Managing Diversity and Many Objectives in Evolutionary Design

Sheikh Faishal Basher

Department of Computer Science

Submitted in partial fulfilment
of the requirements for the degree of

Master of Science

Faculty of Mathematics and Science, Brock University
St. Catharines, Ontario

Abstract

This thesis proposes a new approach to evolving a diversity of high-quality solutions for problems having many objectives. Mouret and Clune's MAP-Elites algorithm has been proposed as a way to evolve an assortment of diverse solutions to a problem. We extend MAP-Elites in a number of ways. Firstly, we introduce a many-objective strategy called sum-of-ranks, which enables problems with many objectives (4 and more) to be considered in the MAP. Secondly, we enhance MAP-Elites by extending it with multiple solutions per "grid" cell (the original MAP-Elites saves only a single solution per cell). A few different ways of selecting cell members for reproduction are also considered. We test the new MAP-Elites strategies on the evolutionary art application of image generation. Using procedural textures, genetic programming is used with upwards of 15 lightweight image features to guide fitness. The goal is to evolve images that share image features with a given target image. Our experiments show that the new MAP-Elites algorithms produce a large number of diverse solutions of varying quality. The extended MAP-Elites algorithm is also statistically competitive compared to vanilla GP in this application domain.

Acknowledgements

First, I would like to express my gratitude to Professor Brian J. Ross for his guidance and support for my thesis work and throughout my graduate program. Then I would like to thank my parents, my brother and my friends for their support. I would also like to thank the department of Computer Science and its staff for their support and help.

Contents

1 Introduction **1**
1.1 Goals and Motivation . . . 3
1.2 Main Contributions . . . 3
1.3 Thesis Structure . . . 4

2 Background **5**
2.1 Genetic Programming . . . 5
2.1.1 Reproduction Operation . . . 7
2.1.2 Selection Strategy . . . 9
2.2 Diversity Search and MAP-Elites . . . 11
2.3 Multi-objective and Many-objective Optimization . . . 14
2.3.1 Pareto Ranking . . . 15
2.3.2 Sum Of Ranks . . . 16
2.4 Image Features . . . 17
2.4.1 Feature Definition . . . 17
2.4.2 Unique Colour Triplets . . . 21
2.4.3 Mean Intensity . . . 21
2.4.4 Image Entropy . . . 23
2.4.5 Sobel Edge Detection . . . 25

3 Related Work **28**
3.1 Quality Diversity Search . . . 28
3.2 Image Classification . . . 29
3.3 Evolutionary Art . . . 31
3.3.1 Procedural Textures . . . 32
3.3.2 Procedural Textures Using MOP . . . 33
3.4 Challenges with Existing Systems . . . 33

4 System Design **34**
4.1 GP System 34
4.1.1 GP System Parameters 36
4.1.2 GP Language 37
4.2 Novel MAP-Elites Algorithm 38
4.2.1 Proposed MAP-Elites Algorithms 39
4.2.2 MAP-Elites System Parameters 41
4.3 GP and MAP-Elites System Integration 42
4.4 Fitness Function 42
4.4.1 Sum-of-ranks and MAP-Elites 43

5 Experiments and Results: Grayscale Image **44**
5.1 Experiment 1 48
5.1.1 Basic Genetic Programming (GP) 48
5.1.2 Base MAP-Elites: Single Individual Bin (MAP_Base) 51
5.1.3 MAP-Elites: Multiple individuals Bin - Random Selection (MAP_B5_R) 60
5.1.4 MAP-Elites: Multiple individuals Bin - Tournament Selection (MAP_B5_T2) 64
5.1.5 Analysis of experiment 1 67
5.2 Experiment 2 71
5.2.1 Basic Genetic Programming (GP) 71
5.2.2 MAP-Elites: Single Individual Bin (MAP_Base) 73
5.2.3 MAP-Elites: Multiple individuals Bin - Random Selection (MAP_B5_R) 74
5.2.4 MAP-Elites: Multiple individuals Bin - Tournament Selection (MAP_B5_T2) 79
5.2.5 Analysis of experiment 2 80
5.3 Experiment 3 84
5.3.1 Basic Genetic Programming (GP) 84
5.3.2 MAP-Elites: Single Individual Bin (MAP_Base) 87
5.3.3 MAP-Elites: Multiple individuals Bin - Random Selection (MAP_B5_R) 93
5.3.4 MAP-Elites: Multiple individuals Bin - Tournament Selection (MAP_B5_T2) 95
5.3.5 Analysis of experiment 3 97

6 Experiments and Results: RGB Image **102**
6.1 Experiment 4 106
6.1.1 Basic Genetic Programming (GP) 106

6.1.2 MAP-Elites: Single Individual Bin (MAP_Base) 109
6.1.3 MAP-Elites: Multiple individuals Bin - Random Selection (MAP_B5_R) 118
6.1.4 MAP-Elites: Multiple individuals Bin - Tournament Selection (MAP_B5_T2) 121
6.1.5 Analysis of experiment 4 124
6.2 Experiment 5 135
6.2.1 Basic Genetic Programming (GP) 135
6.2.2 MAP-Elites: Single Individual Bin (MAP_Base) 138
6.2.3 MAP-Elites: Multiple individuals Bin - Random Selection (MAP_B5_R) 142
6.2.4 MAP-Elites: Multiple individuals Bin - Tournament Selection (MAP_B5_T2)) 144
6.2.5 Analysis of experiment 5 146

7 Discussion **156**
7.1 Objective Behaviours 156
7.2 Overall Performance 158
7.3 Some of Our Preferred Solutions 159
7.4 Challenges of the Proposed Algorithms 160

8 Conclusion **161**
8.1 Conclusion 161
8.2 Comparison to related work 163
8.3 Future Work 163

Bibliography **173**

Appendices **174**

A Additional Analysis **174**
A.1 System Testing 174

List of Tables

2.1 Pareto ranking and sum of ranks. 15
2.2 First preliminary feature set (RGB) 18
2.3 Second preliminary feature set (HSV) 19

4.1 GP system parameters 36
4.2 Genetic programming function set 37
4.3 Genetic programming terminal set 38
4.4 MAP-Elites system parameters 41

5.1 GP parameter for grayscale experiment 45
5.2 MAP-Elites parameter for grayscale experiment 45
5.3 Grayscale Image Features 46
5.4 Grayscale feature set - 1 47
5.5 Grayscale feature set - 2 47
5.6 Grayscale feature set - 3 47
5.7 P-Value table for Experiment 1 67
5.8 P-Value table for Experiment 2 81
5.9 P-Value table for Experiment 3 97

6.1 GP parameter for grayscale experiment 103
6.2 MAP-Elites parameter for grayscale experiment 103
6.3 RGB (Colour) Image Features 104
6.4 RGB feature set 105
6.5 P-value table for experiment 4 124
6.6 P-value table for experiment 5 146

7.1 Objectives count with significant difference with at least one other algorithm 156
7.2 Average sum-of-ranks for all solutions from all the experiments 157

A.1 GP parameter for breast cancer experiment 174
A.2 GP function set for breast cancer experiment 175
A.3 GP terminal set for breast cancer experiment 176
A.4 Result of 10 run average of GP system and MAP-Elites system 179

List of Figures

2.1 A simple GP expression tree. 5

2.2 Crossover operation on parents. 8

2.3 Resulted offspring after crossover operation. 8

2.4 Original tree (left) and mutated offspring (right). 9

2.5 Roulette Wheel selection. 10

2.6 Tournament selection. 11

2.7 MAP-Elites MAP (2D) [X-axis: tree depth (range: 1- 17); Y-axis: number of nodes (range: 1- 400)]. 12

2.8 HSV colour model [6]. 20

2.9 RGB colour model [13] 21

2.10 RGB (top) and grayscale (bottom) version of the same image ("Mountain river" ©*Sheikh Faishal Basher*, 2016.) 22

2.11 High (top) and low (bottom) entropy images 24

2.12 Sobel edge detection 26

4.1 The architecture of the system. 35

5.1 Grayscale target images 46

5.2 Best result of each GP run for target image 1 and grayscale feature set 1 (20 runs). 49

5.3 Average mean fitness of GP population over 20 runs (lower is better). 50

5.4 Average best fitness of GP population over 20 runs (lower is better). . 50

5.5 Best results of base MAP-Elites runs for target image- 1 and grayscale feature set -1 (20 runs). 52

5.6 Sum-of-ranks MAP of run-19 of experiment-1 (higher is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)]. 53

5.7 Final Solution MAP of run-19 of experiment-1 [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)]. 54

5.8 Activity MAP of run-19 of experiment-1 (higher is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)]. 55
5.9 Raw feature difference of run-19 of experiment-1 (lower is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)] 57
5.10 Average mean fitness of base MAP-Elites MAP over 20 runs (lower is better). 58
5.11 Average best fitness of base MAP-Elites MAP over 20 runs (lower is better). 58
5.12 Best results of MAP-Elites (multiple individuals bin - random selection) runs for target image- 1 and grayscale feature set -1 (20 runs). . 61
5.13 Sum-of-ranks MAP of run-2 of experiment-1 (higher is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)]. 62
5.14 Average mean fitness of MAP-Elites (multiple individuals bin - random selection) MAP over 20 runs (lower is better). 63
5.15 Average best fitness of MAP-Elites (multiple individuals bin - random selection) MAP over 20 runs (lower is better). 63
5.16 Best results of MAP-Elites (multiple individuals bin - tournament selection) runs for target image- 1 and grayscale feature set -1 (20 runs). 65
5.17 Average mean fitness of MAP-Elites (multiple individuals bin - tournament selection) MAP over 20 runs (lower is better). 66
5.18 Average best fitness of MAP-Elites (multiple individuals bin - tournament selection) MAP over 20 runs (lower is better). 66
5.19 Error plot of mean minimum pixel value difference for different algorithms 68
5.20 Error plot of mean maximum pixel value difference for different algorithms 68
5.21 Error plot of mean mean pixel value difference for different algorithms 69
5.22 Error plot of mean median pixel value difference for different algorithms 69
5.23 Error plot of mean standard deviation value difference for different algorithms 70
5.24 Best results of GP runs for target image- 1 and grayscale feature set -2 (10 runs). 72
5.25 Best results of base MAP-Elites runs for target image- 1 and grayscale feature set -2 (10 runs). 73
5.26 Best results of MAP-Elites (Multiple individuals bin - random selection) runs for target image- 1 and grayscale feature set -2 (10 runs). . 75

5.27 Sum-of-ranks MAP of run-2 of experiment-2 (higher is better) [X-axis: standard deviation difference (0-1); Y-axis: mean difference (0-1)]. . . 76
5.28 Raw feature difference of run-2 of experiment-2 (lower is better) [X-axis: standard deviation difference (0-1); Y-axis: mean difference (0-1)] 77
5.29 Activity MAP of run-2 of experiment-1 (higher is better) [X-axis: standard deviation difference (0-1); Y-axis: mean difference (0-1)]. 78
5.30 Best results of MAP-Elites (multiple individuals bin - tournament selection) runs for target image- 1 and grayscale feature set -2 (10 runs). 79
5.31 Error plot of mean minimum pixel value difference for different algorithms 81
5.32 Error plot of mean entropy value difference for different algorithms . 82
5.33 Error plot of mean median pixel value difference for different algorithms 82
5.34 Best results of GP runs for target image- 2 and grayscale feature set -3 (10 runs). 85
5.35 Average mean fitness of GP population over 10 runs (lower is better). 86
5.36 Best results of base MAP-Elites runs for target image- 2 and grayscale feature set -1 (20 runs). 87
5.37 Sum-of-ranks MAP of run-5 of experiment-3 (higher is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)]. 88
5.38 Raw feature difference of run-5 of experiment-3 (lower is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)] 90
5.39 Average mean fitness of base MAP-Elites MAP over 10 runs (lower is better). 91
5.40 Activity MAP of run-5 of experiment-3 (higher is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)]. 92
5.41 Best results of MAP-Elites (Multiple individuals bin - random selection) runs for target image- 1 and grayscale feature set -3 (10 runs). . 93
5.42 Average mean fitness of MAP_B5_R MAP over 10 runs (lower is better). 94
5.43 Best results of MAP-Elites (multiple individuals bin - tournament selection) runs for target image- 2 and grayscale feature set -3 (10 runs). 95
5.44 Average mean fitness of MAP_B5_T2 MAP over 10 runs (lower is better). 96
5.45 Error plot of mean maximum pixel value difference for different algorithms 98
5.46 Error plot of mean mean pixel value difference for different algorithms 99
5.47 Error plot of mean median pixel value difference for different algorithms 99
5.48 Error plot of mean standard deviation difference for different algorithms 101
5.49 Error plot of edge count difference value for different algorithms . . . 101

6.1 RGB target images 104
6.2 Best result of each GP run for RGB target image- 1 and RGB feature set (10 runs). 107
6.3 Average mean fitness of GP over 10 runs (lower is better). 108
6.4 Average best fitness of GP over 10 runs (lower is better). 108
6.5 Best results of base MAP-Elites runs for RGB target image- 1 and RGB feature set (10 runs). 109
6.6 Sum-of-ranks MAP of run-6 of experiment 4 (higher is better) [X-axis: unique RGB value (0-65536); Y-axis: entropy (0-8)]. 110
6.7 Raw feature difference of red channel of run-6 of experiment 4 (lower is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)] . 112
6.8 Raw feature difference of green channel of run-6 of experiment 4 (lower is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)] . 113
6.9 Raw feature difference of blue channel of run-6 of experiment 4 (lower is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)] . 114
6.10 Activity MAP of run-6 of experiment 4 (higher is better) [X-axis: unique RGB value (0-65536); Y-axis: entropy (0-8)]. 115
6.11 Average mean fitness of base MAP-Elites MAP over 10 runs (lower is better). 116
6.12 Average best fitness of base MAP-Elites MAP over 10 runs (lower is better). 116
6.13 Best results of base MAP-Elites runs for RGB target image -1 and RGB feature set (10 runs). 118
6.14 Average mean fitness of MAP-Elites (multiple individuals bin - random selection) MAP over 10 runs (lower is better). 119
6.15 Average best fitness of MAP-Elites (multiple individuals bin - random selection) MAP over 10 runs (lower is better). 119
6.16 Best results of base MAP-Elites runs for RGB target image- 1 and RGB feature set (10 runs). 121
6.17 Average mean fitness of MAP-Elites (multiple individuals bin - tournament selection) MAP over 10 runs (lower is better). 122
6.18 Average best fitness of MAP-Elites (multiple individuals bin - tournament selection) MAP over 10 runs (lower is better). 122
6.19 Error plot of mean minimum pixel value difference of red channel for different algorithms 126

6.20 Error plot of mean maximum pixel value difference of red channel for different algorithms 126
6.21 Error plot of mean mean pixel value difference of red channel for different algorithms 127
6.22 Error plot of mean median pixel value difference of red channel for different algorithms 127
6.23 Error plot of mean standard deviation value difference of red channel for different algorithms 128
6.24 Error plot of mean minimum pixel value difference of green channel for different algorithms 129
6.25 Error plot of mean maximum pixel value difference of green channel for different algorithms 129
6.26 Error plot of mean mean pixel value difference of green channel for different algorithms 130
6.27 Error plot of mean median pixel value difference of green channel for different algorithms 130
6.28 Error plot of mean standard deviation value difference of green channel for different algorithms 131
6.29 Error plot of mean minimum pixel value difference of blue channel for different algorithms 132
6.30 Error plot of mean maximum pixel value difference of blue channel for different algorithms 132
6.31 Error plot of mean mean pixel value difference of blue channel for different algorithms 133
6.32 Error plot of mean median pixel value difference of blue channel for different algorithms 133
6.33 Error plot of mean standard deviation value difference of blue channel for different algorithms 134
6.34 Best result of each GP run for RGB target image- 2 and RGB feature set (10 runs). 136
6.35 Average mean fitness of GP over 10 runs (lower is better). 137
6.36 Best results of base MAP-Elites runs for RGB target image- 2 and RGB feature set (10 runs). 138
6.37 Sum-of-ranks MAP of run-6 of experiment 5 (higher is better) [X-axis: unique RGB value (0-65536); Y-axis: entropy (0-8)]. 139

6.38 Activity MAP of run-6 of experiment 5 (higher is better) [X-axis: unique RGB value (0-65536); Y-axis: entropy (0-8)]. 140
6.39 Average mean fitness of base MAP-Elites MAP over 10 runs (lower is better). 141
6.40 Best results of base MAP-Elites runs for RGB target image- 2 and RGB feature set (10 runs). 142
6.41 Average mean fitness of MAP-Elites (multiple individuals bin - random selection) MAP over 10 runs (lower is better). 143
6.42 Best results of base MAP-Elites runs for RGB target image 2 and RGB feature set (10 runs). 144
6.43 Average mean fitness of MAP-Elites (multiple individuals bin - tournament selection) MAP over 10 runs (lower is better). 145
6.44 Error plot of mean maximum pixel value difference of red channel for different algorithms 148
6.45 Error plot of mean mean pixel value difference of red channel for different algorithms 148
6.46 Error plot of mean median pixel value difference of red channel for different algorithms 149
6.47 Error plot of mean standard deviation value difference of red channel for different algorithms 149
6.48 Error plot of mean maximum pixel value difference of green channel for different algorithms 151
6.49 Error plot of mean mean pixel value difference of green channel for different algorithms 151
6.50 Error plot of mean median pixel value difference of green channel for different algorithms 152
6.51 Error plot of mean standard deviation value difference of green channel for different algorithms 152
6.52 Error plot of mean maximum pixel value difference of blue channel for different algorithms 154
6.53 Error plot of mean mean pixel value difference of blue channel for different algorithms 154
6.54 Error plot of mean standard deviation value difference of blue channel for different algorithms 155

7.1 Four of our favourite evolved images from RGB experiments. 159

A.1 A single run 17×20 MAP of breast cancer experiment [X-axis: tree depth (range: 1 - 17); Y-axis: total number of nodes (range: 1 - 400)]. 178

List of Algorithms

1 Genetic Programming Algorithm. 6
2 MAP-Elites Algorithm. 13
3 Proposed MAP-Elites Algorithms. 39

Chapter 1

Introduction

Evolutionary art (Evo Art) uses evolutionary algorithms (EA) to create art [21]. It is used for various art applications including computer generated images via procedural textures [21]. One of the key advantages in modern Evo Art is to generate artistic images in an automated process without needing to have continuous user feedback, as opposed to the original interactive systems [45].

Genetic programming (GP) is a widely used EA that has many applications, as it can generate acceptable solutions for various types of problems. It has long been used for image classification, for example, [24]. It has also been used for 2D image evolution based on image features. Computer vision played a major role in automatic Evo Art systems. Integration of computer vision algorithms with evolutionary algorithms has made it possible to automate image evaluation significantly [50]. Gentropy [91] and Genshade [52] are two examples of automatic Evo Art systems.

One of the issues with GP is that it only evolves one solution per run. Many runs are needed to generate a number of different solutions. GP tends to converge over multiple generations, and those solutions can often be similar. Also, GP evolution is time and resource expensive.

Diversity search is a new area that uses special algorithms to explore a diversity of solutions, rather than converging to one as typically the case [61, 62]. Diversity algorithms like novelty search can help with these issues by producing diverse solutions [61, 62].

Algorithms that combines fitness with diversity are called quality diversity (QD) algorithms [43]. Quality diversity algorithms solve the issues with both GP and novelty search. Gomez *et al.* showed that finding quality diverse solution is possible by combining fitness and novelty [43]. One of the recently developed QD algorithms is MAP-Elites [72], where a MAP grid holds multiple results and each individual in the MAP is a solution. The MAP is defined to hold an assortment of results exhibiting different combination of features.

Multi-objective and many-objective problems are another significant part of evolutionary computation [26]. Most real life problems have more than one objective to optimize. Some examples of many-objective optimization algorithms can be found in [63]. How we handle and optimize objectives determines the quality of the solution. There are various techniques to optimize multiple-objective and many-objective problems. One of the most popular and widely used technique is Pareto ranking [36, 74, 93, 95]. Pareto ranking faces challenges when there are more than four objectives. Therefore, much research is being undertaken to extend Pareto-based analysis towards many-objective optimization [63]. Another strategy is sum-of-ranks, which can handle many objectives.

In this research, we have designed a system that combines MAP-Elites and GP by introducing into it the many-objective sum-of-ranks (or averaged rank) strategy [22, 33]. We also propose two completely new versions of MAP-Elites that use multiple individual bins compared to the single individual bin in the original MAP-Elites [72]. We have used sum-of-ranks as fitness measure to handle many-objective problems. We apply the new algorithms to procedural image evolution using GP. Here, we have used up to 15 objectives in our experiments, so a many-objective approach is necessary.

To generate procedural texture images, Lombardi *et al.*'s [64] lightweight image feature set is used. Salimi [81] used the same feature set to produce procedural texture images using GP. We have used GP to generate images as a benchmark, and then used MAP-Elites algorithm combined with GP, to generate a diversity of images.

1.1 Goals and Motivation

This thesis is mainly inspired by the research of Mouret and Clune [72]. It extends their work and proposes two new version of MAP-Elites algorithm. The main goal of this research is to design a new approach to find diverse and high quality solution.

Main goals are as follows:

- Introducing many objective problems to MAP-Elites by introducing sum of ranks.
- Extend MAP-Elites by permitting multiple solutions per MAP cell. We also examine two different selection strategies for selecting individuals from cells during reproduction.
- Generating a diverse number of solutions compared to GP, giving the user a large selection to choose from.

The problem set that has been tested for this experiment is an evolutionary art problem for which GP has been used traditionally. The main drawback of that is GP only provides one solution after each run. Using MAP-Elites help create diverse solutions. Lombardi *et al.'s* lightweight image feature sets [64] is used to define image features as objectives, as the proposed feature set provides many features to choose from. This approach will help create a diverse set of procedural texture images in each single run.

1.2 Main Contributions

The main contributions of this thesis are as follows:

- Perform a literature review of some of the previous work done with many-objective problems, different selection and fitness strategy.
- Combines sum-of-ranks fitness measure for many-objective problems with MAP-Elites.
- Propose two new versions of the original MAP-Elites to find more diverse and better solution than the original one.

- Performing detail experiment with procedural texture generation from digital images to provide the user with a variety of choice to choose from instead of just one provided by GP.
- Performing details performance analysis among GP, original MAP-Elites and the two proposed algorithms.

1.3 Thesis Structure

The structure of this thesis is as follows:

- Chapter 2 contains background information about genetic programming (GP), multi-objective and many-objective problems, procedural texture, image features and diversity search.
- Chapter 3 presents a literature review of quality diversity algorithms, multi-objective optimization, and evolutionary art.
- Chapter 4 discusses the system architecture including algorithms, parameters, and evaluation functions. GP and MAP-Elites parameters are discussed. Two new versions of the MAP-Elites algorithm are described.
- Chapter 5 describes the results of experiments and performance analysis of different experiments using grayscale images. Three different cases are performed. Performance of all four algorithms used in this research, including GP, original MAP-Elites and the two new MAP-Elites strategies, are analyzed.
- Chapter 6 describes the results of experiments and performance analysis RGB images.
- Chapter 7 summarizes all experiments. It compares their overall performance.
- Chapter 8 concludes the thesis and discusses future work.

Chapter 2

Background

2.1 Genetic Programming

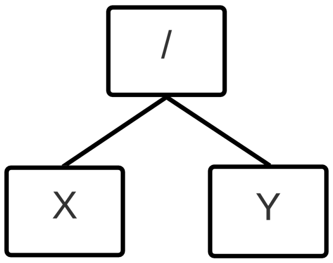


Figure 2.1: A simple GP expression tree.

Genetic programming (GP) is a evolutionary algorithm that uses the ideas of Darwinian evolution to generate programs that solve complex problems [59, 60, 76]. It is an evolutionary computation (EC) approach that extends genetic algorithms (GA) [51]. It was first proposed by John Koza in 1992 [59], which introduced a way to evolve tree-based structures. GP evolves a population over number of generations to find better offspring and ultimately find a suitable program that solves a problem at hand.

Algorithm 1 Genetic Programming Algorithm.

```
$P \leftarrow population\ size$;
$N \leftarrow maximum\ number\ of\ generations$;
$mutpb \leftarrow mutation\ probability$;
$cxpb \leftarrow crossover\ probability$;

randomly initialize a population of size P;

while (! terminal condition) do
    evaluate fitness of each individual;
    $i \leftarrow 0$
    while ($i \leq N$) do
        select an operator: op;
        if (mutpb % of time) then
            select an individual based on its fitness;
            perform mutation operation with probability mutpb;
        else
            if (cxpb % of time) then
                select two individuals based on their fitness;
                perform crossover operation with probability cxpb;
            end if
        end if
        insert offspring into next generation population;
        i++;
    end while
end while
return (most fit individual);
```

All of the evolved programs (individuals) are represented in a tree structure (see Figure 2.1). Each tree contains functions as intermediate nodes and terminals as leaf nodes. The GP language is comprised of these functions and terminal sets. These sets are specially defined for the problem at hand. Functions can include mathematical, trigonometric, logical, or conditional operators, and terminals can simply be a constant or some variable value. For example, in the GP individual expression (see Figure 2.1), X and Y are terminals or leaf nodes, and / is a function meaning it is an intermediate node, and in this example the root node. This denotes the expression the expression X/Y.

The genetic programming algorithm is a simple one (see Algorithm 1). It first produces a random population using the GP language provided by the user. It then reproduces offspring over number of generations using various genetic operations to find an acceptable solution [60].

Algorithm 1 shows the genetic programming algorithm. First user defined number of trees are created by randomly using the functions and terminal set. It is very unlikely that the initially generated population will be able to provide a good solution. All the individuals in a population go through a fitness evaluation, which helps to find with individuals with good fitness.

All individuals (trees) are assigned a fitness score after the fitness evolution. Genetic operators (mutation and crossover) are probabilistically chosen for use. Selection methods use the fitness score of individuals to select parents for reproduction: the higher the fitness of an individual, the higher the chance it will be selected for reproduction. Newly produced populations replace the old population during each generation. This process continues until the terminal condition is met. At the end of the run, the individual with the best fitness from the final generation is returned. More details on the reproduction and selection strategy is provided in Section 2.1.1.

2.1.1 Reproduction Operation

GP uses different reproduction operations to produce offspring for next generation of population.

Crossover:

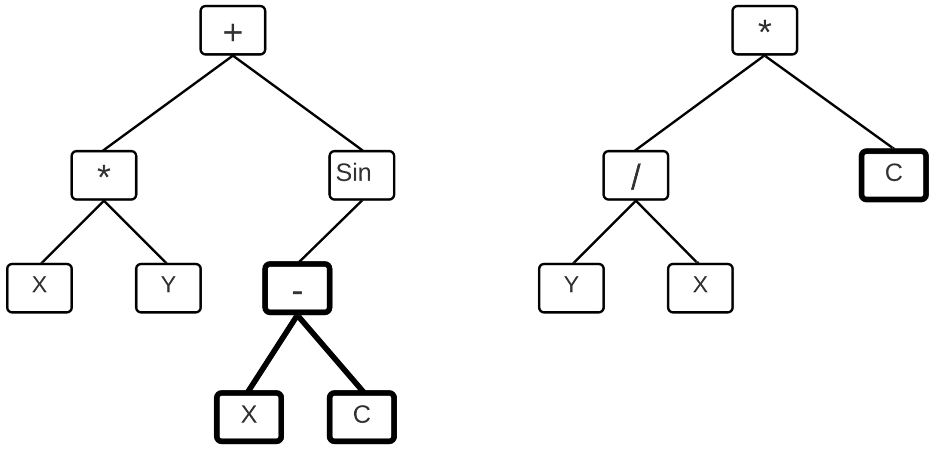


Figure 2.2: Crossover operation on parents.

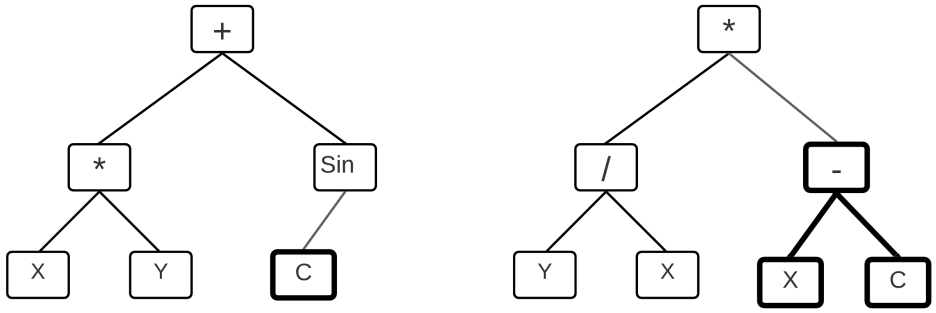


Figure 2.3: Resulted offspring after crossover operation.

One of the key reproduction operations is crossover. It is inspired by the biological crossover where offspring inherit features from both parents. It is the most widely used reproduction operation in GP. To perform a crossover operation, two parents are selected from the current generation of population based on fitness. Then a random node from each tree is selected. The subtrees are split from those nodes and then exchanged between trees, creating two new trees that are different than the parents. Figure 2.2 shows the crossover operation where two subtrees are selected from the parents. Then these subtrees are swapped creating two new offspring, as shown in Figure 2.3.

Mutation

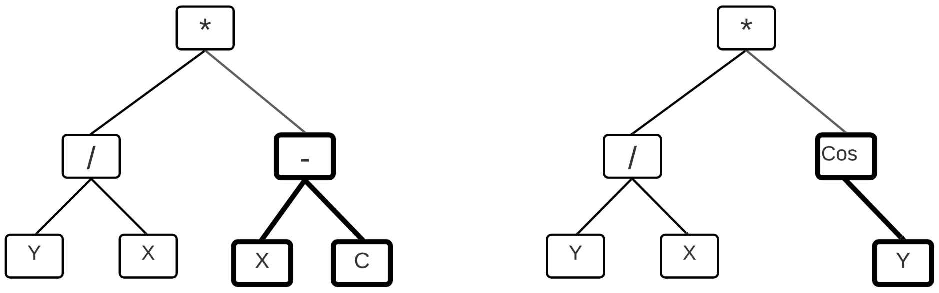


Figure 2.4: Original tree (left) and mutated offspring (right).

Mutation is another reproduction operation. Though in most applications it is not used very frequently, it helps create more diverse offspring. For mutation, only one parent is selected based on fitness. Then a random subtree is selected from the parent which is then replaced with a randomly generated tree. The new random tree generation follows the same procedure as the initial population generation (see Section 2.1). In Figure 2.4, the left tree is the original tree, and the right tree is a mutated offspring where the randomly selected subtree in the original tree has been replaced with a newly generated subtree.

2.1.2 Selection Strategy

There are different fitness based selection strategies that are used in GP to find fit individual from a population for reproduction. Most commonly used strategies include roulette selection and tournament selection. In the roulette selection, selected individuals are assigned a percentage of the roulette wheel [42]. The bigger the fitness value the more space it is assigned on the wheel. Then the wheel is spun to decide which one is selected. Bigger fitness values mean higher portions on the wheel and a higher selection probability [94]. Figure 2.5 shows a simple roulette wheel with 6 individuals. Here, Ind 4 has the highest fitness value and Ind 2 has the lowest fitness value. So, the probability of Ind 4 being selected is higher compared to Ind 2. And since Ind 1 and Ind 5 has similar fitness value, probability of them being selected compared to each other is very similar.

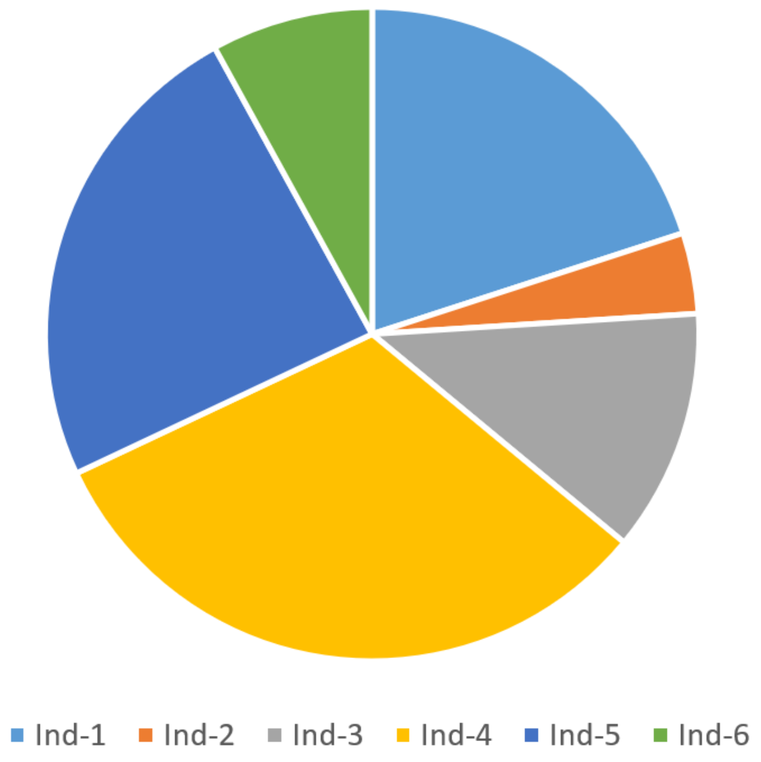


Figure 2.5: Roulette Wheel selection.

Tournament selection is another widely used selection strategy. N number of individuals are selected from a population at random. The individual with the highest fitness among those randomly chosen individuals is selected for reproduction. Since only one individual is selected from a tournament, mutation only needs one tournament selection, where crossover needs two tournaments to get two parents [70]. Figure 2.6 shows the fitness of N individuals that makes up the population. Four individuals from them are chosen at random. Since Ind 1 has the highest fitness value of 5 among the chosen individuals, it is selected for reproduction.

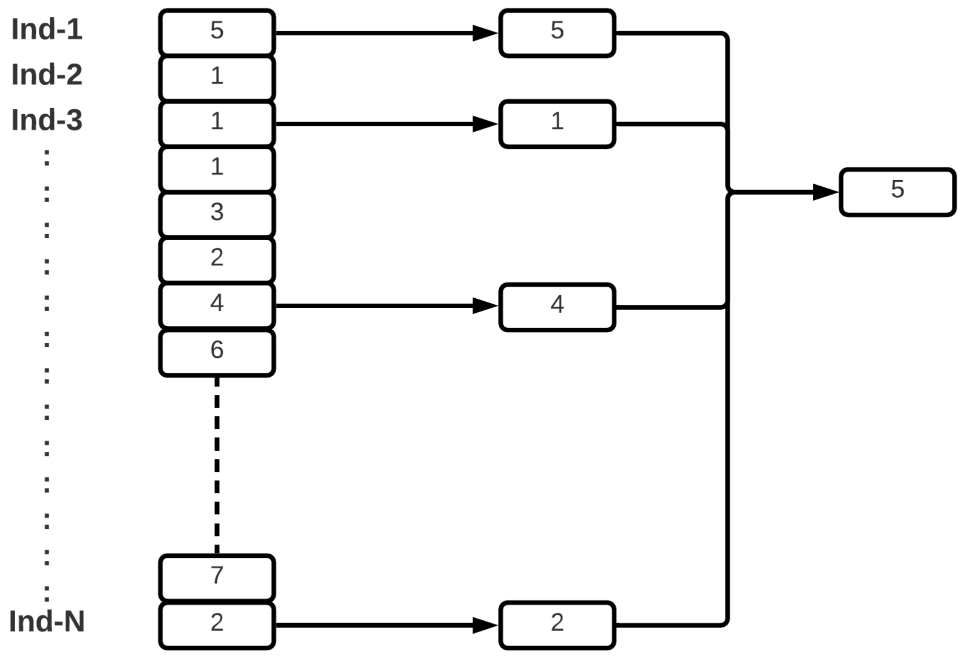


Figure 2.6: Tournament selection.

2.2 Diversity Search and MAP-Elites

Diversity search algorithms consider diversity as an important factor during search. The first diversity search algorithm is novelty search [61] which uses a measured diversity of individual from rest of the population.

MAP-Elites is another diversity search algorithm, known as a quality diversity (QD) algorithm, and was first proposed in 2015 by Mouret and Clune [72]. It is a new algorithm for finding multiple high-quality solutions that exhibit different phenotype behaviours. Each cell of the map contains the chromosome of the most fit individual found so far exhibiting the behaviour saved in that cell. For example, a 10-by-10 2D grid can represent a maximum of 100 possible solutions. Each dimension (X, Y) of the grid represents identified combinations of useful behaviour measurements (X and Y) as defined by the user. Figure 2.7 shows a 10-by-10 heat MAP of final solutions. Empty bins mean no solution were found in that space of the MAP.

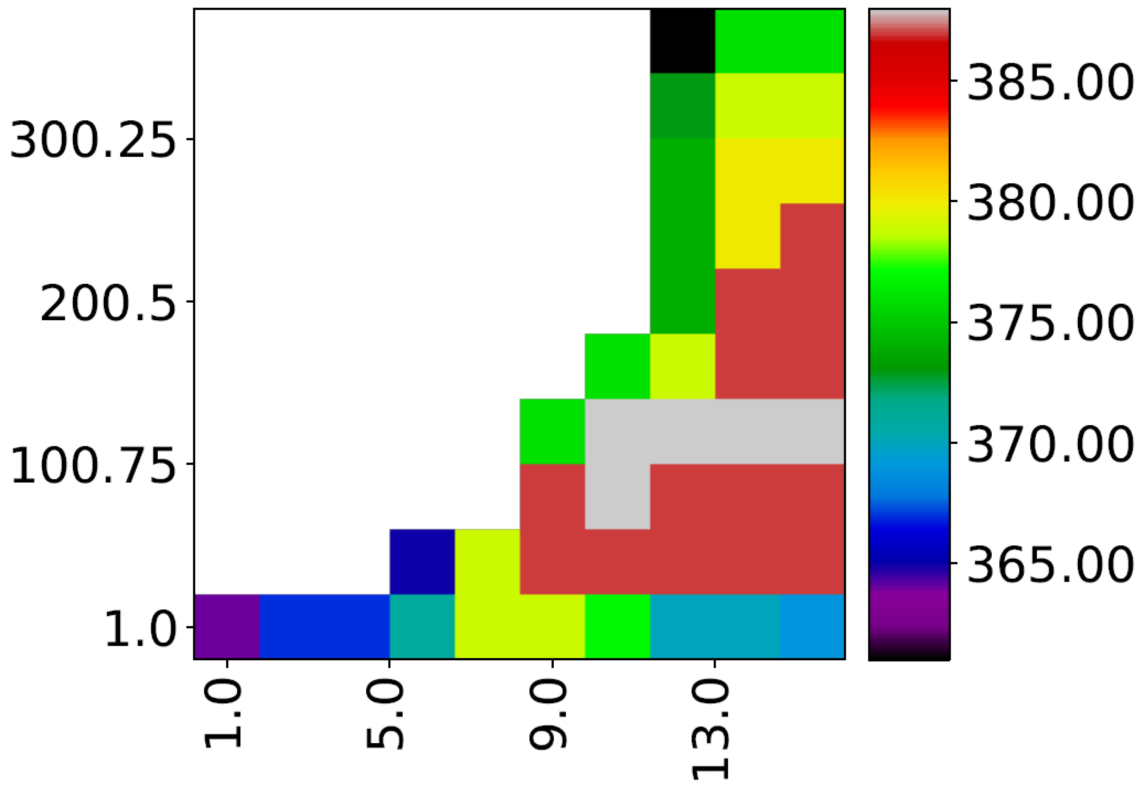


Figure 2.7: MAP-Elites MAP (2D) [X-axis: tree depth (range: 1- 17); Y-axis: number of nodes (range: 1- 400)].

The basic MAP-Elites algorithm is fairly simple (see Algorithm 2). First a MAP of N-dimension is created. Each dimension has its own behaviour feature. On the first iteration, P number of solutions are created and they are placed in different bins on the map based on their feature values. In each iteration a new individual is created using crossover and based on its feature values, it is assigned a bin on the MAP. If that cell (bin) is empty then the new individual is placed in that cell. If the cell is already occupied, the fitness value of current occupant and the new individual is compared. If the new individual has a better fitness value than the occupant of that cell, it replaces the current occupant and takes its place in the map. But, if the new individual has lower fitness than the current occupant, it is discarded. This process continues until the termination condition is met. At the end of a run all the individuals in the bins are presented as solutions. The user can decide on the most appropriate one.

Algorithm 2 MAP-Elites Algorithm.

```
$M \leftarrow MAP\ dimension$;
$N \leftarrow maximum\ number\ of\ iterations$;
$x \leftarrow a\ random\ elites\ on\ the\ MAP$;
$y \leftarrow current\ occupant\ of\ a\ cell$;
$p(x) \leftarrow fitness\ of\ x$;
$f(y) \leftarrow feature\ description\ of\ x\ (Used\ to\ find\ a\ cell\ on\ the\ MAP)$;

randomly initialize a P number of solutions;
place these solutions on MAP based on their features;

$i \leftarrow 0$
while $(i \leq N)$ do
    selects individual z from the elites of the MAP;
    using crossover creates a new randomly modified version of z: z';
    checks feature description of z': f(z');
    checks the fitness of x': p(z');

    if (f(z') == empty ) then
        place z' in MAP space f(z');
    else
        if (p ( z') >p (y) ) then
            replace y with z';
        else
            discard z';
        end if
    end if
    i++;
end while
return (final MAP individuals)
```

2.3 Multi-objective and Many-objective Optimization

Many modern day problems have more than one objective to be satisfied or optimized while finding a solution. Such problems are called multi-objective problems (MOP). If there are more than four objectives to satisfy, then the problem is called a many objective problem (MaOP) [38, 54]. While it can be easy to optimize a single objective, optimizing multiple objective at once is often tricky. And when these multi-objective problems become many-objective problems (number of objectives are five or more), it becomes even more complicated. Optimizing one objective can lead to bad performance in others. Balancing these issues is a major challenge in any multi-objective problem. The goal of MOP or MaOP optimization is to satisfy all the objectives to find a solution [26].

Many different algorithms have been proposed to better optimize many-objective problems. There are many strategies for ranking multi-objective problems [41, 31]. Pareto ranking is one of the most used ranking method for multi-objective optimization [36, 74, 93, 95]. But Pareto ranking fails when there is five or more objectives, as it fails to find Pareto dominance in most case. Sum-of-ranks (also called average rank) can help solve this issue [23, 49].

2.3.1 Pareto Ranking

Table 2.1: Pareto ranking and sum of ranks.

Ind	**Raw fitness**				**PR**	**Ranks**				**SOR**	**RR**	**Norm. SOR**	**RR**
	A	**B**	**C**	**D**		**A**	**B**	**C**	**D**				
1	1	9	5	4	1	2	1	2	2	7	1	1.47	1
2	2	30	4	8	1	3	2	1	3	9	2	2.03	2
3	10	9	9	10	2	4	1	4	4	13	4	2.60	5
4	16	30	8	4	2	5	2	3	2	12	3	2.57	4
5	16	9	40	0	1	5	1	5	1	12	3	2.37	3
6	0	50	50	50	1	1	3	6	5	15	5	3.20	6
					Max rank =	5	3	6	5				

(a) In this example lower value is preferred for both fitness and rank.

(b) PR: Pareto ranking; SOR: Sum-of-ranks; RR: Re-ranked.

Pareto ranking (PR) is based upon Pareto dominance [42]. If a solution is dominating in one objective and remains undominated in others, it is given a rank and then the process continues for the rest of the solutions until all the solutions are assigned a rank [22]. Once a solution receives a rank it is no longer considered while ranking the rest of the solutions. The first undominated solutions receive a rank of one, the undominated solutions in the next iteration receives a rank of two, and so on. This is an effective process for multi-objective problems. It work best when there are two or three objectives [53]. But it usually fails when there are more than four objectives [42, 74, 95].

Table 2.1 shows an example of Pareto ranking. The first column represents 6 individuals that have been assigned name 1, 2, 3, 4, 5 and 6. Raw fitness column shows raw fitness of four objectives (A, B, C and D) of those mentioned individuals. Column PR, shows the Pareto ranking of those individuals. We can see 4 individuals received a rank of 1, meaning those four individuals dominate in one of the objectives and were undominated in other three. In the second iteration the remaining two were evaluated and as they dominate the other one in one objective and remains undominated in other three, they both received a rank of 2.

2.3.2 Sum Of Ranks

Sum-of-ranks (SOR, also known as average rank space) is technique for MaOP. It helps to overcome the problem with Pareto ranking [33], since Pareto ranking usually fails when there are four or more objectives to consider. All the individuals are ranked for each of their objectives based on their raw fitness value for that objective. If higher value objectives are preferred, the individual with highest fitness value is assigned a rank of 1 for that objective, the second highest fitness value receives a rank of 2 and so on. If a lower value is preferred then the individual with lowest fitness score for an objective receives rank of 1 for that objective and so on. Once all the individuals are ranked for all the objectives, the ranks for different objectives are added. This gives all the individual a sum of their ranks [49].

Different modified version of sum-of-ranks can be used. To reduce bias with objectives with larger rank values than others, normalized sum-of-ranks is used where all the ranks for an objective are divided by the highest rank for that objective. If one objective is more important and preferable, then a weighted sum of rank is used, where a simple weight is multiplied to the normalized values.

Table 2.1 shows different ranking techniques that include Pareto ranking, sum-of-ranks and normalized sum-of-ranks. Here, a lower value is preferred. From the table we can see the difference between Pareto ranking (PR), sum-of-ranks (SOR) and normalized sum-of-ranks (Norm. SOR). We can notice a difference between sum-of-ranks and normalized sum-of-ranks. This is a result of different range of ranks for different objectives. If not normalized, it can make the solutions biased toward a few the objectives at the expense of others. Normalizing them ensures that the difference in rank range will not make the solution biased towards the objective that has a higher rank range. If we want to assign different weights to different objectives, we can assign weights to the normalized ranks to prevent unwanted behaviour. The fourth to last column shows the raw sum-of ranks and second to last column shows the normalized sum-of-ranks. For normalization we divide the ranks of an objective with the highest rank for that objective. We have re-ranked the SOR in third to last column and normalized SOR in last column. From these two column we can see that the rank of raw sum-of-ranks is not same as the rank of normalized sum-of-ranks. Which shows the benefits of the normalized sum-of-ranks

2.4 Image Features

This chapter describe different features of the images that are used in this research. The original feature set used here is from Lombardi *et al* [64]. Key reasons for choosing these features is their wide use in similar research in image retrieval, evolutionary art for their simplicity and ease to compute [81].

2.4.1 Feature Definition

Lombardi *et al.* [64] used lightweight features for comparison and classification of art work. It showed promising results as art from same artists are shown to have closer feature characteristics in cases studied. In their research, two types of features has been considered for the test image set: palette and canvas features. Palette features are related to the colour space of an image, and canvas features are related to frequency measurement of an image.

In their research, Lombardi *et al.* [64] have used two sets of feature derived from the colour images. The first set is based on the RGB [13, 83] colour model, and the second set is mainly based on the HSV [6, 30, 83] colour model. The first feature set has 16 features, including 1 palette feature called *palette scope*, which is the total number of unique RGB triplets present in an image. 15 canvas features are the min, max, mean, median and standard deviation value of the 3 channels of the RGB model of an image. Table 2.2 shows the first feature set used.

Table 2.2: First preliminary feature set (RGB)

Feature Name	**Type**	**Description**
Palette Scope	Palette	The total number of unique RGB triples in an image.
Red Max	Canvas	The maximum value in the Red channel.
Red Min	Canvas	The minimum value in the Red channel.
Red Mean	Canvas	The arithmetic mean of the values in the Red channel.
Red Median	Canvas	The median of the values in the Red channel.
Red Std. Dev.	Canvas	The standard deviation of the values in the Red channel.
Green Max	Canvas	The maximum value in the Green channel.
Green Min	Canvas	The minimum value in the Green channel.
Green Mean	Canvas	The arithmetic mean of the values in the Green channel.
Green Median	Canvas	The median of the values in the Green channel.
Green Std. Dev.	Canvas	The standard deviation of the values in the Green channel.
Blue Max	Canvas	The maximum value in the Blue channel.
Blue Min	Canvas	The minimum value in the Blue channel.
Blue Mean	Canvas	The arithmetic mean of the values in the Blue channel.
Blue Median	Canvas	The median of the values in the Blue channel.
Blue Std. Dev.	Canvas	The standard deviation of the values in the Blue channel.

Table 2.3: Second preliminary feature set (HSV)

Feature Name	**Type**	**Description**
Hue Max	Canvas	The maximum value in the Hue channel.
Hue Min	Canvas	The minimum value in the Hue channel.
Hue Mean	Canvas	The arithmetic mean of the values in the Hue channel.
Hue Median	Canvas	The median of the values in the Hue channel.
Hue Std. Dev.	Canvas	The standard deviation of the values in the Hue channel.
Saturation Max	Canvas	The maximum value in the Saturation channel.
Saturation Min	Canvas	The minimum value in the Saturation channel.
Saturation Mean	Canvas	The arithmetic mean of the values in the Saturation channel.
Saturation Median	Canvas	The median of the values in the Saturation channel.
Saturation Std. Dev.	Canvas	The standard deviation of the values in the Saturation channel.
Value Max	Canvas	The maximum value in the Value channel.
Value Min	Canvas	The minimum value in the Value channel.
Value Mean	Canvas	The arithmetic mean of the values in the Value channel.
Value Median	Canvas	The median of the values in the Value channel.
Value Std. Dev.	Canvas	The standard deviation of the values in the Value channel.
Intensity Mean	Canvas	The global brightness of an image.
Colour Entropy	Canvas	The degree of disorder in the frequency distribution of colours.
Line Count	Canvas	The number of lines detected by the Sobel edge detector.

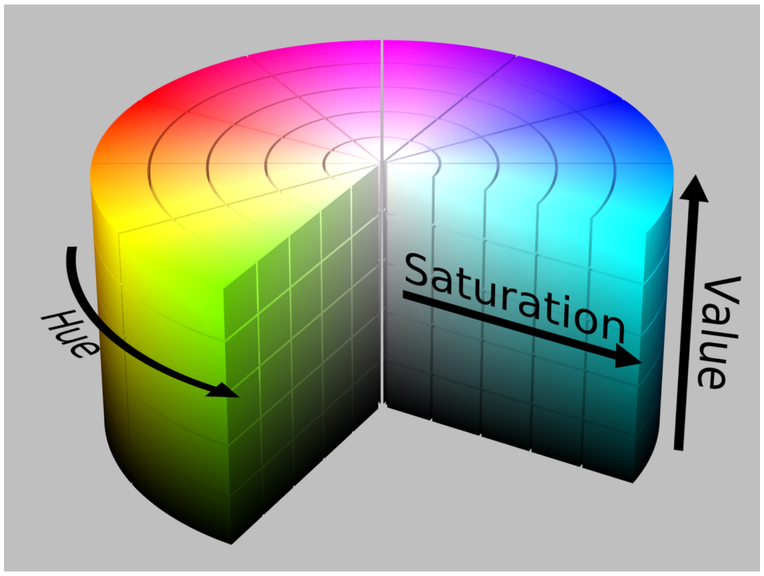


Figure 2.8: HSV colour model [6].

The Min and Max refers to the minimum and maximum value of the channel. Mean refers to average value of a channel, and Median refers to the median value of a channel. *Std Dev* refers to the standard deviation of values of the corresponding channel in a colour model.

The second set of feature is showed in Table 2.3. This set has 18 features which includes 15 values extracted from the HSV [30, 6] colour model of an image and 3 different features. The 15 HSV model values include the minimum, maximum, average, median and standard deviation values of hue, saturation and value channels of the image. Other values include the minimum intensity, colour entropy, and line count found by Sobel edge detection [14] which is described later in this chapter. Figure 2.8 shows a sample HSV colour model.

2.4.2 Unique Colour Triplets

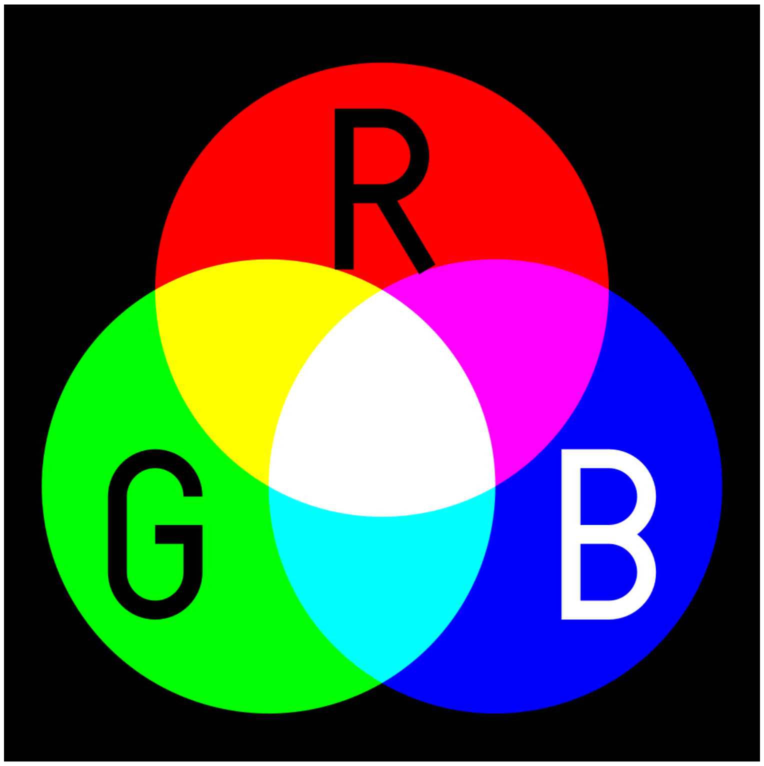


Figure 2.9: RGB colour model [13]

Unique triplets refers to the number of unique pixel value combinations in a RGB image. In the RGB colour model, the red, green and blue channel, all can be treated as separate grayscale image. And all pixels of each channel have their own values. Different combinations of the pixel values in these 3 different channels create different colours [77]. Figure 2.9 shows a sample of RGB colour model. Different shades of colour values come from the different RGB combinations.

