaluation.

Models	Features	Features Selected	Training	LOOCV$_{F1}$	Validation
RFC	*All*	50	0.95 (± 0.006)	0.313 (± 0.014)	0.287 (± 0.039)
	CM	8	0.937 (± 0.02)	0.313 (± 0.054)	0.287 (± 0.073)
	GCM Min	14	0.935 (± 0.008)	0.324 (± 0.046)	0.33 (± 0.079)
	GCM Mean	15	0.93 (± 0.031)	0.311 (± 0.032)	0.278 (± 0.09)
	GCM Near	16	0.939 (± 0.021)	0.3 (± 0.02)	0.296 (± 0.072)
MLR	*All*	67	0.515 (± 0.021)	0.357 (± 0.026)	0.348 (± 0.053)
	CM	7	0.3 (± 0.029)	0.237 (± 0.021)	0.243 (± 0.039)
	GCM Min	11	0.322 (± 0.026)	0.289 (± 0.029)	0.278 (± 0.024)
	GCM Mean	11	0.374 (± 0.022)	0.282 (± 0.045)	0.339 (± 0.084)
	GCM Near	4	0.274 (± 0.029)	0.237 (± 0.024)	0.226 (± 0.065)

Table 4.5: Averaged results over 5 runs of RFE-LOOCV for classification models along with the number of features selected at least 50%. The values in the brackets indicate the standard deviation. Red color indicates the worst results in a column, while green color indicates the best results in a column overall. The number of features in *All* is 74, *CM* is 8 and 22 features for each *GCM* approach.