2.4.3 Mean Intensity

Mean intensity refers to the average intensity of a grayscale image. It can be calculated by converting a RGB image into grayscale image and then taking the average intensity of the grayscale image. Figure 2.10 shows the RGB and grayscale version of the same image.

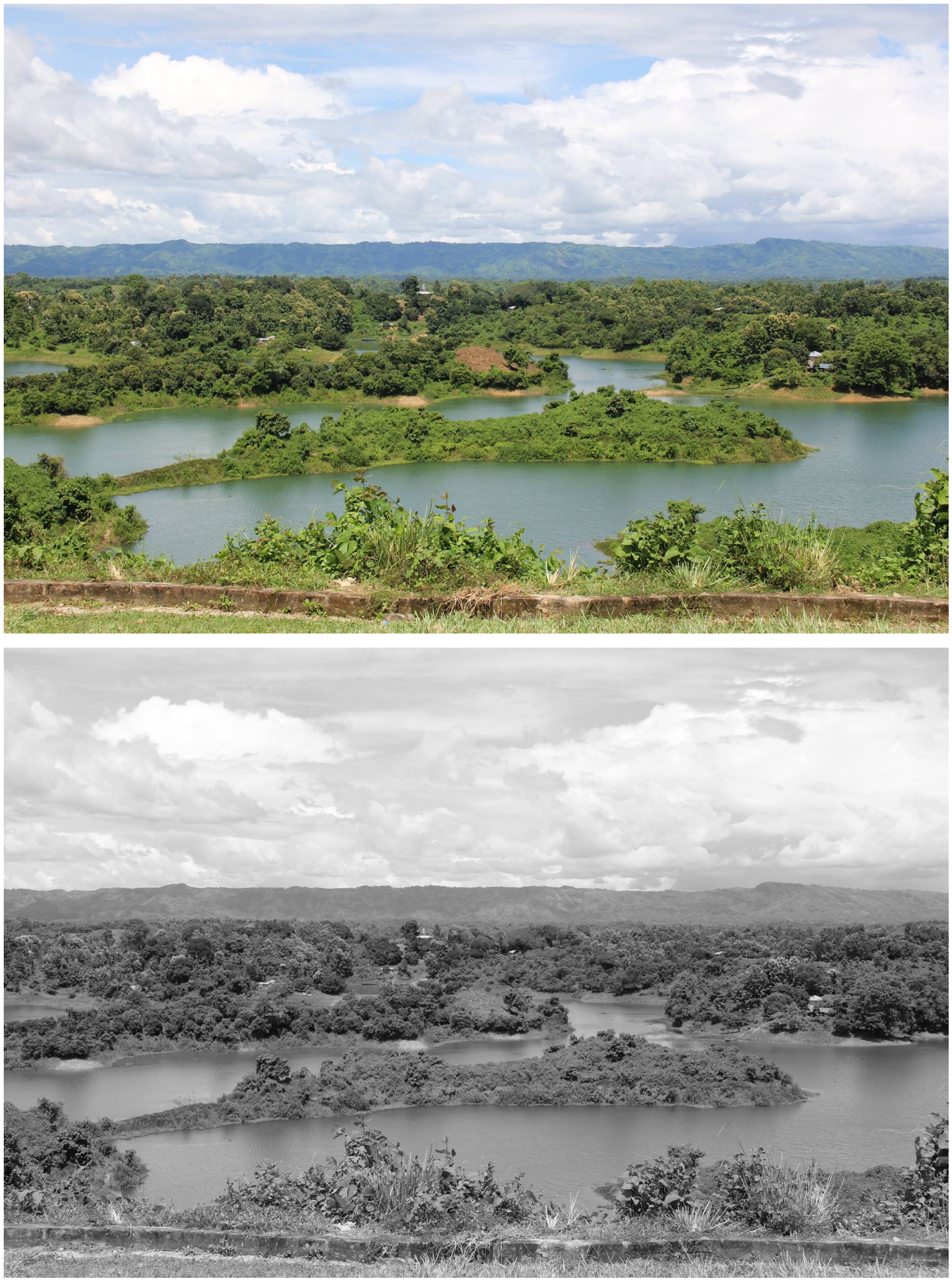


Figure 2.10: RGB (top) and grayscale (bottom) version of the same image (“Mountain river” ©*Sheikh Faishal Basher*, 2016.)

2.4.4 Image Entropy

Image entropy [4, 5] is a very useful feature of computer vision. It refers to the statistical randomness of information content of an image [44, 16]. Shannon entropy has been used in this research. The idea was proposed by Claude Shannon as a part of the field of information theory in 1948 [27].

The probability of the occurrence of different grayscale values is taken into account to calculate the entropy. Entropy can simply be described as a measure of uncertainty. If the occurrence of different events are similar then less information can be extracted from the probability distribution of each event. But, if the occurrence varies widely, more information can be extracted. In first case, as little information can be received it will have a low entropy. And in second case, entropy values will be higher as there are is more information present [92].

Figure 2.11 shows two images with high and low entropy. From the top figure, we can see that how rapidly varying colour and intensity can increase the entropy value. On the other hand, the bottom image has one colour and intensity value, for which it has a uniform probability distribution, meaning it has a very low entropy compared to the top image.

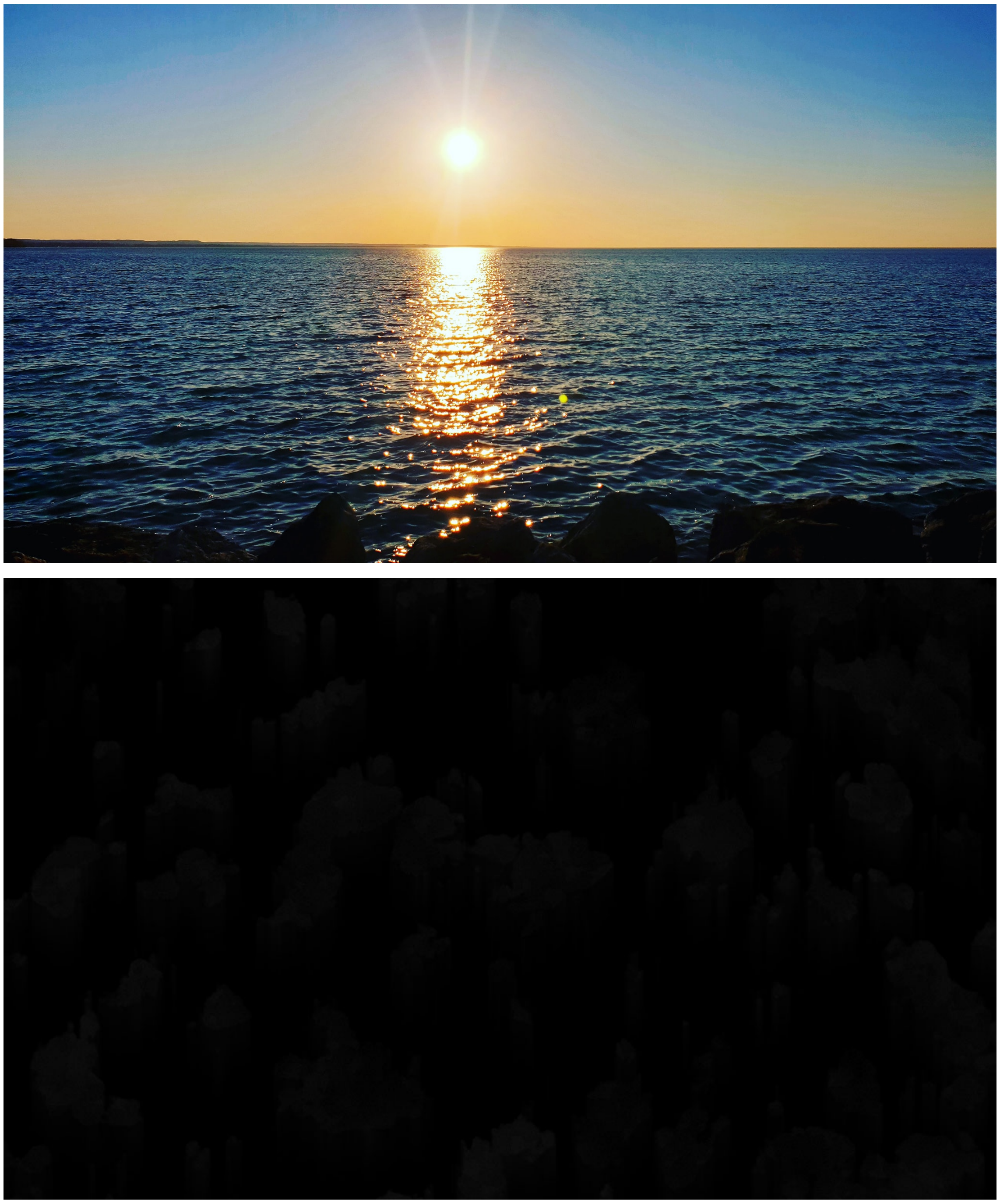


Figure 2.11: High (top) and low (bottom) entropy images

2.4.5 Sobel Edge Detection

Edge detection is a technique in image processing, where boundaries are found based on discontinuity of brightness within an image [2, 3]. Edge detection provides valuable information about an image by detecting different edges and rapid change of brightness in an image. It is used in digital image processing [44], and computer vision for image segmentation [7, 8, 47]. The most common edge detection methods include Sobel [14] , Kenny [29], Prewitt, Roberts and fuzzy edge detection [46, 84].

Sobel edge detection [14] is one of the widely used edge detection techniques. Lombardi *et al.* [64] used Sobel edge detector for the line count feature of the second preliminary feature set. Line count is interpreted as number of edges detected by the Sobel edge detector algorithm for a user provided threshold value. The algorithm takes a pixel as input and checks its neighbouring pixels for a measured change in brightness.

Sobel filters use different $N \times N$ convolution filter matrices, where each value in the matrix is used for multiplying a same size of window pixels in the input image. Equations 2.1 and 2.2 shows Gx and Gy, which are two matrices used for horizontal and vertical edge detection respectively.

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \times (InputImage) \tag{2.1}$$

$$Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \times (InputImage) \tag{2.2}$$

If we consider *Im* (Equation 3.2) to be a input image segment: each *P* represents a pixel, and *(i,j)* represents the coordinate. Each pixel value is multiplied with the matrices (equations 2.1 and 2.2). This process is continued through the whole image to find *Gx*, and *Gy*. These values are then used to calculate $|G|$ (see Equation 2.4).

$$Im = \begin{bmatrix} P_{(i-1,j-1)} & P_{(i-1,j)} & P_{(i-1,j+1)} \\ P_{(i,j-1)} & P_{(i,j)} & P_{(i,j+1)} \\ P_{(i+1,j-1)} & P_{(i+1,j)} & P_{(i+1,j+1)} \end{bmatrix} \tag{2.3}$$

$$|G| = \sqrt{Gx^2 + Gy^2} \tag{2.4}$$

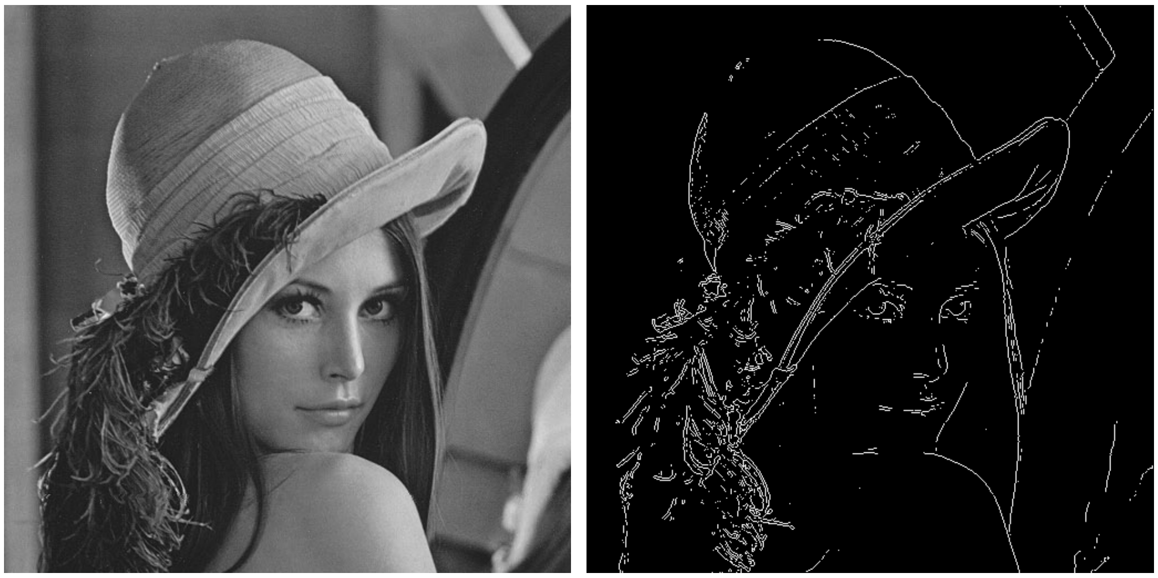

(a) Original image (b) Sobel filter with 0.1 threshold

Figure 2.12: Sobel edge detection

Figure 2.12 shows the grayscale image (a) and its Sobel-filtered version (b). From the figure we can clearly see that some of the pixels selected as edge might not be actual edges. Also, some edge pixels have have not been detected. This is because of how the Sobel edge detector algorithm detects edges. We can change the threshold value for intensity difference to detect edges. This threshold value can be changed based on user's preference.

Chapter 3

Related Work

3.1 Quality Diversity Search

Diversity search, as its name implies, focuses on increasing the diversity of solutions. There are different ways to increase the diversity of a solution. Some algorithms only focus on diversity by rewarding behaviours that help to create diverse solutions [61]. Novelty search is one such approach, where novel behaviours are rewarded and similar behaviours to previous solutions are penalized. This helps to increase the diversity by promoting novel behaviours, meaning, solutions that are not similar to previously found solutions [73]. Novelty search can be used to overcome local minima issues in deceptive problems (for example, maze navigation problems). It can also be used for multi-objective problems as well [71]. So, search for novelty is helpful to find solutions more effectively. But this approach of only valuing the novel behaviours can lead to other issues. Some problems require fitness to be considered, and only novelty alone will produce diverse results that are useless. For these types of problems no solution may be found if one solution reaches close to the final destination. The future solutions may go completely opposite direction to reach the goal to increase the novelty, thus creating useless diverse solutions.

This issue with using just novelty can be solved by combining novelty and fitness. By doing so, we are rewarding both behaviours. So, the algorithm will be motivated to go towards the actual solution while increasing diversity. When we add any fitness measurement to the diversity it becomes a quality diversity (QD) algorithm. Gomez *et al.* showed how combining fitness with novelty can help to find adequate solutions while keeping them diverse [43].

MAP-Elites [72] is a relatively new quality diversity algorithm that diversify solution based on some selective features. One of its advantages is that the user has full control over what features are selected and used for diversification. MAP-Elites helps to solve some of the issues with traditional evolutionary algorithms. It has been used for various problem sets in evolutionary computation and showed promising results even with multiple objectives [82].

Tarapore *et al.* used MAP-Elites to enhance the control of robots by evolving closed loop controller where robots uses sensors for communications [89]. Khalifa *et al.* used a modified version of MAP-Elites for bullet hell games to generate instructions for the agent [58]. Fontaine *et al.* used MESB (MAP-Elites with Sliding Boundaries) for re-balancing purpose of a card game called Hearthstone [39]. In their paper they used a sliding boundary instead of a fixed one for MAP-Elites, and the boundaries can change over time. MAP-Elites has also been used for evolutionary dungeon design for role-playing games by Alvarez *et al.* [17, 18]. Other uses of MAP-Elites include genetic programming, deep neuroevolution, adaptive sampling for noise domains, constrain optimization, and many more [32, 35, 37, 56].

3.2 Image Classification

In recent years, there has been a surge in digital images with the emergence of easier ways to make digital photos. In terms of digital art work, there has also been a boom as art from painters are being turned into digital copies, and new digital art is being created by artists. Many new techniques are being developed to analyze and classify digital art [87].

Various algorithms have been developed over the years that can use visual features of an image to retrieve and classify an image. Both supervised and unsupervised training from the extracted data of available images has been used for image classification [57]. Stricker and Orengo introduced a Colour moment system that uses the HSV model to find similarities between images using three features [88], mean colour value, standard deviation of colour distribution and asymmetry of colour distribution. Smith and Chang proposed a colour set, which can be as effective as colour histogram for RGB and HSV images [86].

Another major image classification is artist identification. Johnson *et al.* [23] used brush stroke characteristics of paintings on a set of grayscale scans of Van Gogh and Kroller Muller museum paintings to identify and classify the artist [55]. Edge detection was used to see the brush stroke pattern by each artist.

There are also various commercial image retrieval systems that use different visual and statistical image features to retrieve or classify them. Netra by UCSB [65], content based image retrieval system Virage [19], neural networks based system Retrievalware [28], are to name a few. These system takes various features including colour, shape, texture, colour brightness to classify images [67, 80].

Lombardi *et al.* [64] designed an image retrieval system to classify images in a small light-weight set of image characteristics. They used two sets of features for RGB and HSV models. The RGB model consists of 16 features that can be classified in two groups: palette scope (total number of unique RGB triplets) and canvas (maximum, minimum, mean, median and standard deviation of pixel values of red, green and blue channel) (see Table 2.2). The HSV model has 18 canvas features. This includes the maximum, minimum, mean, median and standard deviation value of of hue, saturation and value channel as well as the mean intensity, colour entropy and line count by Sobel edge detector (see Table 2.3). A graphical user interface is also provided for the system that contains two different windows for comparison and classification. It also allows users to modify feature values to see similarities or differences between images. It has an overall accuracy rating of 56%. We have used a slightly modified version of RGB feature set for our experiments.

3.3 Evolutionary Art

Evolutionary algorithms have been used to create art. Dawkins was the first to evolve 2D images of "creatures" with an interactive GA [34]. Karl Sims introduced the concept of texture and 3D structure generation using evolutionary computation [85]. He used genetic algorithms (GAs) to produce different textures and complex 3D models. He proved that evolutionary algorithms (EAs) can be used to successfully produce interesting arts.

Gentropy [91] is another GP based system that can successfully generate 2D textures . It uses a combination of mathematical functions and image features to generate procedural texture images. The user feeds one or multiple target images to the system, and an image feature set along with a function set for procedural texture. The GP system evolves a final procedural texture at the end of the run. The fitness evaluation process includes different image analysis and statistical comparisons. The generated image is not identical to the target image, but has very similar feature to the target image. Weins and Ross [91] used island model evolution for their experiments.

Genshade [52] is another texture evolution system very similar to Gentropy, and also evolves images based on the provided target image features. Where Gentropy uses a tree-based GP, Genshade uses an acyclic directed graph for evolving texture. Male (higher illumination scores) and Female (higher chromaticity score) shaders are ordered accordingly and then used for reproduction operation. The texture language for the experiments consists of float X-value and Y-value, representing the current coordinates, an ephemeral constant, luminosity, mean of two arguments, repeating tile patterns, texture effects, conditional function, and perform iterative processing on vectors.

3.3.1 Procedural Textures

Procedural texture generation [12, 15] is an important technique in computer graphics. It helps developers to create different realistic textures for graphics applications. These textures are created using mathematical functions. They are in contrast with bitmap texture, which are fixed in size and based on digital images. One example for procedural texture is given below:

$$R = f_r(x, y) \tag{3.1}$$

$$G = f_g(x, y) \tag{3.2}$$

$$B = f_b(x, y) \tag{3.3}$$

$$RGB = (R, G, B) \tag{3.4}$$

Equations 3.1, 3.2 and 3.3 represent the procedural texture generation functions for red, green and blue channel. Equation 3.4 represents the final function that combines the three channels to produce a RGB image. For grayscale image just one single channel function is sufficient as R, G, and B themselves are grayscale images when examined separately.

Evolutionary Art (Evo Art) is the use of evolutionary algorithms such as GA and GP to create objectives of art. Procedural textures have been a popular topic in Evo Art, as GP is capable of evolving procedural texture expressions [91]. Early systems relied on interactive fitness evolution by the user. Later, automatic fitness was studied. Such systems use image features analysis for fitness evolution [81].

3.3.2 Procedural Textures Using MOP

When it comes to procedural textures, some algorithms are very successful in evolving images with similar features to the target image [52, 91]. There have been multiple examples using different genetic algorithms and genetic programming to evolve images with similar features of a target image. Ross and Zhu [78] used multi-objective optimization in their GP based procedural texture generation research. Using Pareto ranking with diversity, they dealt with the premature convergence problem and generated a diverse population. Their automated texture evolution approach showed promising results for multi-objective optimization. Baniasadi also used GP for non-photorealistic rendering [20], where the proposed system tried to evolve an image that looks like a painting of the target image.

In her research Salimi [81] used GP to produce colour textures. She used the two feature sets (see Table 2.2 and 2.3) proposed by Lombardi *et al.* [64] for RGB and HSV images for her experiments to select image features. She also used a few modified versions of these feature sets. Using only simple mathematical functions and images features from target images, she was able to produce good procedural texture images for a variety of target images. She successfully used sum-of-ranks for many objective optimization (up to 17 objectives in their experiment).

3.4 Challenges with Existing Systems

The new proposed approaches in this thesis aim to use the advantage of MAP-Elites to create diverse high quality solutions for multi-objective and many-objective problems by combining MAP-Elites with the many-objective sum-of-ranks fitness strategy. The new system can also be useful for other many-objective problems where diverse solutions are needed.

Most existing systems for procedural texture generation using vanilla GP generate one result per run. To get a large number of diverse solutions, many runs are needed, which are computationally expensive and time consuming. The MAP-Elites algorithm is fairly new and has not been used in many-objective problems, can help solve the lack of diversity problems with the traditional GP.

Chapter 4

System Design

This chapter describes the system design for this research. The details about system design, system parameters, programming platform and all the external libraries and tools are described. It also includes the GP and MAP-Elites parameters, and the implementation of new versions of MAP-Elites.

4.1 GP System

Python-3.6 [79] has been used for the programming and system implementation. The genetic programming system used is a Python library named DEAP [40], which stands for Distributed Evolutionary Algorithm for Python. It is a very popular genetic programming library that supports easy modification and system integration with other Python libraries. Other important libraries that are used include NumPy [48], PANDAS [68], and OpenCV [25].

Using DEAP [40] and other mentioned libraries, a system has been designed which takes an image as input, and then through evaluation over a number of generation creates a new procedural texture image which has similar feature values as the original input image. Figure 4.1 shows the basic design of the system, which will be described below.

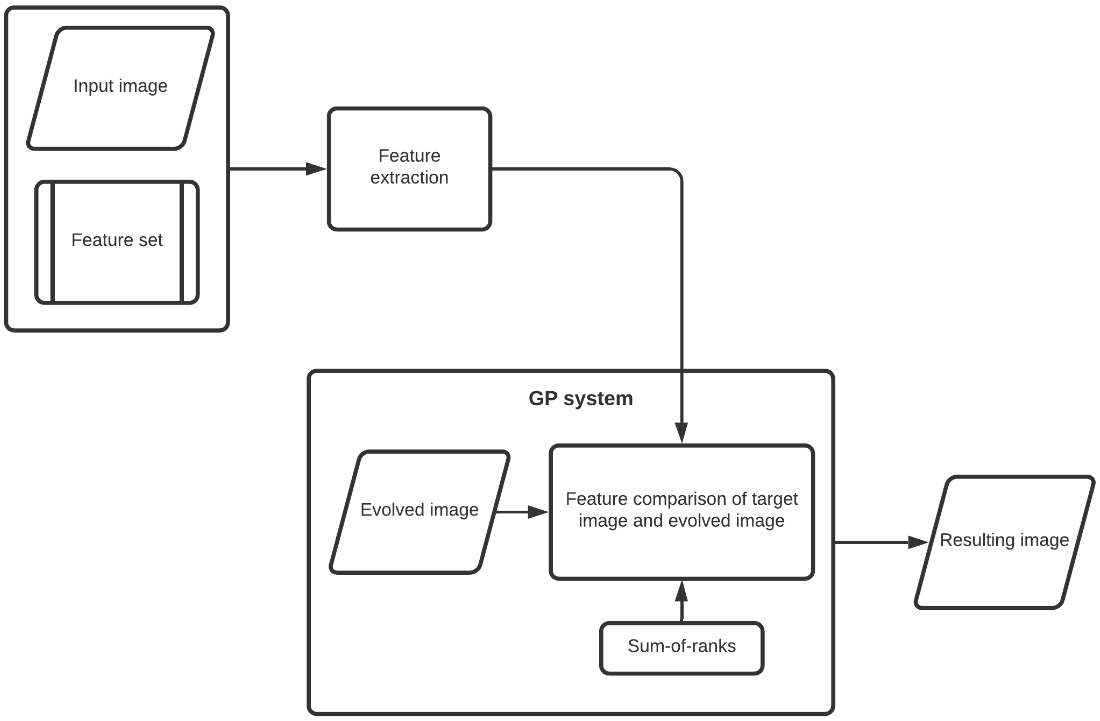


Figure 4.1: The architecture of the system.

First, a target image is selected and provided as input for the system. The GP system is also given the GP parameters, GP function set. The system first extracts and saves the feature values from the target image, which is treated as the base value against which all the feature values for evolved images will be compared. The GP system then creates a random population which is used to evolve images and compare their feature values with the target image feature values. To scale the original and generated image feature values, all the feature values are normalized between 0 and 1. For example, instead of the conventional grayscale values between 0 and 255, for each RGB channel normalized values between 0 and 1 are used.

After comparing the evolved image and target image feature values each individual is assigned a fitness value by the fitness function (see Section 2.3.2). After generating and comparing all population, all individuals are ranked using "normalized sum-of-ranks" (see Chapter 2). These ranks are used for tournament selection while selecting parents to produce next generation of individuals. This process of evaluation, feature comparison and rank assignment continues until the end of the run. The best individual from the final generation is the resulting evolved image which, if evolution was successful, has similar feature values as the original image.

4.1.1 GP System Parameters

Table 4.1: GP system parameters

Name	**Description**
Number of generations	The total number of generations in each run.
Population size	The total number of individuals created in each generation.
Tree initializer	Method for creating a variety of random trees.
Initial min/max tree depth	Sets the minimum/maximum size of initialized random trees.
Crossover probability	Probability of using crossover for reproduction.
Mutation probability	Probability of using mutation for reproduction.
Max tree depth	Maximum allowed size of a GP tree during the run.
Tournament size	Number of individuals selected randomly for tournament selection method.

GP needs some user supplied parameters to operate. These parameters tell GP how to evolve a program, how to create new individual, how many individuals there should be in each generation, or how long it needs to evolve to find a solution. User needs to provide values for the basic GP parameters. Table 4.1 shows the required GP parameters that needs to be provided. More information about these parameters can be found in [76].

4.1.2 GP Language

Table 4.2: Genetic programming function set

Name	**Operation**	**Input/ Output Data type**
add	Arithmetic sum function on two operands. It takes two operands as input and returns a single value.	float
sub	Arithmetic subtraction function on two operands. It takes two operands as input and returns a single value.	float
mul	Arithmetic multiplication function on two operands. It takes two operands as input and returns a single value.	float
protectedDiv	Protected arithmetic division function on two operands. It takes two operands as input and returns a single value. If the divisor operand value is 0, it returns 0, to avoid any error.	float
protectedPow	Protected power function on two operands. It takes a single operand as input and returns a single value. Since the negative square root generates an imaginary number. This function first checks the value of the second operand. If that value is less than 1, then it takes the absolute value of the first operand.	float
protectedLog	Protected natural logarithm function on a single operand. It takes one operand as input and returns a single value. Since the natural logarithm can not be performed on a negative value, it takes the absolute value of the input operand.	float
sin	Trigonometric sine function on a single operand. It takes a single operand as input and returns a single value.	float
cos	Trigonometric cosine function on a single operand. It takes one operand as input and returns a single value.	float

Table 4.3: Genetic programming terminal set

Name	**Description**	**Input/ Output Data type**
x	X-coordinate values	float
y	Y-coordinate values	float
rand1	Ephemeral constant	float

The GP language used in this research is similar to others used in procedural texture generation. The 2D coordinate (X, Y) of an image has been used as a terminal. These co-ordinate values are between 0 and 1 along with the actual pixel values. GP function set includes different mathematical and trigonometric operations. Ephemeral constant is used, which is a randomly generated floating point. A list of basic GP functions is provided in Table 4.2. Only the Mod function from [81] has not been used, as all the values are float and the modulus operation will never be used on float values. Table 4.3 shows the GP terminal set.

4.2 Novel MAP-Elites Algorithm

New MAP-Elites algorithms are designed for this experiment. They are based on the basic MAP-Elites described in Section 2.2. The new algorithms take the basic parameter to create a MAP. Here only a 2D MAP is used for all the experiments. Increasing number of dimension above 2 will increase the MAP size exponentially, which makes the system computationally expensive, resource consuming, and impractical to use. More details are provided in the following sections.

4.2.1 Proposed MAP-Elites Algorithms

Algorithm 3 Proposed MAP-Elites Algorithms.

```
$M \leftarrow MAP\ dimension$;
$N \leftarrow maximum\ number\ of\ iterations$;
$X \leftarrow bin\ size$;
$P \leftarrow initial\ batch\ size$;
$B \leftarrow batch\ size$;
$x \leftarrow a\ random\ elites\ on\ the\ MAP$;
$y \leftarrow current\ occupant\ of\ a\ cell$;
$p(x) \leftarrow fitness\ of\ x$;
$f(y) \leftarrow feature\ description\ of\ x\ (Used\ to\ find\ a\ cell\ on\ the\ MAP)$;

randomly initialize a P number of solutions;
place these solutions on MAP based on their features;

$i \leftarrow 0$
while ($i \leq N$) do
    $j \leftarrow 0$

    while ($j \leq B$) do
        selects individual z from the elites of the MAP;
        using crossover and/or mutation creates a new randomly modified version of z: z';

        checks feature description of z': f(z');
        checks the fitness of x': p(z');

        if (size (f(z')) <X ) then
            place z' in MAP space f(z');
        else
            if (p ( z') >p (y) ) then
                replace y with z';
            else
                discard z';
            end if
        end if
        j++;
    end while
    i++;
end while
return (final MAP individuals)
```

The original MAP-Elites algorithm uses a single-individual bin, meaning it can only hold one individual in each cell of the MAP. In this thesis we propose two new approaches which are very similar (see Algorithm 3). Both approaches have common characteristics, the bin size can be set by the user. Instead of having only one individual in a bin, it is possible to have multiple individuals (N) in a bin. This allows for N number of best solutions for that feature space of the MAP to be stored in a single bin. When N+1 number individual for a bin arrives, its fitness is compared to the occupants of that bin. If the new individual has a better fitness than at least one of the occupants of the bin, it replaces the least fit individual from the bin.

Both approaches use random selection to select a bin for reproduction. But how a individual from a bin is selected varies. The first approach uses random selection to select one of the occupants of the bin for reproduction. The second approach uses a tournament selection of size T (set by the user), where T number of individuals are chosen for a tournament and the winner of that tournament is selected for reproduction. The multiple-individual bin is used in a hope that it will increase the diversity by giving some of the less fit solutions a chance to be a part of reproduction.

4.2.2 MAP-Elites System Parameters

Table 4.4: MAP-Elites system parameters

Name	Description
Number of iteration	The total number of iterations in each run.
Bin feature	Number of feature that is considered during the creation of the MAP grid,
Number of bins	The dimension of the Grid (eg: 2D, 3D). It takes the highest value for each dimension. For example, for 2D grid it will take M and N value, where M is number of bins along X-axis and N is number of bins along Y-axis. Total number of bins is equal to the multiplication of M and N.
Items per bin:	Number of maximum individual that can be stored in a single bin.
Feature domain	Feature domain refers to the range of each feature used during the MAP creation. It takes lower and upper range for each feature and then divides those feature values by number of bins to assign each bin a feature range on the feature axis.
Fitness domain	Minimum and maximum fitness that needed to be considered while evaluating individuals.
Initial batch size	Number of initially created individuals.
Batch size	Number of individuals created in each iteration from the Elites of the MAP.

MAP-Elites [72] needs some user supplied parameters to function (see Table 4.4). These parameters tell the system all the necessary information, including MAP dimension, shape and size of the MAP, the number of bins need to be created, now many items there should be in each bin, and number of features that have to be taken into account while creating the MAP container. The range of each feature value, number of initial individuals, maximum number of iterations and number of new individuals created in each iteration.

4.3 GP and MAP-Elites System Integration

The MAP-Elites algorithm and the genetic programming system have been integrated. In the original MAP-Elites paper [72], a GA was used for the evolution of individuals. Here, we used tree based GP for that purpose, meaning each MAP-Elites individual is a GP tree. Both MAP-Elites parameters and GP parameters are required for the system to work. MAP-Elites creates the initial MAP (2D grid). GP takes the initial batch size parameter and creates specified number of random individuals. Those individuals are placed in appropriate bins in the MAP. Genetic operations are used for reproduction purposes. The number of new individuals created in each iteration is determined by the batch size parameter of MAP-Elites.

4.4 Fitness Function

As mentioned in the earlier sections, the goal is not to create an identical copy of the target image, but to create a new image that has similar feature characteristics as the target. For that purpose, an automated fitness function has been created to evaluate the fitness of each individual. For this process, the difference between each pair of feature of the original image and each evolved image is calculated. Using normalized sum-of-ranks (see Chapter-2) each of the differences is then assigned a rank via sum-of-ranks (see Section 2.3.2). These ranks are used while selecting individuals for reproduction. The feature difference values for each feature for every generation (GP) and (every iteration for MAP-Elites) is stored for plotting and result analysis purpose.

4.4.1 Sum-of-ranks and MAP-Elites

In GP, the implementation is very simple as it ranks the whole population at once and then assign them ranks. But, in the case of our three MAP-Elites algorithm it becomes quite tricky. In each iteration when N-number of new solution is created, they are each given a relative ranks compared to the individual(s) in the bin with same feature space. and once the new individuals are placed in their respective bins (if the bin is empty or they are better than the existing N-number of allowed solution(s) in those bins) or discarded (if not better than the existing N-number of allowed solution(s) in those bins), the whole MAP-is re-ranked again. Another thing to remember that in single individual MAP-Elites only the best of the bins are stored, so ranking them takes relatively smaller time compared to multi-individual bins where all the individuals in the MAP are also ranked, but, there can be N-times more individuals to rank.

A preliminary system testing was performed to see if the MAP-Elites algorithm and the new proposed versions of it are working properly. Details about the system testing is available in Appendix A (Additional Analysis).

Chapter 5

Experiments and Results: Grayscale Image

Our newly designed algorithms are tested and their performances are compared with base GP. These experiments can be divided into two major groups: Grayscale image experiments and RGB image experiments. In both cases, a target image is provided to the system. Evolution tries to generate a procedural texture image which has similar feature characteristics as this target image. The system tries to minimize the feature difference between target image and newly created images. Four different systems are compared: Base GP, single individual bin MAP-Elites (symbol: MAP_Base), multiple individual bin MAP-Elites with random selection (symbol: MAP_B5_R), and multiple individual bin MAP-Elites with tournament selection (symbol: MAP_B5_T2). This chapter only deals with the experiments for grayscale images, and the following Chapter 6 will present experiments with RGB images.

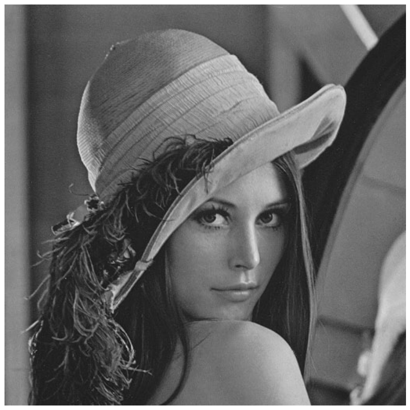
Three different experiments are performed with two different target images and 3 different feature sets. Figure 5.1 shows the two target images used for the grayscale images. The first target image (Lena) has been used for a lot of computer-vision and image processing research. Figure 5.1 (b) is from the personal repository of the author. Target image 1 has been used for experiments 1 and 2, and target image 2 is used for experiment 3. Their feature values are shown in Table 5.3. We can see from the table that the image size and their features are quite different from each other. The second target image has been chosen because of its colour distribution, to make the features more diverse. Table 5.4, 5.5 and 5.6 show the 3 feature sets used in the experiments.

Table 5.1: GP parameter for grayscale experiment

Parameter Name	**Value**
Population Size	300
No. of Generation	50
Tree Initializer	Half and Half
Initial Tree Depth	Min: 2 Max: 7
Max Tree Depth	17
Crossover	100%
Mutation	20%
Selection	Tournament
Tournament Size	3
Number of Runs	20 / 10 / 10

Table 5.2: MAP-Elites parameter for grayscale experiment

Parameter Name	**Value**
MAP size	8×8
Initial batch size	30
Batch size	15
No. of Iteration	1000
No. of features	2
Bin size	1 / 5
Evaluating algo.	GP
Tree Initializer	Half and Half
Initial Tree Depth	Min: 2 Max: 7
Max Tree Depth	17
Crossover	100%
Mutation	20%
Selection	Random / Tournament
Tournament Size	2
Number of Runs	20 / 10 / 10


(a) Target image-1 (Lena)

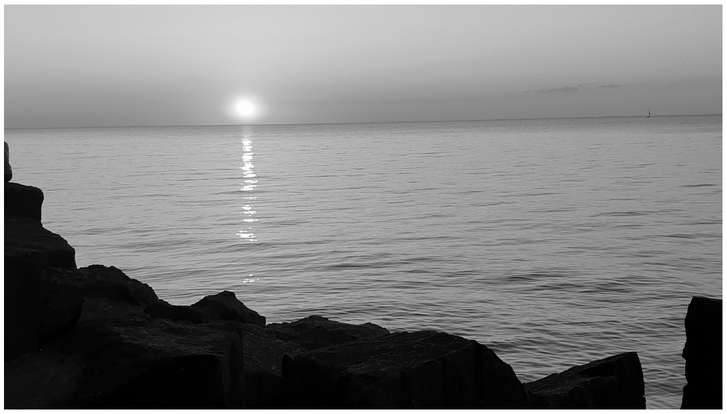

(b) Target image-2 (“Setting Sun” ©*Sheikh Faishal Basher*, 2020.)

Figure 5.1: Grayscale target images

Table 5.3: Grayscale Image Features

Feature Name	**Target Image -1 (Lena)**	**Target Image -2 (Setting Sun)**
Size	512×512	1024×576
Gray Max	0.945	1.000
Gray Min	0.039	0.000
Gray Mean	0.375	0.491
Gray Median	0.380	0.592
Gray Std. Dev.	0.172	0.269
Edge	0.006	0.004
Entropy	0.904	0.894
Unique Gray	0.871	1.000

Table 5.4: Grayscale feature set - 1

Feature Name	**Type**	**Description**
Gray Max	Canvas	The maximum grayscale value in the image.
Gray Min	Canvas	The minimum grayscale value in the image.
Gray Mean	Canvas	The arithmetic mean of the grayscale values in the image.
Gray Median	Canvas	The median of the values in the grayscale image.
Gray Std. Dev.	Canvas	The standard deviation of the values in the grayscale image.

Table 5.5: Grayscale feature set - 2

Feature Name	**Type**	**Description**
Gray Max	Canvas	The maximum grayscale value in the image.
Gray Min	Canvas	The minimum grayscale value in the image.
Entropy	Canvas	The entropy value of the grayscale image.
Gray Median	Canvas	The median of the values in the grayscale image.
Unique Gray	Pallet	The standard deviation of the values in the grayscale image.

Table 5.6: Grayscale feature set - 3

Feature Name	**Type**	**Description**
Gray Max	Canvas	The maximum grayscale value in the image.
Gray Min	Canvas	The minimum grayscale value in the image.
Gray Mean	Canvas	The arithmetic mean of the grayscale values in the image.
Gray Median	Canvas	The median of the values in the grayscale image.
Gray Std. Dev.	Canvas	The standard deviation of the values in the grayscale image.
Edge	Canvas	Number of Sobel edges in the image with 0.2 threshold.

5.1 Experiment 1

Table 5.4 shows the image features selected as objective for this experiment. For this experiment, 20 runs are performed with the same parameters and same target image (Target image 1), producing 20 different solutions. Different versions of the MAP-Elites algorithm generated hundreds of solutions for each run. Each solution is a procedural texture image.

5.1.1 Basic Genetic Programming (GP)

The base GP system parameters are in Table 5.1. Using target image 1 as the system target, GP generated some interesting solutions. Figure 5.2 shows the best solution of the 20 GP runs. Some of the solutions may look similar visually, for example, images that look almost pure black or white may only have a few absolute black or white pixels only, and the rest of the pixel values are very close to the black or white pixel values. But they may look very similar, as there are 256 gray levels between black and white pixel values, and for a human eye it is hard to distinguish between the close pixel values of different pixels.

Figure 5.2 shows the 20 solutions from 20 GP runs over 50 generations. Figure 5.3 shows the average mean fitness of the GP population over 50 generations for 5 different objectives averaged over 20 GP runs. We can see that these values are not always going down or up continuously as a single objective plots usually behave. Instead, the population average starts to converge quickly as is often expected with GP. Similar behaviour can be seen in Figure 5.4, which shows the average best performing individual's fitness plots for 5 objectives over 50 generations and 20 runs. More on this will be discussed in the analysis section of experiment 1 (Section 5.1.5).

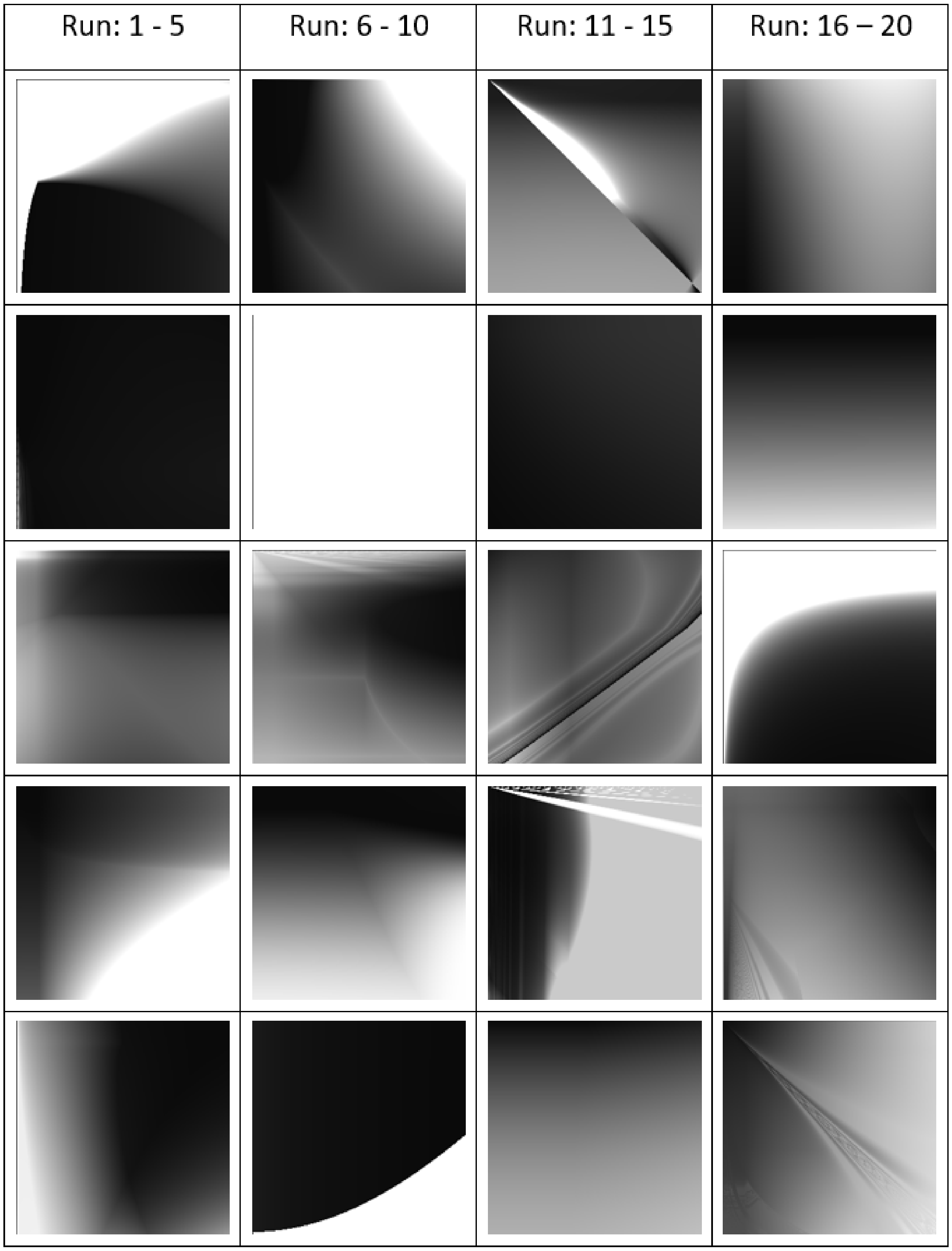


Figure 5.2: Best result of each GP run for target image 1 and grayscale feature set 1 (20 runs).

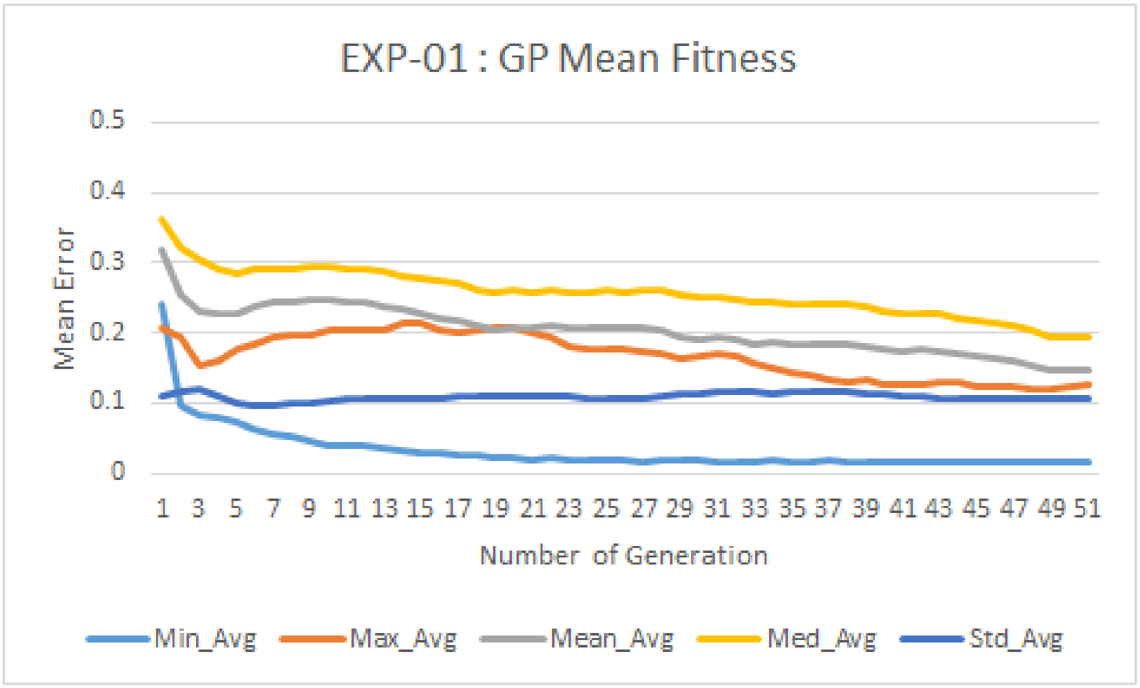


Figure 5.3: Average mean fitness of GP population over 20 runs (lower is better).

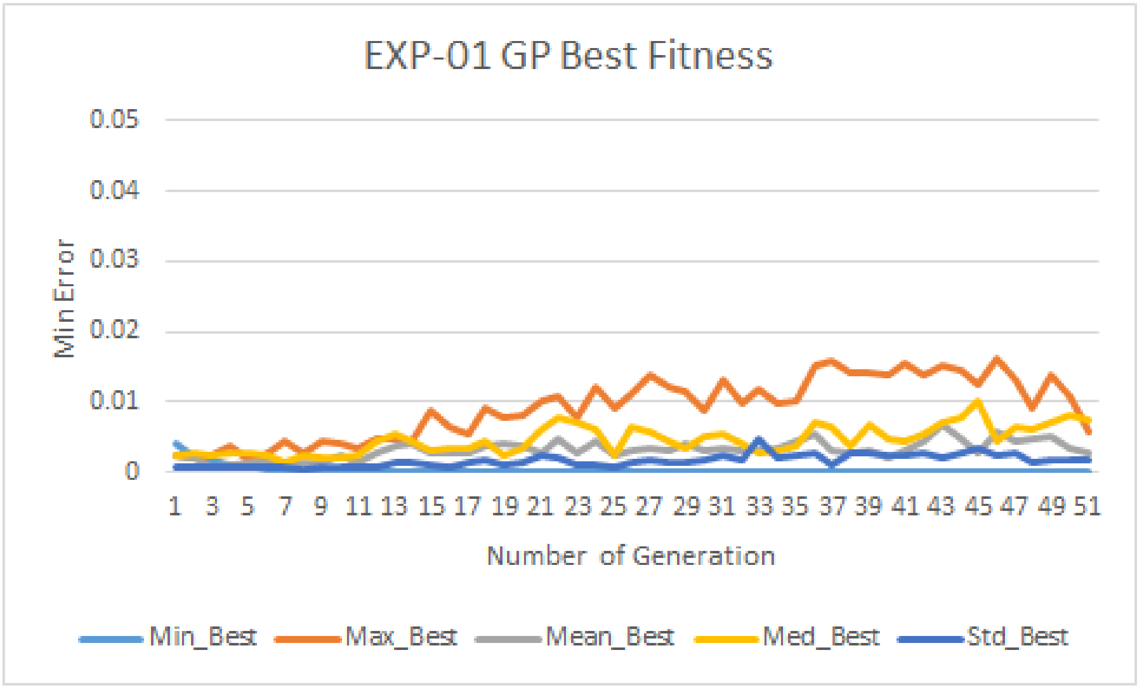


Figure 5.4: Average best fitness of GP population over 20 runs (lower is better).

5.1.2 Base MAP-Elites: Single Individual Bin (MAP_Base)

This experiment uses the standard MAP-Elites algorithm (bin size = 1, and parents are selected randomly from the MAP for reproduction). Other parameters for MAP-Elites are in Table 5.2. The GP parameters are in Table 5.1. The base GP algorithm parameter has a population size of 300, where in all the MAP-Elites experiments for grayscale images the initial batch size (which can be compared to initial population of GP) is set to 30 and batch size is set to 15. The GP system may use all of the newly created individual for the next generation, where MAP-Elites solutions compete with each other for a place on the MAP if more than one newly created solution falls into the same feature set cell.

For this experiment, unique gray values and entropy are used as MAP features. The number of generations in the GP experiment is set to 50, while the number of iterations for the MAP-Elites algorithm is set to 1000. This may be confusing, but the reason to do this is to keep the total number of evaluated individuals equivalent between base GP and MAP-Elites algorithms. GP has a larger population, thus evaluation evolves a very large number of individuals in just 50 generations. MAP-Elites has a small number of new individuals created in each iteration and takes a large number of MAP-Elites iteration to evaluate the same number of individuals as GP.

Figure 5.5 shows the best result of 20 runs. The final MAP produced more than 55 solutions for every single run for this experiment and only the best ranked solution from each run is shown in Figure 5.5. This figure represents the best solution of each MAP from 20 different runs. Each column header represent the run numbers of the solutions. For example, the first column of the figure shows the best solutions from run 1, 2, 3, 4 and 5 respectively.

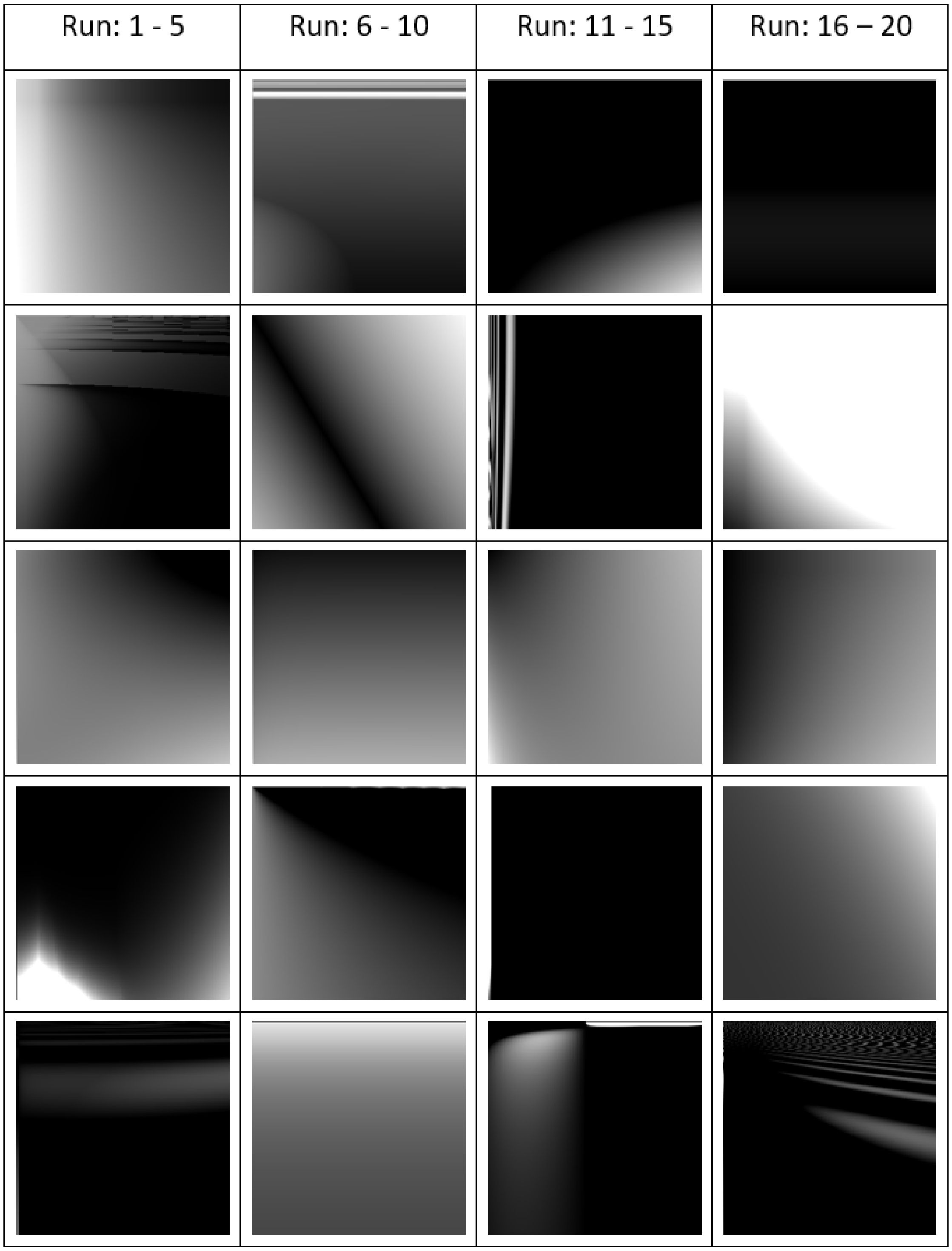


Figure 5.5: Best results of base MAP-Elites runs for target image- 1 and grayscale feature set -1 (20 runs).

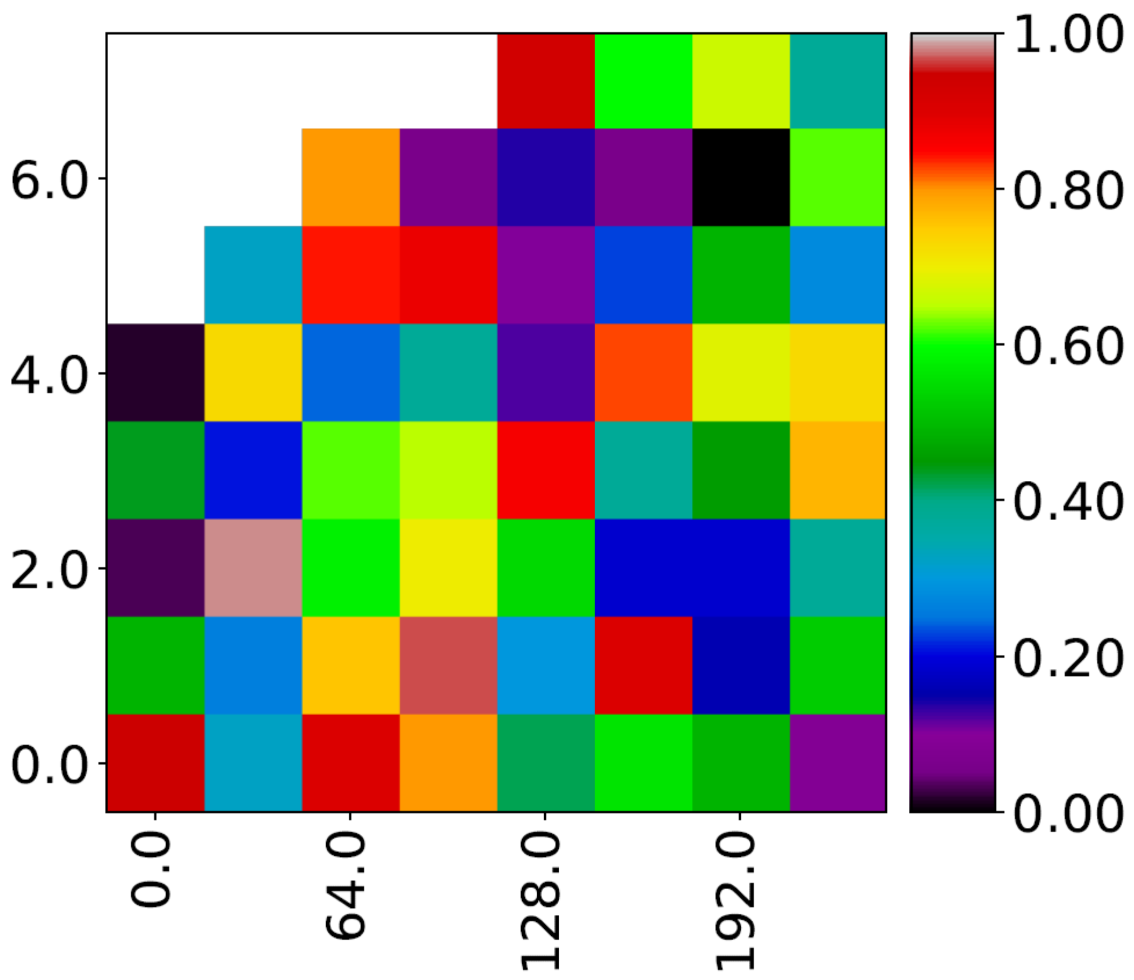


Figure 5.6: Sum-of-ranks MAP of run-19 of experiment-1 (higher is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)].

Figure 5.6 shows the final solution ranking of the MAP (sum-of-rank) for one of the MAP-Elites run (run-19) of base MAP-Elites. The X-axis represents number of unique gray values and the Y-axis represents the entropy values. In the sum-of-ranks MAP, all the ranks are mapped between 0 and 1, and then inversed to make the one with the best ranking value have the value of 1 and the worst ranked individual have a value of 0. The reason for mapping them between 0 and 1 is to normalize all the MAPs of the experiments between 0 and 1. Without this normalization some maps may have a range of 1-64 and some may have 1-5 or even lower based on solutions. For the objectives, our goal is to minimize the difference between feature values of the target image and the evolved images.

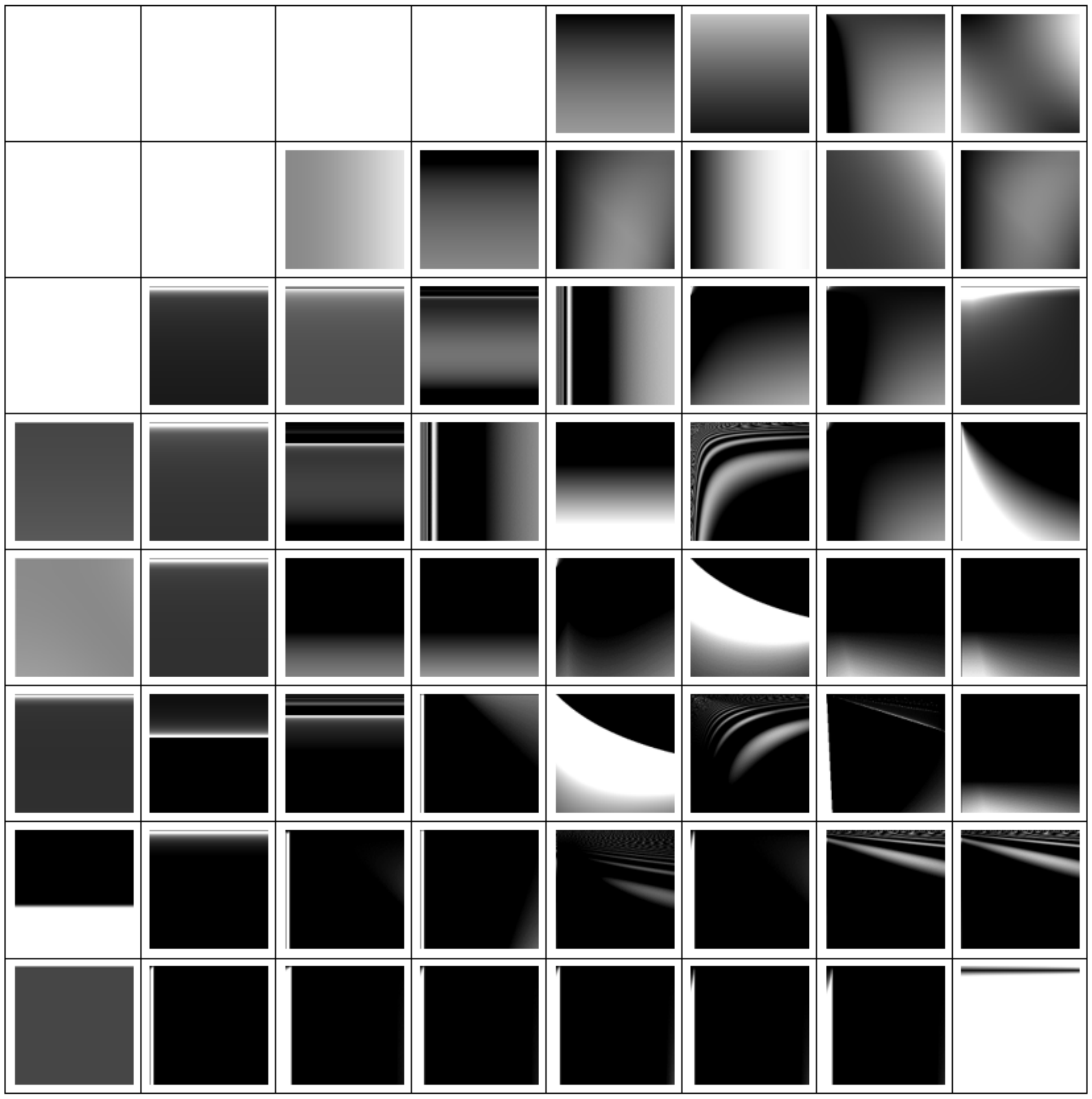


Figure 5.7: Final Solution MAP of run-19 of experiment-1 [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)].

We have also mapped the values of grayscale images between 0 and 1, where 0 represents the grayscale value 0 (black) and 1 represents grayscale value 255 (white). The rest of the gray levels are mapped accordingly. This gives us more precise differences between our target and new image. During unique grayscale value calculations and entropy calculations, we used unsigned integer values between 0 and 255, for grayscale image as there are 256 unique grayscale values. Using a float in this case can result in millions of unique grayscale values, which is not practical.

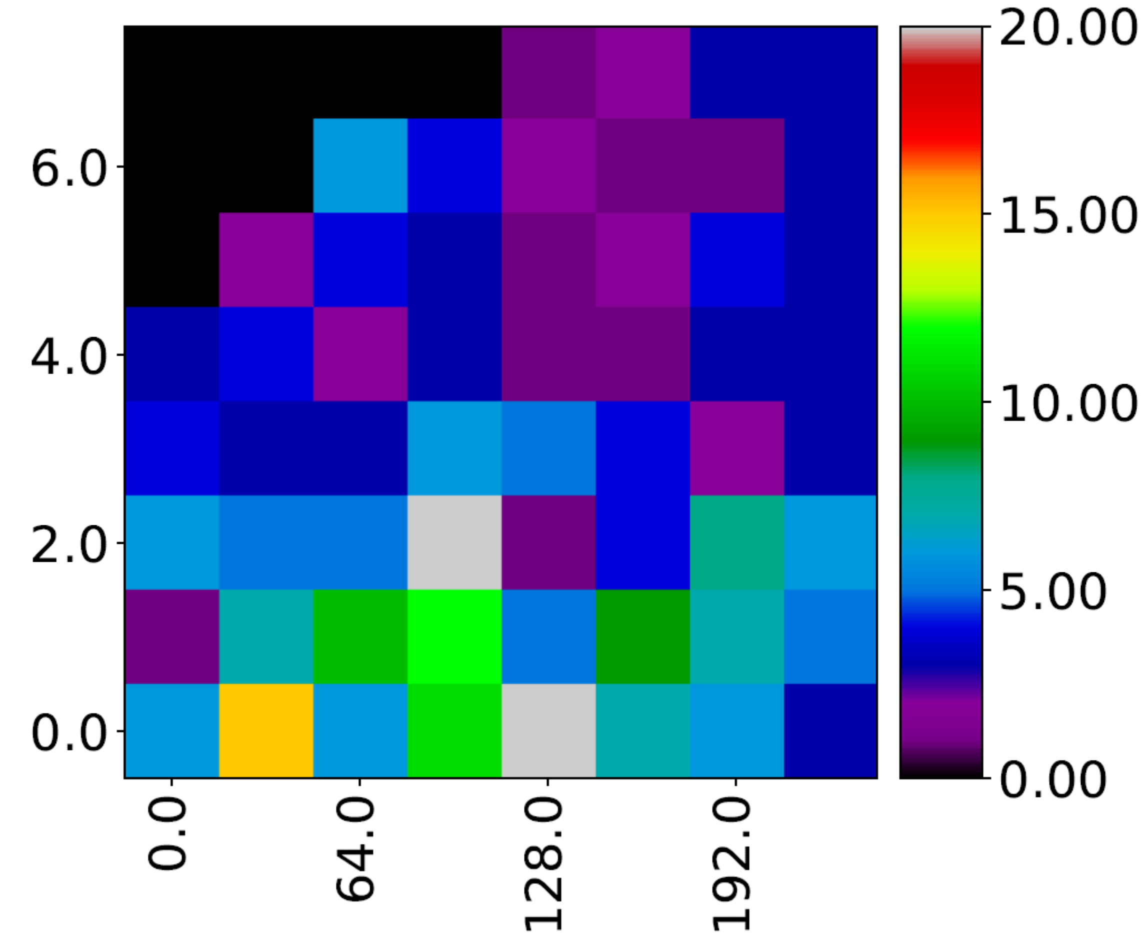


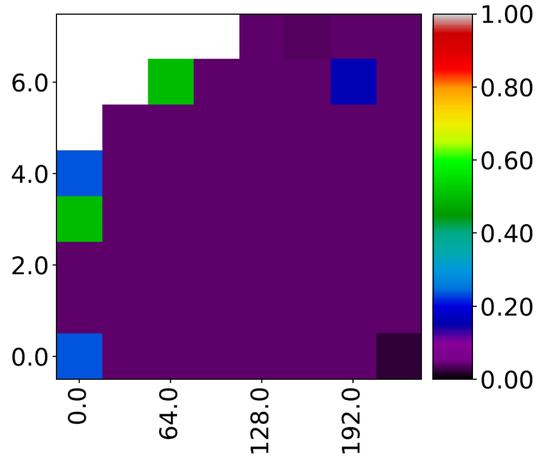
Figure 5.8: Activity MAP of run-19 of experiment-1 (higher is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)].

Figure 5.7 shows the final solution in each bins of a MAP-Elites run (run-19) of base MAP-Elites. Here the X-axis represents number of unique gray values and the Y-axis represents the entropy values. Figure 5.6 can be compared with Figure 5.7, as the represents the same run. The white bins in sum-of-ranks MAP and this figure both mean that there were no solutions found for those bins (those bins never got filled).

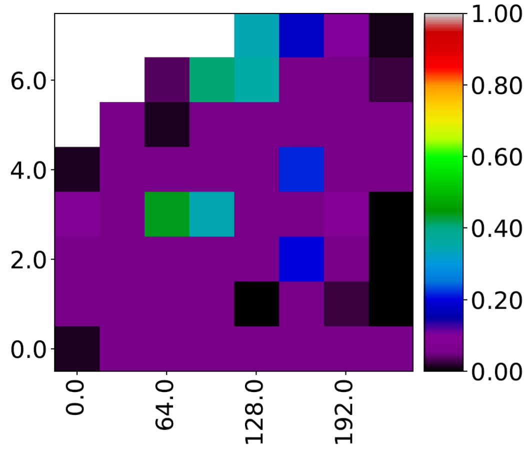
Figure 5.8 shows the activity grid of the MAP. These values represents how many times a bin has been updated. Higher value of bins mean the individuals in those bins have been replaced more times than the bins with lower values. Bins with 0 values mean they have never been used. In these experiments we used five image features as objectives (see Table 5.4). The closer the feature values of the new image is to the original image, the lower the differences of those objectives. By looking at the colour bar next to the heat MAP for activity grid, we can see how many times each bin has been updated. Even though there were 1000 iterations with each creating 15 new individuals, we can see the total number of times bins are updated is significantly lower.

It is easier for a newly created individual to be placed in a bin if that bin is empty. But for the next individual with similar feature set, it becomes a bit difficult as now the new solution needs to have a better fitness than the existing solution in that bin. Over time, replacing the occupying individuals becomes harder as the occupying individual may have achieved a very good fitness which is hard to beat. The black bins are empty bins, meaning there were no solutions found for those feature values. This is understandable, because an image with very low unique gray level values is not very likely to have a very high entropy.

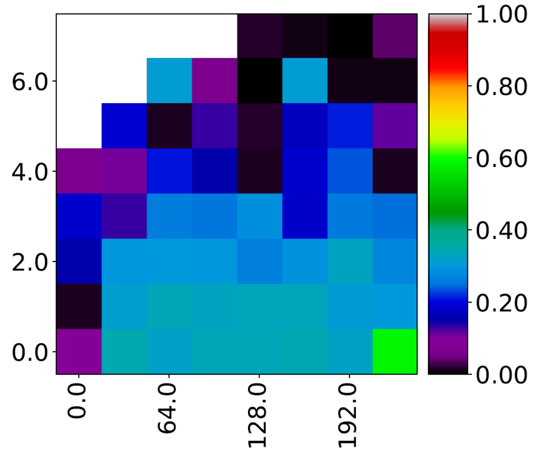
In Figure 5.9 the raw differences of between five objectives (Min, Max, Mean, Median, Standard deviation) values are shown respectively for run-19 of experiment-1 of the base MAP-Elites experiments. We can see the objective differences for different solutions reach zero or very close zero, but the overall ranking may show other solutions with better ranking. This is because, normalizing the scores increases difference between some objectives.



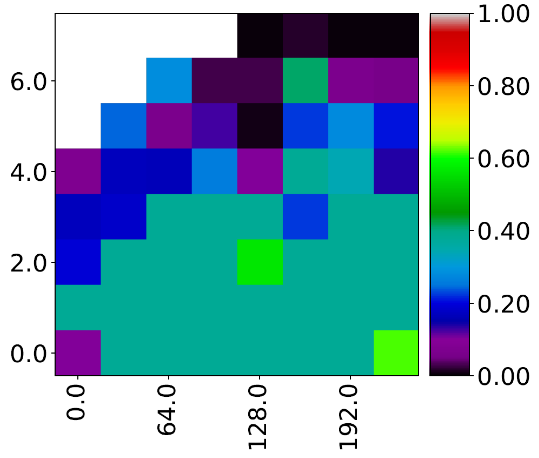
(a) Min difference



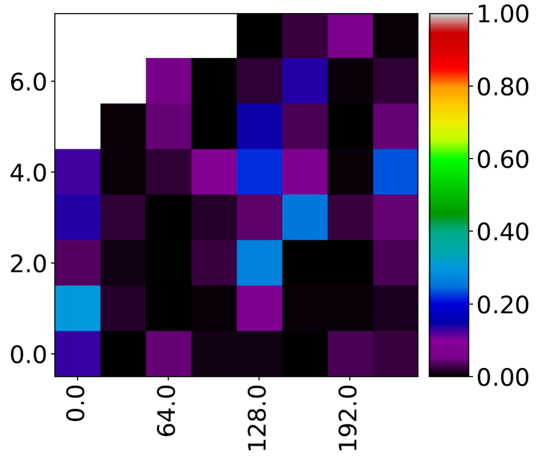
(b) Max difference



(c) Mean difference



(d) Median difference



(e) Std. Dev. difference

Figure 5.9: Raw feature difference of run-19 of experiment-1 (lower is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)]

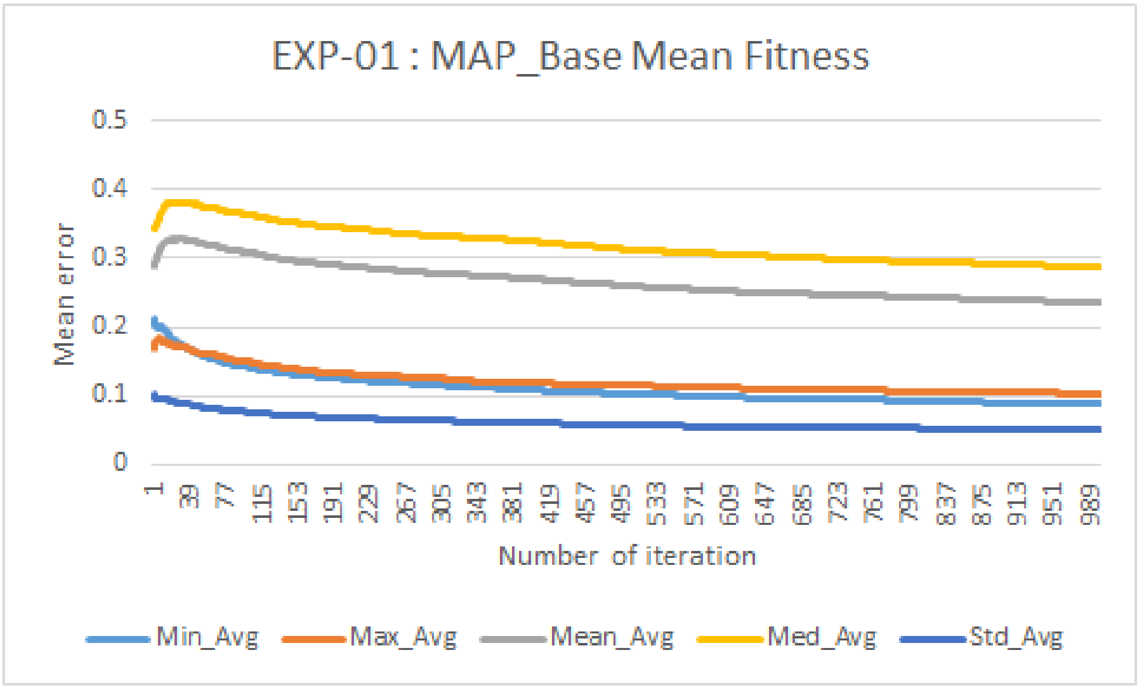


Figure 5.10: Average mean fitness of base MAP-Elites MAP over 20 runs (lower is better).

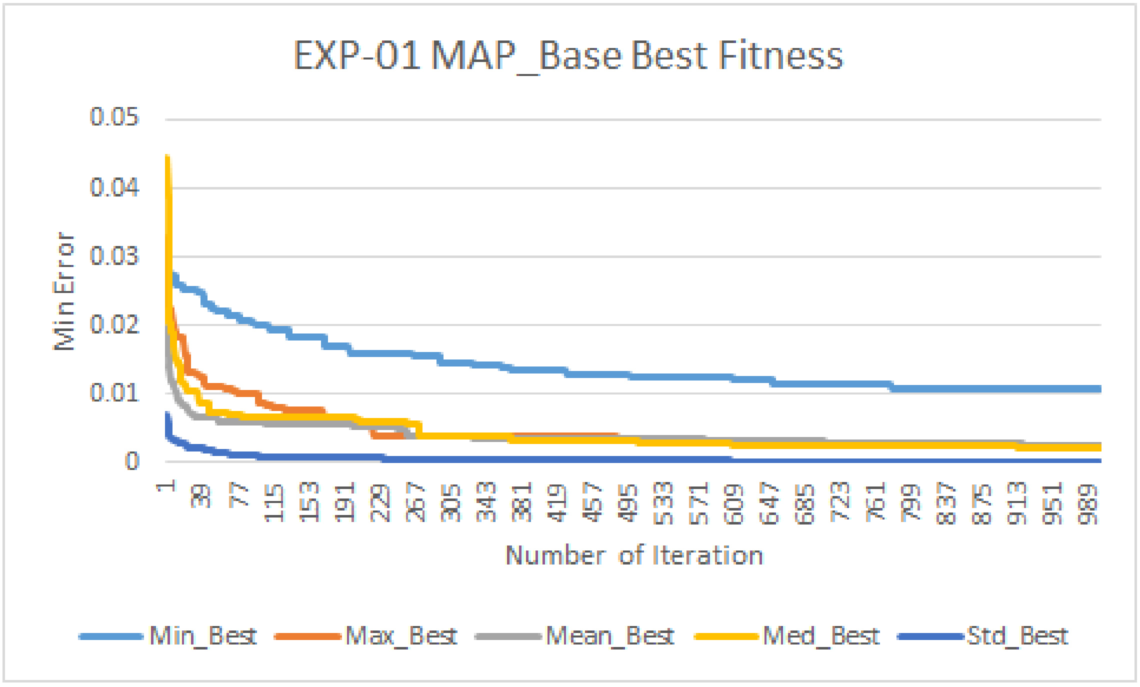


Figure 5.11: Average best fitness of base MAP-Elites MAP over 20 runs (lower is better).

Figure 5.10 shows the mean fitness of base MAP-Elites MAP for all five objectives over 1000 iterations averaged over 20 runs. If we compare this figure to Figure 5.3, we can see some clear differences. The GP fitness (mean error) average of the population improves rapidly for the first few generations and then the solution starts to converge slowly. But in the case of base MAP-Elites, the average fitness (mean error) deteriorates for first few iterations and then starts to improve over time. This is caused by how the MAP-Elites works. GP starts with a large population and the population size remains the same throughout the run. On the other hand, MAP-Elites starts with a few randomly generated solutions on the MAP, which is a very small portion of the MAP. Over time the number of solutions increase.

For first few iterations, the newly generated solutions do not necessarily become better than existing MAP solutions. When they fall into unoccupied bins of the MAP based on their behaviour they do not compete for that bin. That is why the average objective value get worse for first few iterations. But later, when most bins of the MAP are occupied, the new and better solutions replace previous solutions that have been occupying bins with their corresponding feature space in the MAP. This behaviour can more clearly be observed from Figure 5.11, which shows the plots of the objectives of average best solutions over 1000 iterations and 20 runs.

Here, we can see that the solution does not get worse for first few iteration (as was the case for average), but steeply get better. This is because if the new solutions do not have better fitness than the existing best solution of the MAP, they do not replace the current best, so the objective values remains unchanged. And the objective values only change when a new solution in the MAP has better fitness than the current best of the MAP. The value of mean and median feature of the image improve at a faster rate compared with other objectives for the best individuals. This behaviour is quite different from the traditional GP evolution.

5.1.3 MAP-Elites: Multiple individuals Bin - Random Selection (MAP_B5_R)

Here, all the parameters are kept the same as Section 5.1.2 except the bin size, which is set to 5. Thus each bin can store up to 5 best solutions for that feature set. If a bin has no solution or less than 5 solutions, a newly created individual that will be added to that bin without competition. But if there are already 5 solutions, meaning the bin is full, the new solution is compared with the existing solution in terms of fitness. If the new individual has a better fitness than the least fit individual in the bin, it replaces that least fit one. But, if the new individual is the least fit individual compared with the existing individual then it is discarded. For reproduction, individuals are selected at random.

Figure 5.12 shows the best results of 20 runs. We can see the fitness plots of this strategy in Figure 5.14. It shows the average mean fitness of (proposed algorithm-01) MAP-Elites (multiple individual bin - random selection) for all five objectives over 1000 iterations for 20 runs. We can see the graph shows a different pattern here compared with GP or base MAP-Elites. As each bin can hold 5 solutions, some of the bins may have one very good solution, but also some mediocre solutions which contribute to a lower fitness mean and median.

In the base version only the best solution per bin is stored and used for all the statistical calculations over iterations. This explains the increasing average mean and median values over time. This is not ideal to compare the overall solution quality improvements with GP or base MAP-Elites. We can get a more accurate understanding of solution quality improvement through the average best individual plot, which shows a similar behaviour as the base MAP-Elites (see Figure 5.3). In Figure 5.15, we can see a similar behaviour as the base MAP-Elites for the best fitness plots with a slight difference that the majority of improvements for best solution of the MAP happen in first few iterations, and the fitness values are significantly higher (smaller mean error) in the first iterations. As each bin can have five solutions stored in them, it has significantly more solutions compared with the base MAP-Elites. The chances of finding a better solution at earlier iterations is higher compared with base MAP-Elites.

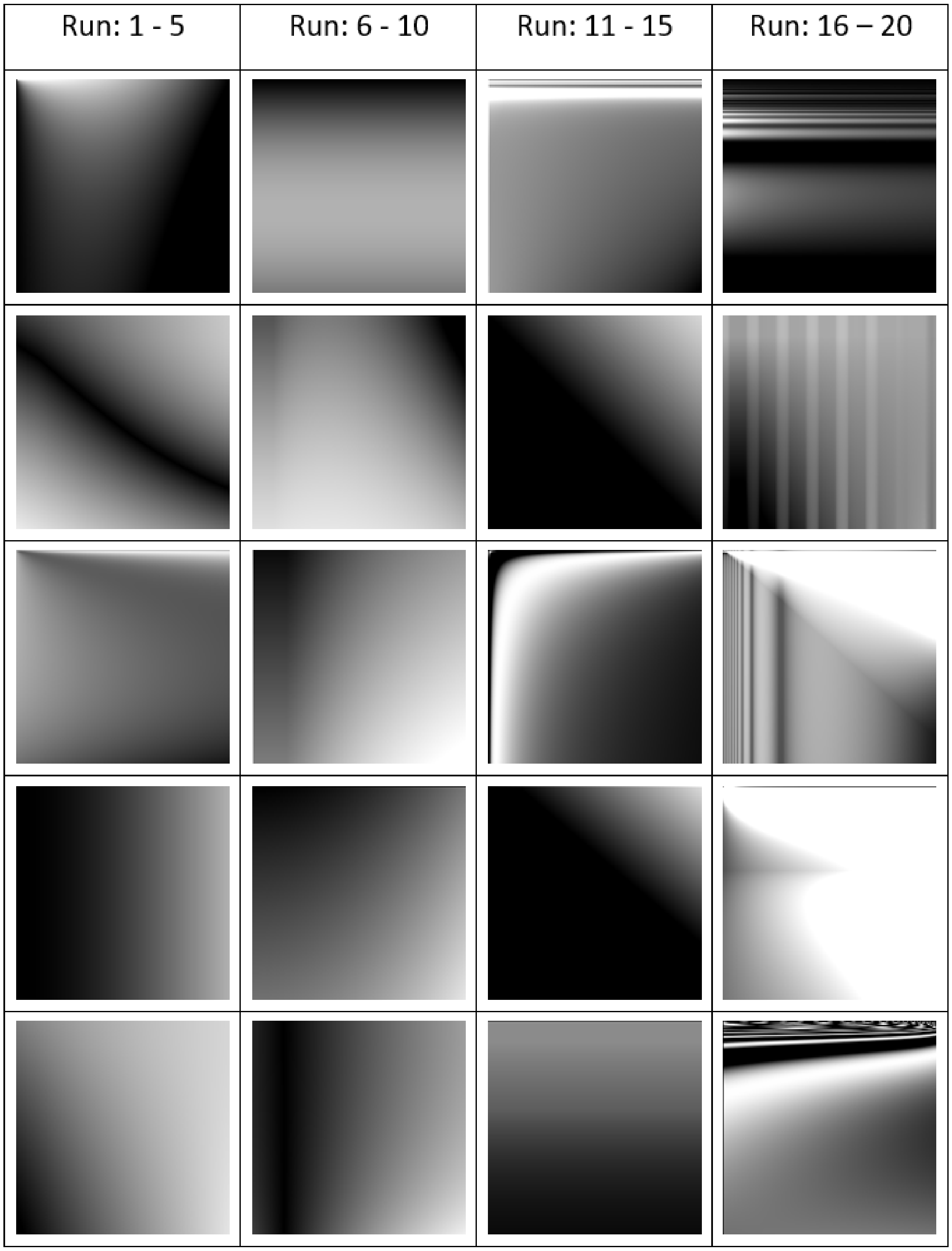


Figure 5.12: Best results of MAP-Elites (multiple individuals bin - random selection) runs for target image- 1 and grayscale feature set -1 (20 runs).

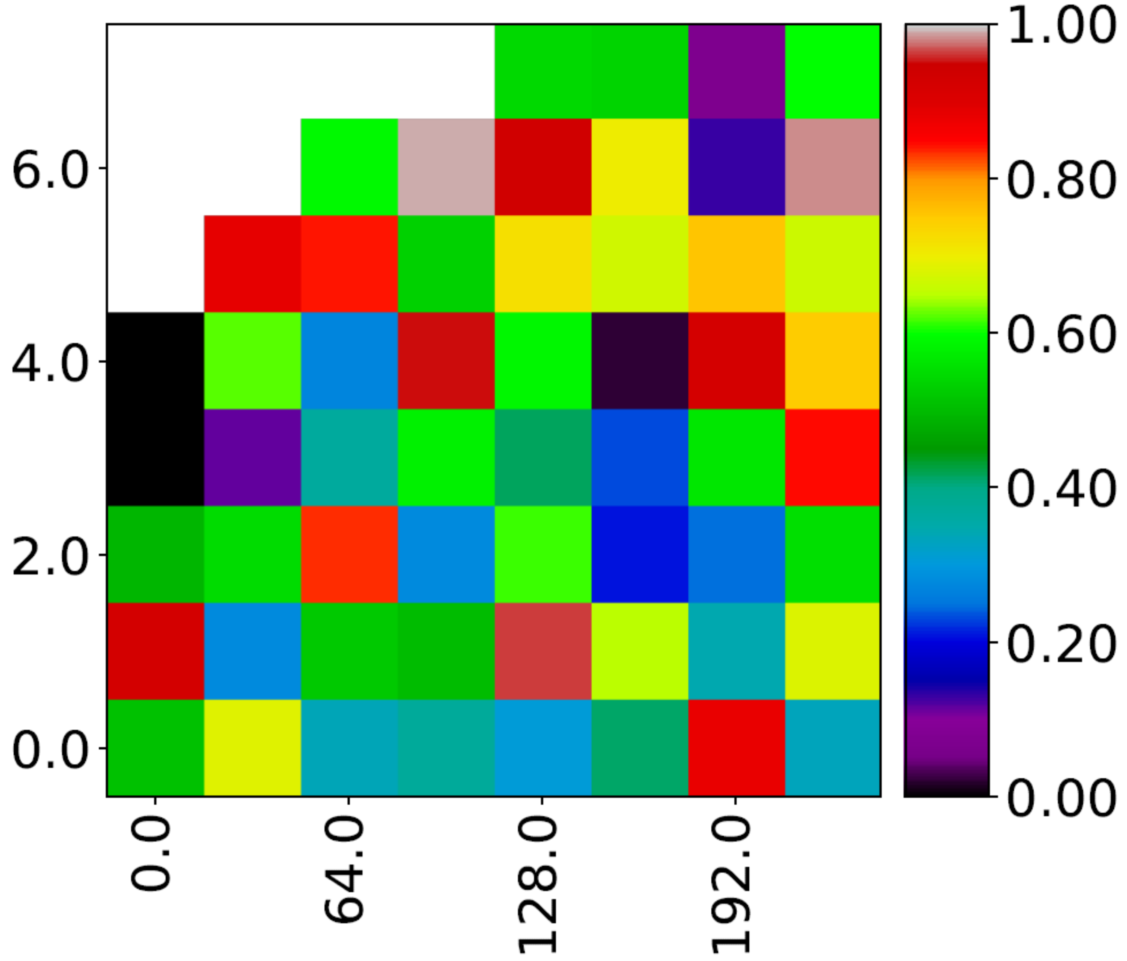


Figure 5.13: Sum-of-ranks MAP of run-2 of experiment-1 (higher is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)].

Figure 5.13 shows the final solution ranking of the MAP (sum-of-rank) for one of the MAP-Elites run (run-2) of MAP_B5_R. The X-axis represents number of unique gray values (range : 0 - 255) and the Y-axis represents the entropy values (range: 0 - 8). When we compare it with Figure 5.6, which represents the same type of MAP for base MAP-Elites. We can see the number of filled bins in the map remains the same despite the bin size. The number of bins filled of a MAP depends mainly on the type of experiment and the feature space of the MAP; here it is unique gray and entropy for images. The bin size does not impact how many bins of the map are going to get filled.

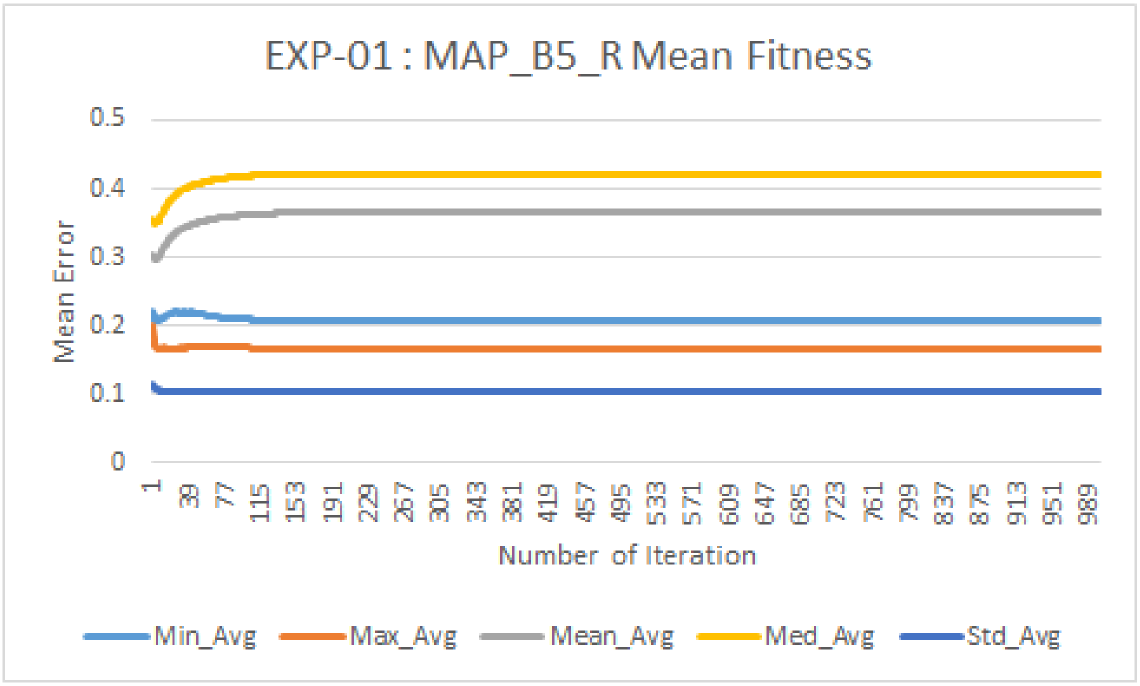


Figure 5.14: Average mean fitness of MAP-Elites (multiple individuals bin - random selection) MAP over 20 runs (lower is better).

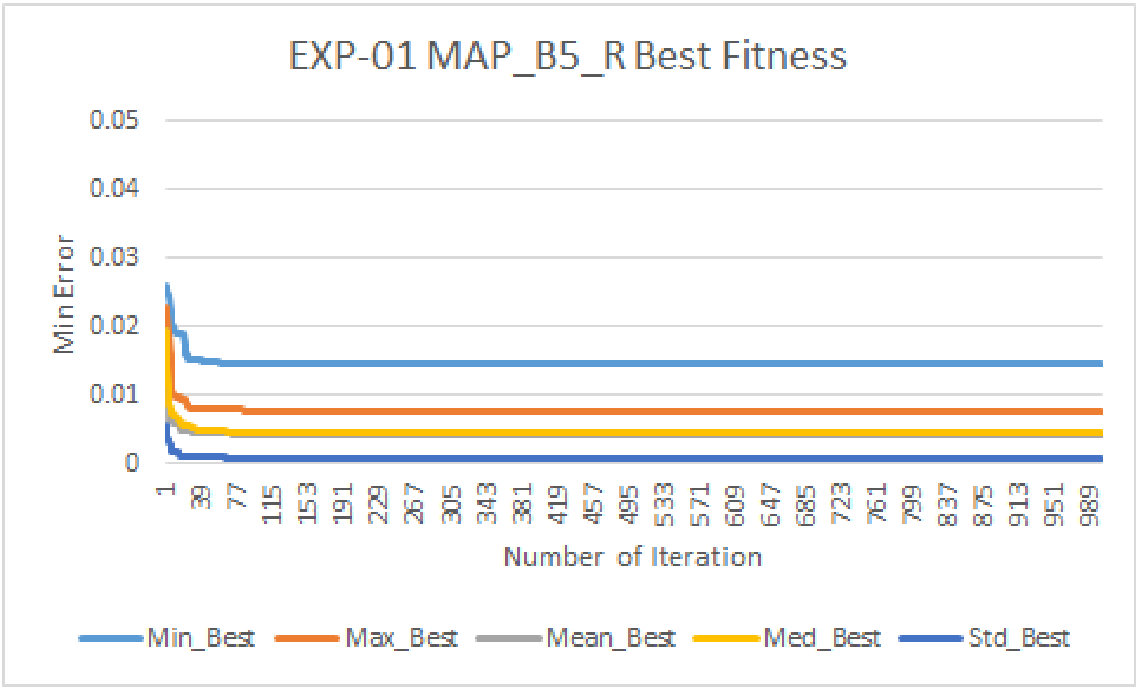


Figure 5.15: Average best fitness of MAP-Elites (multiple individuals bin - random selection) MAP over 20 runs (lower is better).

5.1.4 MAP-Elites: Multiple individuals Bin - Tournament Selection (MAP_B5_T2)

This experiment has similar parameters with Section 5.1.3. The one parameter that is changed is the selection strategy. Instead of selecting individuals at random for reproduction, a tournament selection is used with a tournament size of two. The reason for having such a small bin size. As discussed before, the initial number of individuals in the MAP is set to 30 and bin size to 5. Also, the MAP size is is 8×8, which means each bin may not have five individuals. For reproduction, a bin is selected at random, and then a tournament of 2 is used to select an individual from that bin. Having a large tournament size will reduce the diversity of new individuals. Keeping it a small size does not obstruct the diversity, while allowing for more fit solutions to be created.

Figure 5.16 shows the best results from 20 runs and Figure 5.17 shows the average mean fitness of (proposed algorithm-01) MAP-Elites (multiple individual bin - tournament selection) MAP for all five objectives over 1000 iterations for 20 runs. It shows a similar behaviour as the proposed algorithm-01 (multiple individual bin - random selection). The overall fitness have improved compared with MAP-Elites multiple individual bin - random selection algorithm, as we see lower mean error while using tournament selection instead of the random selection method. Just by changing random selection to tournament selection while keeping all the other parameters same, we allow MAP-Elites to use one advantage of GP, where the probability of selecting a parent with higher fitness for reproduction increases drastically and the impact can be seen in the fitness plots (see Figure 5.14 and Figure 5.17). If we look at the best fitness plotting in Figure 5.18, we see a similar impact in terms of lower mean error compared with the random selection multiple individual bin algorithm.

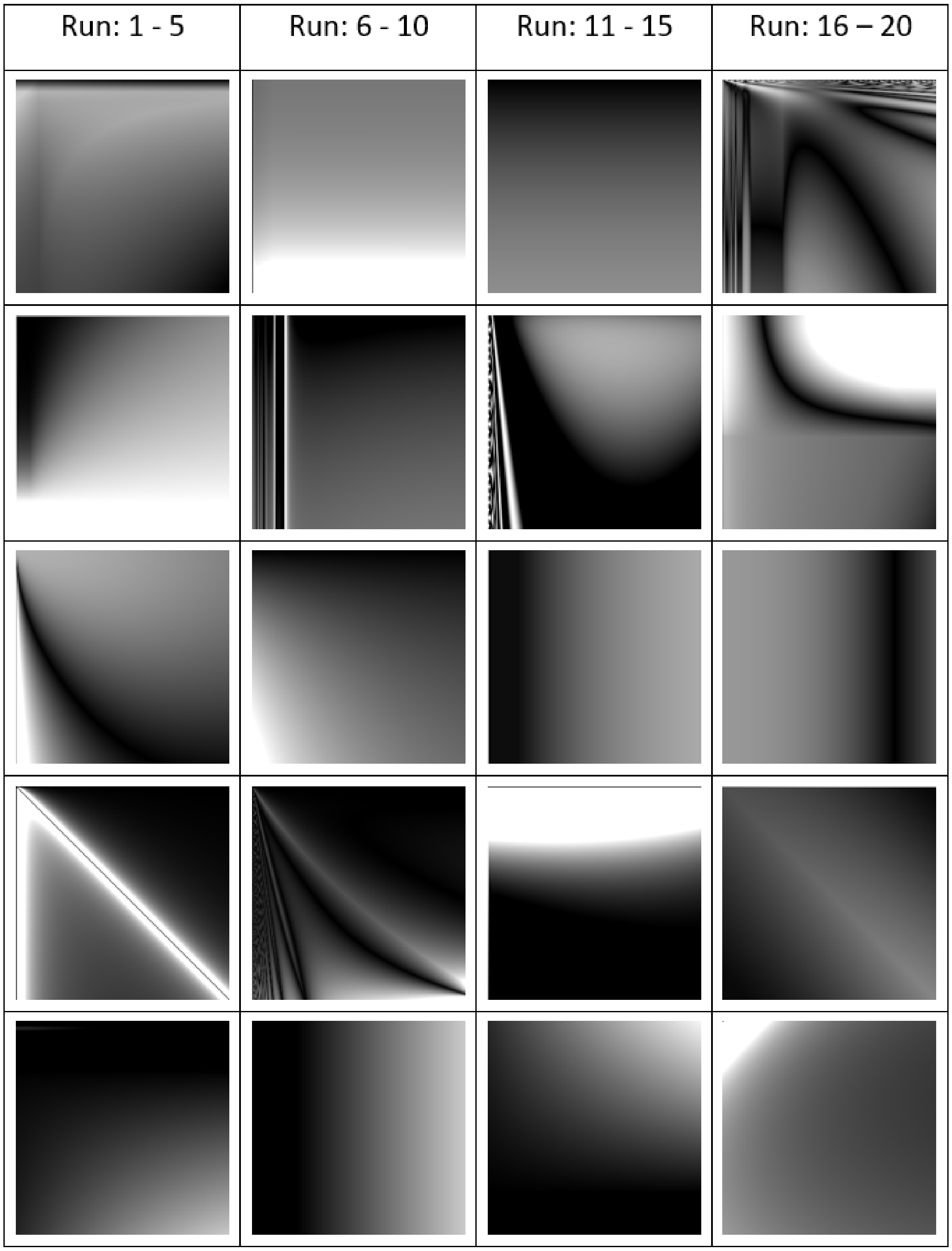


Figure 5.16: Best results of MAP-Elites (multiple individuals bin - tournament selection) runs for target image- 1 and grayscale feature set -1 (20 runs).

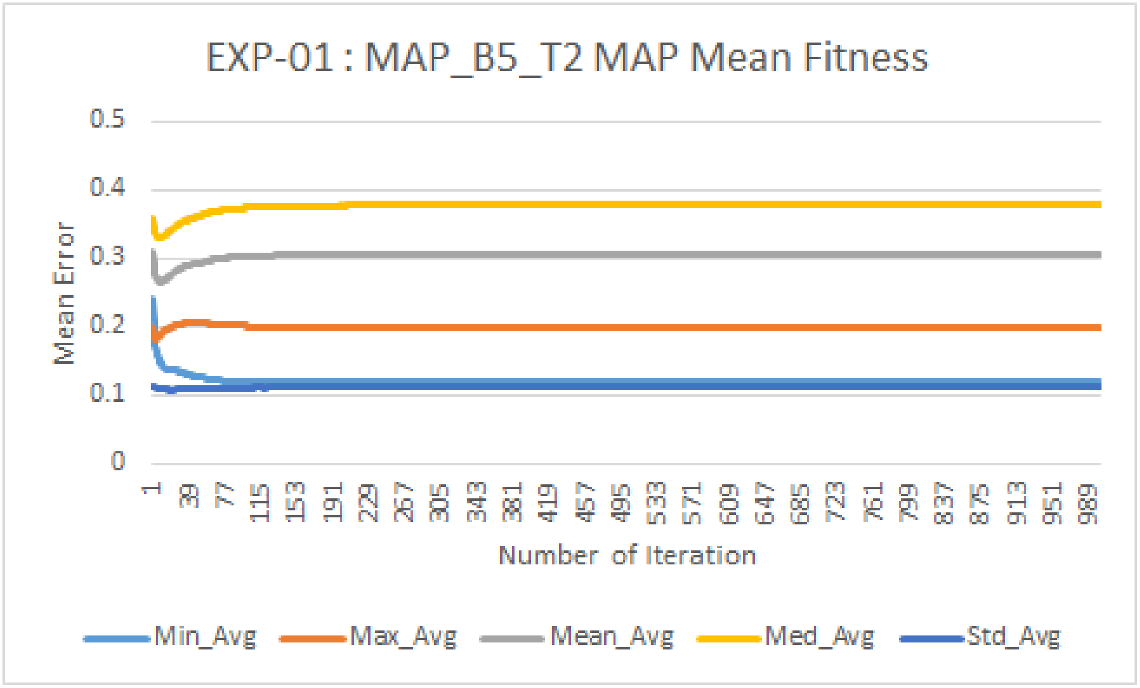


Figure 5.17: Average mean fitness of MAP-Elites (multiple individuals bin - tournament selection) MAP over 20 runs (lower is better).

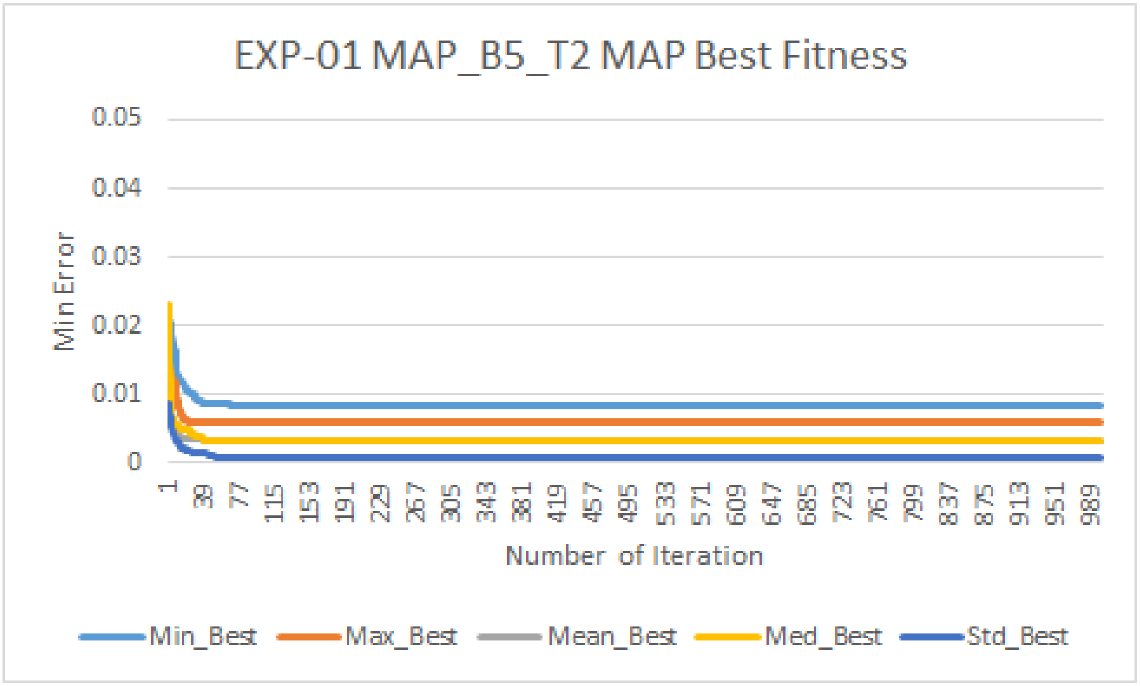


Figure 5.18: Average best fitness of MAP-Elites (multiple individuals bin - tournament selection) MAP over 20 runs (lower is better).

5.1.5 Analysis of experiment 1

Table 5.7: P-Value table for Experiment 1

	Objectives	**MAP_Base**	**MAP_B5_R**	**MAP_B5_T2**
GP	**Min**	1.33E-08 ←	4.97E-07 ←	1.22E-08 ←
	Max	0.156463	0.428268	0.215474
	Mean	0.337507	0.070210	0.347446
	Median	0.424871	0.253703	0.342428
	Std. Dev.	1.65E-05 ↑	0.000507 ↑	0.011135 ↑
MAP_Base	**Min**	—	0.290642	0.424949
	Max	—	0.030523 ←	0.019176 ←
	Mean	—	0.236740	0.220375
	Median	—	0.322586	0.357452
	Std. Dev.	—	0.010367 ←	0.001819 ←
MAP_B5_R	**Min**	—	—	0.379778
	Max	—	—	0.273779
	Mean	—	—	0.038215 ↑
	Median	—	—	0.139580
	Std. Dev.	—	—	0.197085

Some analyses for different algorithms have been discussed in previous sections. In this section we will compare them in more detail. Since the mean error values do not show a normal distribution, non-parametric analysis has been used to calculate the P-values. We used Mann-Whitney U test [66, 69] for the non-parametric analysis. Table 5.7 shows the statistical P-values of the comparison between the best 20 solutions generated by 4 different algorithms. Here, mean error values between the objectives of the original target image and the generated images are used for the analysis to get the P-value.