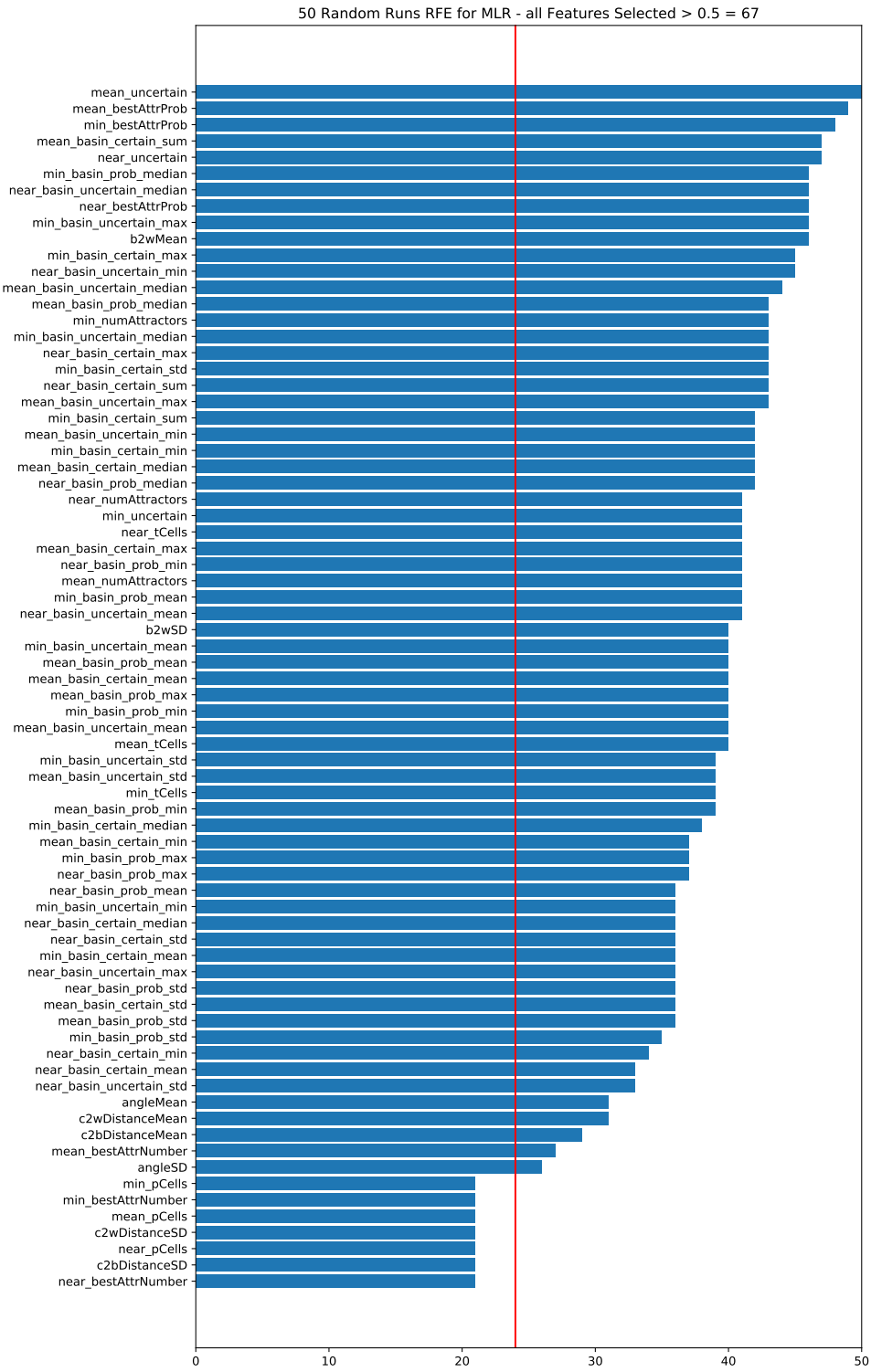

Figure 4.3: The number of times a feature was selected after 50 runs of RFE-LOOCV for MLR_{All}. The red line indicates the threshold, which is 50% and the number features selected is 67.

Because SVC with RBF kernel does not return any coefficients in *sklearn*, it cannot be used with RFE. Therefore, we only conduct RFE for RFC and MLR and the results are as displayed in Table 4.5. We still believe that RFC is overfitting on the training set given the

high F1 score present. In addition, an increase is noticed on the LOOCV$_{F1}$, which we consider a good sign. However, RFC$_{CM}$ selected all the CM features which results in the same result obtained in Table 4.4. The best three models on average on the training set are RFC$_{All}$ 50 features, RFC$_{GCMNear}$ 16 features and RFC$_{CM}$ 8 features while the worst model on average is MLR$_{GCMNear}$ 4 features. All models with the exception of RFC$_{CM}$ 8 features decreased the F1 score on the training set by no more than 0.5%, while the worst model decreased F1 score by 9.72%. Moreover, the top models on average based on LOOCV$_{F1}$ are MLR$_{All}$ 67 features increasing by 1.41% followed by RFC$_{GCMMin}$ 14 features associated with 6.57% decrease and third, RFC$_{All}$ 50 features with 2.52% decrease and RFC$_{CM}$ 8 features with the same value. The worst two models on average on LOOCV$_{F1}$ are MLR$_{GCMNear}$ 4 features even though the F1 score increased by 11.16% and MLR$_{CM}$ 7 features with a decreased score of 11.16%. Furthermore, the best three performing models on average on the validation set are MLR$_{All}$ 67 features with no change in the score followed by MLR$_{GCMMean}$ with 26% increase and RFC$_{GCMMin}$ 14 features with 14.87% decrease while the worst model on average is MLR$_{GCMNear}$ 4 features despite the increase by 8.29%. Given that, we consider only MLR$_{All}$ with 67 features to be one of the best along with RFC$_{GCMMin}$ SVC$_{All}$ from the initial results. To analyze the MLR$_{All}$ 67 feature more, Figure 4.3 displays the features that were chosen after RFE-LOOCV. We notice that 13 out of the 17 features that we consider not unique were selected more than 50%, but we do not see this as something problematic. This is because although MLR$_{All}$ training F1 reduced, it managed to increase the F1 score on CV and maintain the same F1 score on the validation set.

Boruta

We conduct 5 random runs using the seeds with the Boruta algorithm that utilizes Random Forest Classifier to find the features that are the most relevant to the classification tasks. We set the parameters as follows: *n_estimators='auto', random_state=1*, while for the estimator of Boruta, RFC, we set the *max_depth=5* as recommended in the documentation. The purpose of the 5 random seeds is for splitting the data. In addition, we only choose the features that were selected at least 2 times out of 5 (majority voting). Moreover, We did experiment with different parameters settings, but the same results were obtained. The results of the features chosen after Boruta are in Table 4.6.