From Table 5.7, we can see that while comparing the best solutions of GP to the best solutions of three version of MAP-Elites, the mean error of two of the features (minimum and standard deviation) shows significant statistical difference (threshold: $P<0.05$). While comparing base MAP-Elites to the two proposed version of MAP-Elites we can see two features (maximum and standard deviation) shows significant statistical difference (threshold: $P<0.05$). And finally while comparing the two proposed algorithms, we can see a significant statistical difference (threshold: $P<0.05$) between them for one feature (mean). Shaded cells represents significant statistical difference. The arrows next to the P-Values point towards the algorithm that performed better.

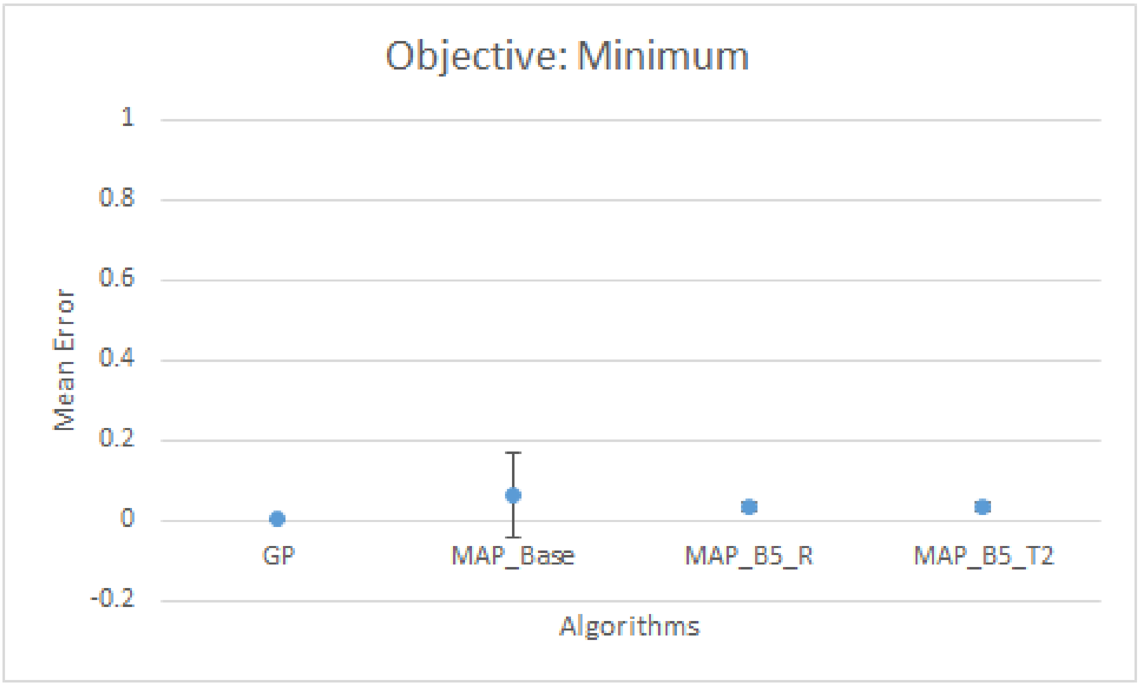


Figure 5.19: Error plot of mean minimum pixel value difference for different algorithms

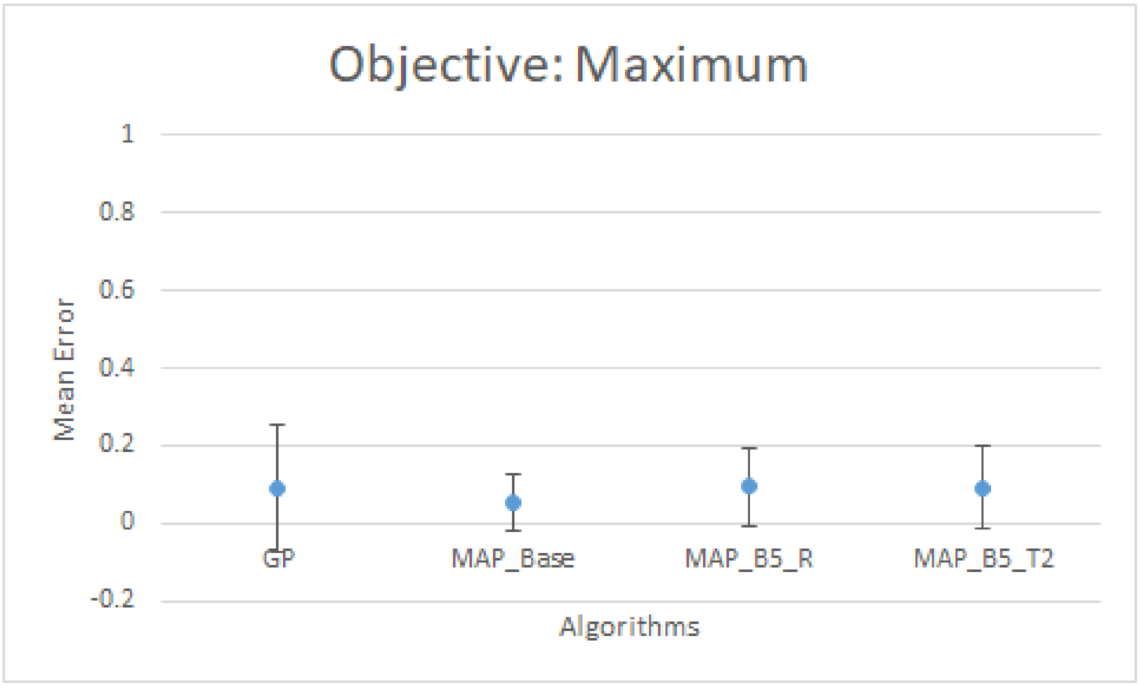


Figure 5.20: Error plot of mean maximum pixel value difference for different algorithms

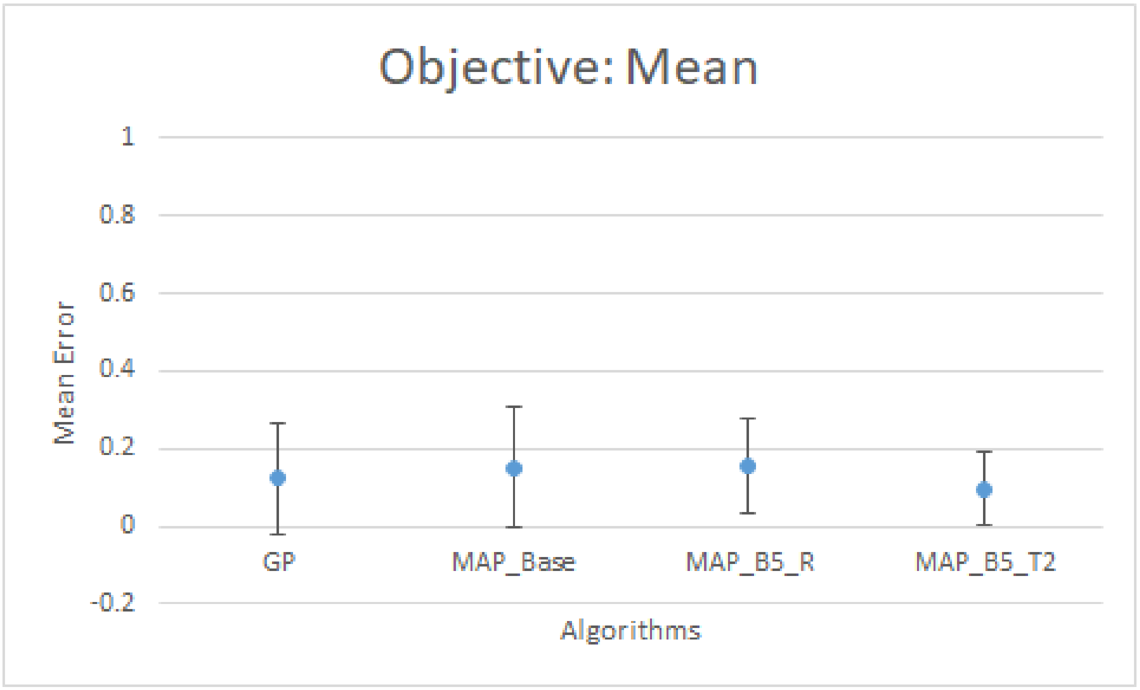


Figure 5.21: Error plot of mean mean pixel value difference for different algorithms

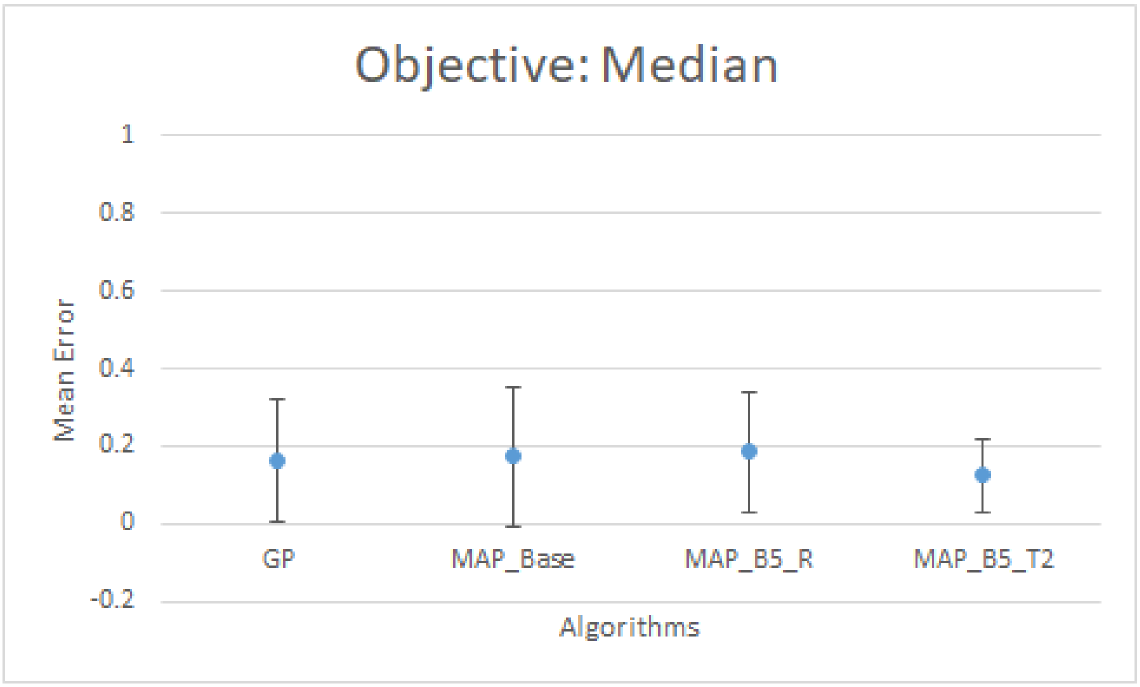


Figure 5.22: Error plot of mean median pixel value difference for different algorithms

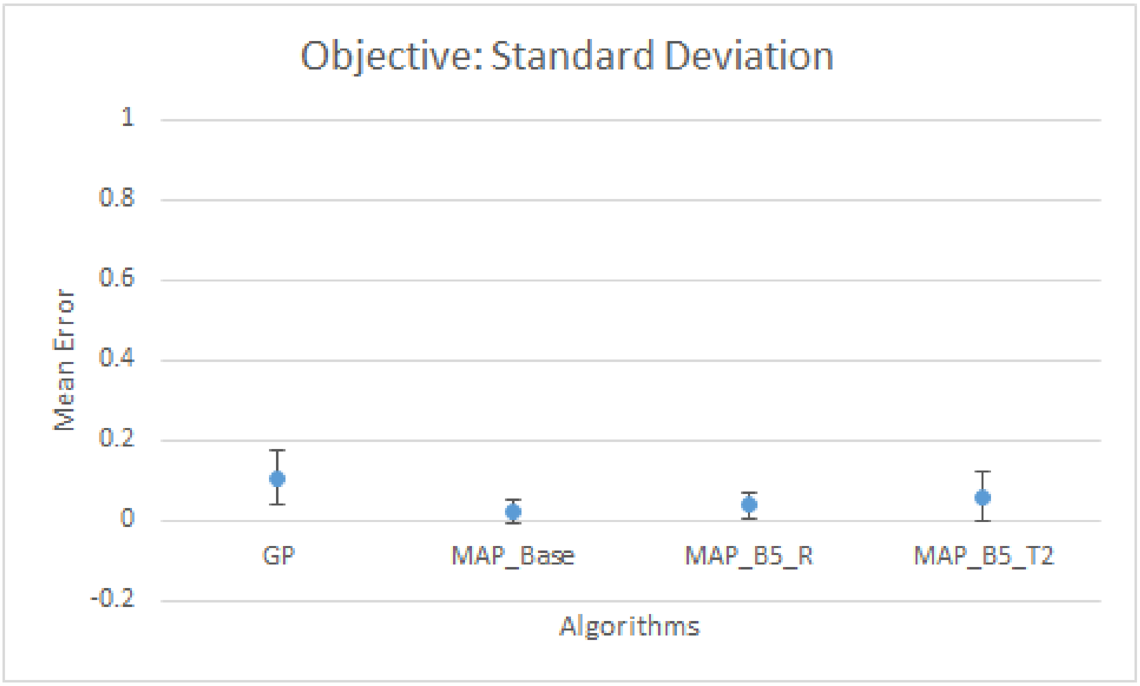


Figure 5.23: Error plot of mean standard deviation value difference for different algorithms

Figures 5.19, 5.20, 5.21, 5.22 and 5.23 show the error plots with standard deviation for the four algorithms. These figures shows the values for five different objectives (minimum, maximum, mean, median and standard deviation, respectively). When we associate these values with the P-value table (Table 5.7), we can see which algorithm performs better in terms of mathematical values. GP outperforms the other three algorithm significantly in terms of minimum (see Figure 5.19) and performance is poor in terms of standard deviation (see Figure 5.23) compared with all three MAP-Elites algorithms. But with other three features, there are no statistically significant difference between GP and other three algorithms. GP outperforms the others as it converges during the run. On the other hand, MAP-Elites solutions diverges as more bins in the MAP are filled.

While comparing the base version of MAP-Elites with the two proposed multi-individual-bin MAP-Elites, we can see base MAP-Elites outperforms the other two version of MAP-Elites algorithm in terms of maximum and standard deviation values (see Figure 5.20 and 5.23). While comparing the random selection - multiple individual bins MAP-Elites to tournament selection - multiple individual bins MAP-Elites, we can see the tournament selection algorithm significantly outperforms the random selection algorithm in terms of mean values.

5.2 Experiment 2

Table 5.5 shows the feature set that has been used for this experiment. If we compare this feature set with the feature set of the first experiment (see Table 5.4), we can see the mean has been replaced by entropy and standard deviation has been replaced with unique gray level. Figure 5.1(a) shows the target image for this experiment.

5.2.1 Basic Genetic Programming (GP)

For this experiment, we kept all the GP parameters the same as the first experiment (see Table 5.1). Figure 5.24 shows the target image in first column and the evolved image for 10 GP runs in the second and third columns. Some of the solutions may look similar visually. The reason for this has been explained in Section 5.1.1.

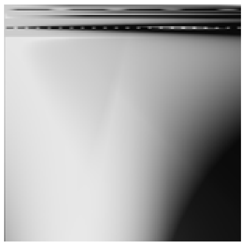
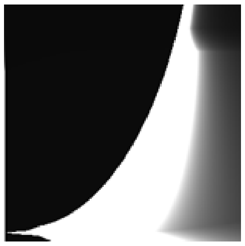
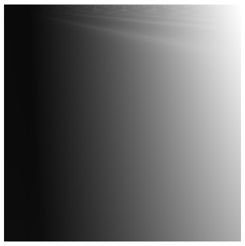
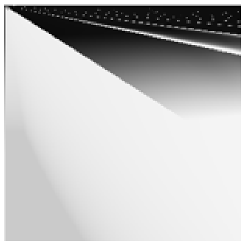
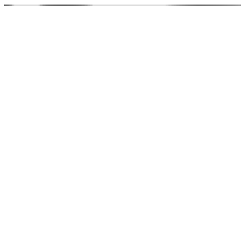
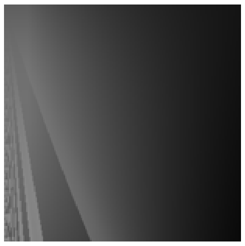
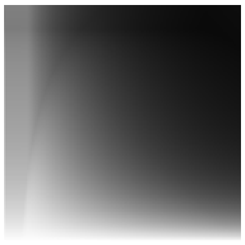
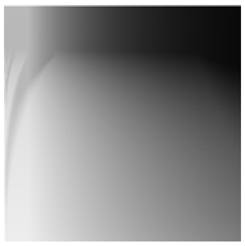
Target Image	Run: 1 - 5	Run: 6 - 10
		
		
		
		
		

Figure 5.24: Best results of GP runs for target image- 1 and grayscale feature set -2 (10 runs).

5.2.2 MAP-Elites: Single Individual Bin (MAP_Base)

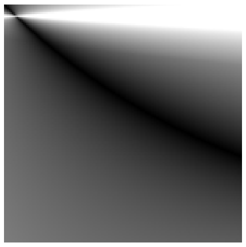
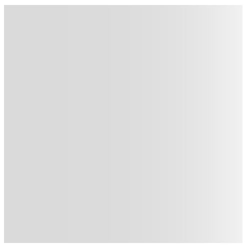
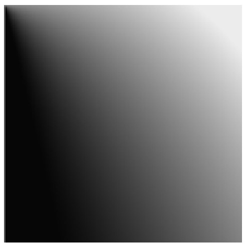
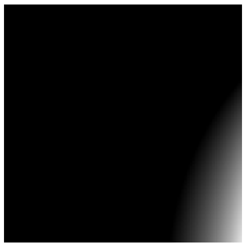
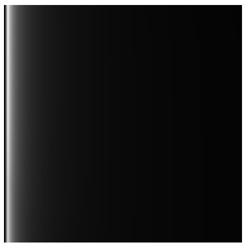
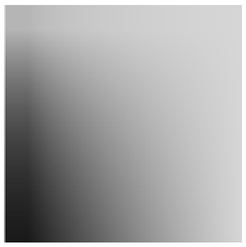
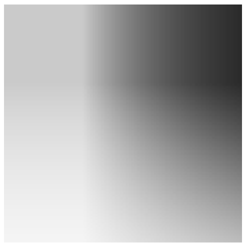
Target Image	Run: 1 - 5	Run: 6 - 10
		
		
		
		
		

Figure 5.25: Best results of base MAP-Elites runs for target image- 1 and grayscale feature set -2 (10 runs).

The MAP-Elites parameters have also been kept the same as experiment 1 (see Table 5.2). The X-axis and Y-axis of the MAP represents the standard deviation difference and mean difference between target image and generated images respectively. Figure 5.25 shows the target image in first column and the evolved image for 10 MAP-Elites run in the second and third columns.

5.2.3 MAP-Elites: Multiple individuals Bin - Random Selection (MAP_B5_R)

All the parameters are kept the same as the previous except the bin size, where it is now set to 5. It has similar behaviour as seen in Section 5.1.3, as the algorithm is the same. Only thing that is changed is the feature space MAP, which is similar to the previous section. The X-axis and Y-axis of the MAP represents the standard deviation difference and mean difference between target image and generated images respectively.

Figure 5.26 shows the target image and the best results from 10 runs. Like the first experiment the objective values are mapped between 0 and 1. The unique gray value and entropy values were kept raw, meaning the unique gray values were 0 to 255 and entropy values were 0 to 8. Here since we are measuring the difference, to make the difference standard for all experiments, this range was set to 0 to 1 for both feature behaviour of the MAP, which is the same range as the objectives.

Figure 5.27 shows the final solution sum of ranks MAP for multiple individual bins - random selection MAP-Elites run (run-2). Here the X-axis represents the standard deviation difference between target image and generated images and the Y-axis represents the mean difference between target image and generated images. The white bins in sum-of-ranks MAP mean that there were no solutions found from those bins (those bins never got filled). If we compare this with Figure 5.6, we can see how changing the behaviour of the MAP from unique gray values and entropy to standard deviation and mean difference affects the MAP. In this experiment we have fewer solutions, as the right half of the MAP remains completely empty. It is the same case for other runs as well, including the base MAP-Elites and multiple individual bins - tournament selection MAP-Elites. This means we have fewer solutions, resulting in less diversity. One of the reasons for the right half of the MAP being empty is because the mean pixel value difference also gets smaller when we try to minimize the differences for the five objectives (see Table 5.5).

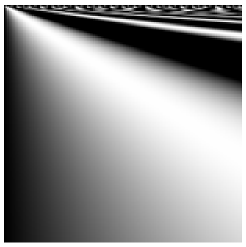
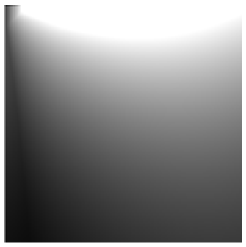
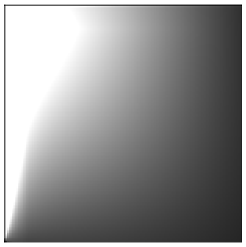
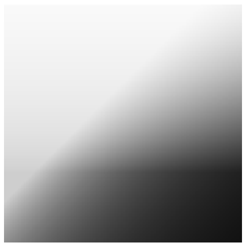
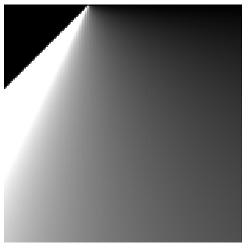
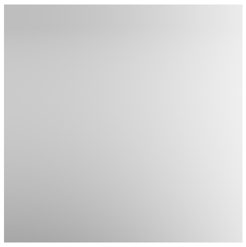
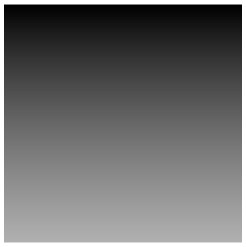
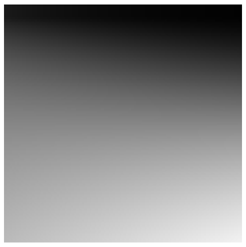
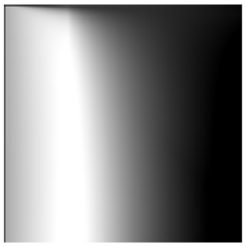
Target Image	Run: 1 - 5	Run: 6 - 10
		
		
		
		
		

Figure 5.26: Best results of MAP-Elites (Multiple individuals bin - random selection) runs for target image- 1 and grayscale feature set -2 (10 runs).

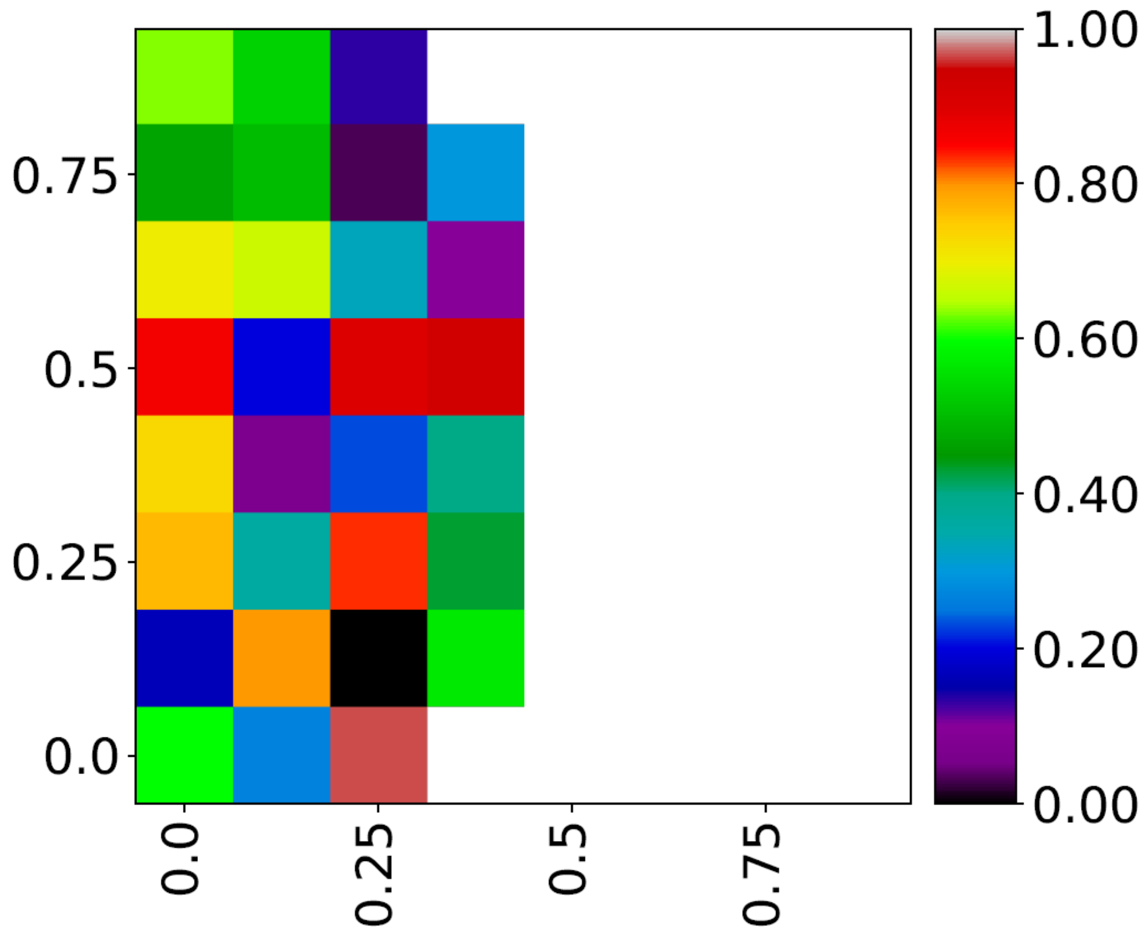
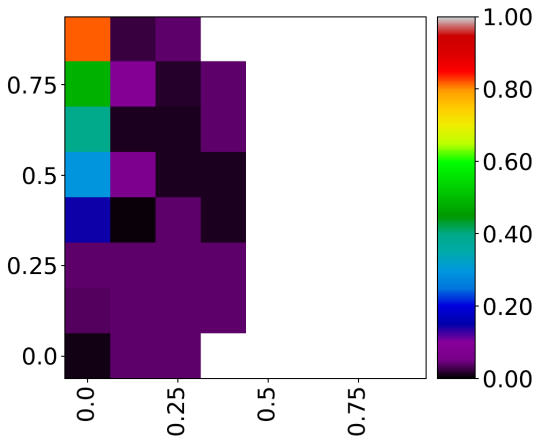
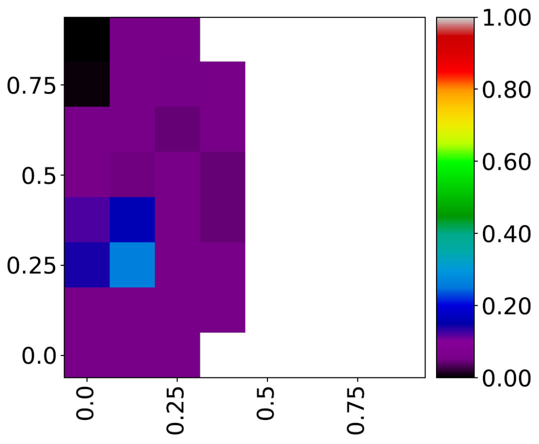


Figure 5.27: Sum-of-ranks MAP of run-2 of experiment-2 (higher is better) [X-axis: standard deviation difference (0-1); Y-axis: mean difference (0-1)].

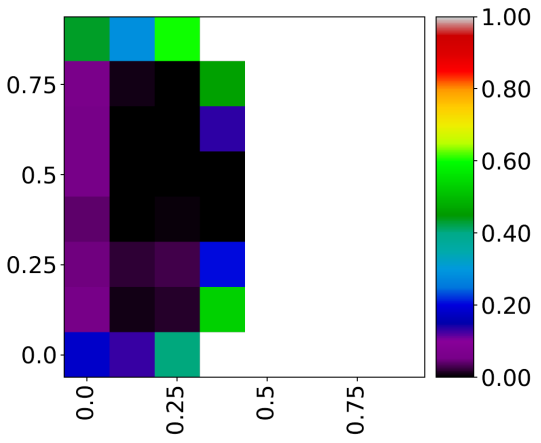
In Figure 5.28 the raw differences of the five objective Min, Max, Entropy, Median, Unique gray values are shown respectively for run-2 of experiment-2 of the base MAP-Elites experiments. We can see different objective differences for different solutions reach zero or very close zero, but the overall ranking may show other solutions with better ranking. Here, we use the same normalized sum-of-ranks technique as experiment 1. For the sum-of-ranks that higher value is preferred and for raw feature difference values the lower is preferred as they are already normalized, as the images are normalized between 0 and 1.



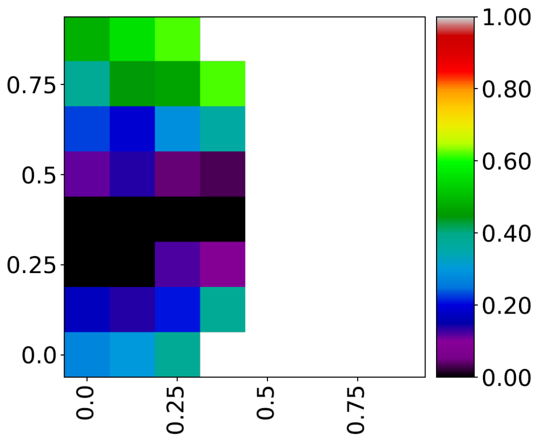
(a) Min difference



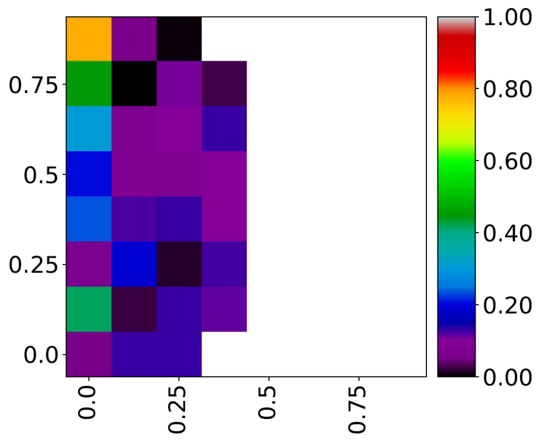
(b) Max difference



(c) Entropy difference



(d) Median difference



(e) Unique gray difference

Figure 5.28: Raw feature difference of run-2 of experiment-2 (lower is better) [X-axis: standard deviation difference (0-1); Y-axis: mean difference (0-1)]

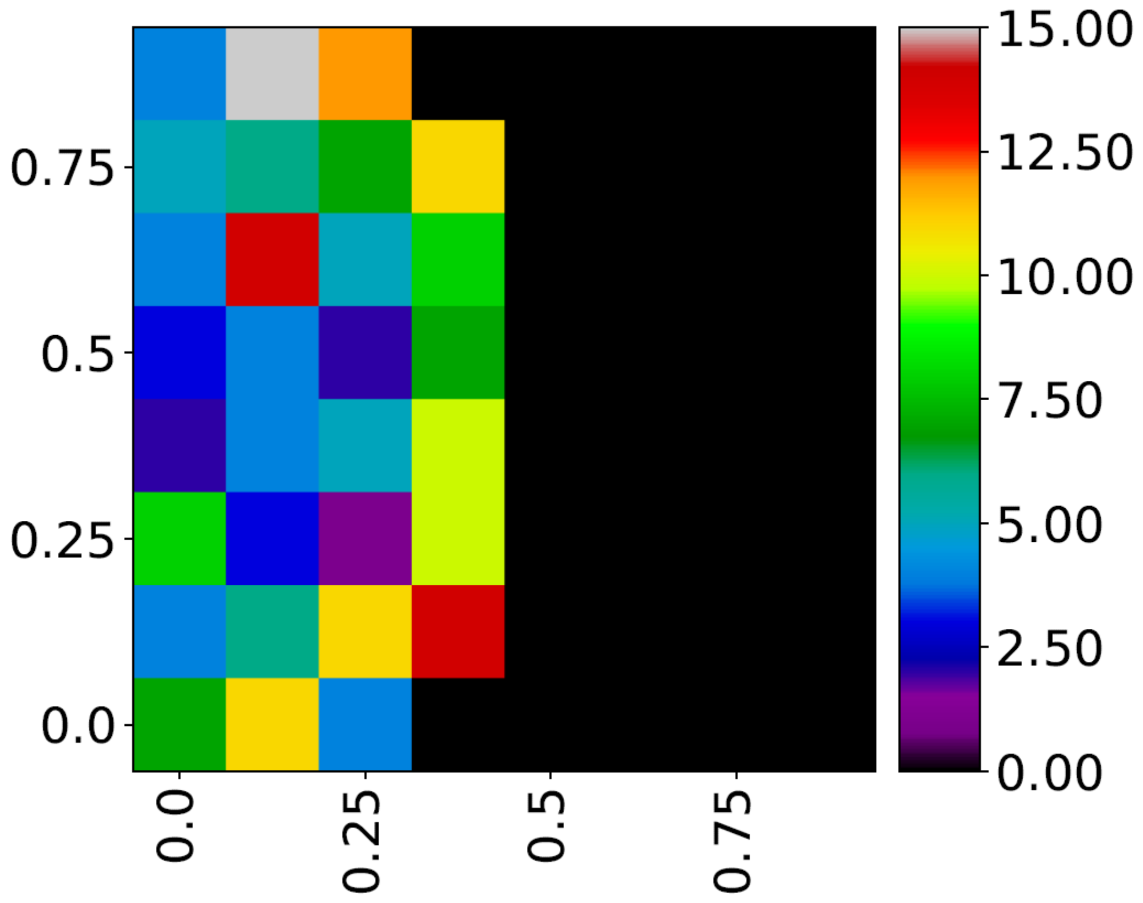


Figure 5.29: Activity MAP of run-2 of experiment-1 (higher is better) [X-axis: standard deviation difference (0-1); Y-axis: mean difference (0-1)].

Figure 5.29 shows the activity grid of the MAP (run 02). These values represents how many times a bin has been updated. Higher value of bins mean the individuals in those bins have been replaced more times than the bins with lower values. Bins with 0 values (black bins) mean they have never been used. If we compare this activity map to the activity map of experiment 1 (see Figure 5.8) we can see individual bins have updated relatively more than in experiment 1. This is because this version of MAP-Elites can hold the top (fittest) 5 solutions per bin, where the base MAP-Elites only holds the best 1 solution. The probability of one of the 5 solution being replaced is higher than replacing the best solution. For this reason, the proposed versions of the MAP-Elites show a higher bin update rate than base the MAP-Elites.

5.2.4 MAP-Elites: Multiple individuals Bin - Tournament Selection (MAP_B5_T2)

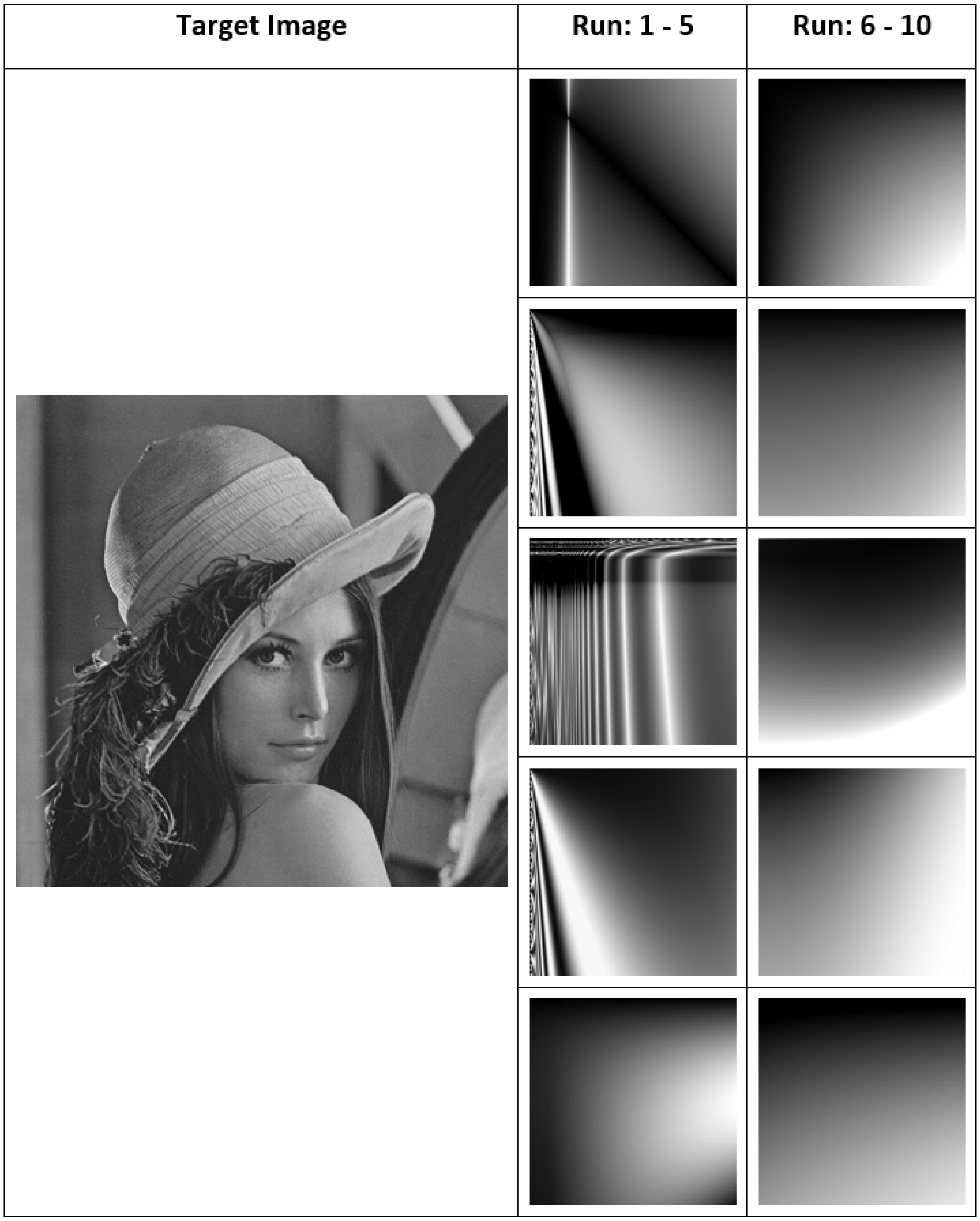


Figure 5.30: Best results of MAP-Elites (multiple individuals bin - tournament selection) runs for target image- 1 and grayscale feature set -2 (10 runs).

This experiment has similar parameters with previous section. The one parameter that is changed is the selection strategy. Instead of selecting individuals at random for reproduction, a tournament selection is used with a tournament size of two. The reason for having such a small tournament has been described in Section 5.1.4 . Figure 5.30 shows the target image and the best results from 10 runs.

5.2.5 Analysis of experiment 2

We used Mann-Whitney U test [66, 69] for the non-parametric analysis. Table 5.8 shows the statistical P-values of the comparison between the best 10 solutions generated by 4 different algorithms. Again, mean error values between the objectives of the original target image and the generated images are used for the analysis to get the P-value.

From Table 5.8, we can see the P-value for the best solution comparison between algorithms. The mean error of minimum difference shows significant statistical difference (threshold: $P<0.05$), when comparing GP to the MAP-Elites algorithm. We can see mean entropy difference shows significant statistical difference (threshold: $P<0.05$) with the proposed versions of MAP-Elites. And finally while comparing the GP algorithm with the second proposed algorithm (MAP_B5_T2) we can see a significant statistical difference (threshold: $P<0.05$) between them for median value difference. In short, GP shows significant difference for only one objective while comparing to base MAP-Elites, two objectives when compared with MAP_B5_R and for three objectives when compared with MAP_B5_T2. Other combinations do not show any significant difference. The arrows next to the P-Values point towards the algorithm that performed better.

Table 5.8: P-Value table for Experiment 2

	Objectives	MAP_Base	MAP_B5_R	MAP_B5_T2
GP	**Min**	0.000109 $\leftarrow$	6.23E-05 $\leftarrow$	4.11E-05 $\leftarrow$
	Max	0.053767	0.437913	0.157672
	Entropy	0.060612	0.037831 $\uparrow$	0.026951 $\uparrow$
	Median	0.105975	0.106147	0.007010 $\uparrow$
	Unique Gray	0.172080	0.352623	00.409594
MAP_Base	**Min**	—	0.328370	0.412255
	Max	—	0.103110	0.189622
	Entropy	—	0.192337	0.395668
	Median	—	0.285230	0.454827
	Unique Gray	—	0.051534	0.063690
MAP_B5_R	**Min**	—	—	0.344741
	Max	—	—	0.156335
	Entropy	—	—	0.311588
	Median	—	—	0.144869
	Unique Gray	—	—	0.178600

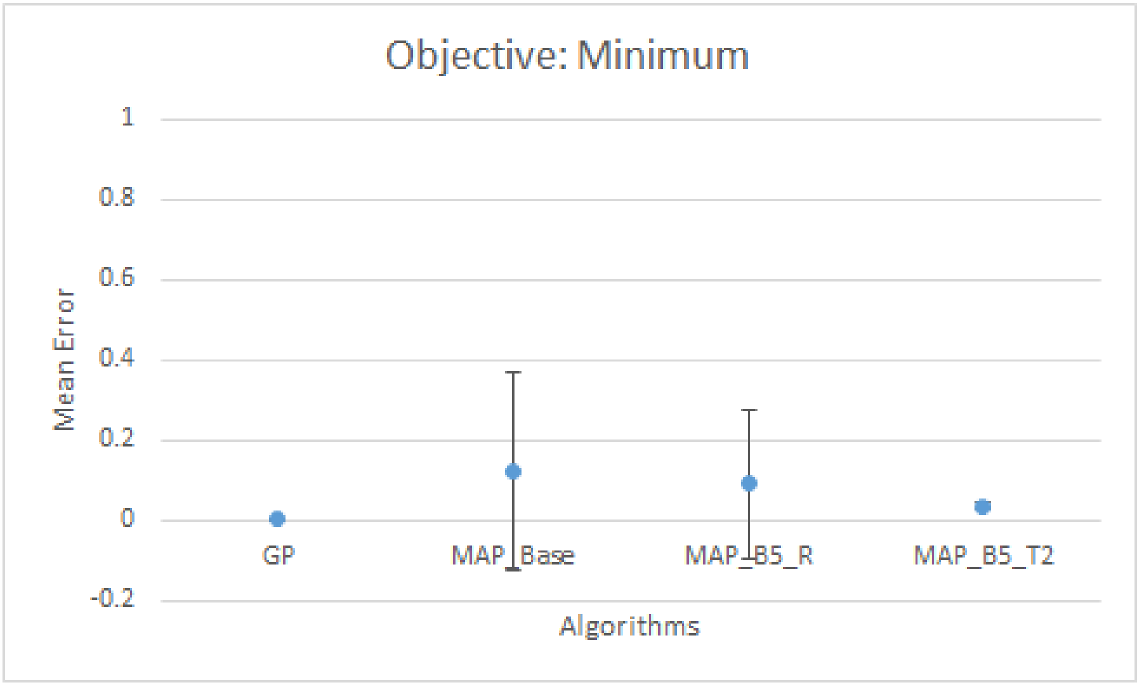


Figure 5.31: Error plot of mean minimum pixel value difference for different algorithms

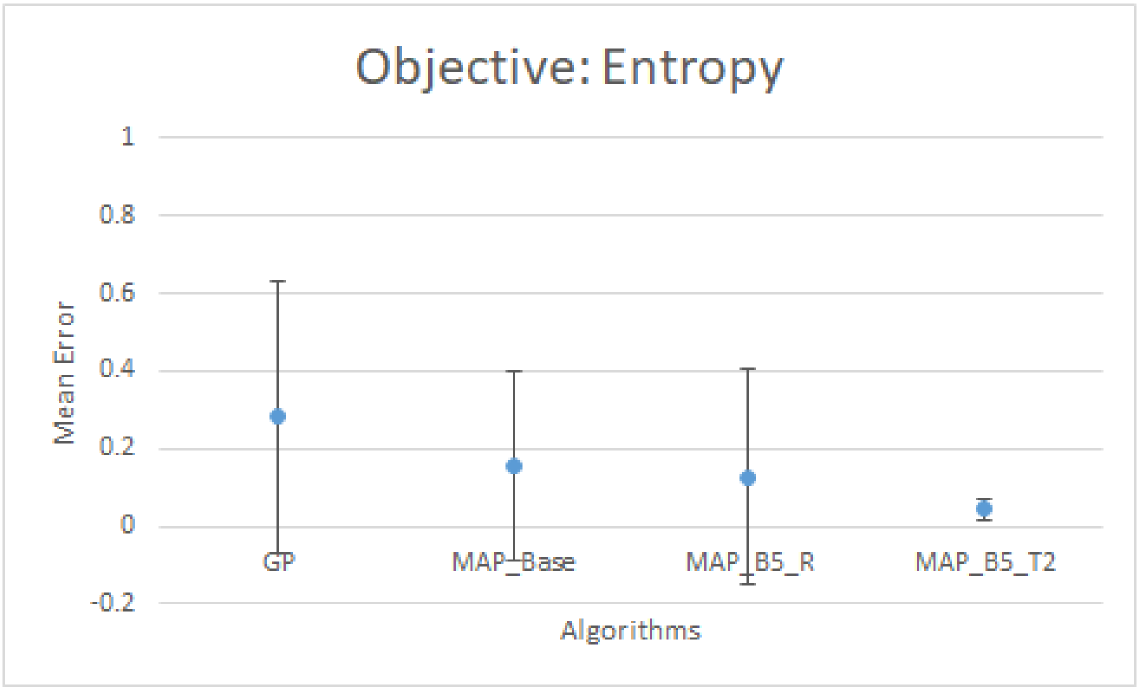


Figure 5.32: Error plot of mean entropy value difference for different algorithms

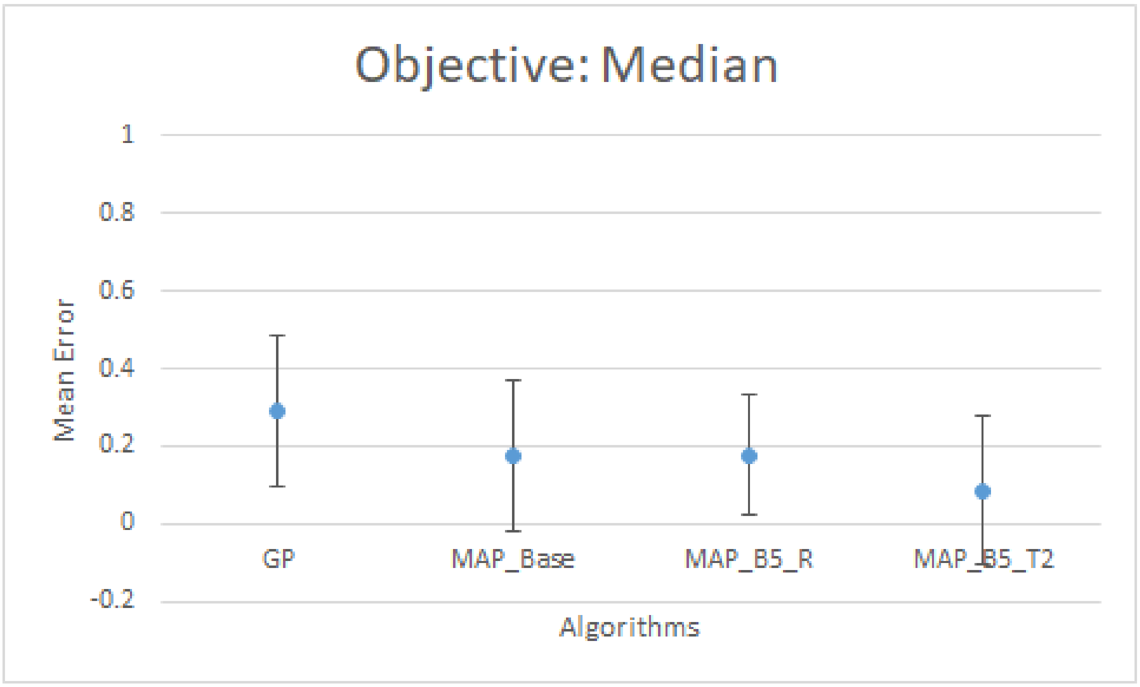


Figure 5.33: Error plot of mean median pixel value difference for different algorithms

Figures 5.31, 5.32 and 5.33 show the error plots with standard deviation for the four algorithms. These figures shows the values for five different objectives (minimum, maximum, entropy, median and unique gray value, respectively). When we associate these figures with the P-value table (Table 5.7), we can see which algorithm performs better in terms of statistical values. Shaded cells represents significant statistical difference and arrows nest to the values points towards the algorithm that is performing better for that value. GP outperforms the other three algorithm significantly in terms of minimum (see Figure 5.31) and performance is poor in terms of entropy and median (see Figure 5.32) when compared with the proposed versions of MAP-Elites, where both proposed algorithms outperforms GP. MAP_B5_T2 outperforms GP also in terms of median value (see Figure 5.33). But with other values and comparison between other algorithm pairs, we cannot see any significant statistical difference (threshold: $P<0.05$).

Here only three error plots are showed for three out of five objectives. The reason behind this is the other two objectives do not show any significant statistical difference between any algorithm (see Table 5.8), and the plots do not provide any significantly useful information. In experiment 1, every single objective provided some significant differences between algorithms, and all of the objectives had their own error plot, which is not the case here.

5.3 Experiment 3

Table 5.6 shows the image features that have been selected as objectives for this experiment. If we look closely, we can see a similarity with Table 5.4 used for experiment 1. The only difference between experiment 1 and 3 is experiment 3 has one additional objective, number of edges in the image. The range of edge is also mapped between 0 and 1 to scale it with other objective values. 10 runs were performed with the same parameters and same target image (Target image 2; see Figure 5.1(b)), producing 10 different solutions. Different versions of the MAP-Elites algorithm generated hundreds of solutions for each run.

5.3.1 Basic Genetic Programming (GP)

The base GP system parameters are in Table 5.1. Using target image-2 as the system target GP generated some pretty interesting solutions. Figure 5.34 shows the best solution of the 10 GP runs. With the addition of edge count as an objective, we can see some interesting characteristics. GP is actively trying to create edges which was not seen in experiment-1 as edge was not an objective for fitness in that experiment.

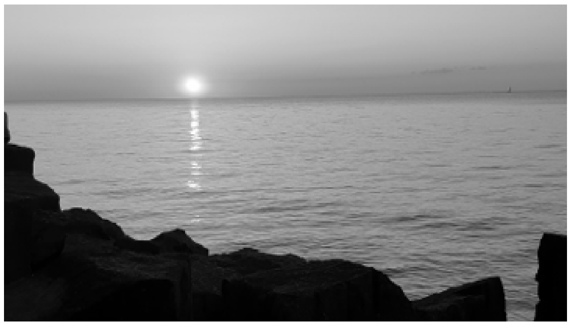
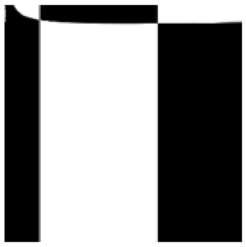
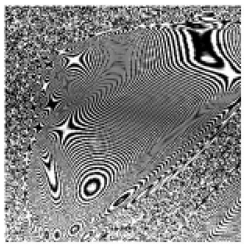
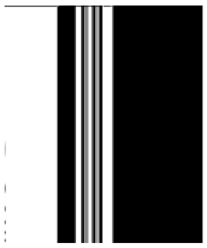
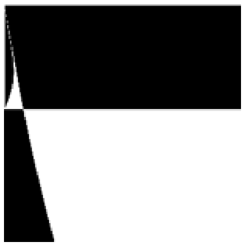
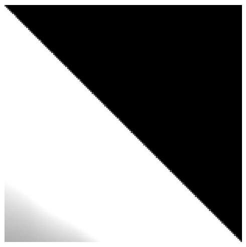
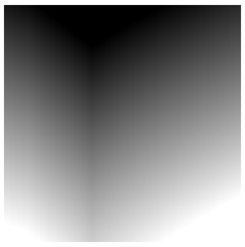
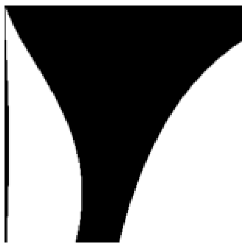
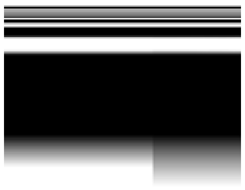
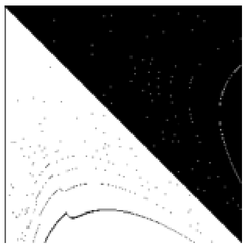
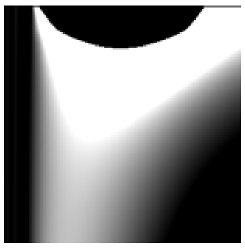
Target Image	Run: 1 - 5	Run: 6 - 10
		
		
		
		
		

Figure 5.34: Best results of GP runs for target image- 2 and grayscale feature set -3 (10 runs).