Features	**Features Selected**
All	c2wDistanceMean, c2wDistanceSD, b2wSD min_basin_prob_max mean_basin_prob_min mean_bestAttrProb near_basin_prob_min
CM	c2wDistanceMean, c2wDistanceSD, b2wSD
GCM Min	min_basin_prob_min min_basin_prob_median min_basin_prob_max
GCM Mean	mean_basin_prob_min
GCM Near	near_basin_prob_min

Table 4.6: The features chosen after Boruta algorithm

We notice that only 7 features were chosen when using all the 74 features and. Following that, 3 features were chosen from CM and GCM Min, while only 1 feature was chosen for both GCM Mean and GCM Near. We can easily observe the overlap with the features selected from the full feature set and the subsets. In addition, the same features that were selected in CM, GCM Mean and GCM Near are selected in All features, but only one feature from GCM min, min_basin_prob_max, was chosen. From here, we take the chosen features and we fit the classification models again with the same settings as before and use CV. The results of the models with the chosen features are shown in Table 4.7. The results shown indicate that none of the models benefited from the features chosen on the validation set F1 score. The models of MLR and SVC scored the worst on training, LOOCV$_{F1}$ and validation. Therefore, we do not consider any of the models to be desirable. From here, we conclude that the best three models that match our criteria are: (1) SVC$_{All}$ *74 features, LOOCV$_{F1}$ = 0.350, validation = 0.348*, (2) RFC$_{GCMMin}$ *22 features, LOOCV$_{F1}$ = 0.346, validation = 0.383* and (3) MLR$_{All}$ *67 features, LOOCV$_{F1}$ = 0.357, validation = 0.348*.

Models	**Features**	**Features Selected**	**Training**	**LOOCV$_{F1}$**	**Validation**
RFC	*All*	7	0.95 (± 0.021)	0.357 (± 0.012)	0.313 (± 0.094)
	CM	3	0.941 (± 0.026)	0.309 (± 0.03)	0.313 (± 0.071)
	GCM Min	3	0.939 (± 0.021)	0.354 (± 0.054)	0.278 (± 0.073)
	GCM Mean	1	0.893 (± 0.021)	0.243 (± 0.038)	0.2 (± 0.073)
	GCM Near	1	0.9 (± 0.025)	0.291 (± 0.021)	0.339 (± 0.057)
MLR	*All*	7	0.341 (± 0.013)	0.3 (± 0.018)	0.33 (± 0.039)
	CM	3	0.261 (± 0.0)	0.259 (± 0.005)	0.261 (± 0.0)
	GCM Min	3	0.344 (± 0.01)	0.317 (± 0.009)	0.313 (± 0.036)
	GCM Mean	1	0.261 (± 0.0)	0.237 (± 0.02)	0.261 (± 0.0)
	GCM Near	1	0.265 (± 0.017)	0.25 (± 0.0)	0.261 (± 0.0)
SVC	*All*	7	0.261 (± 0.0)	0.254 (± 0.01)	0.261 (± 0.0)
	CM	3	0.261 (± 0.0)	0.261 (± 0.0)	0.261 (± 0.0)
	GCM Min	3	0.289 (± 0.019)	0.219 (± 0.009)	0.217 (± 0.053)
	GCM Mean	1	0.261 (± 0.0)	0.261 (± 0.0)	0.261 (± 0.0)
	GCM Near	1	0.261 (± 0.0)	0.252 (± 0.005)	0.261 (± 0.0)

Table 4.7: Averaged results over 5 runs of Boruta for classification models along with the number of features. The values in brackets corresponds to the standard deviation. Red color indicates the worst results in a column, while green color indicates the best results in a column overall. The number of features in *All* is 74, *CM* is 8 and 22 features for each *GCM* approach.

To examine the predictive ability of the aforementioned models, the confusion matrices averaged over 5 runs of the training predicted classes with the validation set predicted classes for the models are displayed in Figures 4.4, 4.5 and 4.6. We do not notice concerning matters except for the results on the validation set for SVC since it is biased towards classifying RLS unlike MLR, which had the same score on validation set. Next we perform hyperparameter tuning for the models in order to obtain a higher F1 score on LOOCV$_{F1}$ and validation.

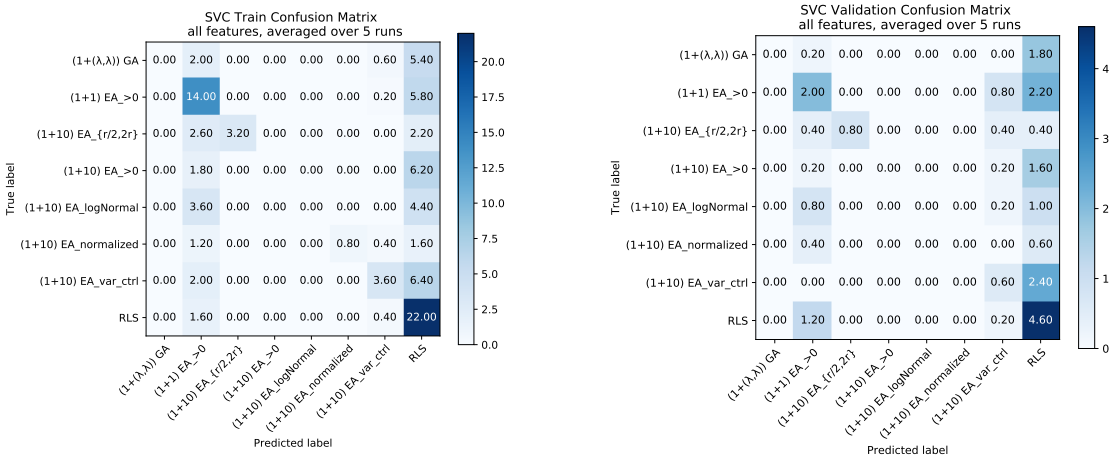

Figure 4.4: Averaged confusion matrices over 5 runs of SVC_{All} model, Left: training set. Right: validation set

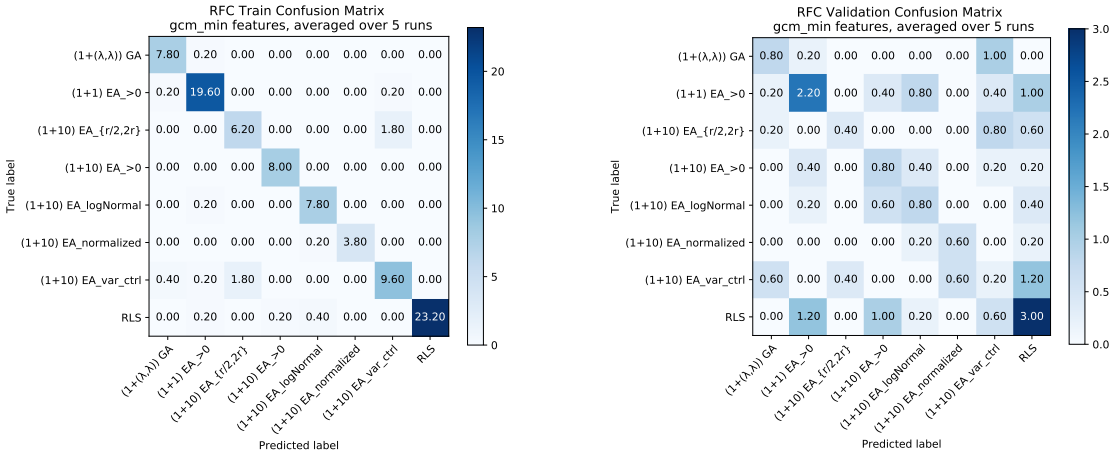

Figure 4.5: averaged confusion matrices over 5 runs of RFC_{GCMMin} model, Left: training set. Right: validation set

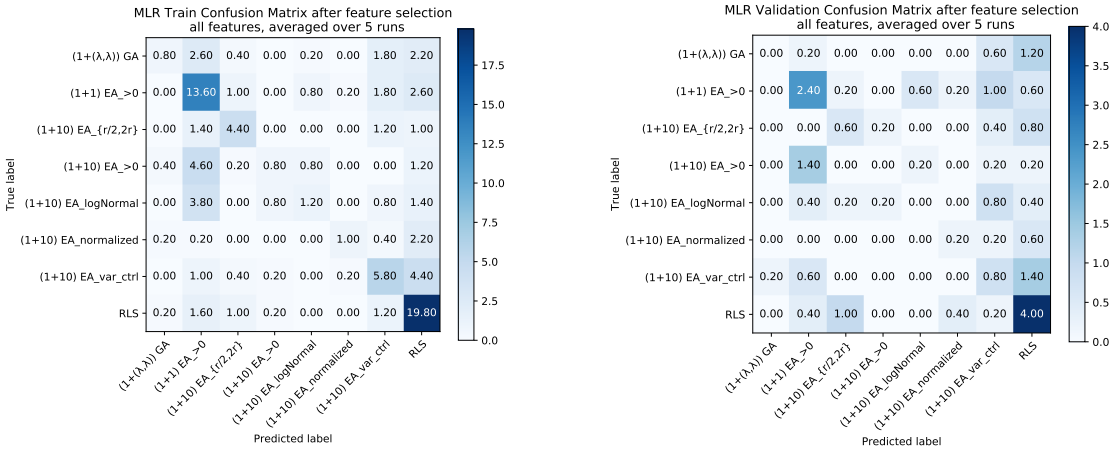

Figure 4.6: Averaged confusion matrices over 5 runs of MLR_{All} 67 features model, Left: training set. Right: validation set