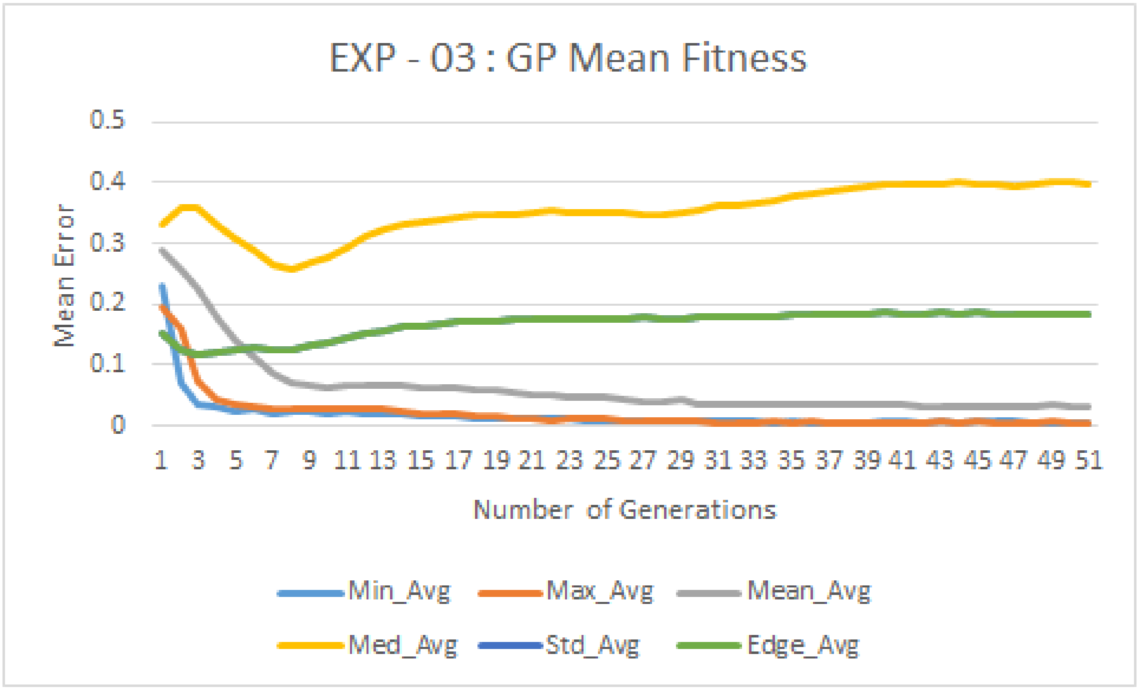


Figure 5.35: Average mean fitness of GP population over 10 runs (lower is better).

Figure 5.35 shows the mean fitness plot of GP averaged over 10 runs. We can see that the mean error values declines rapidly in first few generations for four out of six objectives. But edge and median values decline and then go up. This behaviour is directly related to our fitness evaluation method. Sum-of-ranks consider all the ranks of different objectives of an individual to assign them a final rank. Here, the sum-of-ranks sacrifice some objectives (mean, edge) to benefit the majority.

5.3.2 MAP-Elites: Single Individual Bin (MAP_Base)

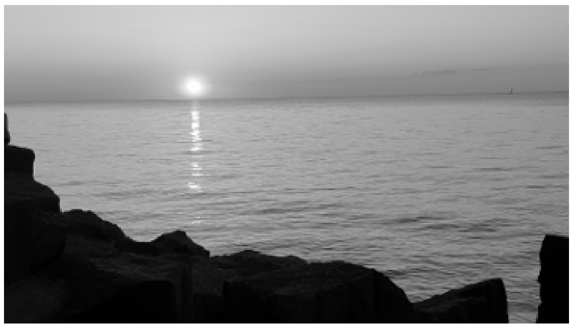
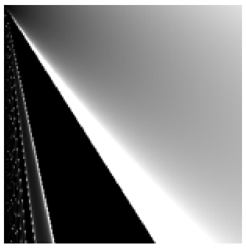
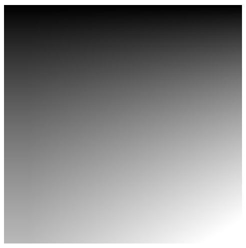
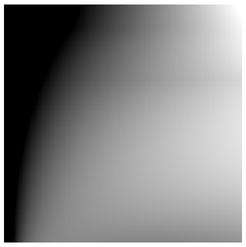
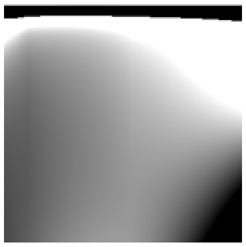
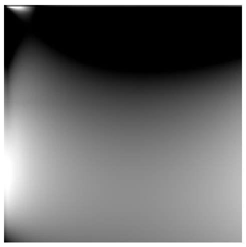
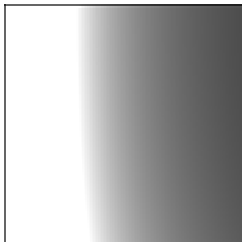
Target Image	Run: 1 - 5	Run: 6 - 10
		
		
		
		
		

Figure 5.36: Best results of base MAP-Elites runs for target image- 2 and grayscale feature set -1 (20 runs).

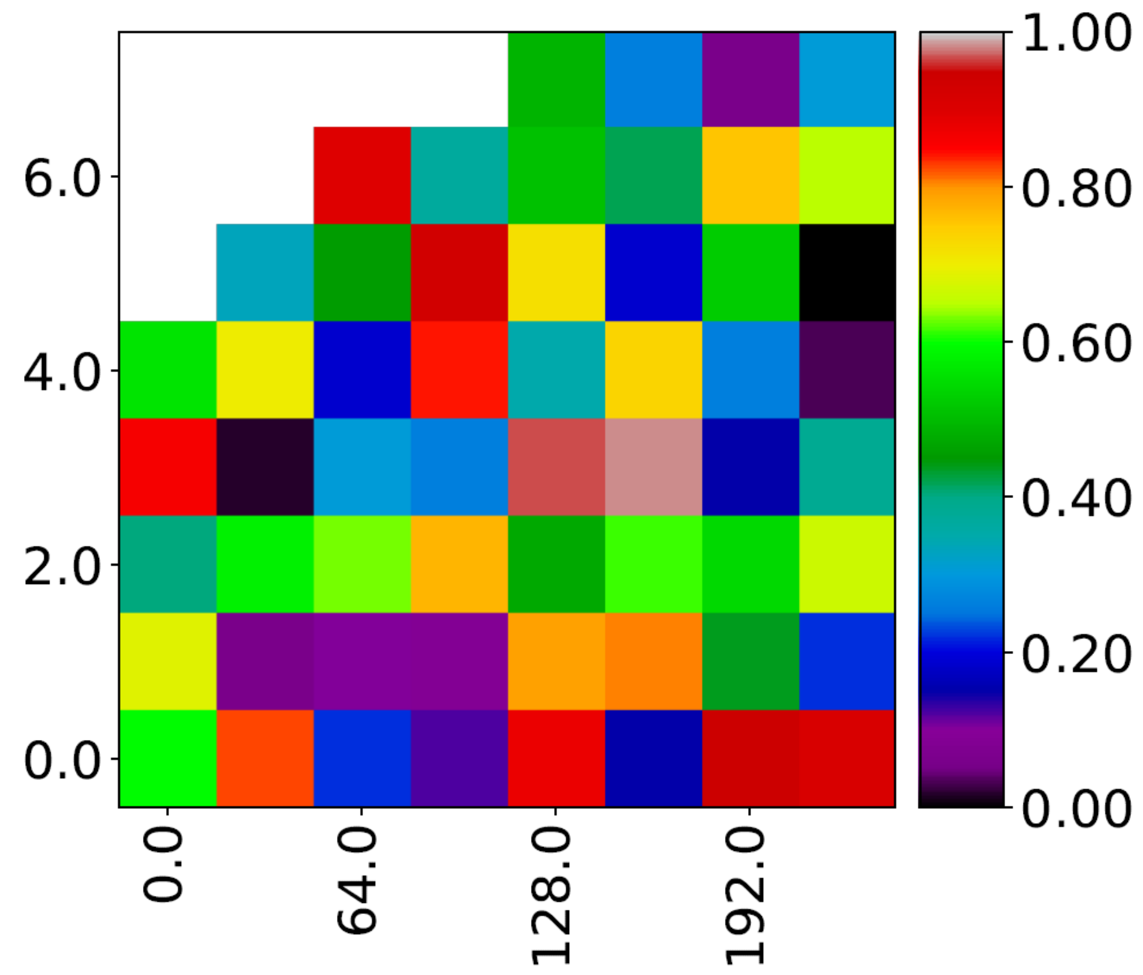
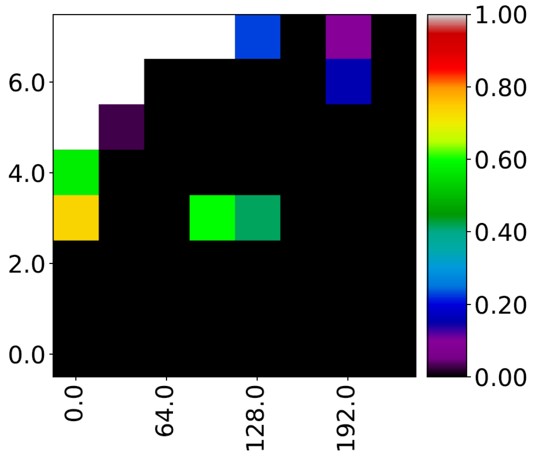


Figure 5.37: Sum-of-ranks MAP of run-5 of experiment-3 (higher is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)].

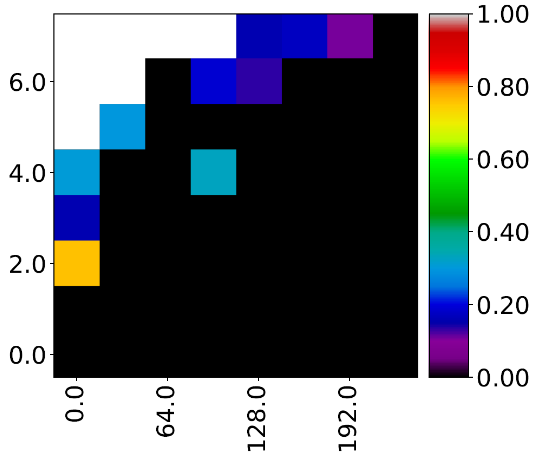
This experiment uses the base MAP-Elites algorithm (bin size = 1). Other parameters for MAP-Elites are in Table 5.2. The MAP feature behaviour is set to unique gray level (range : 0 to 255) and entropy (range : 0 to 8) and the objectives remains the same as previous section (see Table 5.6). Figure 5.36 shows the best results from 10 base MAP-Elites run for target image 2 shown in the left most column of the figure. Values that appears to be absolute black or white, are not necessarily absolute black or white. In most cases they are very close to the pixel value of black or white, and only appear to be absolute black or white visually. The number of generations in the GP experiment is set to 50 with population size of 300, while the number of iterations for the MAP-Elites algorithm is set to 1000 with a batch size of 15, to keep the total number of evaluated individuals equivalent between base GP and MAP-Elites algorithms.

Figure 5.37 shows the final solution ranking of the MAP (sum-of-rank) for one of the MAP-Elites run (run-5). In the sum-of-ranks MAP, all the ranks are mapped between 0 and 1, and then inversed to make the one with the best ranking value have the value of 1 and the worst ranked individual have a value of 0. The reason for mapping them between 0 and 1 is to normalize all the MAPs of the experiments between 0 and 1. We have also mapped the values of grayscale images between 0 and 1, where 0 represents the grayscale value 0 (black) and 1 represents grayscale value 255 (white). The rest of the gray levels are mapped accordingly. During unique grayscale value calculations and entropy calculations, we used unsigned integer values between 0 and 255, for grayscale image as there are 256 unique grayscale values. Using a float in this case can result in millions of unique grayscale values, which is not practical.

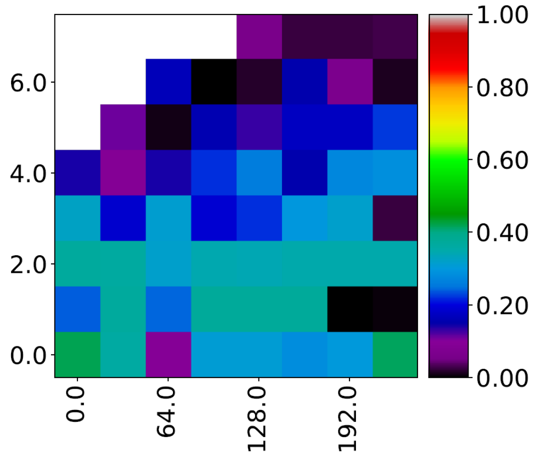
In Figure 5.38, the raw difference is plotted between six objectives; Min, Max, Mean, Median, Standard division and Edge values are shown respectively for run-5 of experiment-3 of the base MAP-Elites experiments. We can see different objective differences for different solutions reach zero or very close zero, but the overall ranking may show other solutions with better ranking (same as experiment 1).



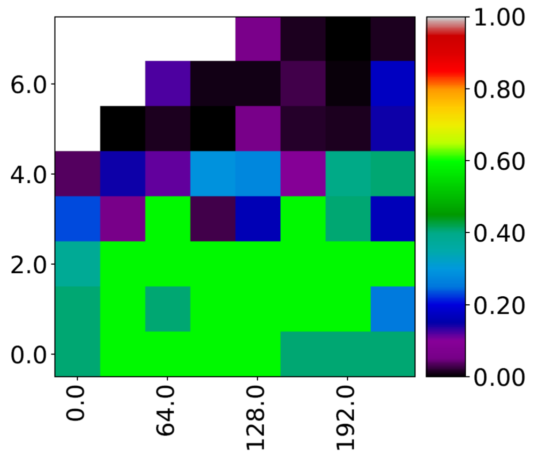
(a) Min difference



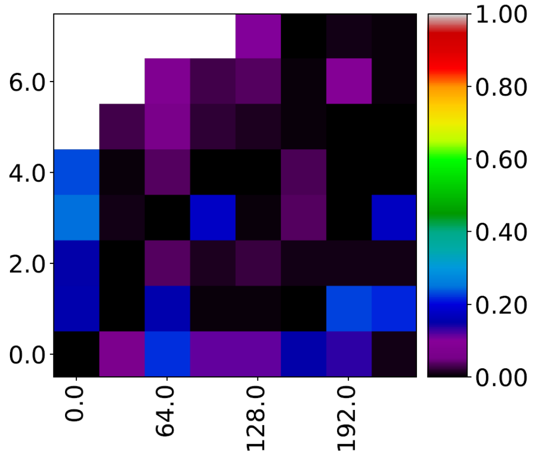
(b) Max difference



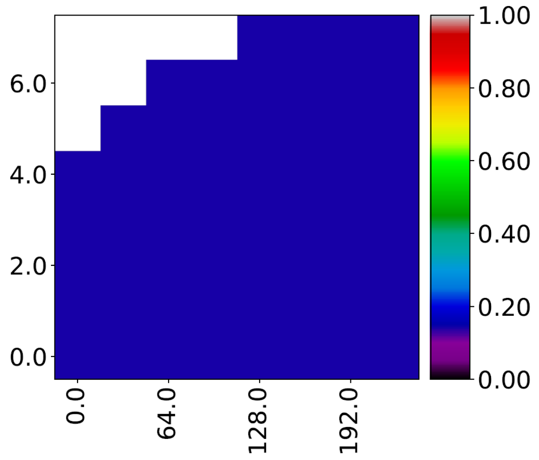
(c) Mean difference



(d) Median difference



(e) Std. Dev. difference



(f) Edge difference

Figure 5.38: Raw feature difference of run-5 of experiment-3 (lower is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)]

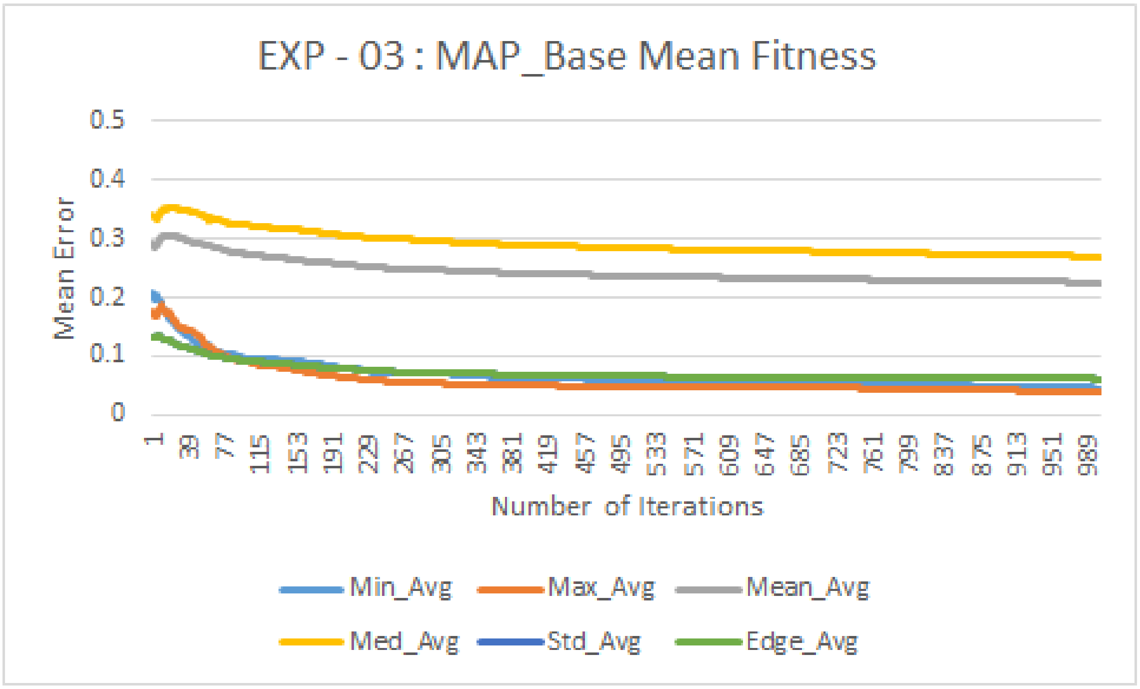


Figure 5.39: Average mean fitness of base MAP-Elites MAP over 10 runs (lower is better).

Figure 5.39 shows the mean fitness of base MAP-Elites MAP for all five objectives over 1000 iterations averaged over 10 runs. If we compare this Figure to Figure 5.35, we can see some clear differences. The GP fitness (mean error) average of the population improves over generations for four of the objectives and gets worse for the other two objectives. But in the case of base MAP-Elites, the average fitness (mean error) improves over iteration for all six objectives. We can also see that, for those four objectives of GP the error rate is lower compared with similar objectives in MAP-Elites. This is because GP starts with a large randomly generated population and converges over generations. On the other hand, MAP-Elites starts with a few randomly generated solutions on the MAP, which is a very small portion of the MAP. Over time the number of solutions increases as does the diversity.

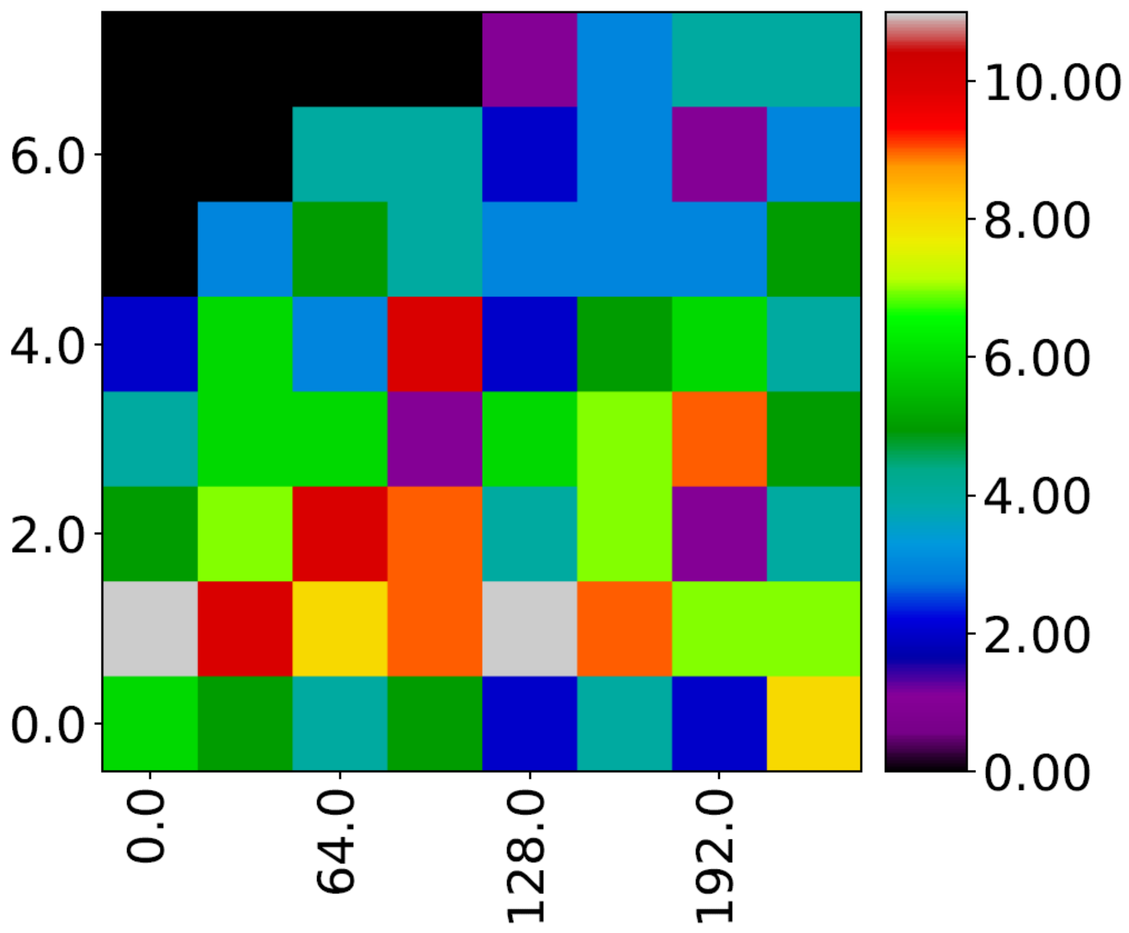


Figure 5.40: Activity MAP of run-5 of experiment-3 (higher is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)].

Figure 5.40 shows the activity grid of the MAP. These values represents how many times a bin has been updated. Higher value of bins mean the individuals in those bins have been replaced more times than the bins with lower values. Bins with 0 values mean they have never been used. In these experiments, we used six image features as objectives (see Table 5.6). By looking at the colour bar next to the heat MAP for activity grid, we can see how many times each bin has been updated. Even though there were 1000 iterations with each creating 15 new individuals, we can see the total number of times bins are updated is significantly lower, proving that all newly evolved solutions are not necessarily better than previous best for a feature space in the MAP. The black bins are empty bins, meaning there were no solutions found for those feature values.

5.3.3 MAP-Elites: Multiple individuals Bin - Random Selection (MAP_B5_R)

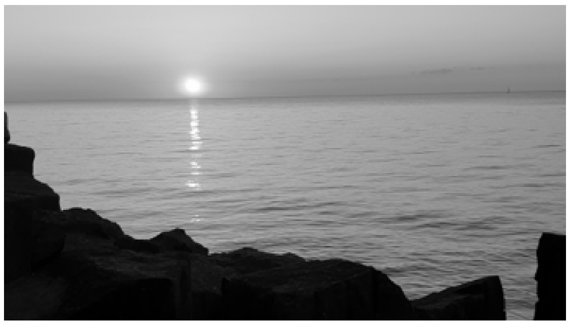
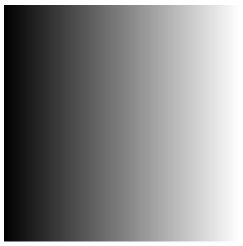
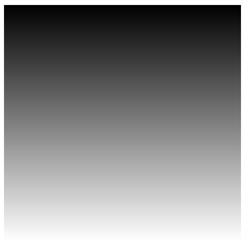
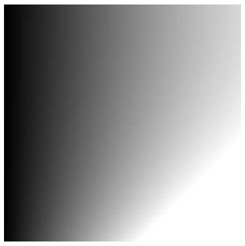
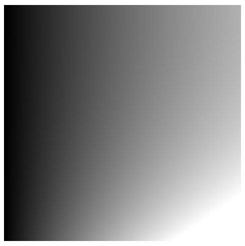
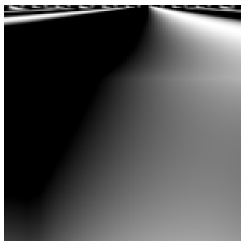
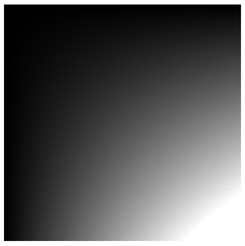
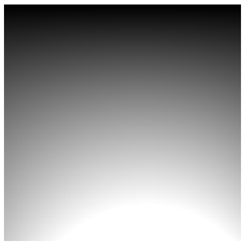
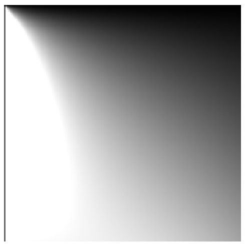
Target Image	Run: 1 - 5	Run: 6 - 10
		
		
		
		
		

Figure 5.41: Best results of MAP-Elites (Multiple individuals bin - random selection) runs for target image- 1 and grayscale feature set -3 (10 runs).

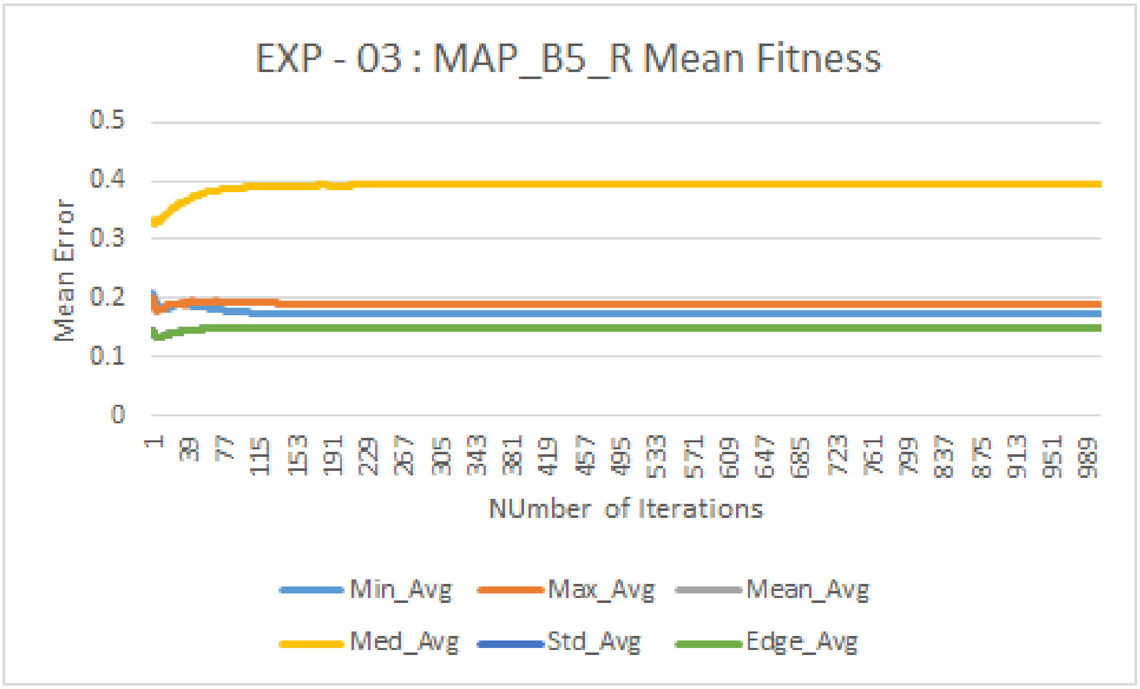


Figure 5.42: Average mean fitness of MAP_B5_R MAP over 10 runs (lower is better).

For this experiment we have used all the same objectives (see Table 5.6) and parameters as the previous section. The only difference is the number of individuals in each bins. Here each bin can store up to best 5 individuals for that feature space compared with a single individual in MAP_Base. Figure 5.41 shows the best solutions from 10 runs of MAP_B5_R.

Figure 5.42 shows the mean fitness of MAP_B5_R MAP for all six objectives over 1000 iterations averaged over 10 runs. If we compare this figure to Figures 5.35 and 5.39, we can see some differences. The GP fitness (mean error) average of the population improves over generations for four of the objectives and gets worse for the other two objectives, the base MAP-Elites gets better in all six objectives, and the MAP_B5_R gets better over five of the objectives out of six. But we can see the average mean error values appears to be higher for this version of MAP-Elites. As mentioned before (Section 5.1.4), this is due to the fact that multiple individual bins MAP-Elites store up to five top solutions for a bin, and not just the best one, which contributes to a lower mean as all of those individuals are considered during mean calculation.

5.3.4 MAP-Elites: Multiple individuals Bin - Tournament Selection (MAP_B5_T2)

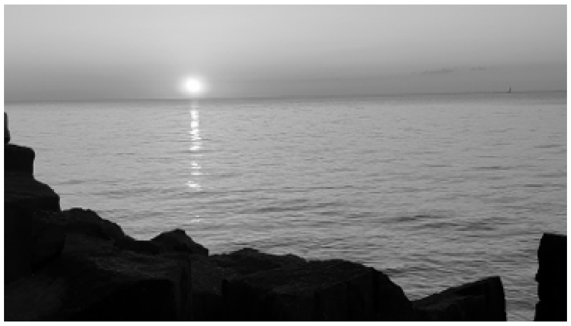
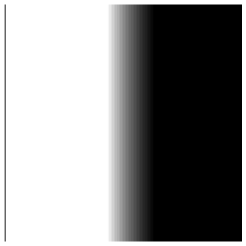
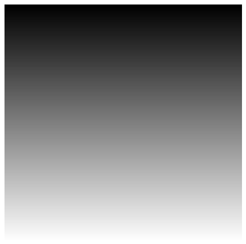
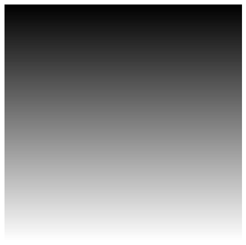
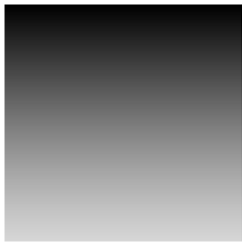
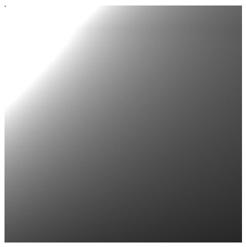
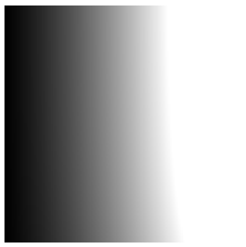
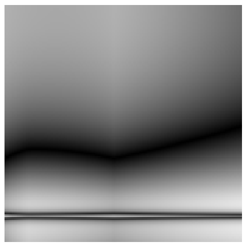
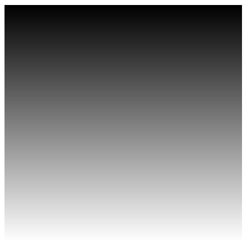
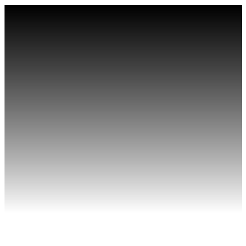
Target Image	Run: 1 - 5	Run: 6 - 10
		
		
		
		
		

Figure 5.43: Best results of MAP-Elites (multiple individuals bin - tournament selection) runs for target image- 2 and grayscale feature set -3 (10 runs).

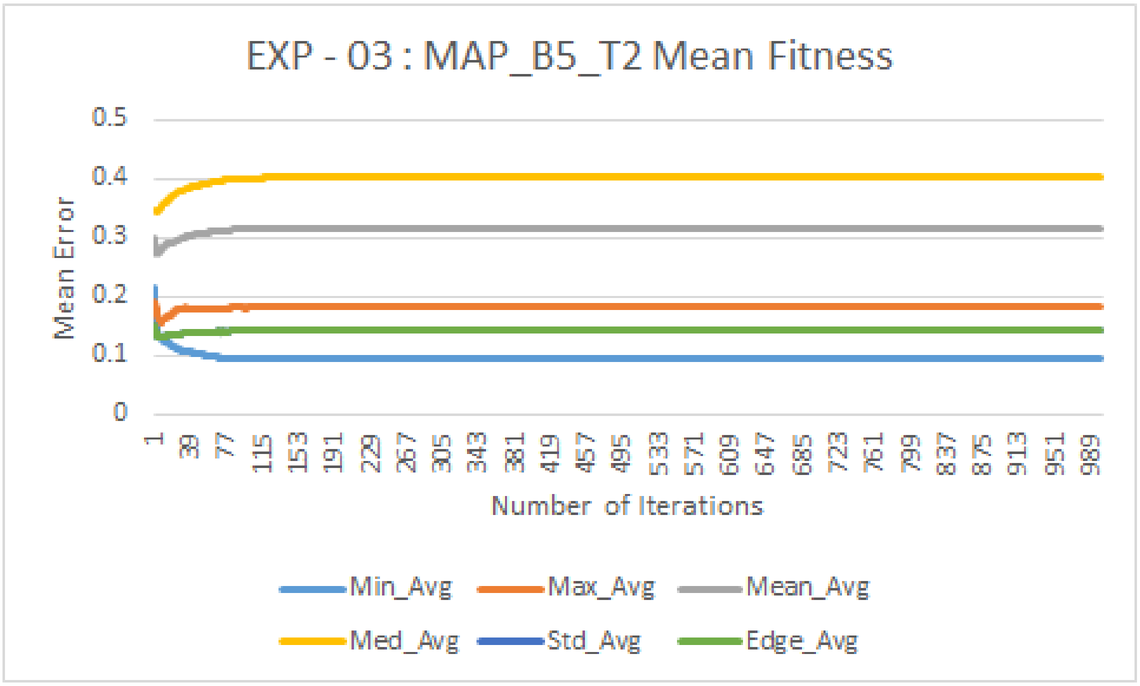


Figure 5.44: Average mean fitness of MAP_B5_T2 MAP over 10 runs (lower is better).

For this experiment we have used all the same objectives (see Table 5.6) and parameters as previous section. The bin size is also same (bin size : 5). The only difference is the selection strategy for selecting parents for reproduction. In previous section random selection is used. Here, a tournament selection is used with a tournament size of two. Figure 5.43 shows the best solutions from 10 runs of MAP_B5_T2.

Figure 5.44 shows the mean fitness of MAP_B5_T2 MAP for all six objectives over 1000 iterations averaged over 10 runs. If we compare this figure to Figure 5.35, 5.39 and 5.42, we can see the average fitness differences between our four algorithms. The fitness plots are highly influenced due to the fact that each bin can store up to five individuals. The downside of this while comparing fitness plots is discussed in previous section (section 5.3.3). But we can see the overall mean error values are lower compared with the multiple individual - random selection strategy. This shows that tournament selection produced better solutions compared with random selection approach for multiple individual bins MAP-Elites.

5.3.5 Analysis of experiment 3

Table 5.9: P-Value table for Experiment 3

	Objectives	**MAP_Base**	**MAP_B5_R**	**MAP_B5_T2**
GP	**Min**	1.000000	1.000000	1.000000
	Max	1.000000	0.184060	0.038792 ←
	Mean	0.000384 ←	0.000089 ←	0.000089 ←
	Median	0.000323 ↑	0.006306 ↑	0.001557 ↑
	Std. Dev.	0.000123 ↑	0.000215 ↑	0.000836 ↑
	Edge	0.022537 ↑	0.051795	0.105717
MAP_Base	**Min**	—	1.000000	1.000000
	Max	—	0.184060	0.038792 ←
	Mean	—	0.060332	0.424914
	Median	—	0.022295 ←	0.012707 ←
	Std. Dev.	—	0.005603 ←	0.000836 ←
	Edge	—	0.015446 ←	0.022214 ←
MAP_B5_R	**Min**	—	—	1.000000
	Max	—	—	0.116953
	Mean	—	—	0.062669
	Median	—	—	0.350791
	Std. Dev.	—	—	0.053766
	Edge	—	—	0.500000

In this section we will compare the different algorithms used in this experiment. We use Mann-Whitney U test [66, 69] for the non-parametric analysis. Table 5.9 shows the statistical P-values of the comparison between the best 10 solutions generated by 4 different algorithms. Here, mean error values between the objectives of the original target image and the generated images are used for the analysis to get the P-value. Shaded cells represents significant statistical difference and arrows nest to the values points towards the algorithm that is performing better for that value.

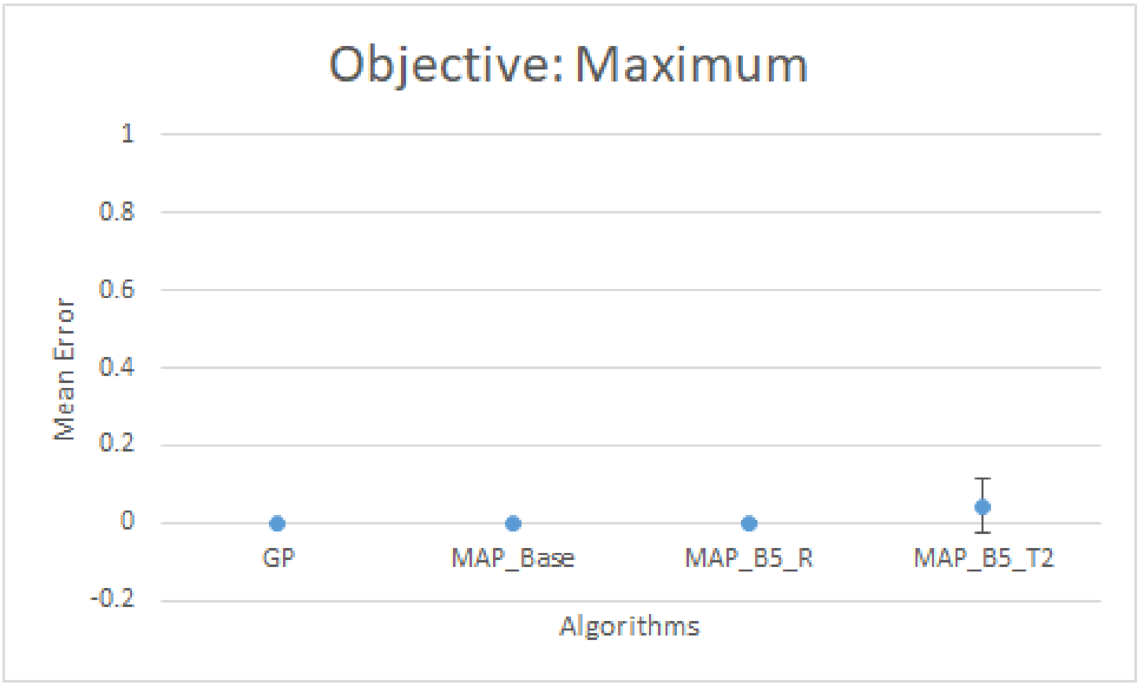


Figure 5.45: Error plot of mean maximum pixel value difference for different algorithms

From Table 5.9, we can see the P-value for the best solution comparison between algorithms. The mean error of maximum difference shows significant statistical difference (threshold: $P<0.05$), when comparing GP to the MAP_B5_T2 algorithm. We can see mean, median and standard deviation shows significant statistical difference (threshold: $P<0.05$) while comparing GP with three version of MAP-Elites algorithms. Also there is a significant statistical difference (threshold: $P<0.05$) between GP and base MAP-Elites in terms of unique gray value. The value 1.0 in the MAP represents that the two sets of data were identical. If we observe closely these are for the difference value for min and max pixel value of the image, which were all zero.

.

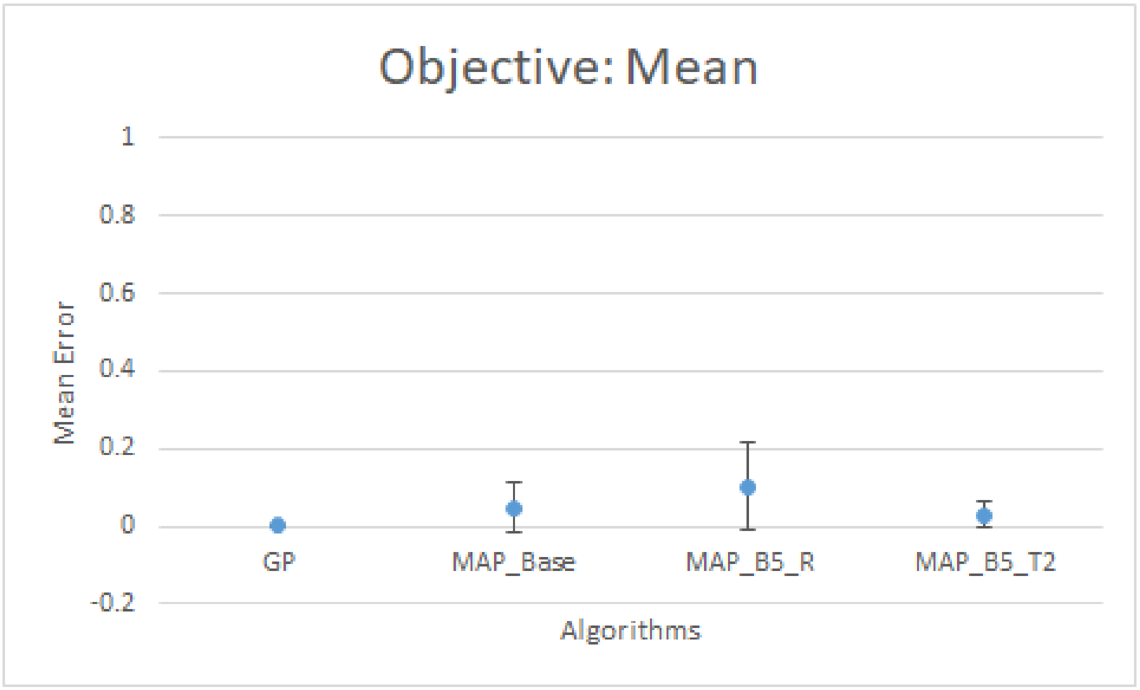


Figure 5.46: Error plot of mean mean pixel value difference for different algorithms

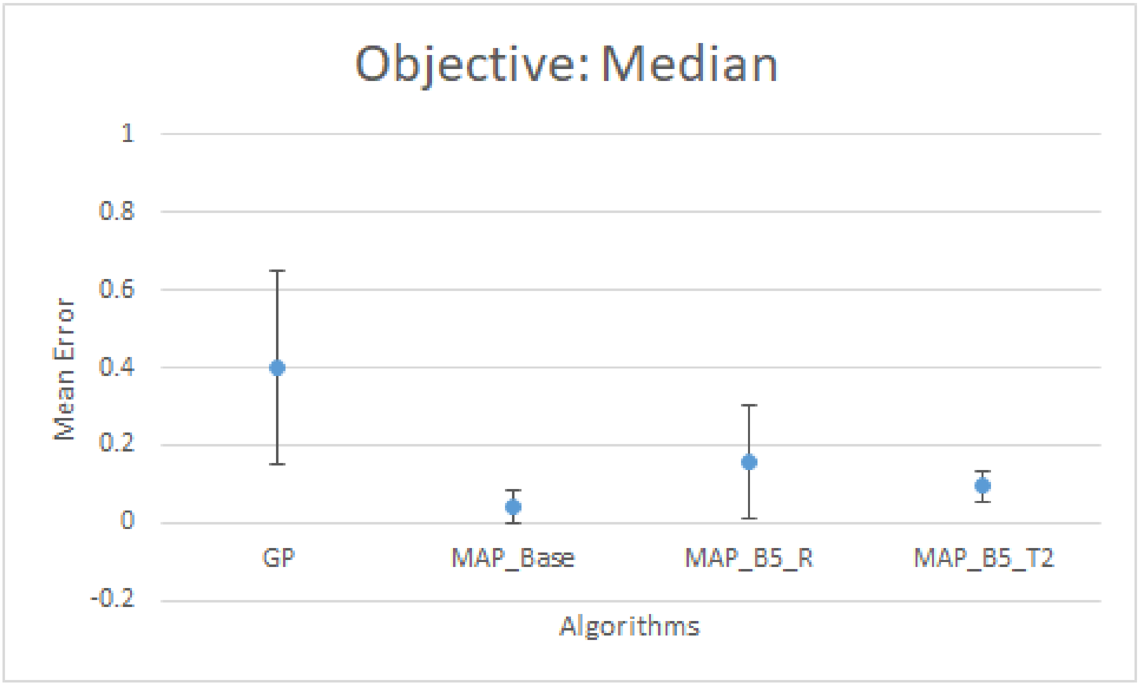


Figure 5.47: Error plot of mean median pixel value difference for different algorithms

When comparing base MAP-Elites with MAP_B5_T2 there is a significant statistical difference (threshold: $P<0.05$) in terms of max difference. Base MAP-Elites also shows significant statistical difference (threshold: $P<0.05$) when compared with the two proposed multiple individual bins MAP-Elites algorithms in terms of median, standard deviation and unique gray value. But this does not tell the whole story. To know which algorithm is performing better in which feature, we have to compare the table with error plots of respective objectives. We did not put the error plot of min as it does not have any statistical significance while comparing algorithms, therefore making it unnecessary for this analysis

Figures 5.45, 5.46, 5.47, 5.48 and 5.49 show the error plots with standard deviation for the four algorithms. These figures shows the values for five (min error plot is not shown here) different objectives (maximum, mean, median, standard deviation and edge, respectively). When we associate these figures with the P-value table (Table 5.9), we can see which algorithm performs better in terms of statistical values.

From Figure 5.45 we can see that GP and base MAP-Elites outperforms MAP_B5_T2 in terms of maximum pixel value difference. Figure 5.46 shows that GP out performs all three MAP-Elites algorithm in terms of mean pixel value difference. Figure 5.47 shows that all three MAP-Elites beat GP and base MAP-Elites beats the proposed two version of MAP-Elites in terms of performance for median value difference. From Figure 5.48 we can see that base MAP-Elites outperforms the other three algorithms and the proposed two MAP-Elites algorithms outperform GP in standard deviation value difference. Finally, Figure 5.49 shows that the base MAP-Elites outperforms the other three algorithms in terms of edge count difference.

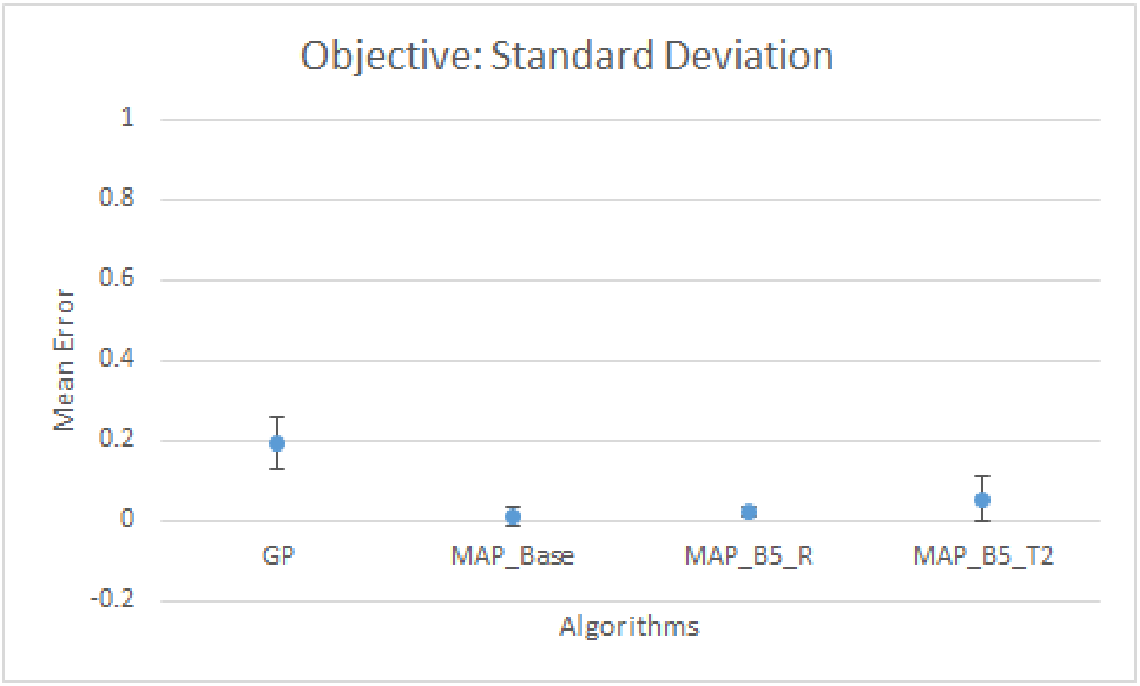


Figure 5.48: Error plot of mean standard deviation difference for different algorithms

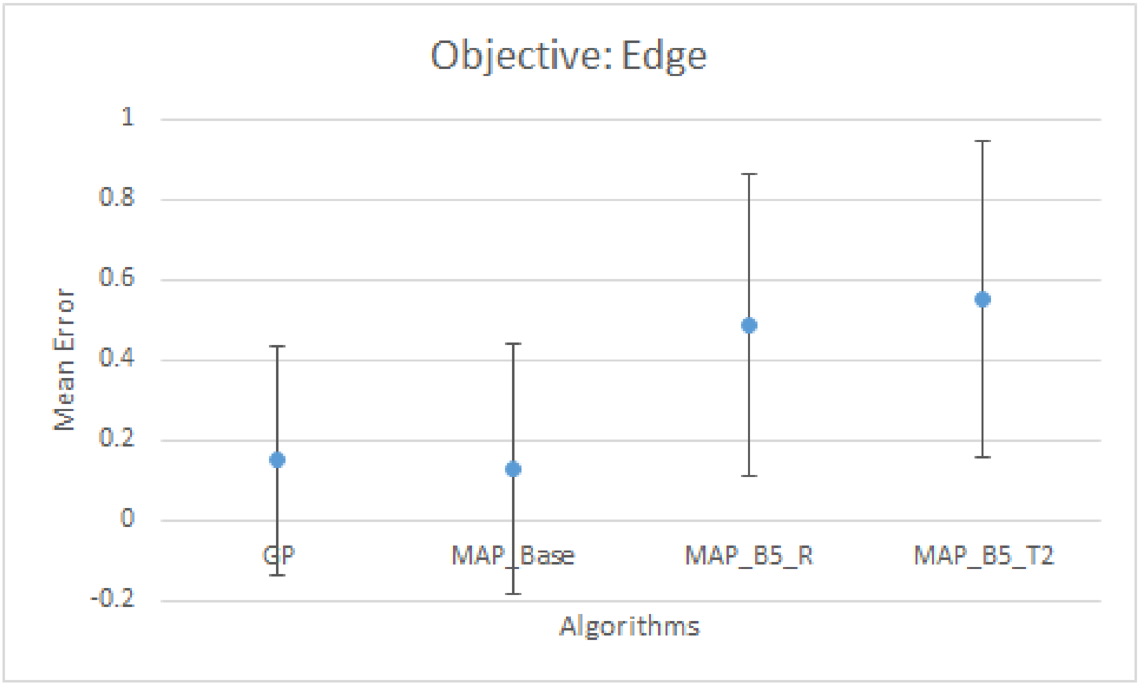


Figure 5.49: Error plot of edge count difference value for different algorithms

Chapter 6

Experiments and Results: RGB Image

In this chapter RGB image experiments are presented and analyzed. RGB images have three colour channels which, when separated, behave similarly to grayscale image. Also, RGB images have many more interesting features compared with grayscale images. Therefore, they are much more challenging to deal with, and shows interesting behaviour. We will examine two RGB image experiments, and compare the performance of the four algorithms used.

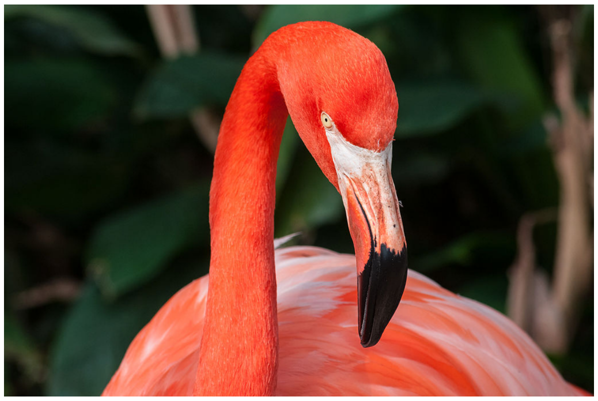
Tables 6.1 and 6.2 show the GP and MAP-Elites parameters for RGB image experiments. When we compare Table 6.1 to Table 5.1, we can see most of the parameters are the same. The only difference here is the number of generations, which is changed from 50 to 30. This change is made considering two factors. The first one is time and resources. RGB image evolution takes much longer time and consumes more resources than evolving grayscale images. The second factor is, within the first 30 generations, GP makes most of the progress. Considering the minor improvements after 50 generations, and the time and resources required, 30 generations is sensible and adequate. When we compare Table 6.2 to Table 5.2, we can see a similar trend. Number of iterations is the only parameter that has been changed from 1000 to 600. Making this change assures that the total number of individuals evaluated by GP and MAP-Elites algorithms remains the same.