4.5.3 Hyperparameter Tuning

In this subsection we take the three models that we identify to be the best models namely, SVC_{All} 74 features, MLR_{All} 67 features and RFC_{GCMMin} 22 features, and utilize the Mixed integer, Parallel - Efficient Global Optimization package or as referred to as MiP-EGO. MiP-EGO is an optimization package that is used to optimize mixed integer optimization problems and it uses a surrogate model (Random Forest), which corresponds to the EGO part and it learned from the evaluations made in the process. We decided to use this package because hyperparameter tuning is a time consuming process. The process begins with defining the search space, which can be Continuous, Ordinal and/or Nominal. Then, a function to be optimized is defined and in this case we aim to maximize the LOOCV_{F1} averaged for each fold. Last, the maximum number of evaluations and iterations is set to 500. For the SVC model we vary the penalty parameter C in a continuous space $[1, 200]$ and the kernel parameter γ in a continuous space $[1, 50]$. In addition, the same space is defined for the penalty parameter C in MLR. Moreover, RFC hyperparameters are all defined in the Ordinal space and they consist of $n\_estimators \in [10, 500]$, $max\_depth \in [5, 50]$, $min\_samples\_leaf \in [2, 30]$ $min\_samples\_split \in [1, 20]$ and $bootstrap \in [0, 1]$ where the bootstrap values corresponds to True (1) and False (0).

Given that we run the hyperparameter tuning over 5 random runs using the seeds, we obtain 15 models where each 5 models correspond to one of the best models. After these models are obtained, we running the fitting and prediction with LOOCV for 5 runs using the seeds and the results are displayed in Table 4.8.

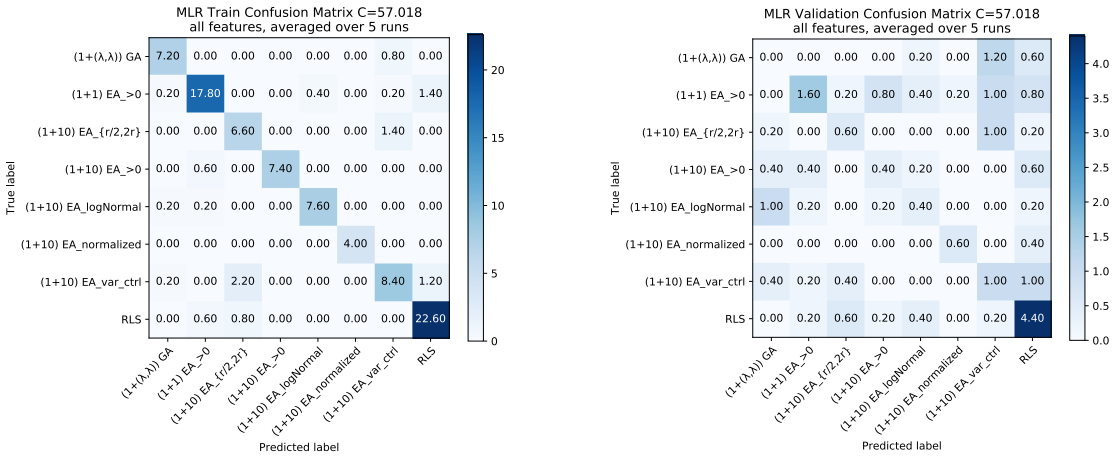

Figure 4.7: Averaged confusion matrices over 5 runs of MLR_{All} model after HPT, Left: training set. Right: validation set

The SVC model benefited the most on the training set across all parameters C and γ that were selected reaching an average F1 score on the training of 0.961 with ($\pm$ 0.009) deviation. Furthermore, there was a decrease on the LOOCV_{F1} score even though it was the value to be maximized. Although the decrease was small, the F1 scores on the validation does not make any of parameters good enough as the F1 score on the validation set is 0.27 with ($\pm$ 0.019) deviation. Therefore, we would still choose the SVC model with the default parameters.
Proceeding to the RFC model, there is indeed a noticeable variation in the F1 score on the training set on average unlike the LOOCV_{F1} scores, which also decreased. Again the results on

Model	Params	Training	LOOCV_{F1}	Validation
$\textbf{RFC}_{GCMMin}$	*n_estimators=493* *max_depth=22* *min_samples_split=18* *min_samples_leaf=18* *bootstrap= False*	0.402 (± 0.043)	0.33 (± 0.013)	0.313 (± 0.057)
	n_estimators=35 *max_depth=19* *min_samples_split=14* *min_samples_leaf=2* *bootstrap=False*	0.748 (± 0.031)	0.315 (± 0.037)	0.339 (± 0.036)
	n_estimators=72 *max_depth=11* *min_samples_split=19* *min_samples_leaf=12* *bootstrap=False*	0.502 (± 0.027)	0.302 (± 0.06)	0.295 (± 0.064)
	n_estimators=32 *max_depth=11* *min_samples_split=19* *min_samples_leaf=12* *bootstrap=True*	0.411 (± 0.031)	0.341 (± 0.033)	0.313 (± 0.036)
	n_estimators=251 *max_depth=21* *min_samples_split=4* *min_samples_leaf=2* *bootstrap=False*	0.937 (± 0.016)	0.339 (± 0.029)	0.33 (± 0.066)
$\textbf{SVC}_{All}$	*C=64.087* *γ=1.131*	0.961 (± 0.012)	0.3 (± 0.009)	0.27 (± 0.019)
	C=35.36 *γ=5*	0.961 (± 0.012)	0.279 (± 0.01)	0.27 (± 0.019)
	C=162.393 *γ=1.094*	0.961 (± 0.012)	0.3 (± 0.009)	0.27 (± 0.019)
	C=32.087 *γ=1.839*	0.961 (± 0.012)	0.3 (± 0.009)	0.27 (± 0.019)
	C=199.789 *γ=2.076*	0.961 (± 0.012)	0.3 (± 0.009)	0.27 (± 0.019)
$\textbf{MLR}_{All}$	*C=108.703*	0.904 (± 0.028)	0.352 (± 0.025)	0.391 (± 0.107)
	C=57.018	0.887 (± 0.035)	0.359 (± 0.025)	0.391 (± 0.092)
	C=64.847	0.889 (± 0.032)	0.361 (± 0.026)	0.391 (± 0.097)
	C=116.921	0.906 (± 0.032)	0.352 (± 0.025)	0.383 (± 0.117)
	C=187.017	0.922 (± 0.025)	0.337 (± 0.03)	0.383 (± 0.117)

Table 4.8: Hyperparameter tuning results on the best three models found using MiP-EGO for 5 runs. The values in the brackets report the standard deviation. Red color indicates the worst results in a column, while green color indicates the best results in a column overall

the validation set were not better than the use of default parameters. Given that, we would still consider the RFC model with the default parameters to be better. Last, the MLR model did witness a slight variation on the training set, but we do not consider it concerning. In addition,

the MLR scores on LOOCV$_{F1}$ were the best among other models and indeed the best on the validation set with three models reaching 0.391 on average. From all the 15 models, we would replace the MLR$_{All}$ 67 features with $C = 1$ (default parameter) to the same model where $C = 57.018$ given the high LOOCV$_{F1}$ and validation set score. Furthermore, Figure 4.7 displays the confusion matrices averaged over 5 runs to classify the best algorithm using the aforementioned hyperparameter. The confusion matrices show that the predictive ability on both the training set and the validation set improved compared to the MLR$_{All}$ model with the default parameters.

4.6 Summary of Results

In this chapter we explained the data generated from feature generation and the ERT/labels obtained from the IOHprofiler. To answer $Q4$, we experimented with various regression models and concluded that predicting the ERT is not feasible at this points given the large deviation within the ERT values and the high MAPE values. Furthermore, we attempted to decrease the MAPE value by selecting features (RFE-LOOCV) and we came to the closure as before. After that, we sought to classify (select) the best algorithm using different machine learning models. In addition, feature selection (RFE-LOOCV and Boruta) and hyperparameter tuning (MiP-EGO) were implemented and the best models evaluated using F1 score on average are as follows:

1. MLR$_{All}$ 67 features , $C = 57.018$, Training: 0.887, LOOCV$_{F1}$: 0.359, Validation 0.391.
2. RFC$_{GCMMin}$ 22 features, default parameters, Training: 0.935, LOOCV$_{F1}$: 0.346, Validation 0.383.
3. SVC$_{All}$ 74 features, default parameters, Training: 0.474, LOOCV$_{F1}$: 0.35, Validation 0.348.