Table 6.1: GP parameter for grayscale experiment

Parameter Name	**Value**
Population Size	300
No. of Generation	30
Tree Initializer	Half and Half
Initial Tree Depth	Min: 2 Max: 7
Max Tree Depth	17
Crossover	100%
Mutation	20%
Selection	Tournament
Tournament Size	3
Number of Runs	10 / 10

Table 6.2: MAP-Elites parameter for grayscale experiment

Parameter Name	**Value**
MAP size	8×8
Initial batch size	30
Batch size	15
No. of Iteration	600
No. of features	2
Bin size	1 / 5
Evaluating algo.	GP
Tree Initializer	Half and Half
Initial Tree Depth	Min: 2 Max: 7
Max Tree Depth	17
Crossover	100%
Mutation	20%
Selection	Random / Tournament
Tournament Size	2
Number of Runs	10 / 10


(a) Target colour image-1 (Phoenicopterus ruber in São Paulo Zoo [11])

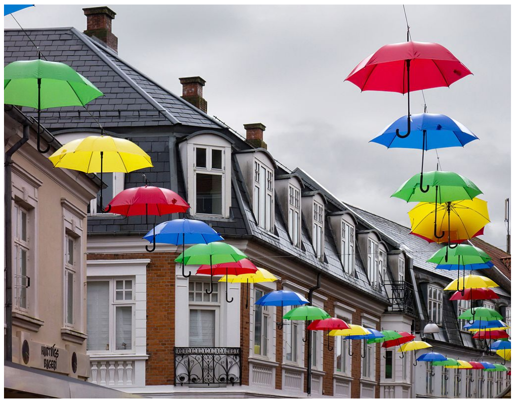

(b) Target colour image-2 (Les Parapluies de Viborg [9])

Figure 6.1: RGB target images

Table 6.3: RGB (Colour) Image Features

Feature Name	**Target Colour Image -1**	**Target Colour Image -2**
Size	1024×680	1024×794
Red Min	0.000	0.000
Red Max	1.000	1.000
Red Mean	0.373	0.558
Red Median	0.133	0.600
Red Std. Dev.	0.386	0.266
Green Min	0.000	0.000
Green Max	1.000	1.000
Green Mean	0.230	0.534
Green Median	0.184	0.557
Green Std. Dev.	0.177	0.276
Blue Min	0.000	0.000
Blue Max	1.000	1.000
Blue Mean	0.169	0.522
Blue Median	0.114	0.522
Blue Std. Dev.	0.172	0.289
Entropy	7.263	7.739
Unique RGB	92629	141438

Table 6.4: RGB feature set

Feature Name	**Type**	**Description**
Red Max	Canvas	The maximum value in the Red channel.
Red Min	Canvas	The minimum value in the Red channel.
Red Mean	Canvas	The arithmetic mean of the values in the Red channel.
Red Median	Canvas	The median of the values in the Red channel.
Red Std. Dev.	Canvas	The standard deviation of the values in the Red channel.
Green Max	Canvas	The maximum value in the Green channel.
Green Min	Canvas	The minimum value in the Green channel.
Green Mean	Canvas	The arithmetic mean of the values in the Green channel.
Green Median	Canvas	The median of the values in the Green channel.
Green Std. Dev.	Canvas	The standard deviation of the values in the Green channel.
Blue Max	Canvas	The maximum value in the Blue channel.
Blue Min	Canvas	The minimum value in the Blue channel.
Blue Mean	Canvas	The arithmetic mean of the values in the Blue channel.
Blue Median	Canvas	The median of the values in the Blue channel.
Blue Std. Dev.	Canvas	The standard deviation of the values in the Blue channel.

For the two RGB experiments we have used two different images (see Figure 6.1). Both images are public domain images from Wikimedia commons. The first image is a picture of an American flamingo called “Phoenicopterus ruber in São Paulo Zoo” [11] and second image is a picture of hanging umbrellas called “Les Parapluies de Viborg” [9]. The two images were chosen because of their different feature values (see Table 6.3). They have different mean, median and standard deviation values for all three channels. Therefore, their objectives will impact the evolved images.

6.1 Experiment 4

Table 6.4 shows the RGB feature set that has been used for this experiment. The first target image is used (see Figure 6.1 (a)). 10 different runs are performed for each of the four algorithms to generate solutions. Each evolved image here is 256 × 256 pixel. Even though the total RGB colour spectrum has over 16 millions unique RGB colour, because of our evolved image size, the maximum possible unique RGB colour can be 65536 (total number of pixels in the evolved images). All the pixel values are mapped between 0 and 1 for all three RGB channels.

6.1.1 Basic Genetic Programming (GP)

Table 6.1 shows the GP parameters used. The GP population size is set to 300 and the total number of generations is set to 30. Sum-of-ranks is used for fitness evaluation.

Figure 6.2 shows the 10 solutions generated by GP. The left most column shows the target image and the right two columns show the best 10 evolved images from each run. These images represents the best evolved images with closest features with the target image.

Figure 6.3 shows the mean fitness of the GP population averaged over 10 runs for all 15 objectives. The X-axis represents number of generations and the Y-axis represents mean error. We can see that GP behaviour remains similar to the grayscale experiments for mean error (see Figure 5.3). It shows that most of the features improves over time and then they remain almost flat, meaning that the population is converging.

Figure 6.4 shows the mean fitness of the best GP individual per generation over 30 generation for 10 runs. We can see that the best fitness plot only shows 9 objectives instead of 15. This is because min and max error difference values of all three RGB channel is zero for the best individuals, and they do not change over time. So they do not show any change over time, and remain flat on the X-axis. Table 6.3 shows that the min and max value for all three channel for the first image is 0 and 1 respectively. It is very likely for a moderate size RGB image to have at least one pixel with 0 and one pixel with 1 value. So, the min and max difference becomes zero for the best solution of a GP population. We can see from the fitness plots of

GP that most features get better while some features get worse over time. This is because sum of ranks sacrifices some of the objectives to improve most others.

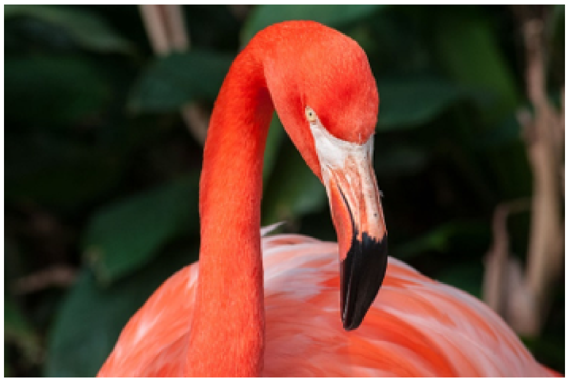
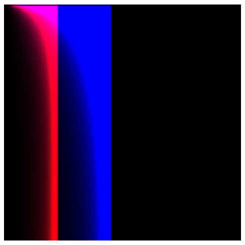
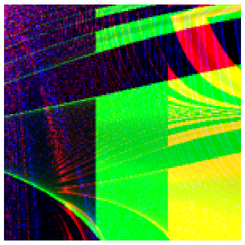
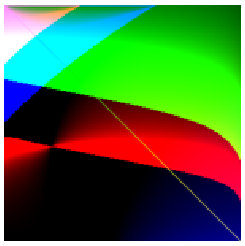
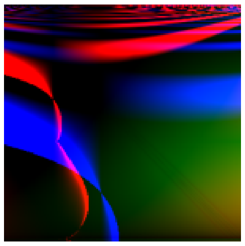
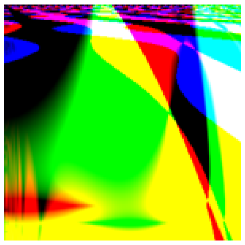
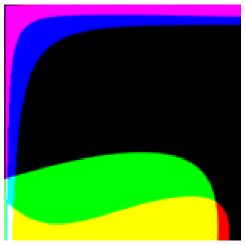
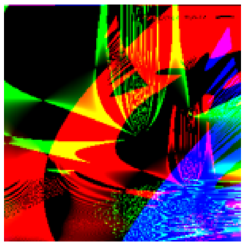
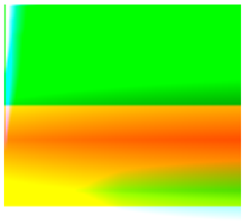
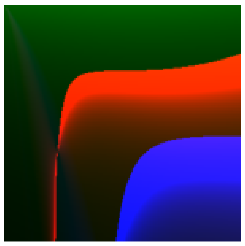
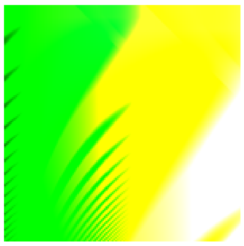
Target Image	Run: 1 - 5	Run: 6 - 10
		
		
		
		
		

Figure 6.2: Best result of each GP run for RGB target image- 1 and RGB feature set (10 runs).

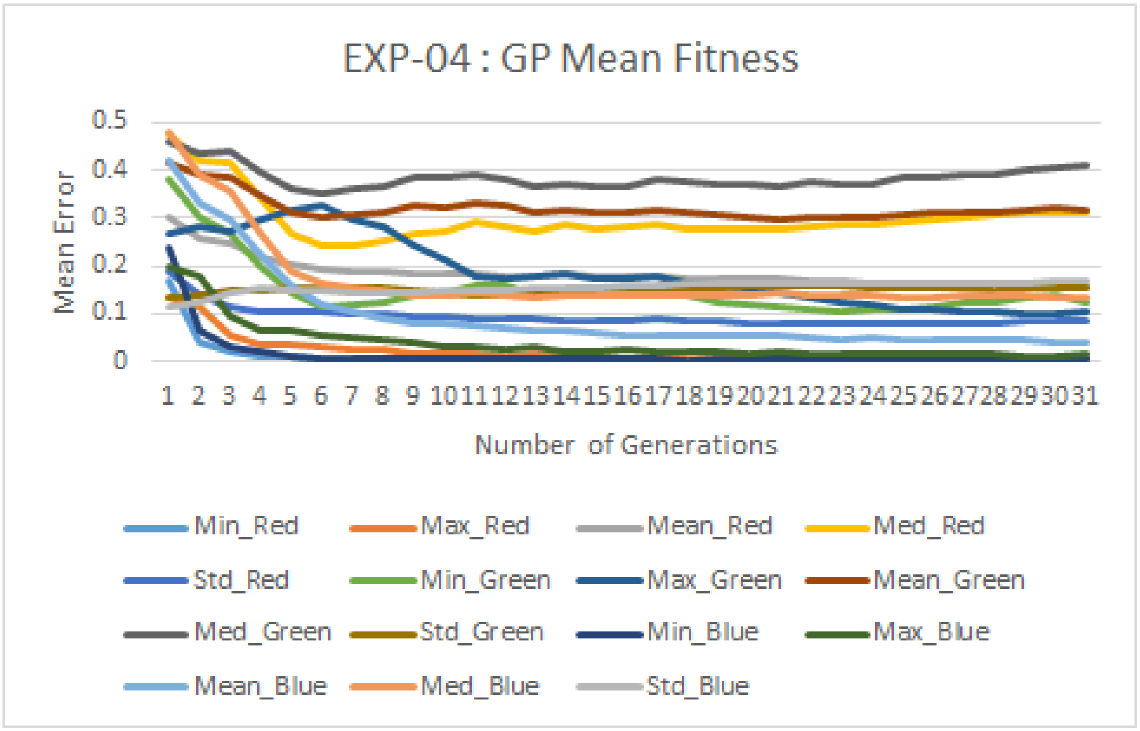


Figure 6.3: Average mean fitness of GP over 10 runs (lower is better).

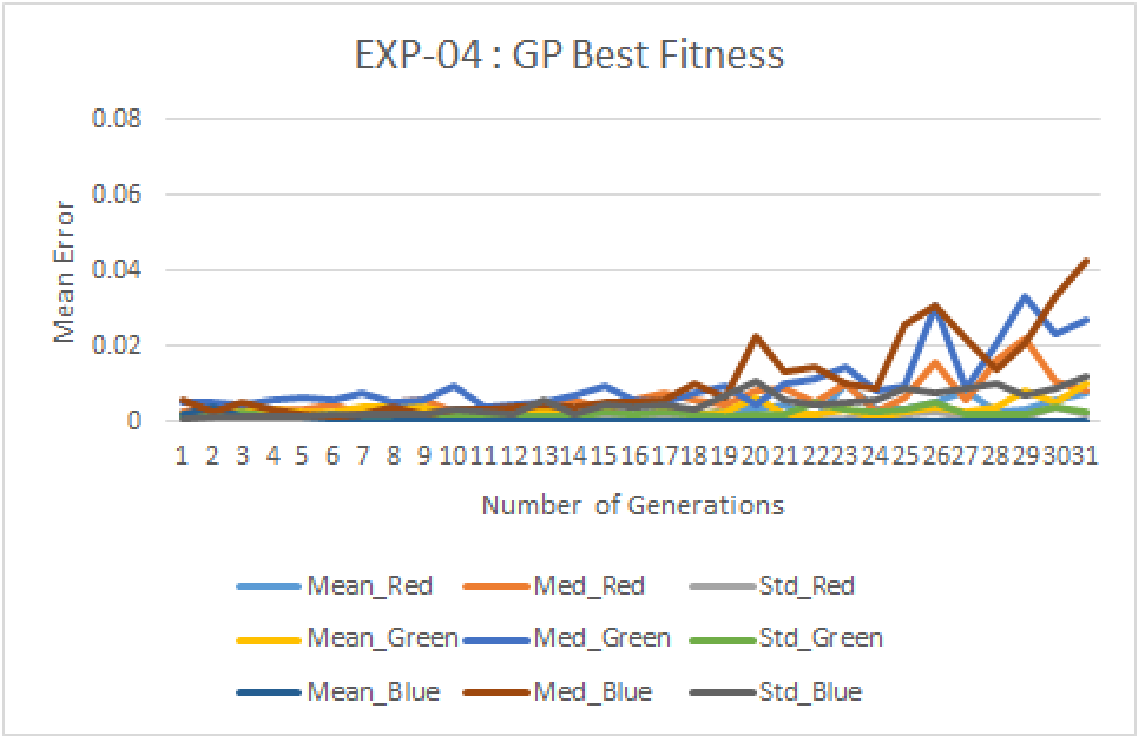


Figure 6.4: Average best fitness of GP over 10 runs (lower is better).

6.1.2 MAP-Elites: Single Individual Bin (MAP_Base)

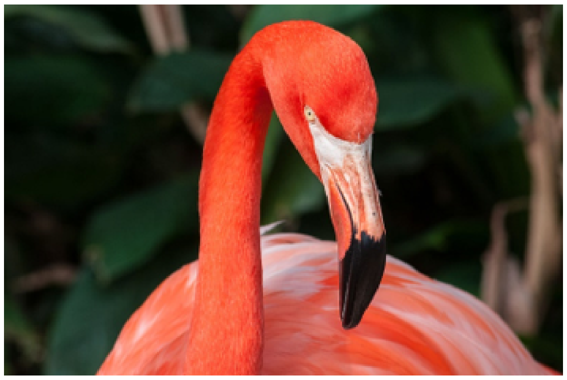
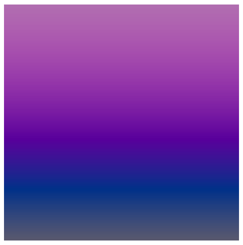
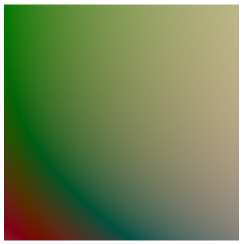
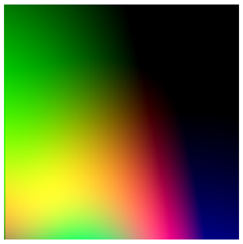
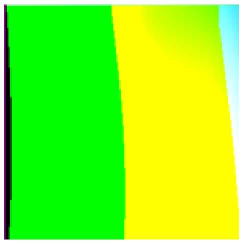
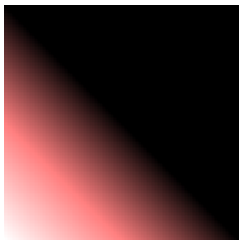
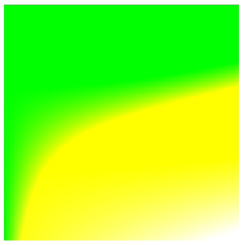
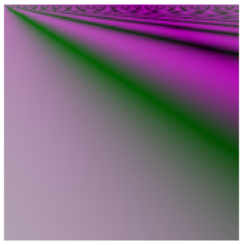
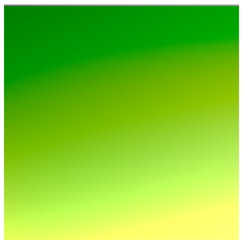
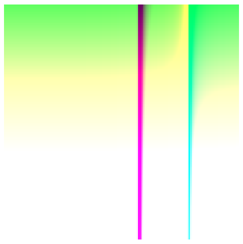
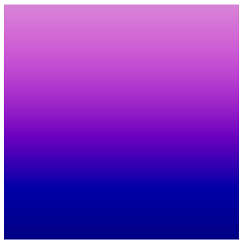
Target Image	Run: 1 - 5	Run: 6 - 10
		
		
		
		
		

Figure 6.5: Best results of base MAP-Elites runs for RGB target image- 1 and RGB feature set (10 runs).

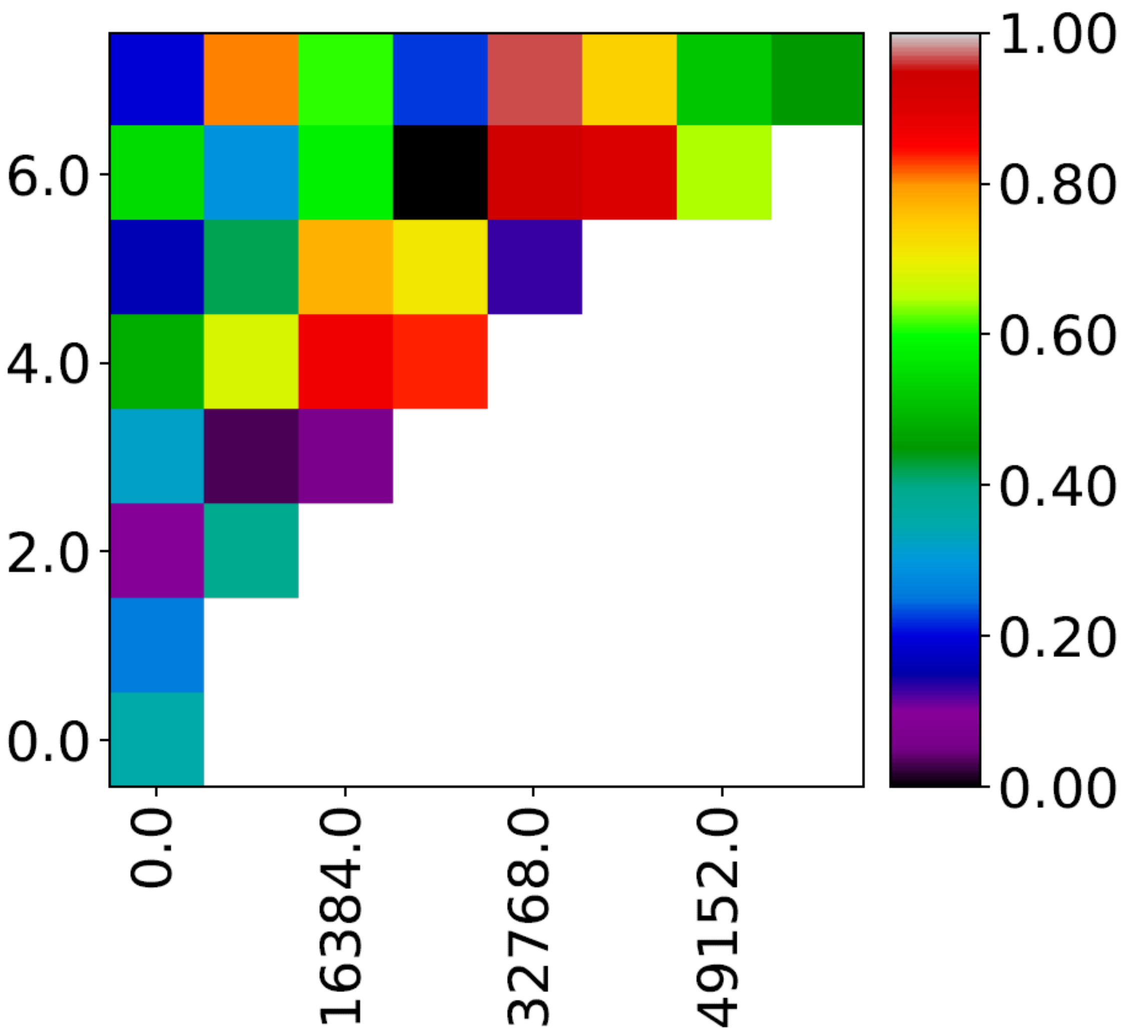
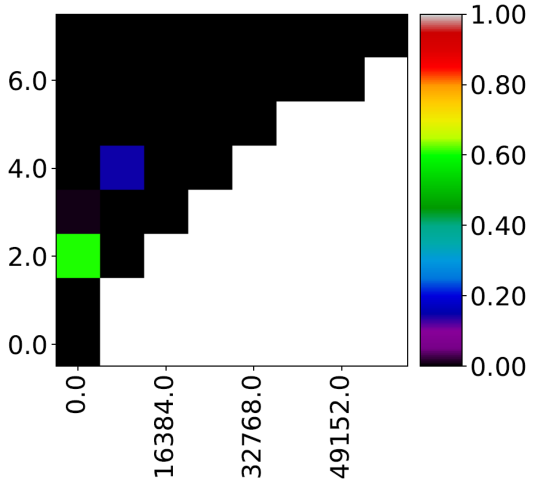


Figure 6.6: Sum-of-ranks MAP of run-6 of experiment 4 (higher is better) [X-axis: unique RGB value (0-65536); Y-axis: entropy (0-8)].

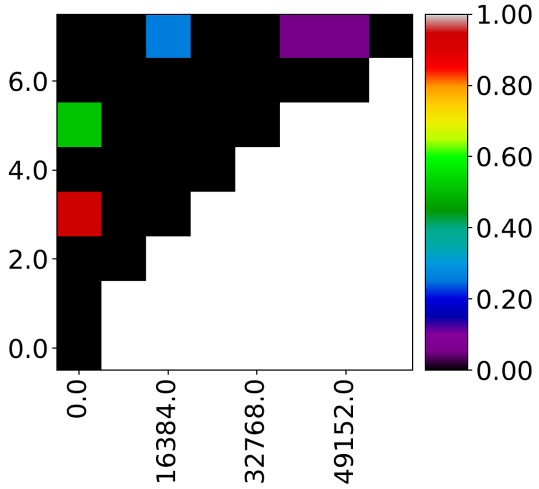
Figure 6.5 shows the best solutions generated by ten MAP-Elites runs. MAP-Elites generated hundreds of solutions. Only the best one from each base MAP-Elites is shown here. To find the best solution per bin all the individuals in the MAP are ranked at the end of final iteration and the best ranked individuals from each bin are then re-ranked with respect to each other. Then their rank is normalized and inversed to make the best ranked individual have a rank of 1 and the worst ranked individual to have a rank of 0. Other individuals ranks are between 1 and 0.

Figure 6.6 shows the final sum-of-ranks MAP for run-6 of experiment 4. X-axis represents the unique RGB values and Y-axis represents the entropy of the solution images. We can see that the range for unique RGB is significantly lower than the total number of unique RGB values (over 16 million) due to the small solution image size.

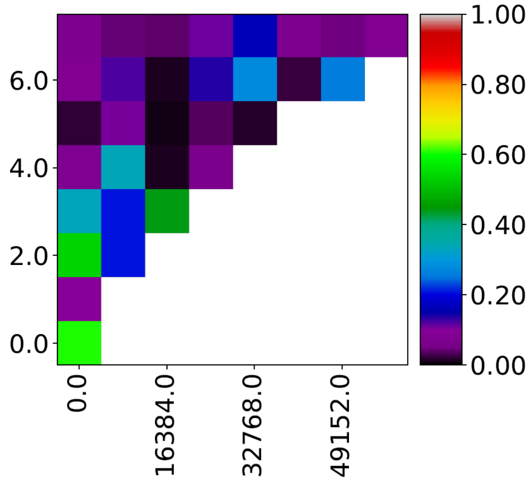
Figure 6.7, 6.8 and 6.9 shows the raw objective values of 15 objectives. Figure 6.7 shows the raw values for the five objectives of red channel, Figure 6.8 shows the raw values for green channel, and Figure 6.9 shows the raw values blue channel. Lower values for an objective of a bin means that, for that solution, the difference between the target image and evolved image for that feature is lower. We can see that the min and max value for all three channels shows the lowest difference between target and evolved images, where the difference values for other objectives are large. This is because the target image had a min pixel value of 0 and max pixel value of 1. So, even if one pixel value of a solution space is 0 the min difference will be zero, similarly if just one pixel value of a solution image is 1, then the max pixel value difference will be zero.



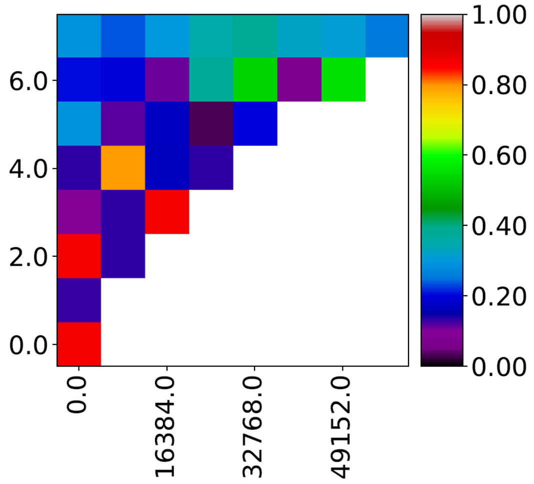
(a) Min red difference



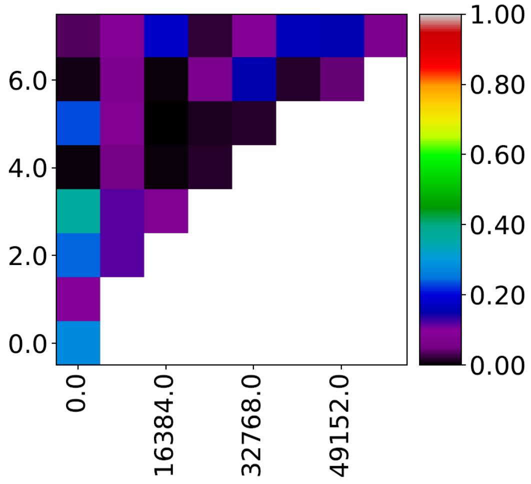
(b) Max red difference



(c) Mean red difference

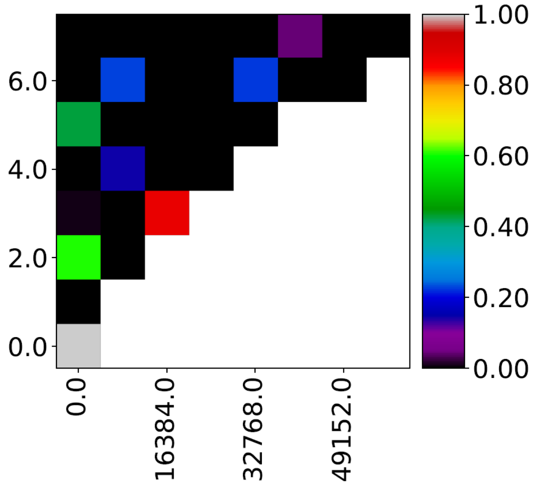


(d) Median red difference

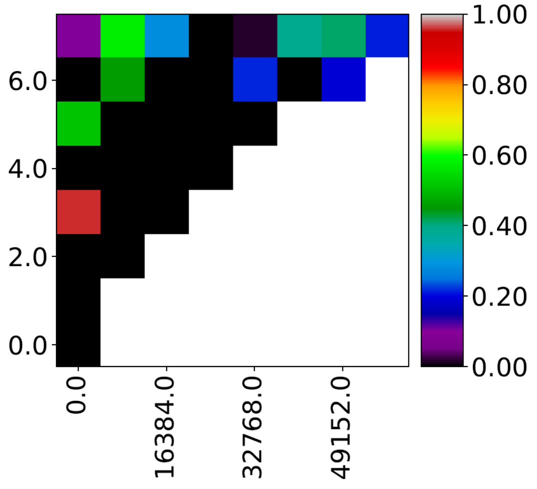


(e) Std. Dev. red difference

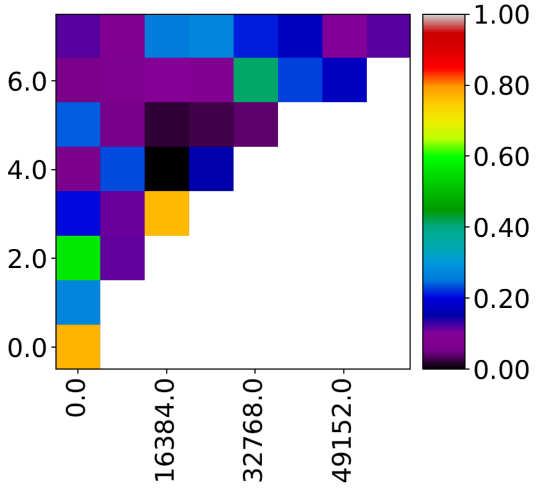
Figure 6.7: Raw feature difference of red channel of run-6 of experiment 4 (lower is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)]



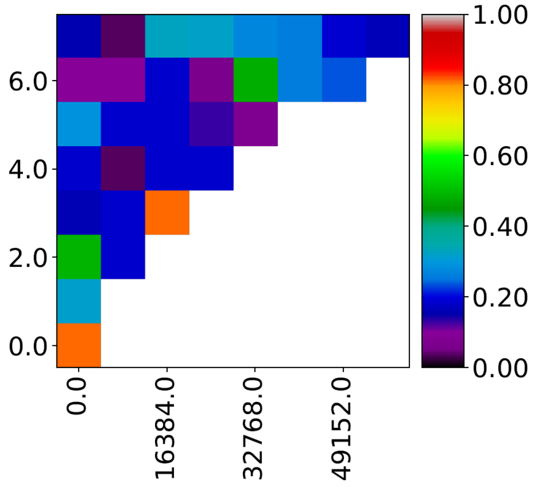
(a) Min green difference



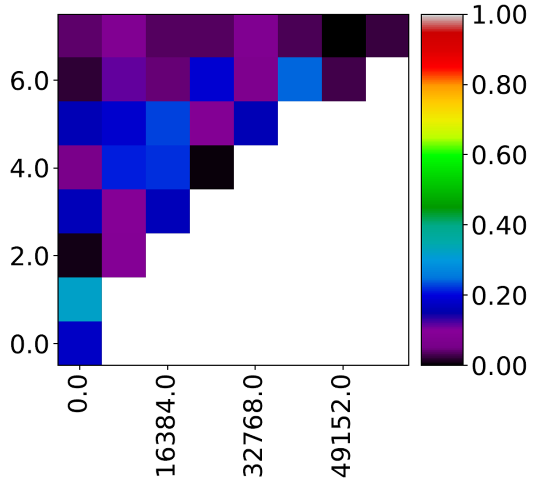
(b) Max green difference



(c) Mean green difference

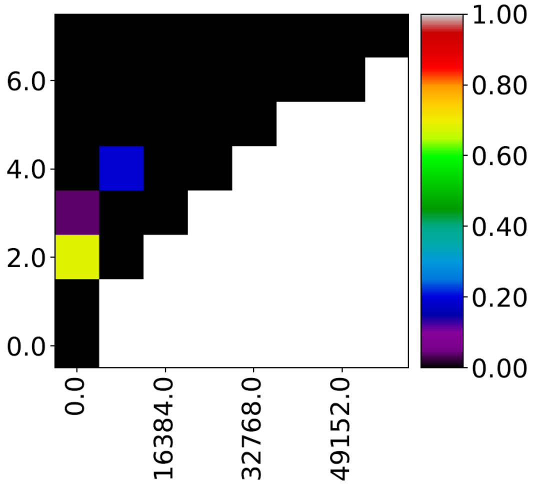


(d) Median green difference

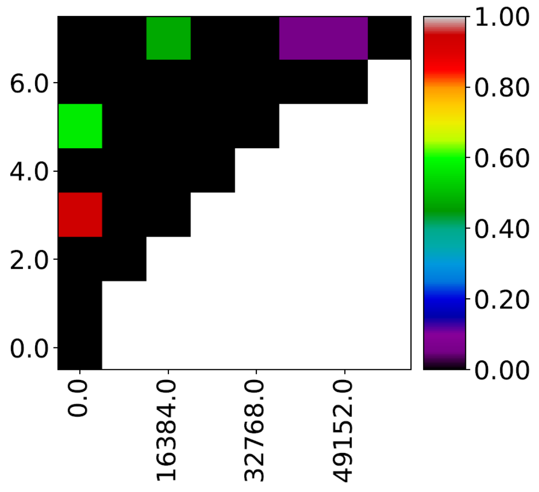


(e) Std. Dev. green difference

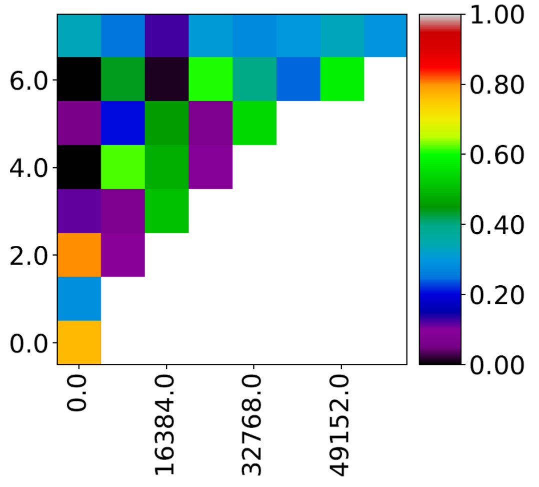
Figure 6.8: Raw feature difference of green channel of run-6 of experiment 4 (lower is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)]



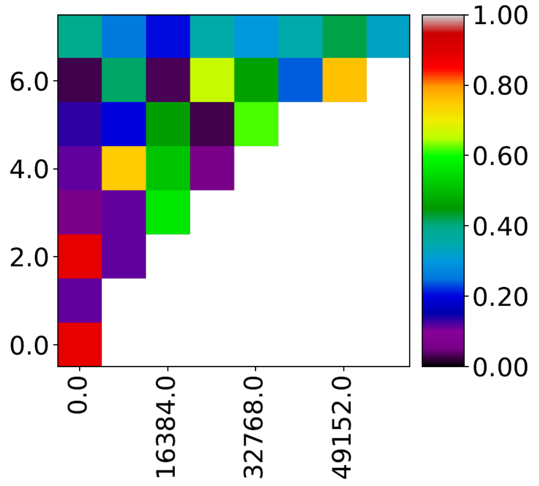
(a) Min blue difference



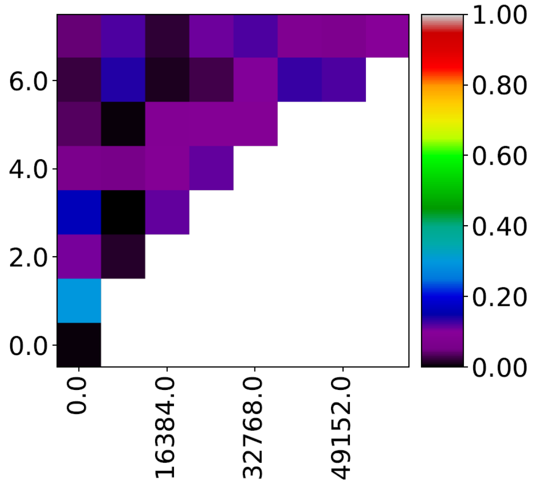
(b) Max blue difference



(c) Mean blue difference



(d) Median blue difference



(e) Std. Dev. blue difference

Figure 6.9: Raw feature difference of blue channel of run-6 of experiment 4 (lower is better) [X-axis: unique gray value (0-255); Y-axis: entropy (0-8)]

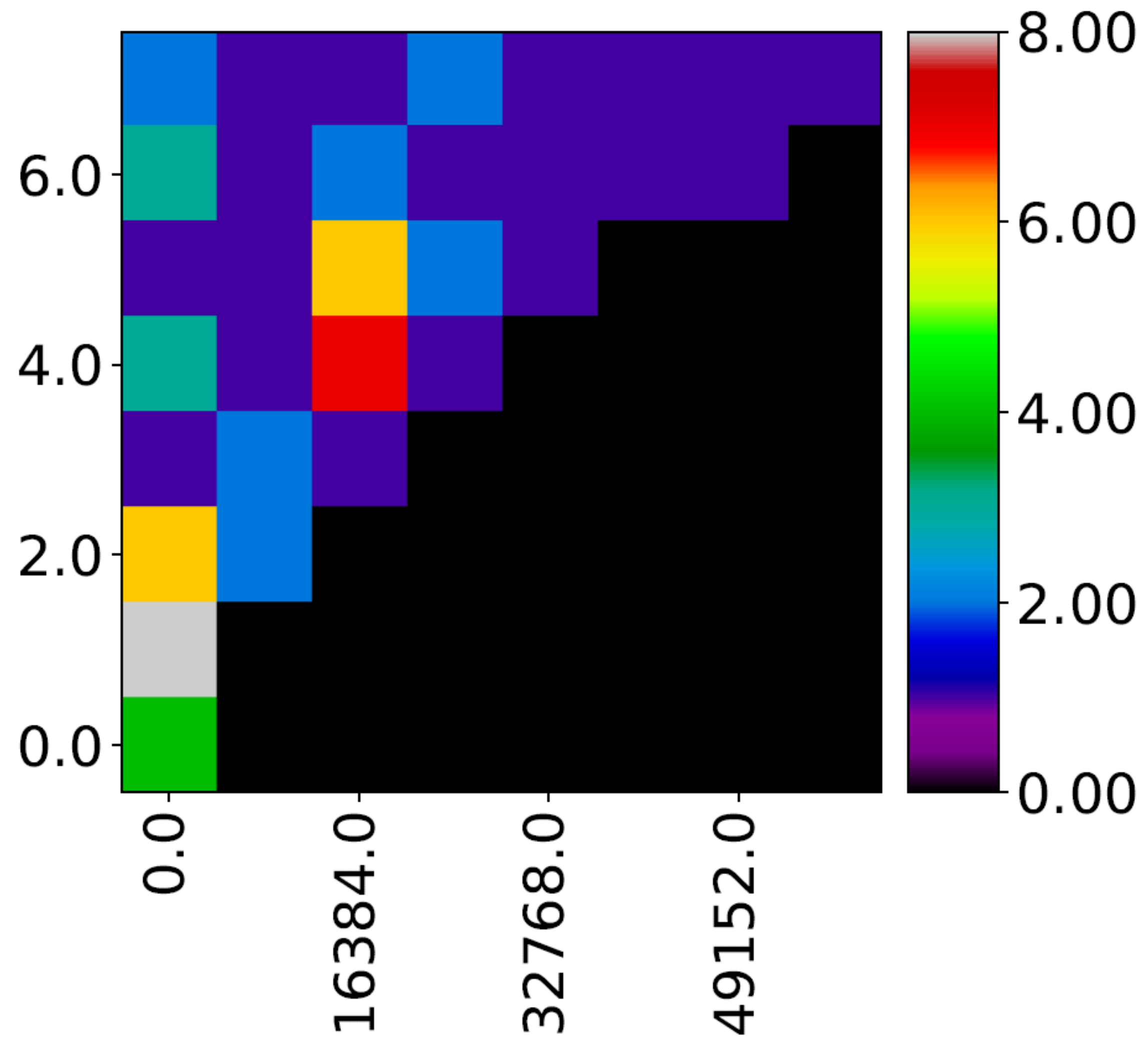


Figure 6.10: Activity MAP of run-6 of experiment 4 (higher is better) [X-axis: unique RGB value (0-65536); Y-axis: entropy (0-8)].

Figure 6.10 Shows the activity MAP of run-6 of experiment 4. The X-axis represents unique RGB value and the Y-axis represents entropy. We can see that most of the bins have been updated only once, meaning that after those bins have been filled ones the new evolved solutions either had poor fitness than the existing solutions of those bins or they did not have the similar MAP feature behaviour, resulting them competing for other bins. Only a very few bins have been updated multiple times. Black bins represents no activity at all, meaning no solutions were found for those feature space as very high unique colour combination and low entropy is very unlikely. When we compare Figure 6.10 to Figure 6.6 we can see the most updated bin in this run has the best fitness among all the MAP solutions for this run (run-6).

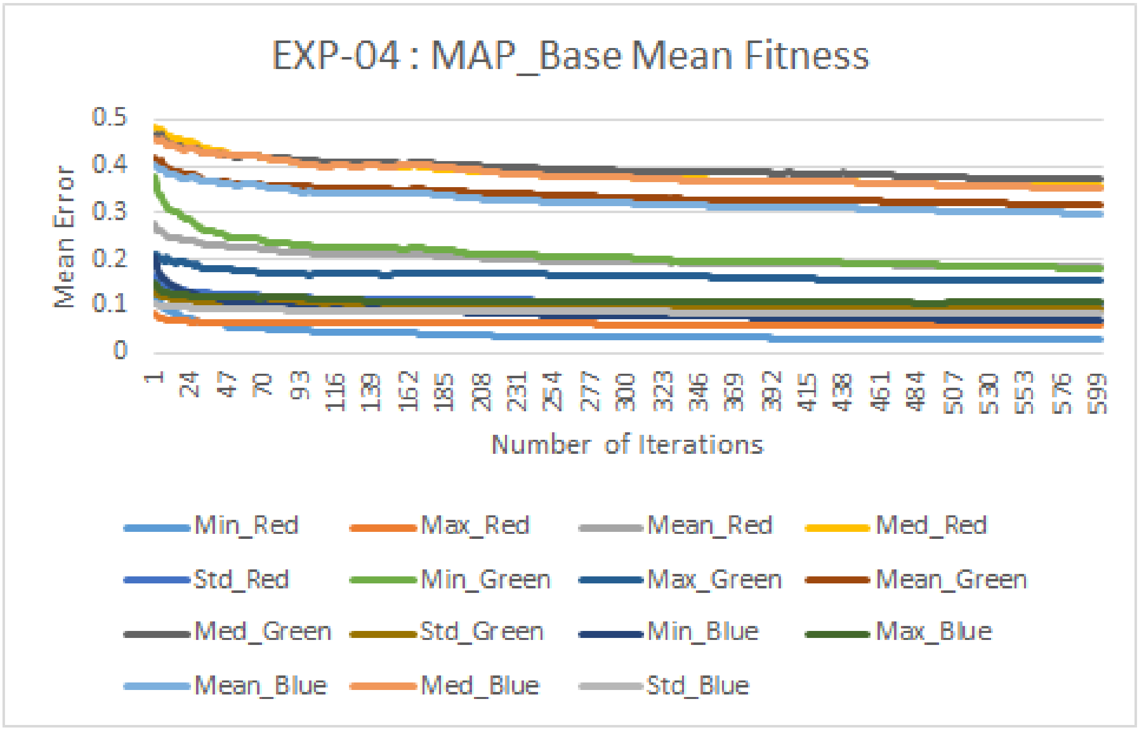


Figure 6.11: Average mean fitness of base MAP-Elites MAP over 10 runs (lower is better).

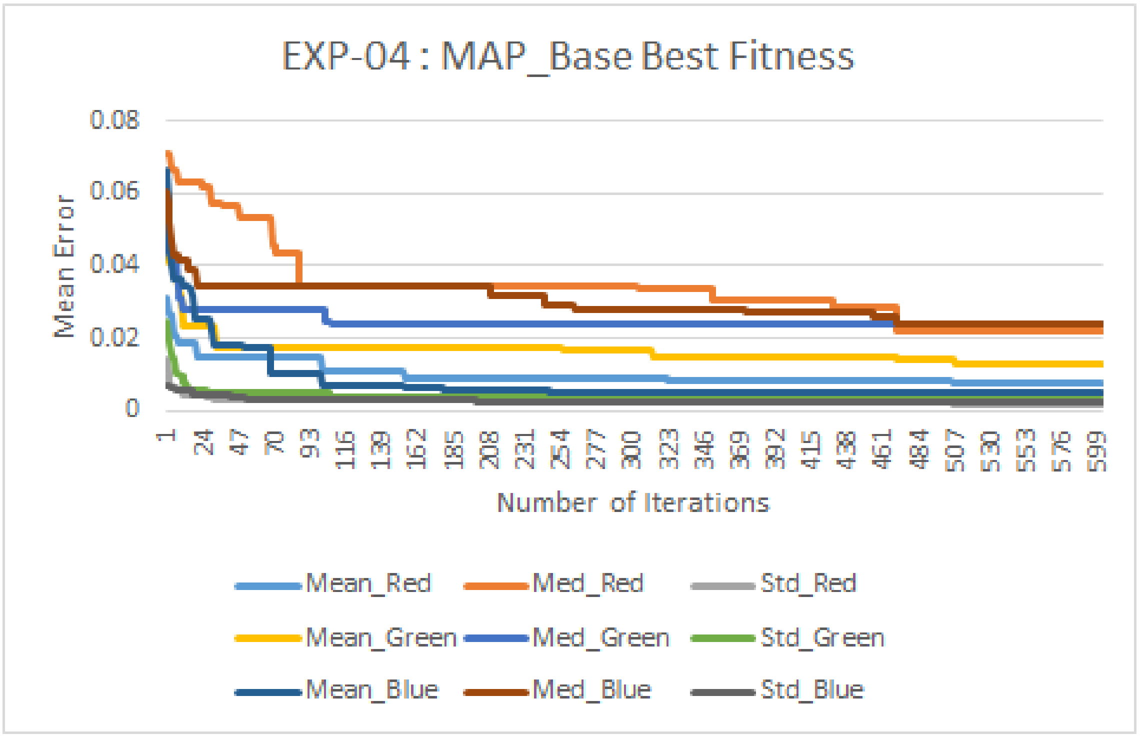


Figure 6.12: Average best fitness of base MAP-Elites MAP over 10 runs (lower is better).

Figure 6.11 shows the mean fitness of the base MAP-Elites MAP averaged over 10 runs for all 15 objectives. The X-axis represents number of Generations and the Y-axis represents mean error. We can see that compared with GP, base MAP-Elites has better fitness is some objectives and similar or worse fitness in a few. The solution space values do not change much as it takes the whole MAP of diverse solutions into account, and solutions diversify more over time. Figure 6.12 shows the best fitness of the base MAP-Elites MAP over 600 iterations.

6.1.3 MAP-Elites: Multiple individuals Bin - Random Selection (MAP_B5_R)

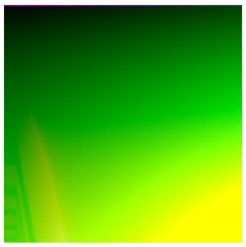
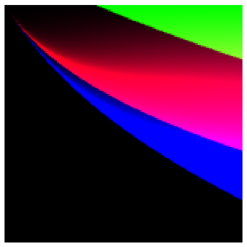
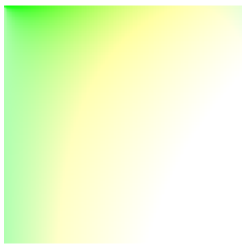
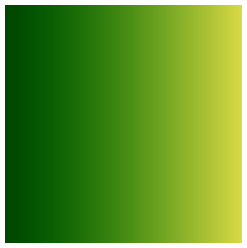
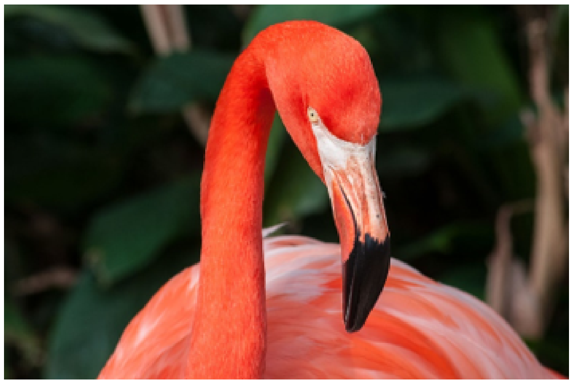
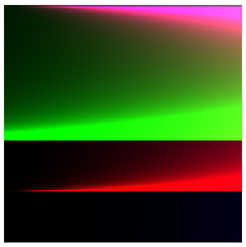
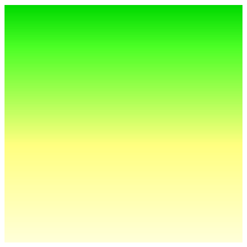
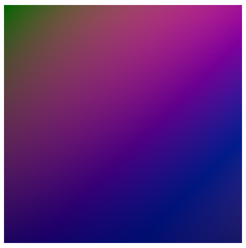
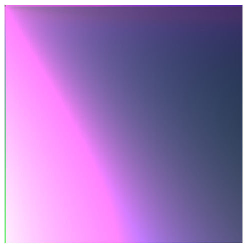
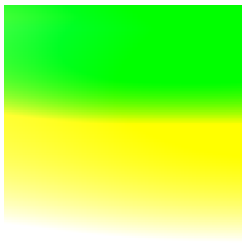
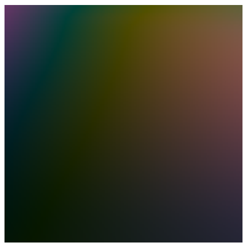
Target Image	Run: 1 - 5	Run: 6 - 10
		
		
		
		
		

Figure 6.13: Best results of base MAP-Elites runs for RGB target image -1 and RGB feature set (10 runs).

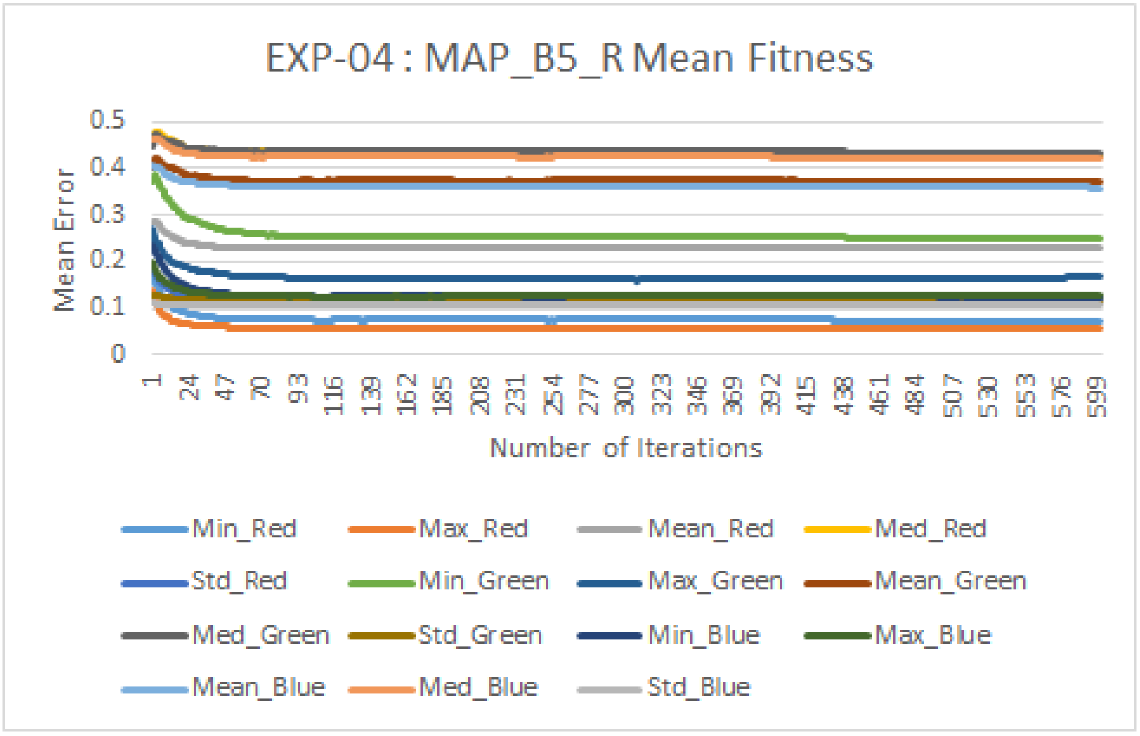


Figure 6.14: Average mean fitness of MAP-Elites (multiple individuals bin - random selection) MAP over 10 runs (lower is better).

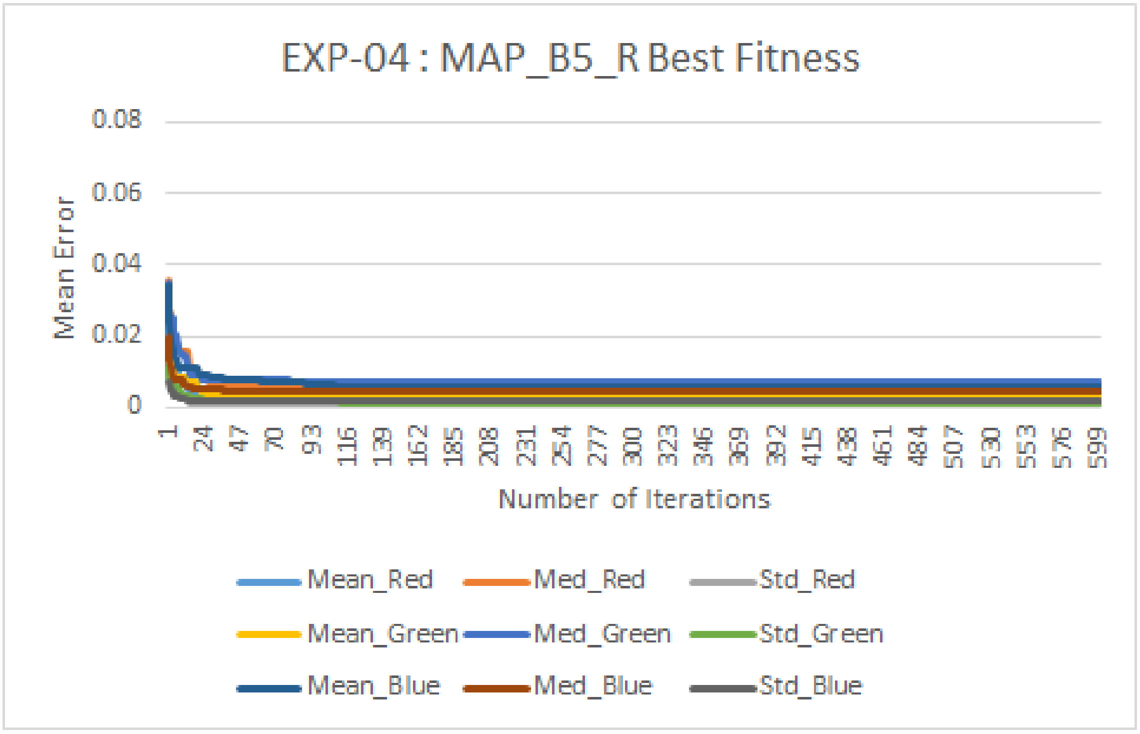


Figure 6.15: Average best fitness of MAP-Elites (multiple individuals bin - random selection) MAP over 10 runs (lower is better).