Chapter 5

Conclusion

Exploratory Landscape Analysis (ELA) has been developing since the early 1990's as a way to characterize optimization problems that are solved using classical heuristic optimization or Evolutionary Algorithm (EA) approaches. Researchers did not attempt to only understand the landscape of an optimization problem, but also attempted to use this characterization to find the best algorithm or predict the performance of an algorithm. As described in section 2.1, researchers developed many methods to discover the properties of continuous optimization problems and recommend the best suited algorithm. Moreover, almost all ways are tested against The Black-Box Optimization Benchmark (BBOB) functions. However, there does not exist a general method to identify the properties in the discrete decision space, but methods that are specific per optimization problem.

In Chapter 1, we defined our problem to be finding current feature extraction methods in the continuous decision space and attempt to adapt it to the discrete decision space. In addition, after generating the features we attempted to predict the Expected Run Time (ERT) or classify the best performing algorithm where the results were obtained from the IOHprofiler. The functions to be tested against were 23 Pseudo-Boolean Functions subject to maximization as mentioned in section 1.3. In Chapter 2, we focused on a relatively new feature generation tool, $flacco$, to find a method that is suitable. Indeed, Cell Mapping angle features and Generalized Cell Mapping (GCM) for their different approaches were suitable. From here, we studied how $flacco$ generated the features in the continuous domain.

In Chapter 3, we proposed an adjustment to the creation of the $FeatureObject$ in a way that operates for binary input. Our feature inspect analysis indicated that the CM angle features do produce different features for each of the functions for different number of observations. Moreover, the GCM set for different approaches does not produce a complete feature set that is unique, but rather some features can be selected from different approaches of GCM. Also, even though we consider all the functions to be completely different, two functions $(f11, f12)$ had exactly the same feature values for all different methods, CM and GCM, across all functions and number of observations.

In Chapter 4, we aimed to predict the ERT using three different regression models, Linear Regression (LR), LASSO and Random Forest Regression (RFR), and the Mean Absolute Percentage Error (MAPE) as the evaluation metric. The models are fitted using 5 different random defined seeds for data split and the results of the training, LOOCV and validation are reported on average along with the standard deviation. Given the high standard deviation, the

huge difference between the minimum and maximum values and the MAPE scores, the models were unable to predict the ERT and therefore not trustworthy. In addition, we attempted to select important features using Recursive Feature Elimination (RFE) with Leave-One-Out Cross-validation (LOOCV) by running it 50 random runs and choosing the features that were selected at least 50% of the time. However, we came the same conclusion as initially stated. Next, we sought to classify (select) the best algorithm using three different classification models, Random Forest Classifier (RFC), Multinomial Logistic Regression (MLR) and Suppervector Classifier with 'RBF' kernel (SVC) and the F1 'micro' score as the evaluation metric. The models are fitted using 5 different random defined seeds for data split and the results of the training, LOOCV and validation are reported on average along with the standard deviation. We discovered that the highest F1 score on the validation set was 0.383 for RFC model using GCM Min features. From here, we ran RFE-LOOCV and Boruta algorithm to find the best subset of the features for each model. Given that, we found out that Boruta is of no use and from RFE-LOOCV only MLR with all the features reduced to 67 out of 74 improved performance on the LOOCV$_{F1}$ and validation set. Afterwards, we moved to hyperparameter tuning using MiP-EGO for the three best selected models: RFC$_{GCMMin}$ 22 features, SVC$_{All}$ 74 features and MLR$_{All}$ 67 features. The process of hyoerparameter tuning was also ran over 5 defined seeds and all the resulted models were evaluated. None of the models exhibited a better performance except for MLR$_{All}$ 67 features. From here, the best three models were MLR$_{All}$ 67 features with the tuned C parameter (0.391 on validation set) followed by RFC$_{GCMMin}$ (0.383 on validation set) and SVC$_{All}$ 74 features (0.348 on validation set).

To conclude, there is a potential in generating features in the discrete decision space and making use of them to classify the best performing algorithm. This is possible if the labels are available as well as if the behaviour of the test functions is already known before hand. This will help to match the expectation of the features generated and the known behaviour of the functions. As for selecting the best algorithm, Multinomial Logistic Regression has proved to be a good enough classifier given its high F1 score on the validation set and LOOCV$_{F1}$. Future work in this domain would include finding more feature generating approaches that are relevant to the problems as well as growing the data set to include more functions along with its labels as well as testing on other Pseudo-Boolean functions.

Bibliography

[1] F. Bursal and C. Hsu. Application of a cell-mapping method to optimal control problems. *International Journal of Control*, 49(5):1505–1522, 1989.

[2] C. Doerr, H. Wang, F. Ye, S. van Rijn, and T. Bäck. Iohprofiler: A benchmarking and profiling tool for iterative optimization heuristics. *arXiv preprint arXiv:1810.05281*, 2018.

[3] C. Flamm, I. L. Hofacker, P. F. Stadler, and M. T. Wolfinger. Barrier trees of degenerate landscapes. *Zeitschrift für physikalische chemie*, 216(2):155, 2002.

[4] N. Hansen, A. Auger, S. Finck, and R. Ros. *Real-parameter black-box optimization benchmarking 2010: Experimental setup*. PhD thesis, INRIA, 2010.

[5] C. Hernández, Y. Naranjani, Y. Sardahi, W. Liang, O. Schütze, and J.-Q. Sun. Simple cell mapping method for multi-objective optimal feedback control design. *International Journal of Dynamics and Control*, 1(3):231–238, 2013.

[6] C. Hernández, O. Schutze, M. Emmerich, F.-R. Xiong, and J.-Q. Sun. Barrier tree for continuous landscapes by means of generalized cell mapping. *Bioinformatics*, 26:299–233, 2014.

[7] C. Hsu and B. Tongue. Cell-to-cell mapping, a method of global analysis for nonlinear systems. *Journal of Applied Mechanics*, 55:749, 1988.

[8] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[9] S. A. Kauffman. *The origins of order: Self-organization and selection in evolution*. OUP USA, 1993.

[10] P. Kerschke. Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the r-package flacco. *arXiv preprint arXiv:1708.05258*, 2017.

[11] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary computation*, 27(1):3–45, 2019.

[12] P. Kerschke, M. Preuss, C. Hernández, O. Schütze, J.-Q. Sun, C. Grimme, G. Rudolph, B. Bischl, and H. Trautmann. Cell mapping techniques for exploratory landscape analysis. In *EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*, pages 115–131. Springer, 2014.

[13] P. Kerschke, M. Preuss, S. Wessing, and H. Trautmann. Detecting funnel structures by means of exploratory landscape analysis. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 265–272. ACM, 2015.

[14] M. Lunacek and D. Whitley. The dispersion metric and the cma evolution strategy. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 477–484. ACM, 2006.

[15] K. M. Malan and A. P. Engelbrecht. Quantifying ruggedness of continuous landscapes using entropy. In *2009 IEEE Congress on evolutionary computation*, pages 1440–1447. IEEE, 2009.

[16] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 829–836, New York, NY, USA, 2011. ACM.

[17] O. Mersmann, M. Preuss, and H. Trautmann. Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. In *International Conference on Parallel Problem Solving from Nature*, pages 73–82. Springer, 2010.

[18] M. A. Muñoz, M. Kirley, and S. K. Halgamuge. Exploratory landscape analysis of continuous space optimization problems using information content. *IEEE Transactions on Evolutionary Computation*, 19(1):74–87, 2014.

[19] B. Naudts, D. Suys, and A. Verschoren. Epistasis as a basic concept in formal landscape analysis. In *ICGA*, pages 65–72. Citeseer, 1997.

[20] E. Pitzer and M. Affenzeller. A comprehensive survey on fitness landscape analysis. In *Recent advances in intelligent engineering systems*, pages 161–191. Springer, 2012.

[21] V. K. Vassilev, T. C. Fogarty, and J. F. Miller. Information characteristics and the structure of landscapes. *Evolutionary computation*, 8(1):31–60, 2000.

[22] H. Wang, B. van Stein, M. Emmerich, and T. Back. A new acquisition function for bayesian optimization based on the moment-generating function. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 507–512. IEEE, 2017.