This experiment has the same parameters (see Table 6.2) as the previous experiment. The only difference is the bin size, which is set to 5.

Figure 6.13 shows the best results from 10 MAP_B5_R runs. These are the best ranked individuals from each final MAP of the runs, meaning they have feature values close to the target image. Figure 6.14 shows the mean fitness of the MAP_B5_R MAP averaged over 10 runs for all 15 objectives. The X-axis represents number of generations and the Y-axis represents mean error. We can see that compared with base MAP-Elites, MAP_B5_R shows similar mean fitness in terms of mean error. More information can be gained from Figure 6.15, which shows the best fitness of the MAP_B5_R MAP over 600 iterations. Features show that for the best individual of the MAP MAP_B5_R shows better mean error than base MAP-Elites from a very early iterations (see Figure 6.12).

6.1.4 MAP-Elites: Multiple individuals Bin - Tournament Selection (MAP_B5_T2)

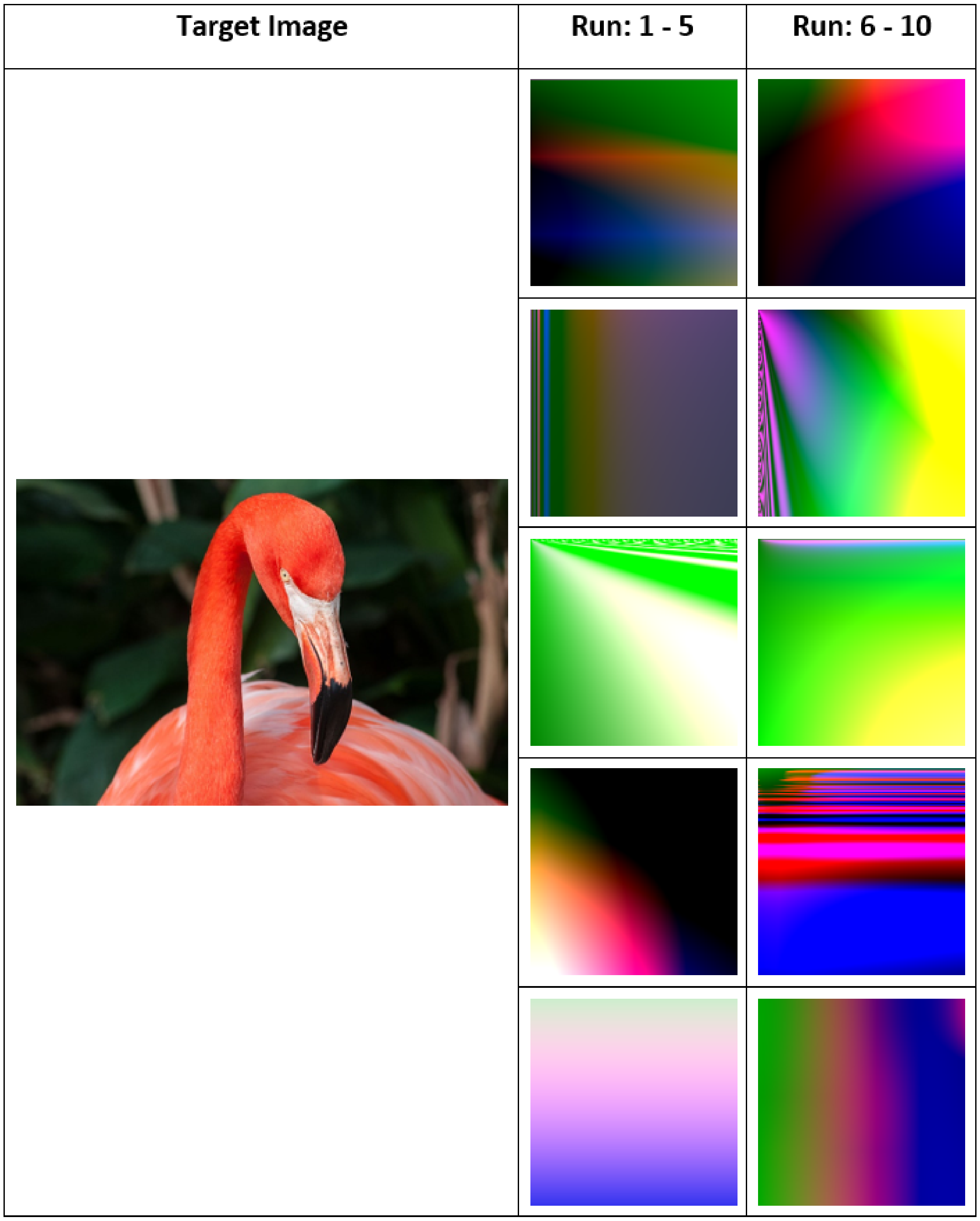


Figure 6.16: Best results of base MAP-Elites runs for RGB target image- 1 and RGB feature set (10 runs).

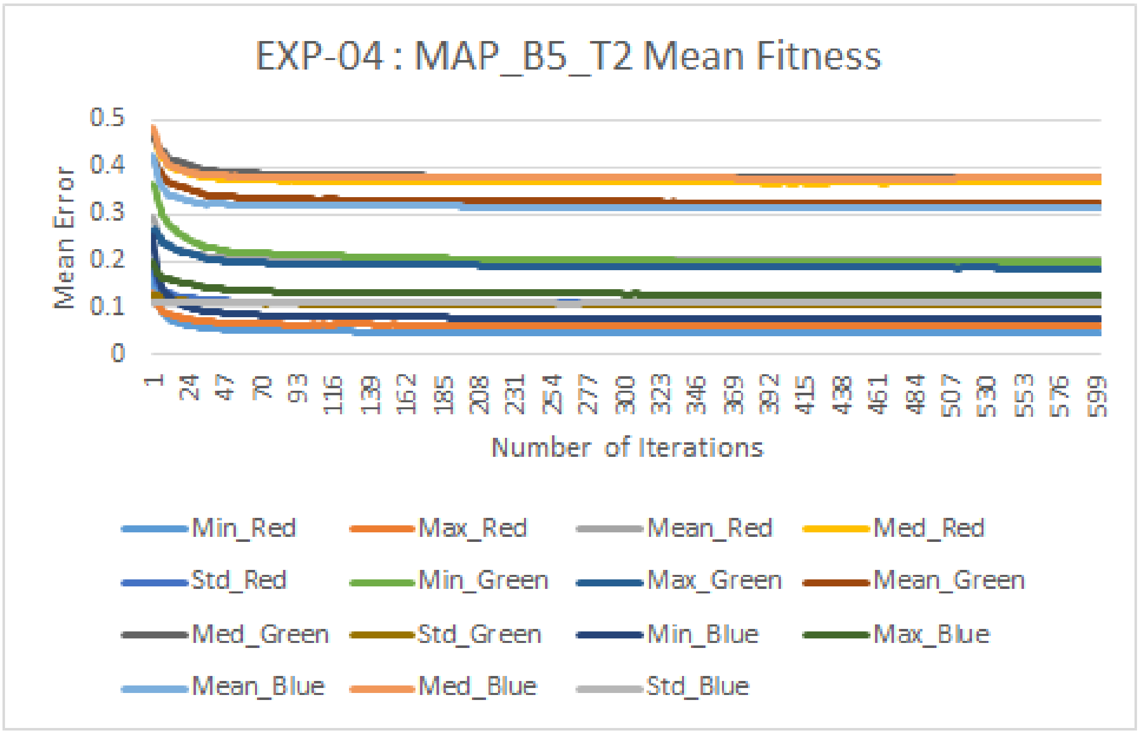


Figure 6.17: Average mean fitness of MAP-Elites (multiple individuals bin - tournament selection) MAP over 10 runs (lower is better).

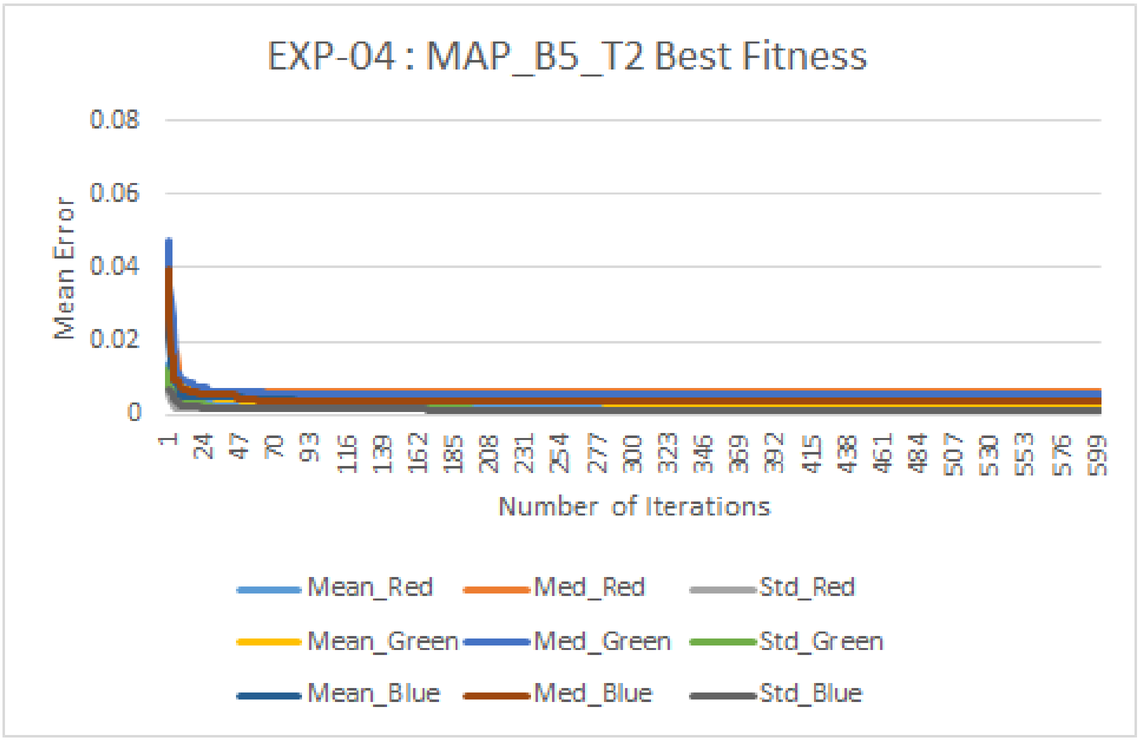


Figure 6.18: Average best fitness of MAP-Elites (multiple individuals bin - tournament selection) MAP over 10 runs (lower is better).

This experiment has same parameters (see Table 6.2) as earlier. The only difference is the selection method. The bin size is 5 and a tournament of size 2 is used for the selection of parents for reproduction.

Figure 6.16 shows the best results from 10 MAP_B5_T2 runs. Figure 6.17 shows the mean fitness of the base MAP-Elites MAP averaged over 10 runs for all 15 objectives. The X-axis represents number of Generations and the Y-axis represents mean error. We can see that compared with MAP_B5_R, MAP_B5_T2 shows a slight better mean fitness in terms of mean error. More information can be gained from Figure 6.18, which shows the best fitness of the MAP_B5_T2 MAP over 600 iterations.

6.1.5 Analysis of experiment 4

Table 6.5: P-value table for experiment 4

	Objectives	**MAP_Base**	**MAP_B5_R**	**MAP_B5_T2**
GP	**Red Min**	1.000000	1.000000	0.184060
	Red Max	0.007433 ←	0.038936 ←	0.017492 ←
	Red Mean	0.080986	0.338792	0.213678
	Red Med	0.054784	0.211570	0.271072
	Red Std	0.236338	0.172352	0.484925
	Green Min	0.401924	0.044929 ←	0.221276
	Green Max	0.398990	0.447078	0.215909
	Green Mean	0.366865	0.311588	0.311588
	Green Med	0.321597	0.408417	0.453519
	Green Std	0.014153 ↑	0.048089 ↑	0.022577 ↑
	Blue Min	0.038936 ←	1.000000	0.184060
	Blue Max	0.007433 ←	0.007466 ←	0.007466 ←
	Blue Mean	9.13E-05 ←	9.13E-05 ←	9.13E-05 ←
	Blue Med	0.022982 ←	0.483499	0.285907
	Blue Std	9.13E-05 ↑	0.000291 ↑	0.008629 ↑
MAP_Base	**Red Min**	—	1.000000	0.184060
	Red Max	—	0.304265	0.401889
	Red Mean	—	0.037831 ←	0.395668
	Red Med	—	0.298005	0.056138
	Red Std	—	0.311588	0.285375
	Green Min	—	0.041066 ←	0.175839
	Green Max	—	0.418058	0.250268
	Green Mean	—	0.410265	0.236338
	Green Med	—	0.424031	0.365347
	Green Std	—	0.086728	0.092938
	Blue Min	—	0.038936	0.193506
	Blue Max	—	0.404231	0.328301
	Blue Mean	—	0.425053	0.338792
	Blue Med	—	0.133325	0.258678
	Blue Std	—	0.032011 ←	0.106147

Table 6.5 continued from previous page

	Objectives	**MAP_Base**	**MAP_B5_R**	**MAP_B5_T2**
MAP_B5_R	**Red Min**	—	—	0.184060
	Red Max	—	—	0.447078
	Red Mean	—	—	0.153745
	Red Med	—	—	0.324951
	Red Std	—	—	0.136518
	Green Min	—	—	0.195144
	Green Max	—	—	0.166112
	Green Mean	—	—	0.236338
	Green Med	—	—	0.409287
	Green Std	—	—	0.106147
	Blue Min	—	—	0.184060
	Blue Max	—	—	0.285907
	Blue Mean	—	—	0.454861
	Blue Med	—	—	0.298005
	Blue Std	—	—	0.395668

Table 6.5 shows the P-value table of experiment 4 comparing four different algorithms for 15 objectives for their best results. The value used for the P-value calculation is the absolute difference between the target image and the best evolved images for the 15 RGB objectives (see Table 6.4). We used Mann-Whitney U test [66, 69] for the non-parametric analysis. The shaded cells of the tables represents significant statistical differences (threshold : $P<0.05$) and the arrow next to the values shows which algorithm is statistically better.

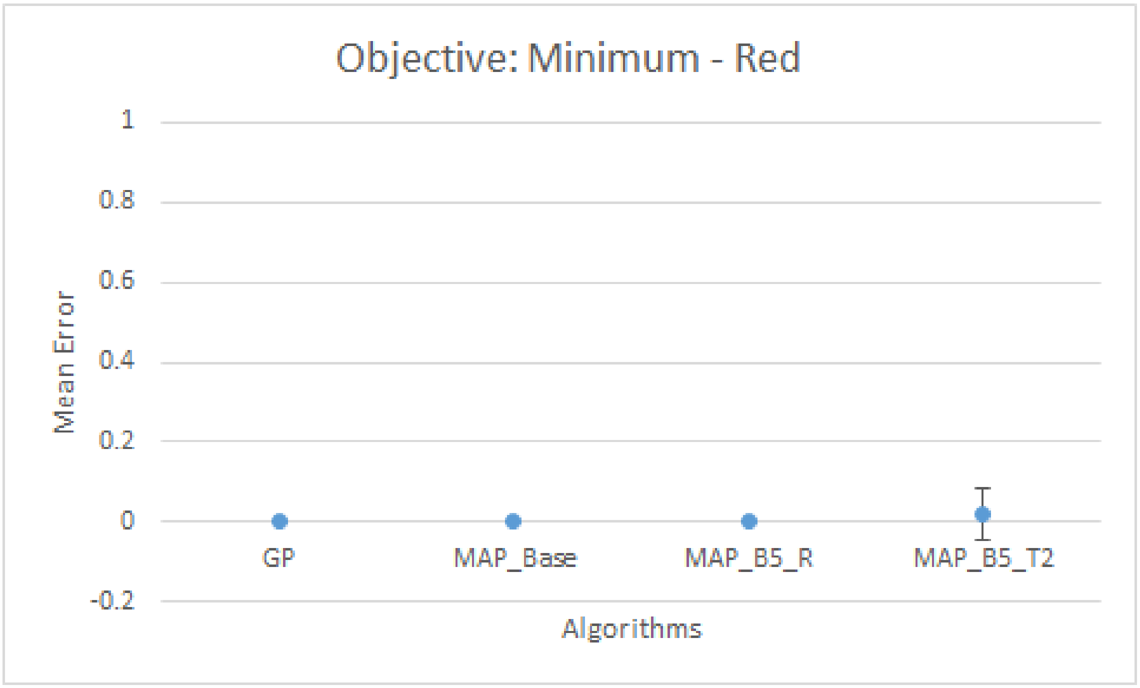


Figure 6.19: Error plot of mean minimum pixel value difference of red channel for different algorithms

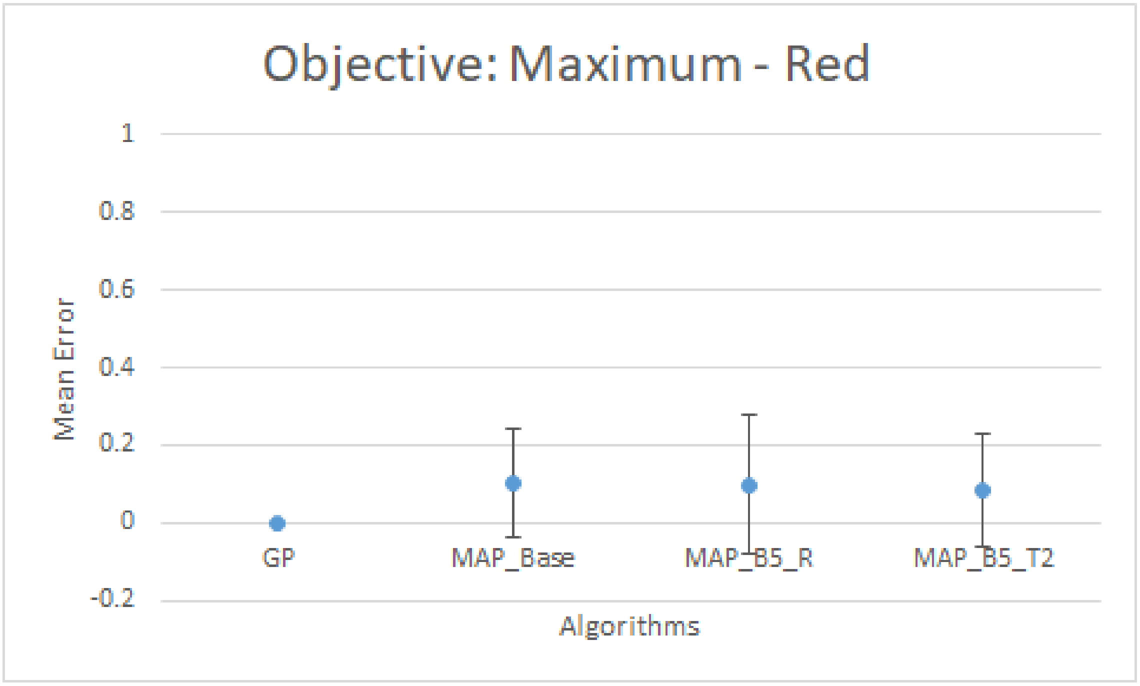


Figure 6.20: Error plot of mean maximum pixel value difference of red channel for different algorithms

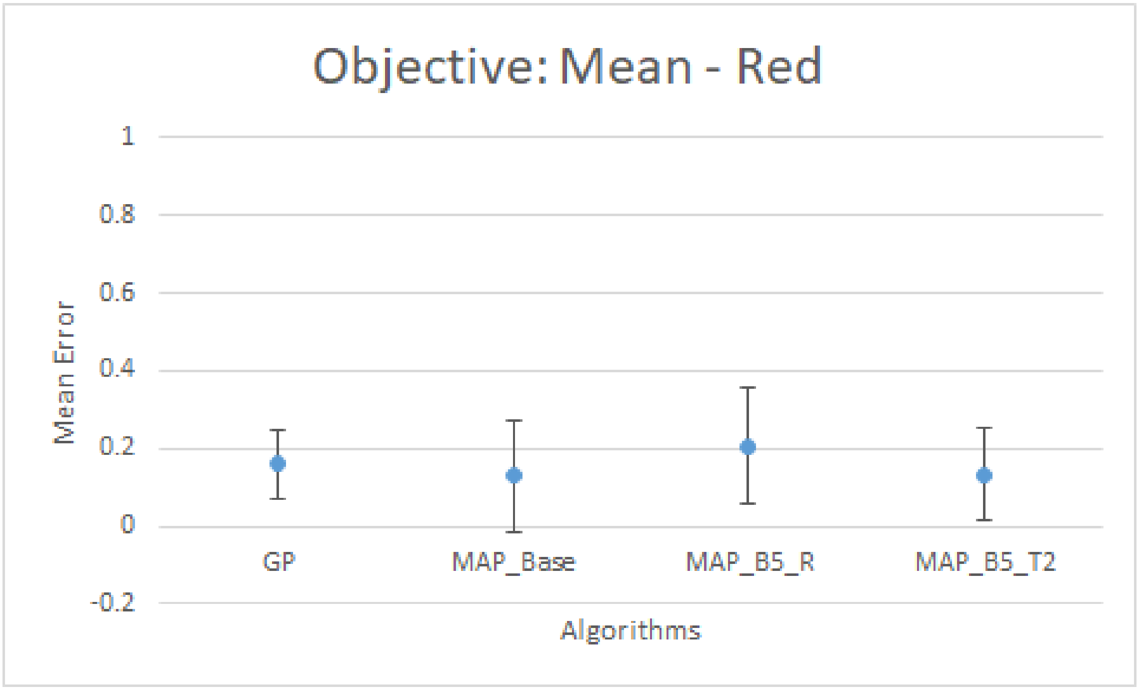


Figure 6.21: Error plot of mean mean pixel value difference of red channel for different algorithms

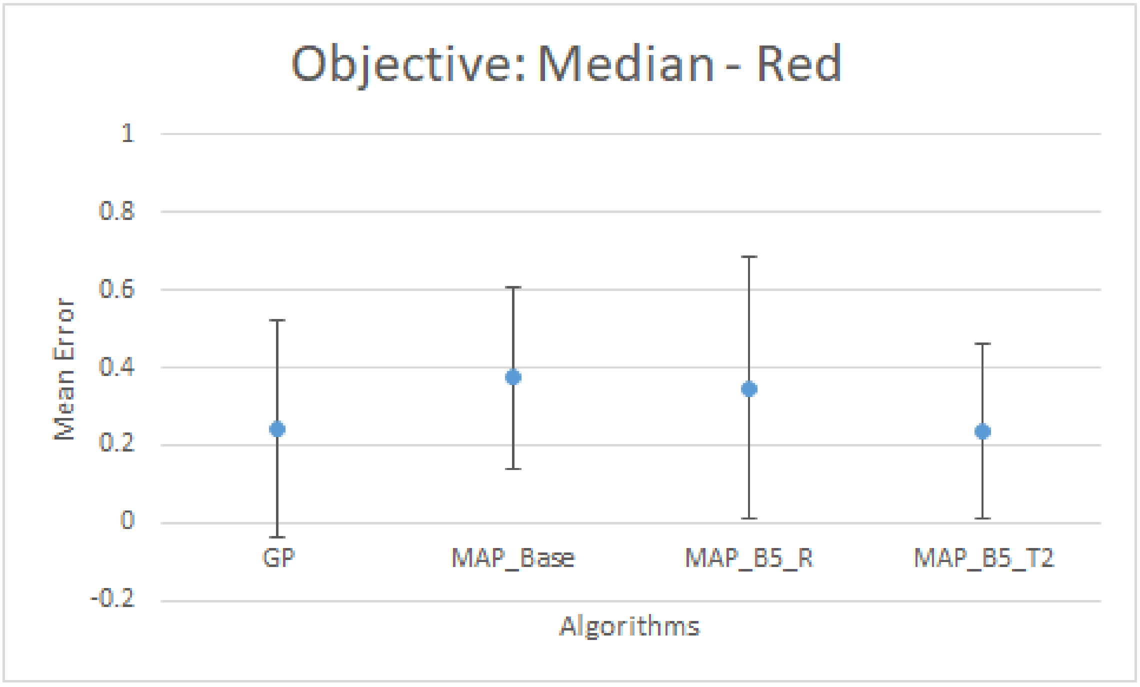


Figure 6.22: Error plot of mean median pixel value difference of red channel for different algorithms

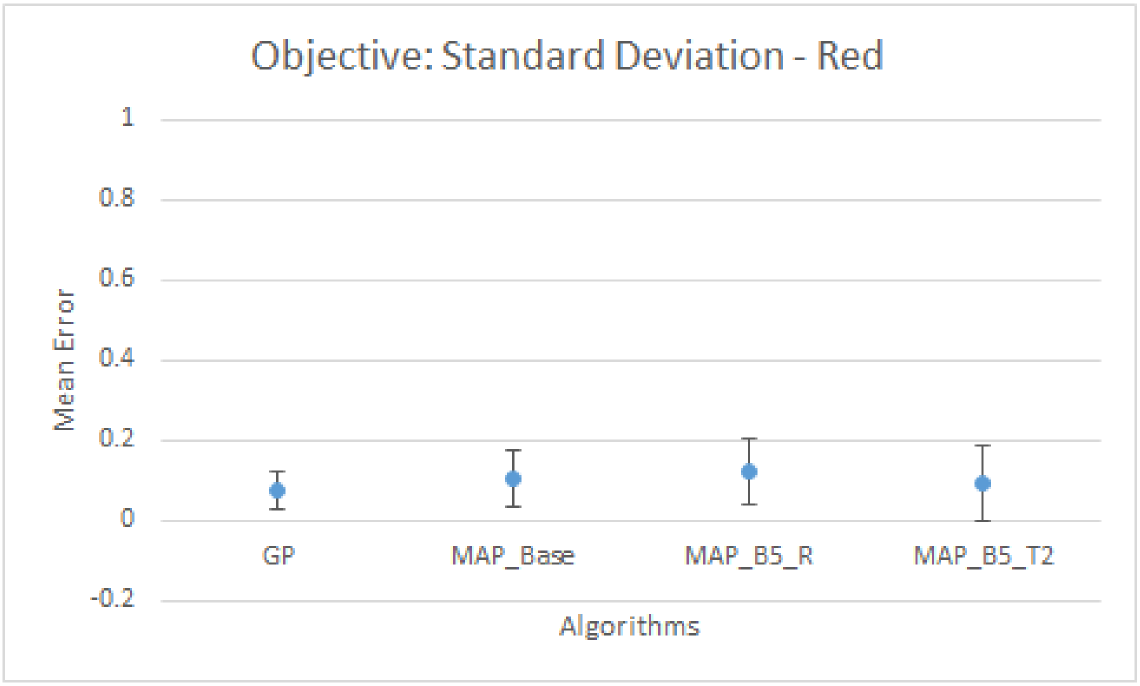


Figure 6.23: Error plot of mean standard deviation value difference of red channel for different algorithms

Figure 6.19 to 6.33 show the error plots with standard deviation for the four algorithms. These figures shows the values for fifteen different objectives (minimum, maximum, mean, median, and standard deviation of red, green and blue channel respectively). Figure 6.19, 6.20, 6.21, 6.22, and 6.23, shows the five objectives of the red channel. When we compare these figures with Table 6.5, we can see which algorithm is performing better in which objective. When GP is compared with other three algorithm in terms of red max objective (Figure 6.20), GP is statistically significantly better than the three version of MAP-Elites. And for the red mean objective (Figure 6.21), base MAP-Elites performs significantly better than MAP_B5_R. Other red channel objectives do not show any significant difference between algorithms.

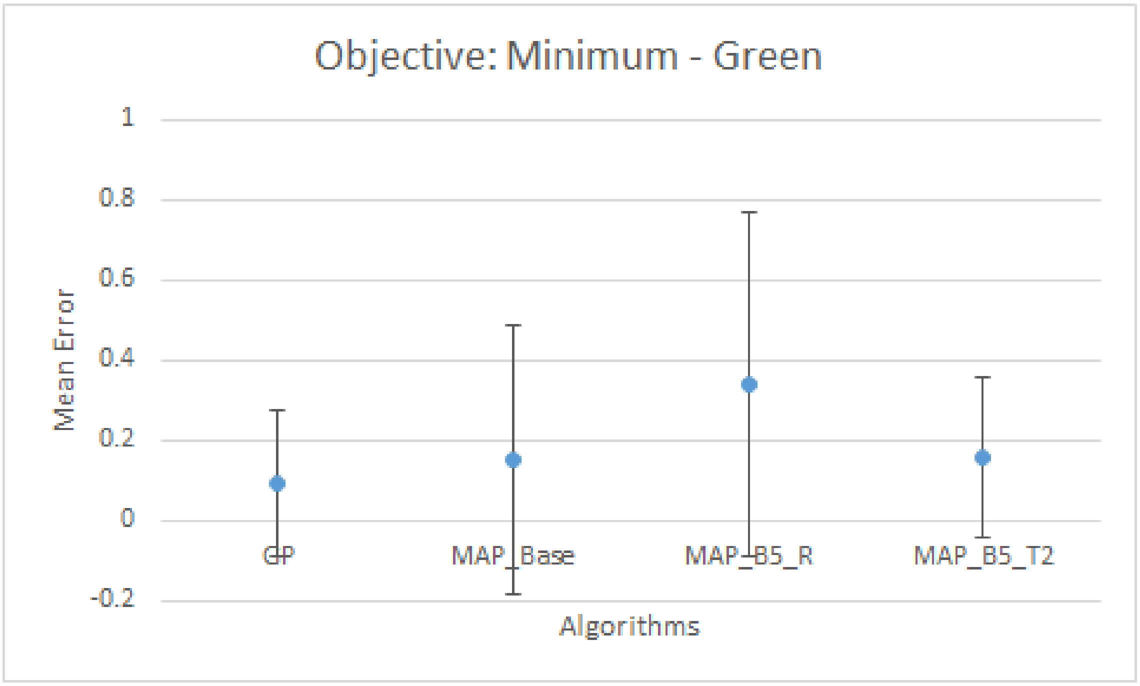


Figure 6.24: Error plot of mean minimum pixel value difference of green channel for different algorithms

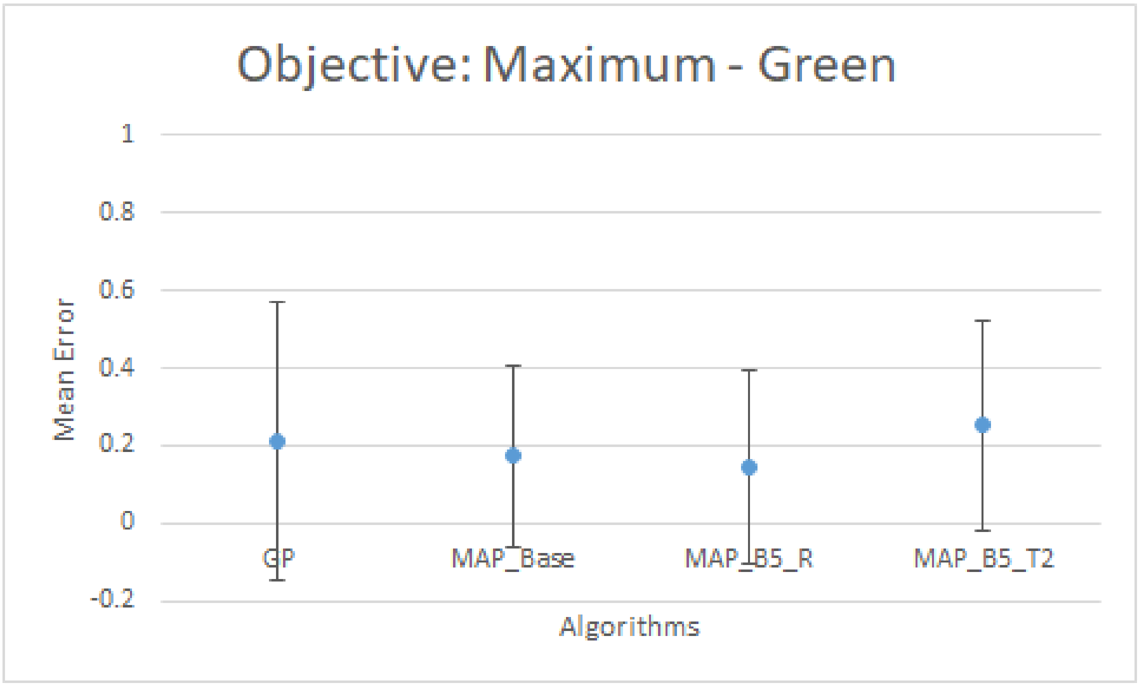


Figure 6.25: Error plot of mean maximum pixel value difference of green channel for different algorithms

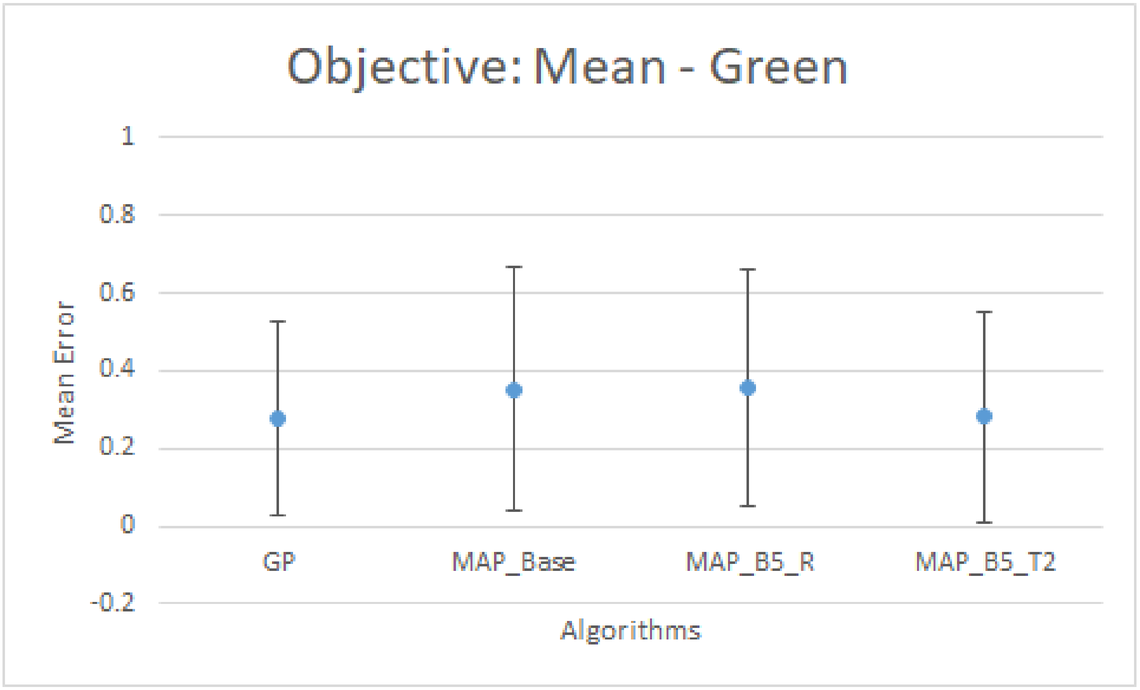


Figure 6.26: Error plot of mean mean pixel value difference of green channel for different algorithms

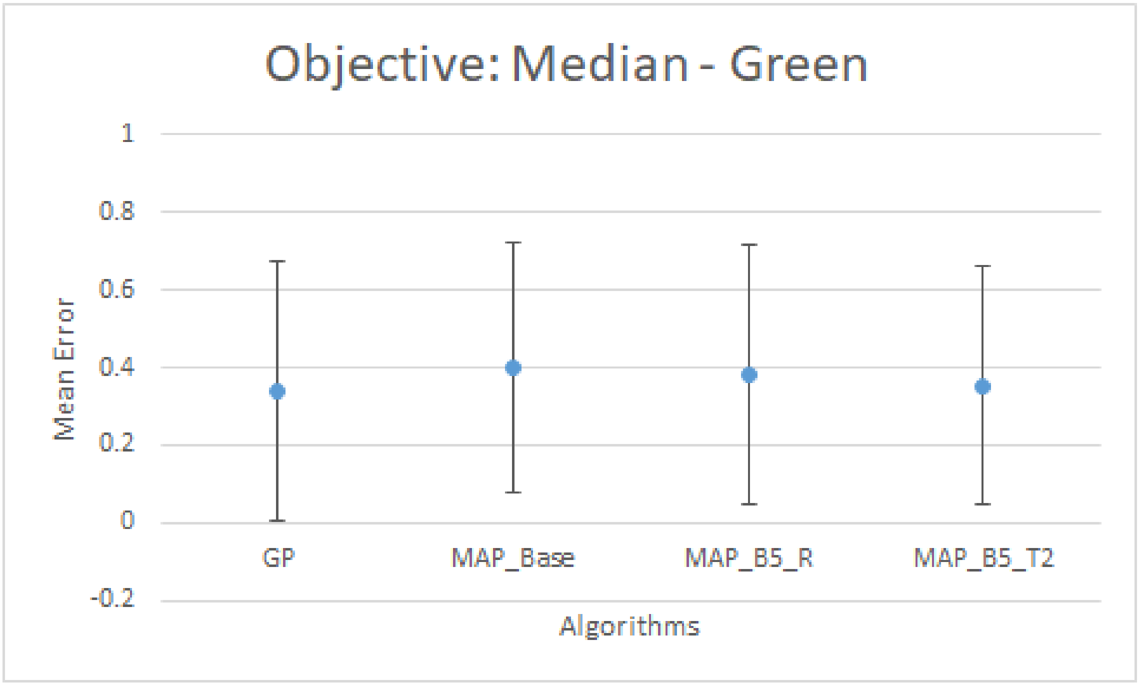


Figure 6.27: Error plot of mean median pixel value difference of green channel for different algorithms

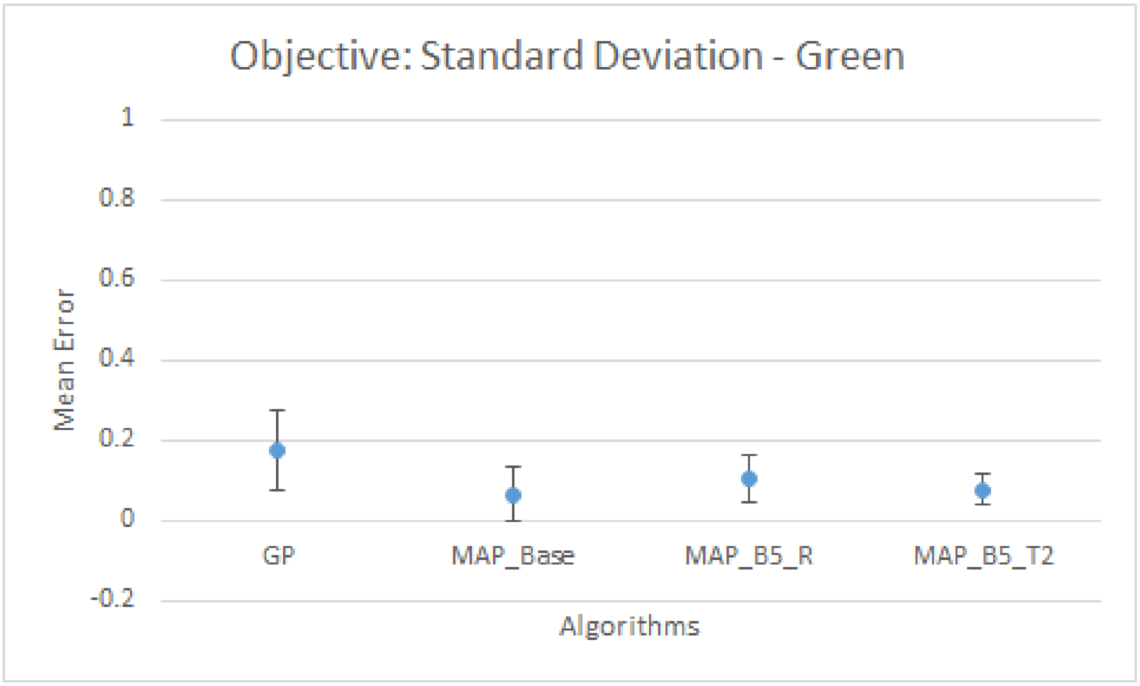


Figure 6.28: Error plot of mean standard deviation value difference of green channel for different algorithms

Similarly Figure 6.24, 6.25, 6.26, 6.27, and 6.28 shows the five objectives for the green channel. When we compare Table 6.5 with these figures we can see which algorithm is better performing in which objective. When GP is compared with MAP_B5_R in terms of green min objective (see Figure 6.24), performance of both GP and base MAP-Elites is statistically significantly better than the performance of MAP_B5_R. And for green standard deviation objective (see Figure 6.28), GP performs significantly worse than the three MAP-Elites algorithm, performs significantly better than MAP_B5_R. Other green channel objectives do not show any significant difference between algorithms.

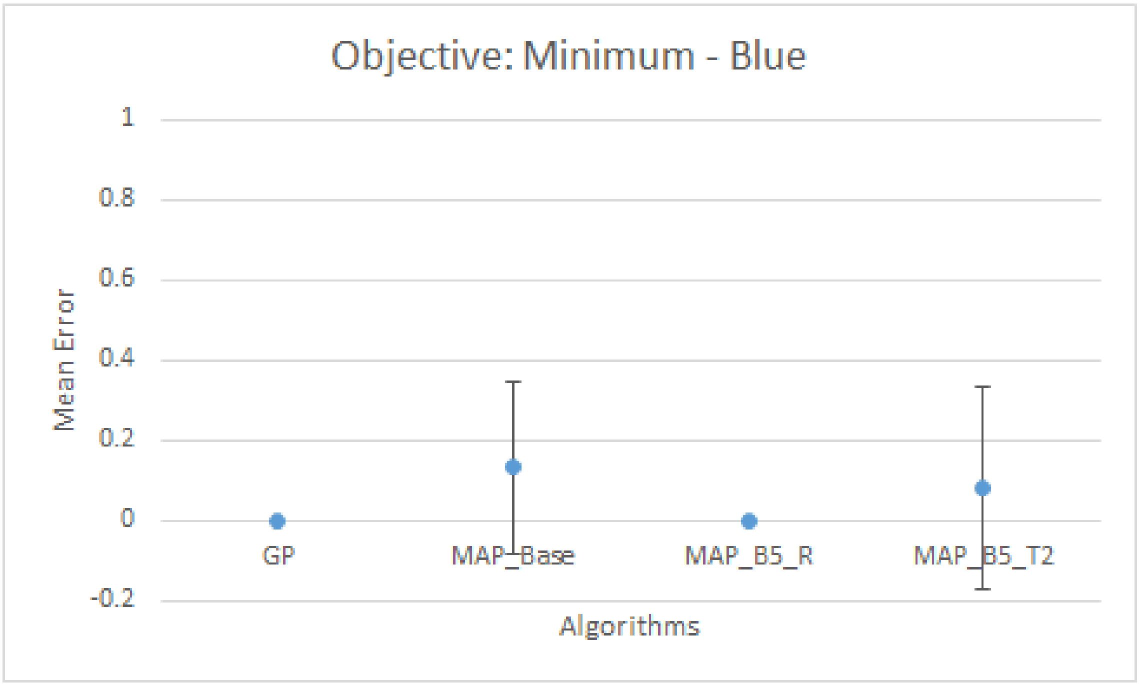


Figure 6.29: Error plot of mean minimum pixel value difference of blue channel for different algorithms

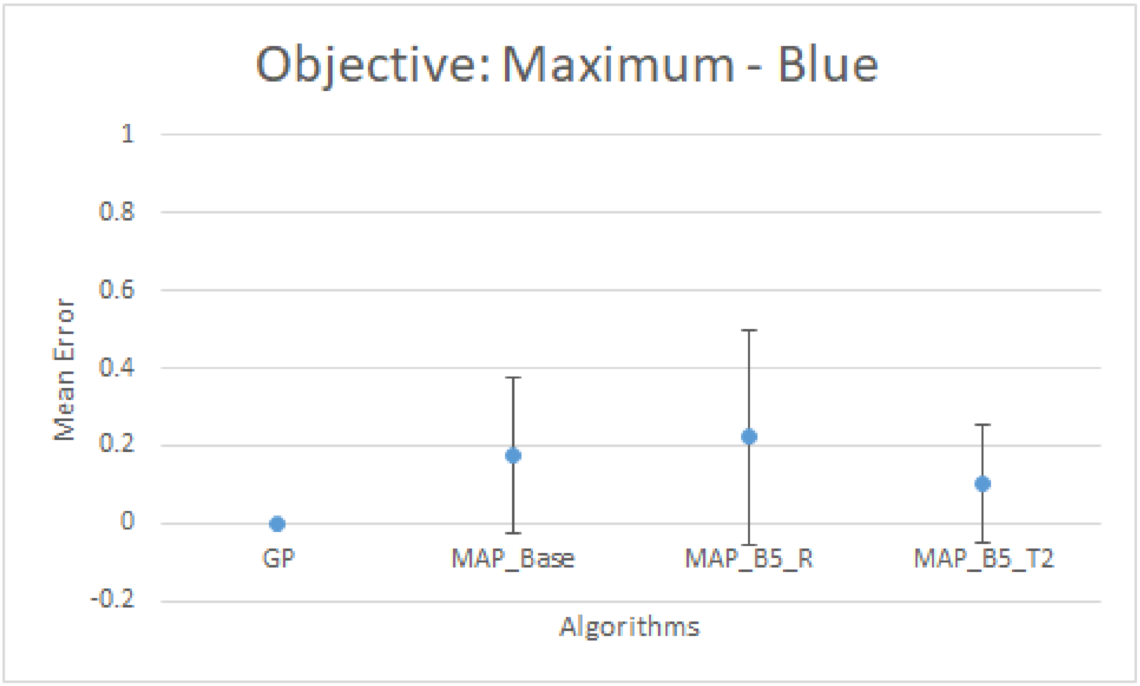


Figure 6.30: Error plot of mean maximum pixel value difference of blue channel for different algorithms

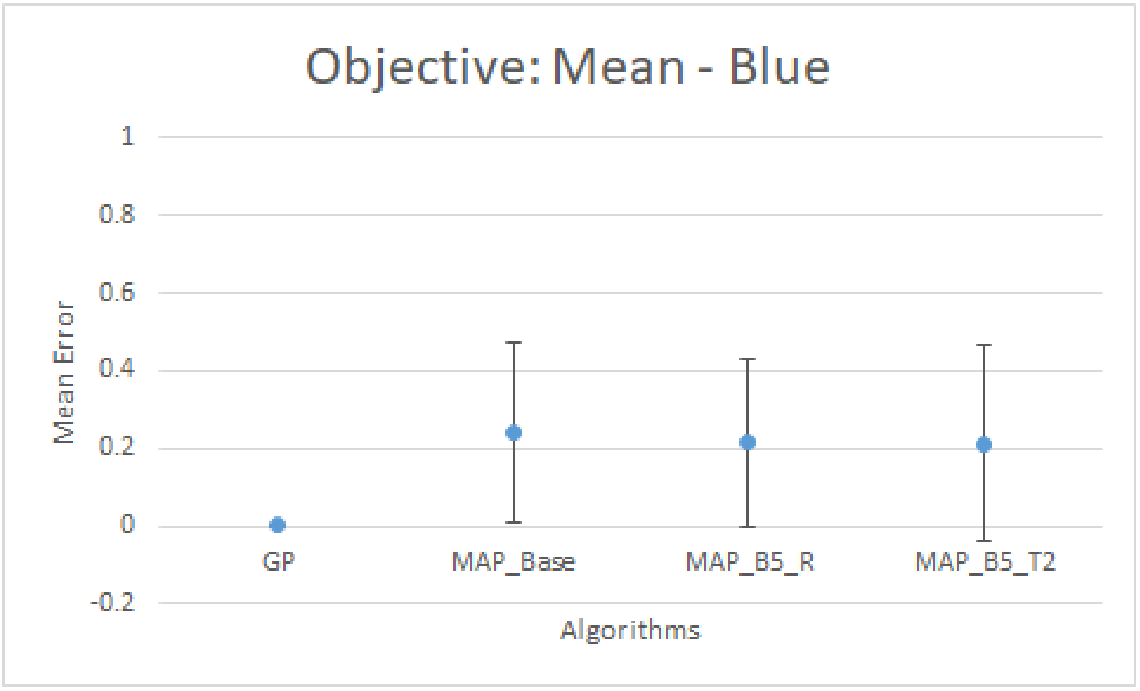


Figure 6.31: Error plot of mean mean pixel value difference of blue channel for different algorithms

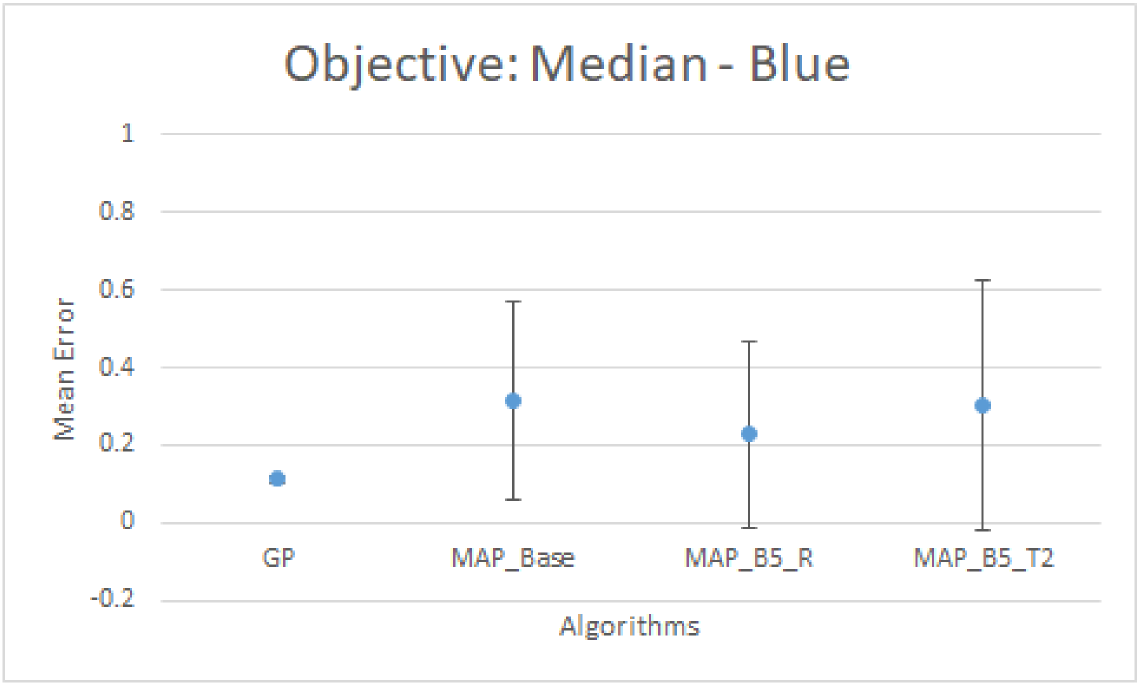


Figure 6.32: Error plot of mean median pixel value difference of blue channel for different algorithms

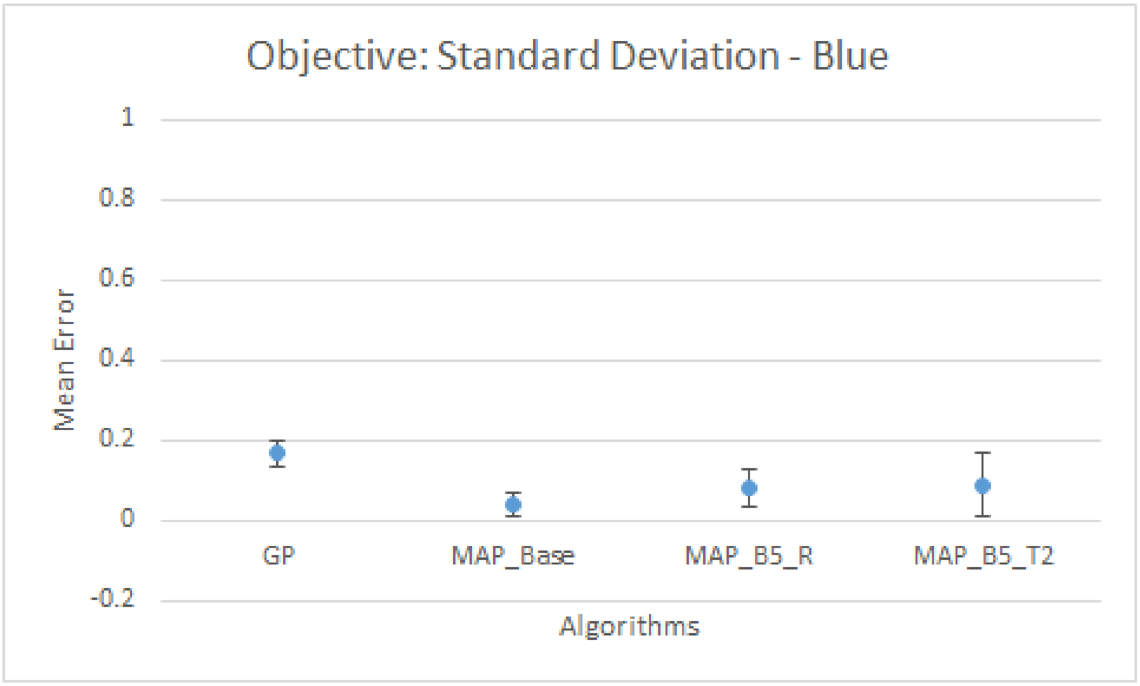


Figure 6.33: Error plot of mean standard deviation value difference of blue channel for different algorithms

Blue channel objectives shows most number of significant difference between algorithms, (Figures 6.29, 6.30, 6.31, 6.32, and 6.33). When we compare Table 6.5 with these figures we can see which algorithm is statistically better. When GP is compared with base MAP-Elites in terms of blue min objective (see Figure 6.29), performance of GP is statistically significantly better (threshold : $p<0.05$). For blue max and mean objective (see Figure 6.30 and 6.31), GP performs significantly better than the three MAP-Elites algorithm. For blue median objective (see Figure 6.32), GP significantly outperforms base MAP-Elites. Finally, for blue standard deviation objective (see Figure 6.33), all three MAP-Elites algorithms outperform GP and base MAP-Elites outperforms MAP_B5_R. The table and figures do not show any other statistical significance.

6.2 Experiment 5

This experiment is very similar to the previous RGB experiment. Table 6.4 shows the RGB feature set that has been used for this experiment. For this experiment the second target image is used (see Figure 6.1 (b)). Ten different runs were performed for each of the four algorithms to generate solutions. Each of the solution here is a 256×256 RGB image. All the pixel values are mapped between 0 and 1 for all three RGB channels.

6.2.1 Basic Genetic Programming (GP)

Table 6.1 shows the GP parameters used for this experiments and Table 6.4 shows the objective set for fitness evaluation.

Figure 6.34 shows the best solutions of ten GP runs. The left most column shows the target image and the right two columns show the evolved images.

Figure 6.35 shows the mean fitness of GP population averaged over 10 runs for all 15 objectives. The X-axis represents number of generations and the Y-axis represents mean error. The GP behaviour remains similar to the grayscale experiments and the previous RGB experiment for mean error. It shows that most of the features converge quickly. Some features get better or worse due to the use of sum-of-ranks as it tries to find optimal solution based on ranking of (fifteen) objectives. So, where some of the features may provide lower rank for an image others may not, resulting the kind of performance graph we are observing.

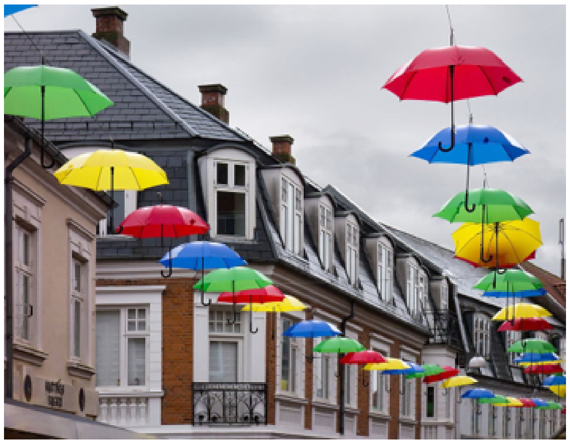
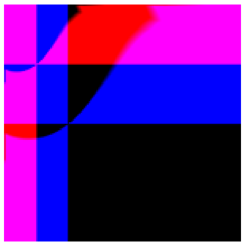
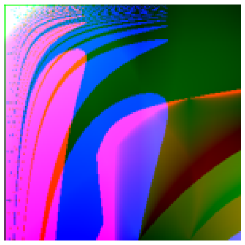
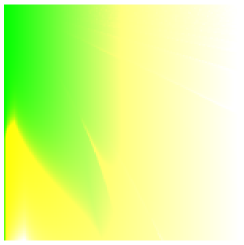
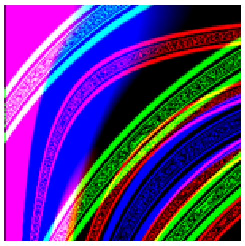
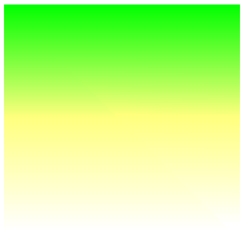
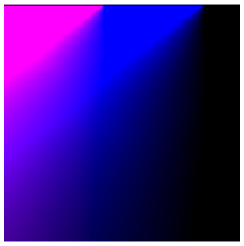
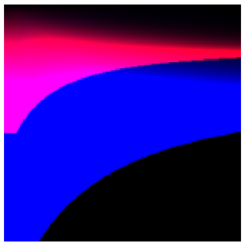
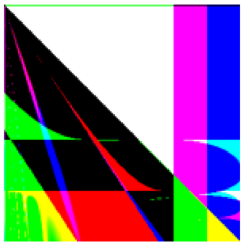
Target Image	Run: 1 - 5	Run: 6 - 10
		
		
		
		
		

Figure 6.34: Best result of each GP run for RGB target image- 2 and RGB feature set (10 runs).

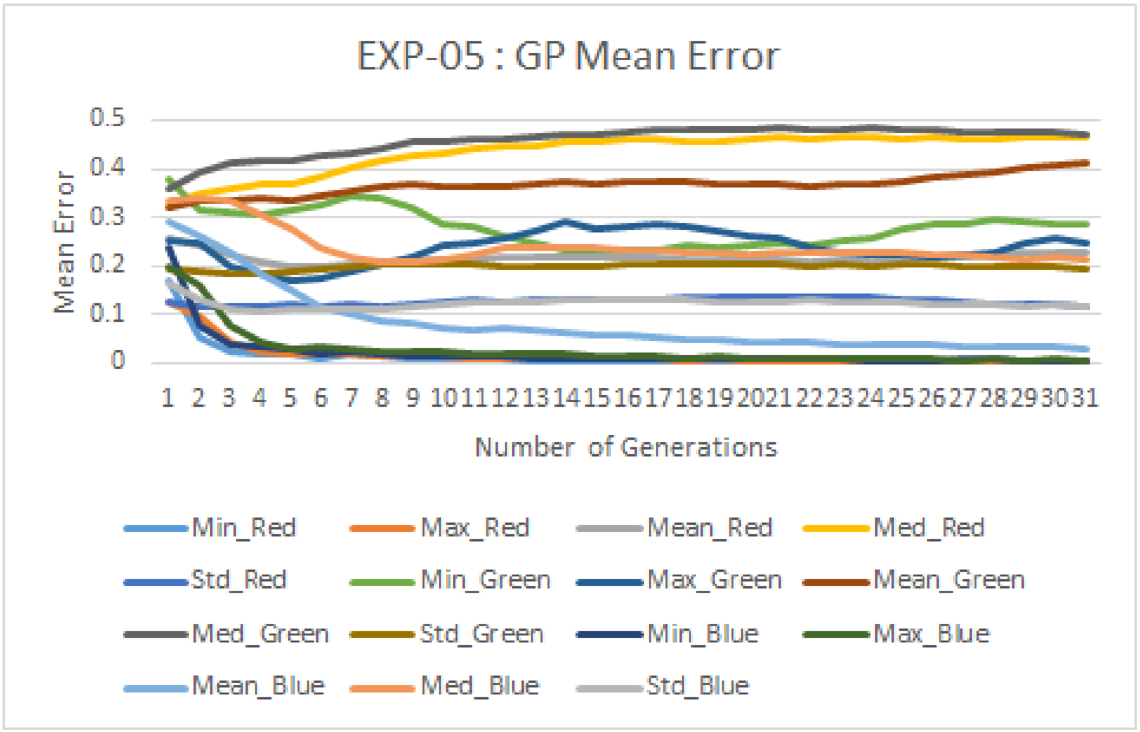


Figure 6.35: Average mean fitness of GP over 10 runs (lower is better).

Table 6.3 shows that the min and max value for all three channels for the second image (see Figure 6.1 (b)), which is 0 and 1 respectively (similar to the first image).

6.2.2 MAP-Elites: Single Individual Bin (MAP_Base)

Figure 6.36 shows the best solutions of ten MAP-Elites runs. The left most column shows the target image and the right two columns show the evolved images.

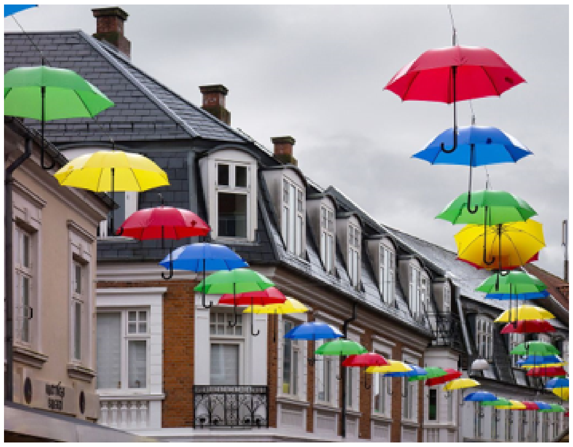
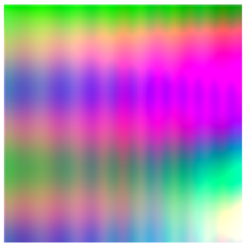
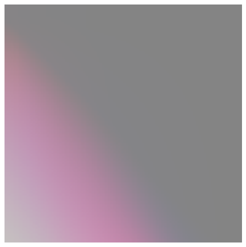
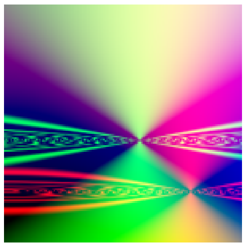
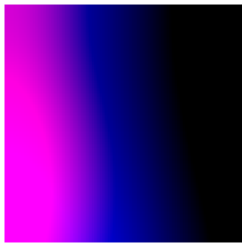
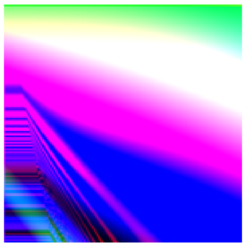
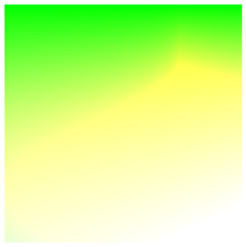
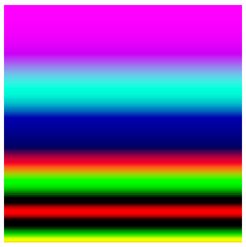
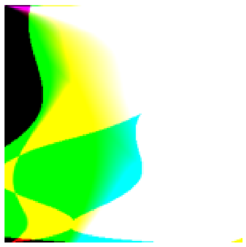
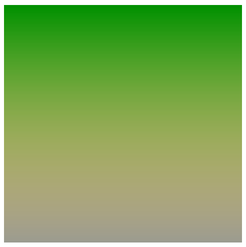
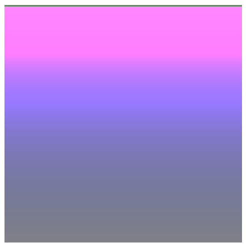
Target Image	Run: 1 - 5	Run: 6 - 10
		
		
		
		
		

Figure 6.36: Best results of base MAP-Elites runs for RGB target image- 2 and RGB feature set (10 runs).

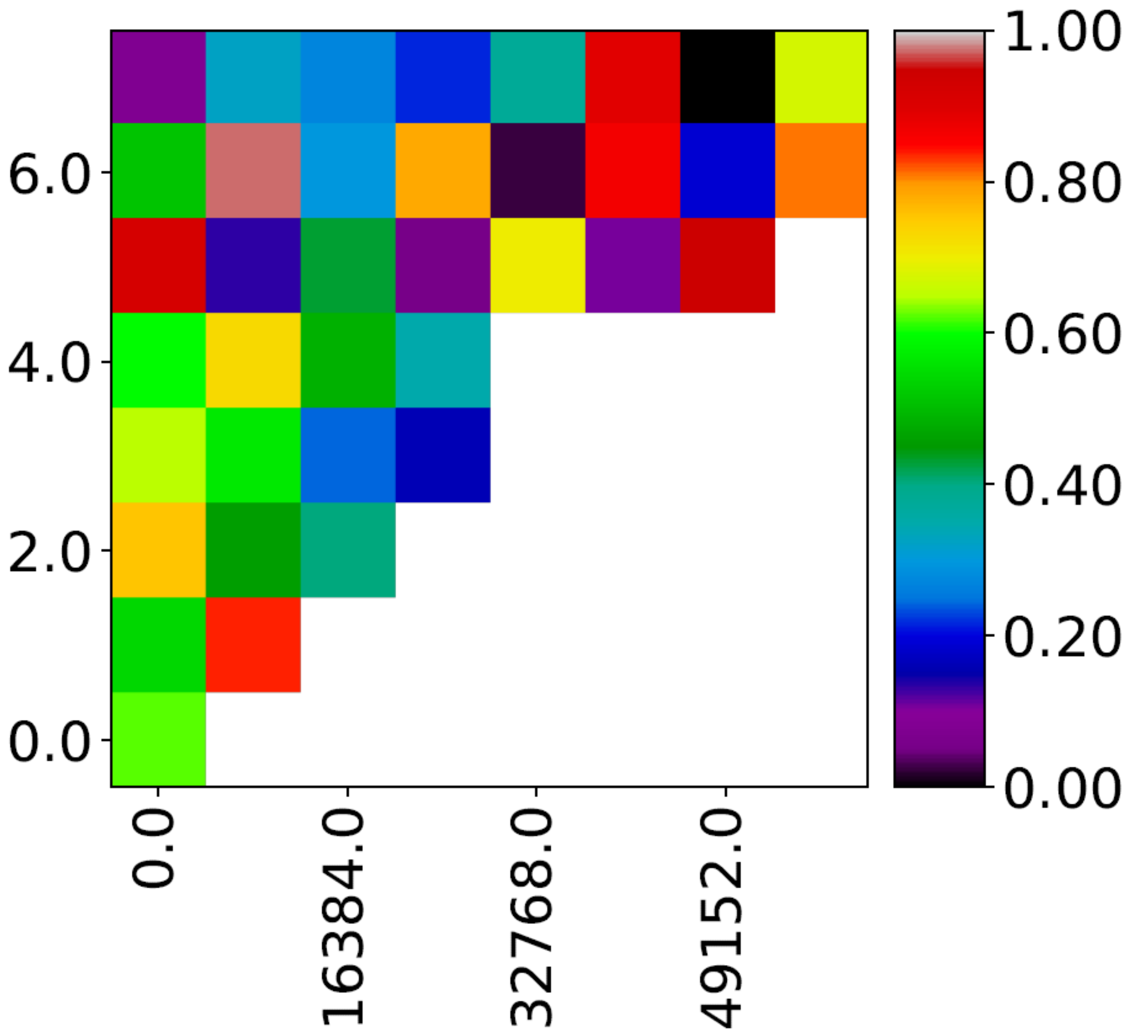


Figure 6.37: Sum-of-ranks MAP of run-6 of experiment 5 (higher is better) [X-axis: unique RGB value (0-65536); Y-axis: entropy (0-8)].

Figure 6.37 shows the final sum-of-ranks MAP for run-6 of experiment 4. X-axis represents the Unique RGB values and Y-axis represents the entropy of the solution images. We can see that the range for unique RGB is significantly lower than the total number of unique RGB value (over 16 millions). The reason behind this is explained in Section 6.1.2.

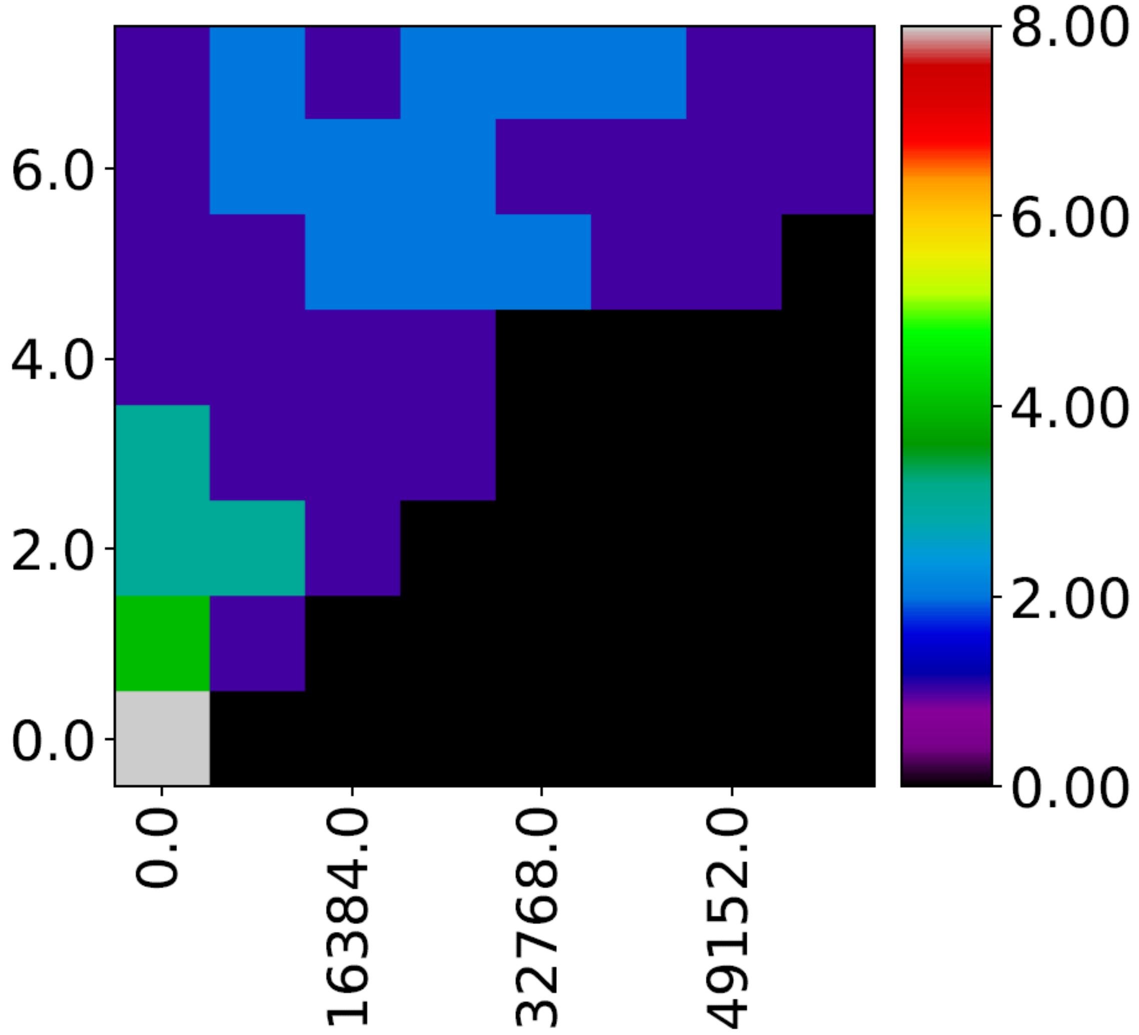


Figure 6.38: Activity MAP of run-6 of experiment 5 (higher is better) [X-axis: unique RGB value (0-65536); Y-axis: entropy (0-8)].

Figure 6.38 shows the activity MAP of run-4 of experiment 5. The X-axis represents unique RGB value and the Y-axis represents entropy. We can see that most of the bins have been updated only once, meaning that after those bins have been filled ones the new evolved solutions either had poor fitness than the existing solutions of those bins or they did not have the similar MAP feature behaviour, resulting them competing for other bins. Only a very few bins have been updated multiple times. Black bins represents no activity at all, meaning no solutions were found for those feature space as very high unique colour combination and low entropy is very unlikely. Here we can see the best solution bin has only been updated once, meaning new solutions either did not match the feature space or did not have better fitness than the existing solution in the bin.

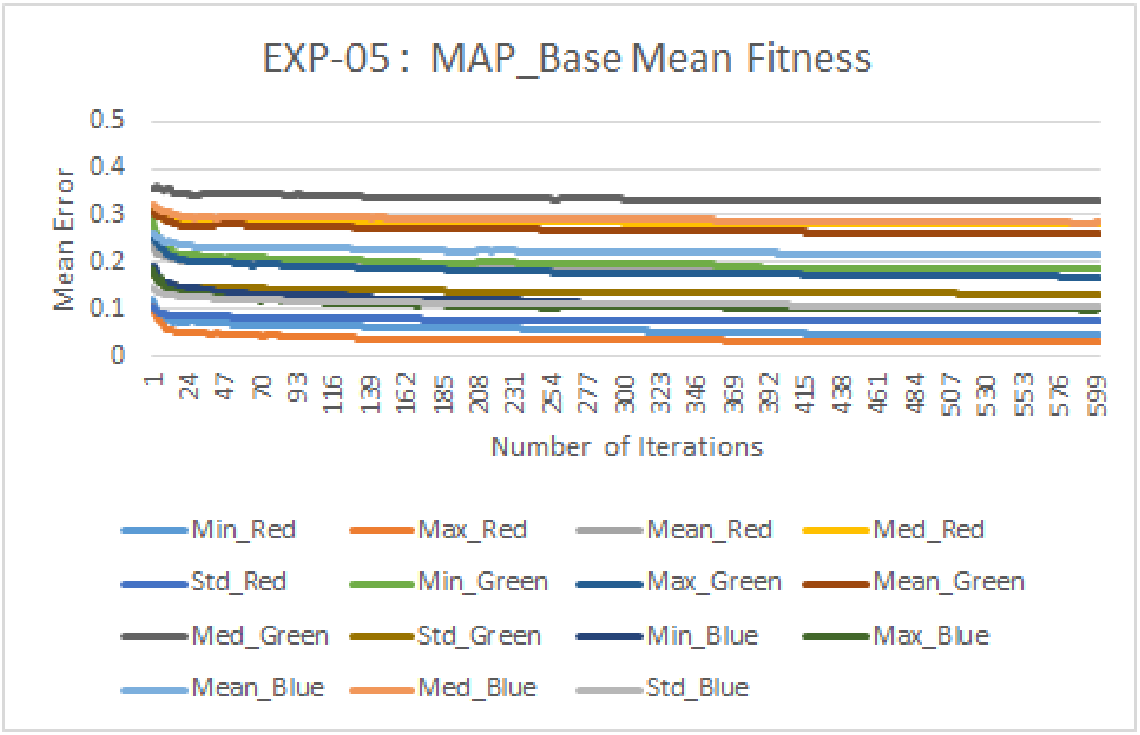


Figure 6.39: Average mean fitness of base MAP-Elites MAP over 10 runs (lower is better).

Figure 6.11 shows the mean fitness of the base MAP-Elites MAP averaged over 10 runs for all 15 objectives. The X-axis represents number of generations and the Y-axis represents mean error. We can see that, compared with GP, base MAP-Elites has better fitness is some objectives and similar or worse fitness in a few. And the solution space values do not change much as it takes the whole MAP of diverse solutions into account and diversify solutions more over time. The best fitness plot behaves similar to the best fitness plot shown in Section 6.1.3 (see Figure 6.12). The reason for not showing the best fitness plot is same as the one described in Section 6.2.1 (does not show any useful information).

6.2.3 MAP-Elites: Multiple individuals Bin - Random Selection (MAP_B5_R)

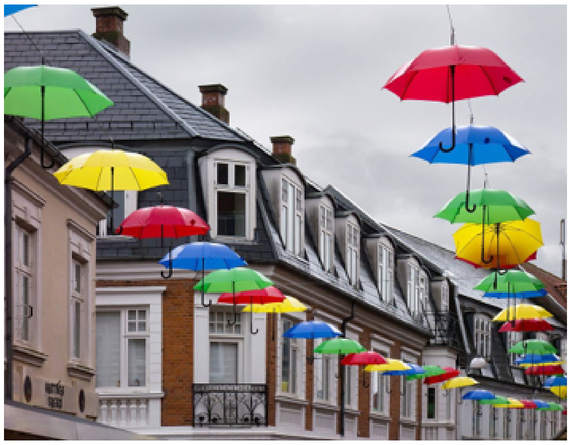
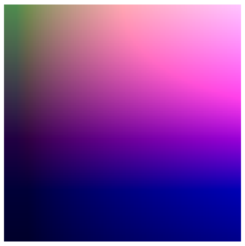
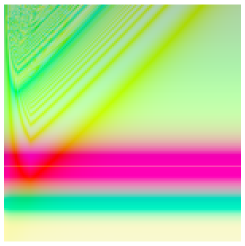
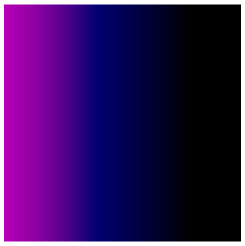
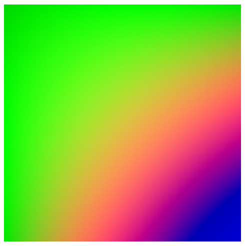
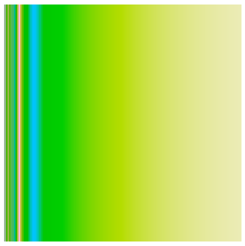
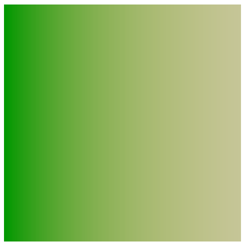
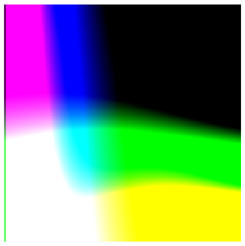
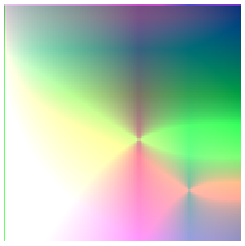
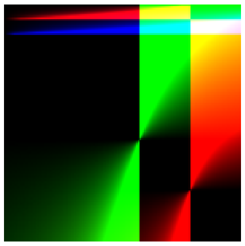
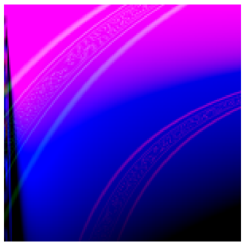
Target Image	Run: 1 - 5	Run: 6 - 10
		
		
		
		
		

Figure 6.40: Best results of base MAP-Elites runs for RGB target image- 2 and RGB feature set (10 runs).

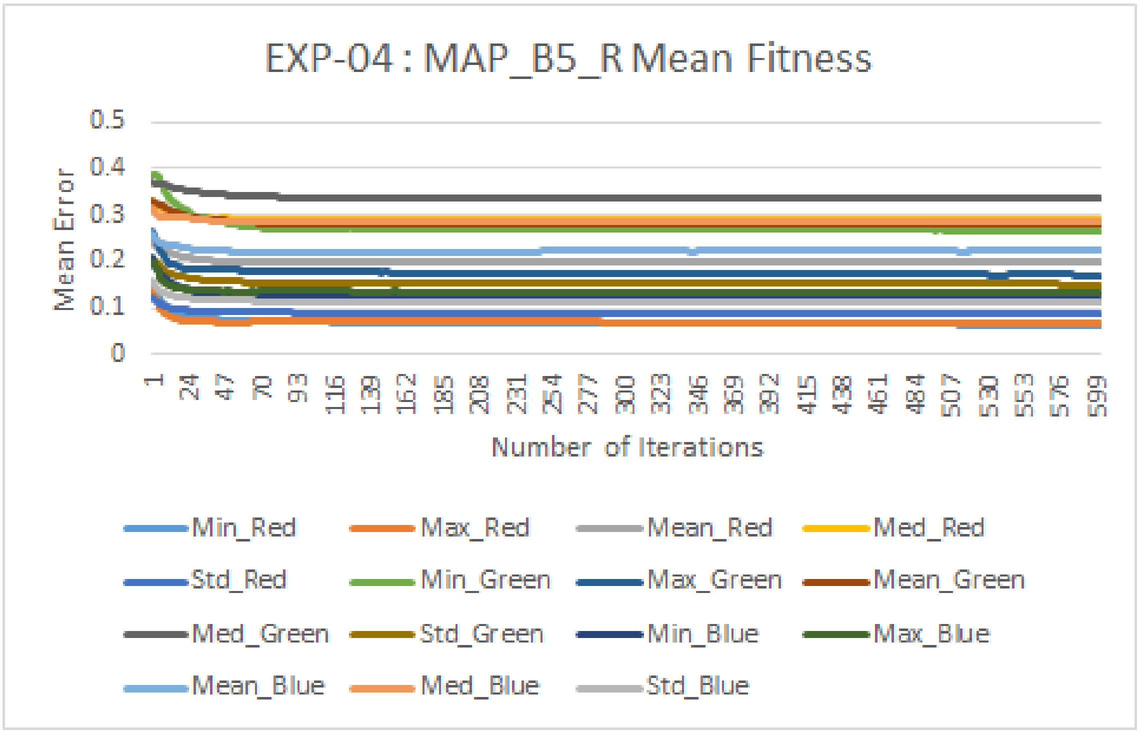


Figure 6.41: Average mean fitness of MAP-Elites (multiple individuals bin - random selection) MAP over 10 runs (lower is better).

This experiments has the same parameters (see Table 6.2) as the previous section. The only difference is the bin size, which is set to 5. Figure 6.40 shows the best results from 10 MAP_B5_R runs. Figure 6.14 shows the mean fitness of the MAP_B5_R MAP averaged over 10 runs for all 15 objectives. The X-axis represents number of generations and the Y-axis represents mean error. We can see that compared with base MAP-Elites, MAP_B5_R shows similar mean fitness in terms of mean error. The best fitness plot behaves similar to the best fitness plot shown in Section 6.1.3 (see Figure 6.15). The reason for not showing the best fitness plot is same as the one described in Section 6.2.1 (does not show any useful information).

6.2.4 MAP-Elites: Multiple individuals Bin - Tournament Selection (MAP_B5_T2))

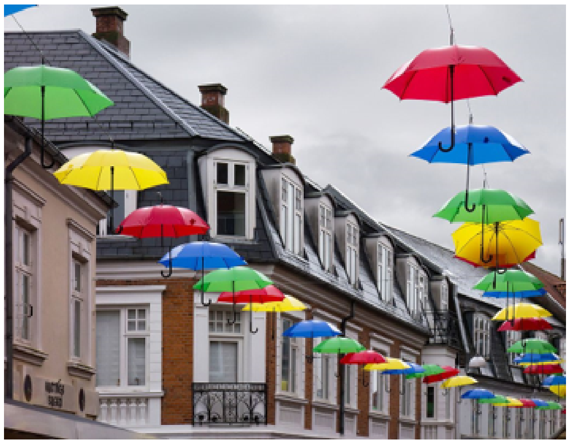
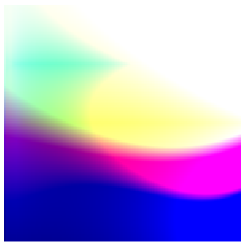
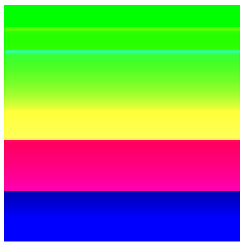
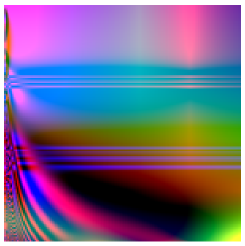
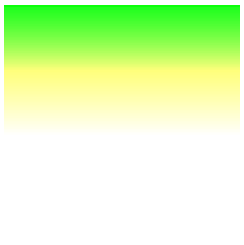
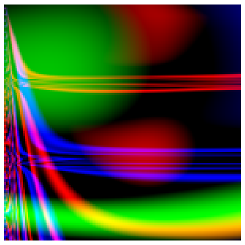
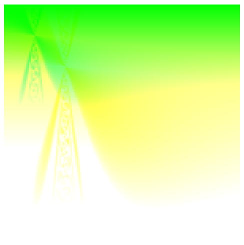
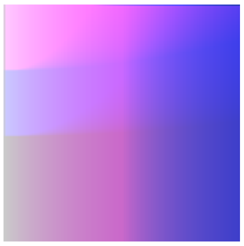
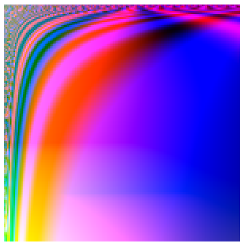
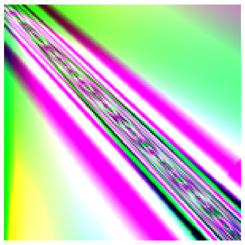
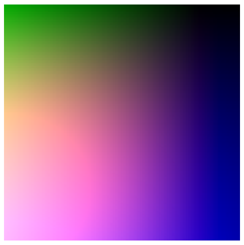
Target Image	Run: 1 - 5	Run: 6 - 10
		
		
		
		
		

Figure 6.42: Best results of base MAP-Elites runs for RGB target image 2 and RGB feature set (10 runs).

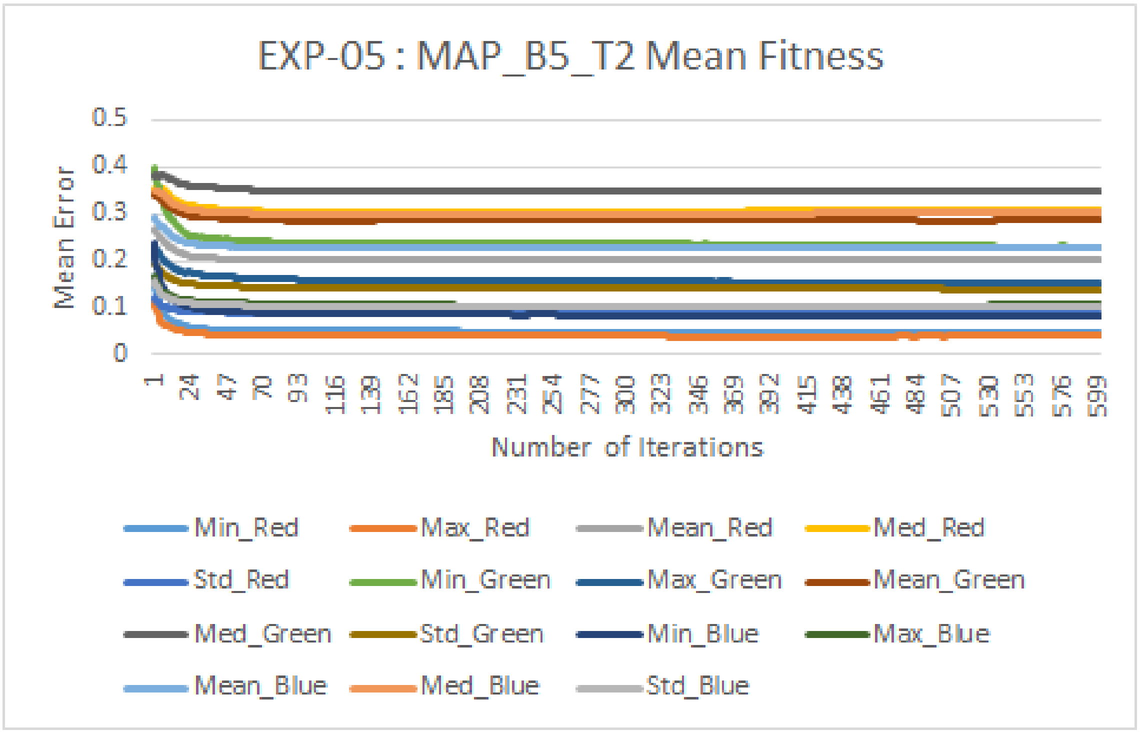


Figure 6.43: Average mean fitness of MAP-Elites (multiple individuals bin - tournament selection) MAP over 10 runs (lower is better).

This experiment has the same parameters (see Table 6.2) as the previous section. The only difference is the selection method. The bin size is 5 and a tournament of size 2 is used for the selection of parents for reproduction. Figure 6.42 shows the best results from 10 MAP_B5_T2 runs. Figure 6.43 shows the mean fitness of the base MAP-Elites MAP averaged over 10 runs for all 15 objectives. The X-axis represents number of Generations and the Y-axis represents mean error. We can see that MAP_B5_R and MAP_B5_T2 shows a similar mean fitness in terms of mean error. The best fitness plot behaves similar to the best fitness plot shown in Section 6.1.4 (see Figure 6.18). The reason for not showing the best fitness plot is same as the one described in Section 6.2.1 (does not show any useful information).

6.2.5 Analysis of experiment 5

Table 6.6: P-value table for experiment 5

	Objectives	**MAP_Base**	**MAP_B5_R**	**MAP_B5_T2**
GP	**Red Min**	0.184060	1.000000	0.184060
	Red Max	0.038936 ←	0.017422 ←	0.184060
	Red Mean	0.037831 ↑	0.120661	0.192337
	Red Med	0.006984 ↑	0.145475	0.014351 ↑
	Red Std	0.136518	0.120661	0.037831 ↑
	Green Min	0.271869	0.440730	0.363169
	Green Max	0.280246	0.197078	0.354335
	Green Mean	0.034087 ↑	0.201864	0.059631
	Green Med	0.057665	0.112569	0.029405 ↑
	Green Std	0.131372	0.039037 ↑	0.041121 ↑
	Blue Min	0.184060	0.184060	0.084039
	Blue Max	0.038936 ←	0.017492 ←	0.184060
	Blue Mean	0.000853 ←	9.13E-05 ←	9.13E-05 ←
	Blue Med	0.500000	0.171350	0.271839
	Blue Std	0.172352	0.044487 ↑	0.012874 ↑
MAP_Base	**Red Min**	—	0.184060	0.500000
	Red Max	—	0.377866	0.127926
	Red Mean	—	0.338792	0.213678
	Red Med	—	0.234488	0.201775
	Red Std	—	0.236338	0.092938
	Green Min	—	0.344893	0.377866
	Green Max	—	0.438354	0.062569
	Green Mean	—	0.128330	0.285085
	Green Med	—	0.295820	0.307543
	Green Std	—	0.213330	0.338217
	Blue Min	—	0.500000	0.292124
	Blue Max	—	0.344893	0.117133
	Blue Mean	—	0.080986	0.153745
	Blue Med	—	0.099129	0.172261
	Blue Std	—	0.192337	0.037831 ↑

Table 6.6 continued from previous page

	Objectives	**MAP_Base**	**MAP_B5_R**	**MAP_B5_T2**
MAP_B5_R	**Red Min**	—	—	0.184060
	Red Max	—	—	0.061304
	Red Mean	—	—	0.484925
	Red Med	—	—	0.394854
	Red Std	—	—	0.311588
	Green Min	—	—	0.426026
	Green Max	—	—	0.031381 ↑
	Green Mean	—	—	0.172261
	Green Med	—	—	0.454450
	Green Std	—	—	0.366672
	Blue Min	—	—	0.251816
	Blue Max	—	—	0.050581
	Blue Mean	—	—	0.484925
	Blue Med	—	—	0.484902
	Blue Std	—	—	0.106147

Some analyses for different algorithms have been discussed in previous sections. In this section we will compare them in more detail. Table 6.6 shows the P-value table of experiment 5 comparing four different algorithms for 15 objectives for their beast results. The value used for P-value calculation is the absolute difference between the target image and the best evolved images for the 15 RGB objectives (see Table 6.4). We used Mann-Whitney U test [66, 69] for the non-parametric analysis. The shaded cells of the tables represents significant statistical differences (threshold : $P<0.05$) and the arrow next to the values shows which algorithm is statistically better.

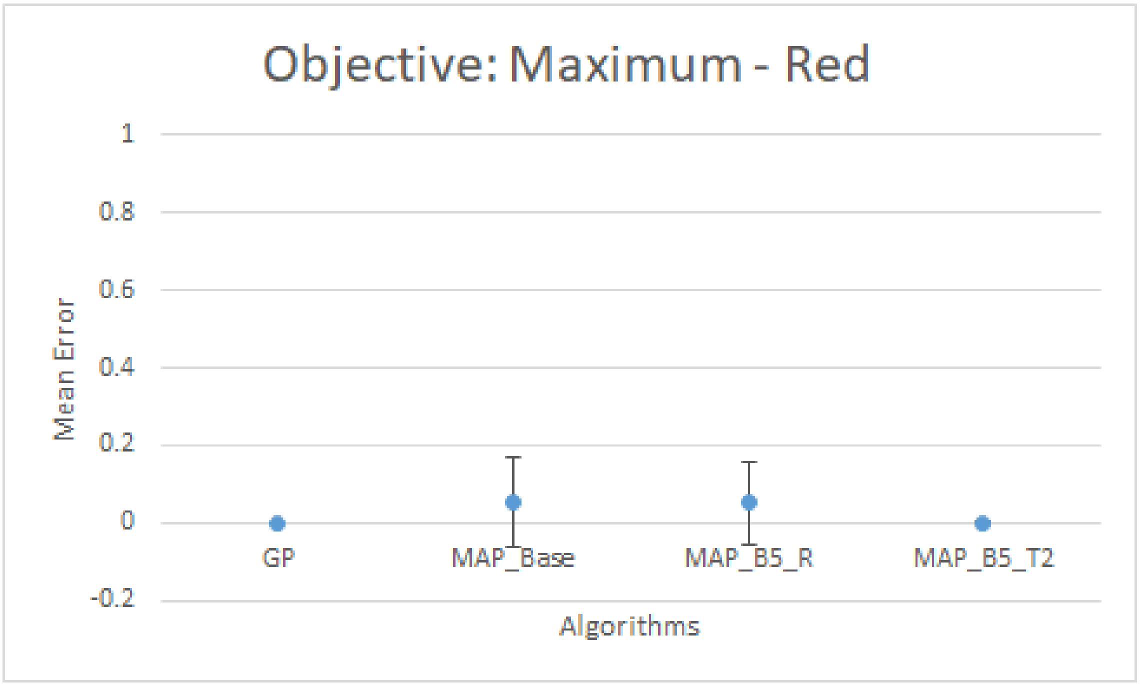


Figure 6.44: Error plot of mean maximum pixel value difference of red channel for different algorithms

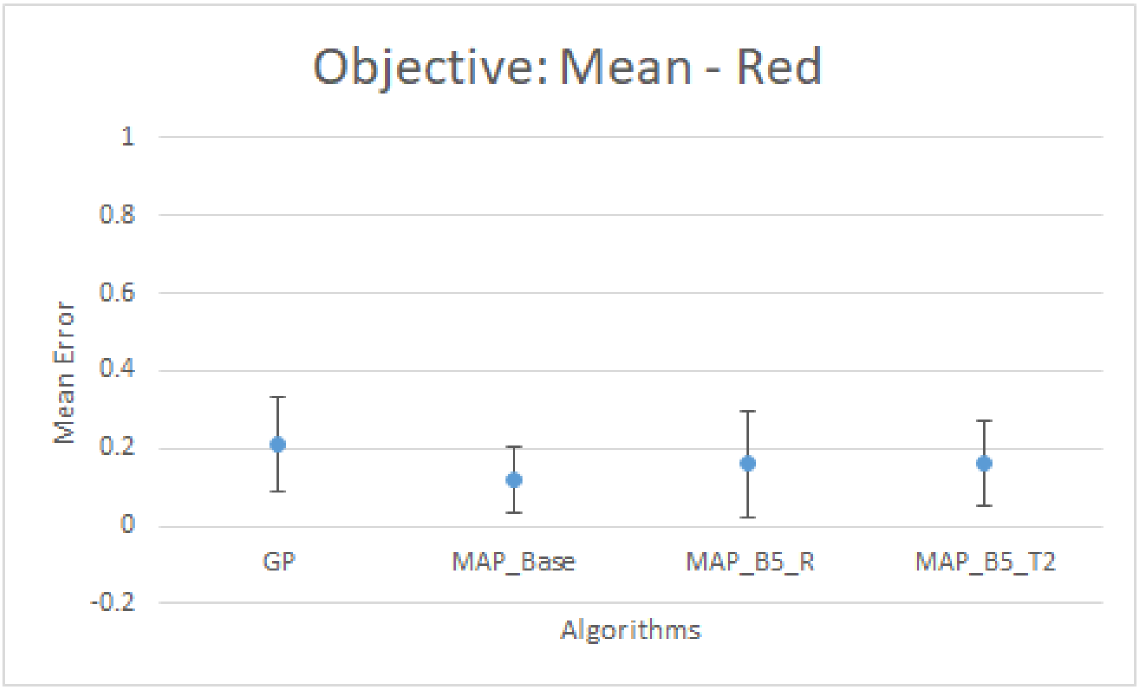


Figure 6.45: Error plot of mean mean pixel value difference of red channel for different algorithms

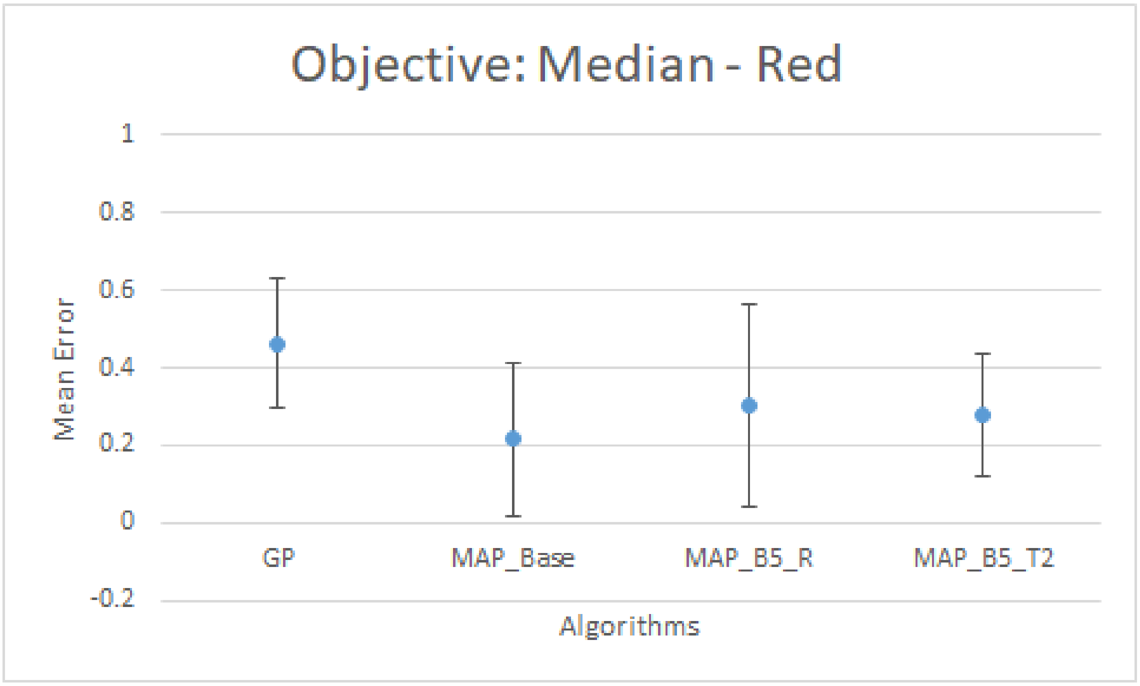


Figure 6.46: Error plot of mean median pixel value difference of red channel for different algorithms

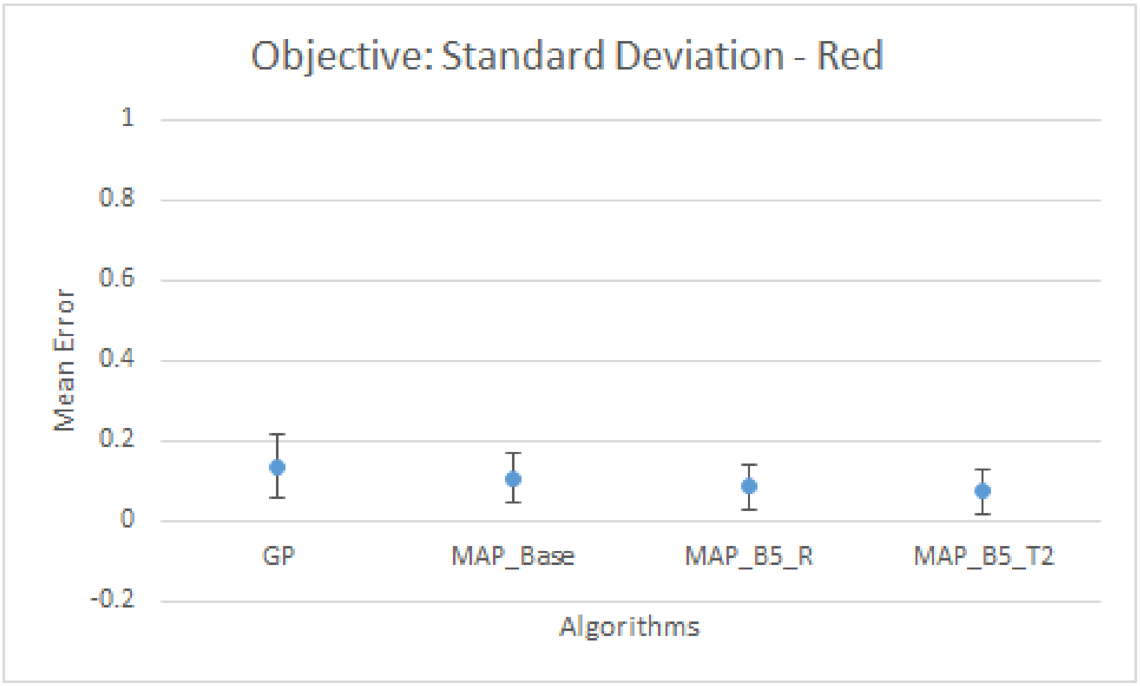


Figure 6.47: Error plot of mean standard deviation value difference of red channel for different algorithms

Figures 6.44, 6.45, 6.46, and 6.47, show the error plots with standard deviation for four objectives (max, mean, median and standard deviation) of red channel for the four algorithms used in this experiment. The error plot for the minimum objective of the red channel is not shown here as it does not have any statistical significance. When we compare these four figures mentioned earlier with Table 6.6 we can see which algorithm is performing better in which objective.

When GP is compared with other three algorithm in terms of red max objective (see Figure 6.44), GP is statistically significantly better (threshold : $p<0.05$) than the base MAP-Elites and MAP_B5_R. But, shows no significant difference with MAP_B5_T2. For red mean and median objectives (see Figure 6.45) base MAP-Elites significantly outperforms GP. For the red median objective base MAP-Elites and MAP_B5_T2 (see Figure 6.46), both shows significantly better results than GP. And for the red standard deviation MAP_B5_T2 (see Figure 6.47), performs significantly better than GP.

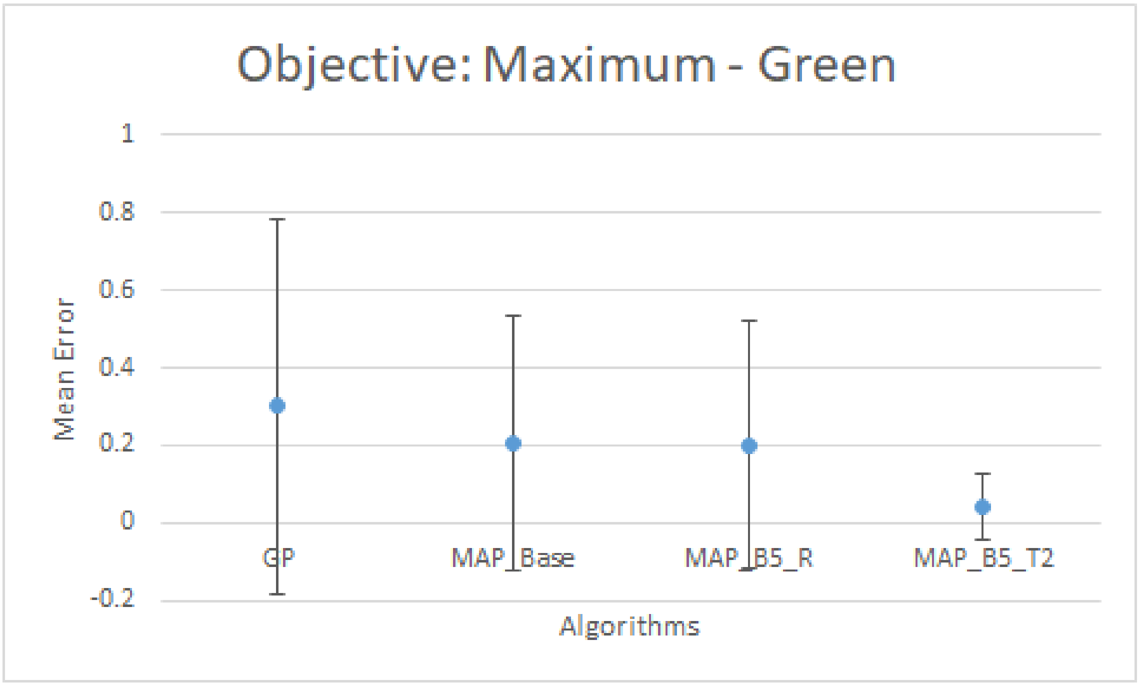


Figure 6.48: Error plot of mean maximum pixel value difference of green channel for different algorithms

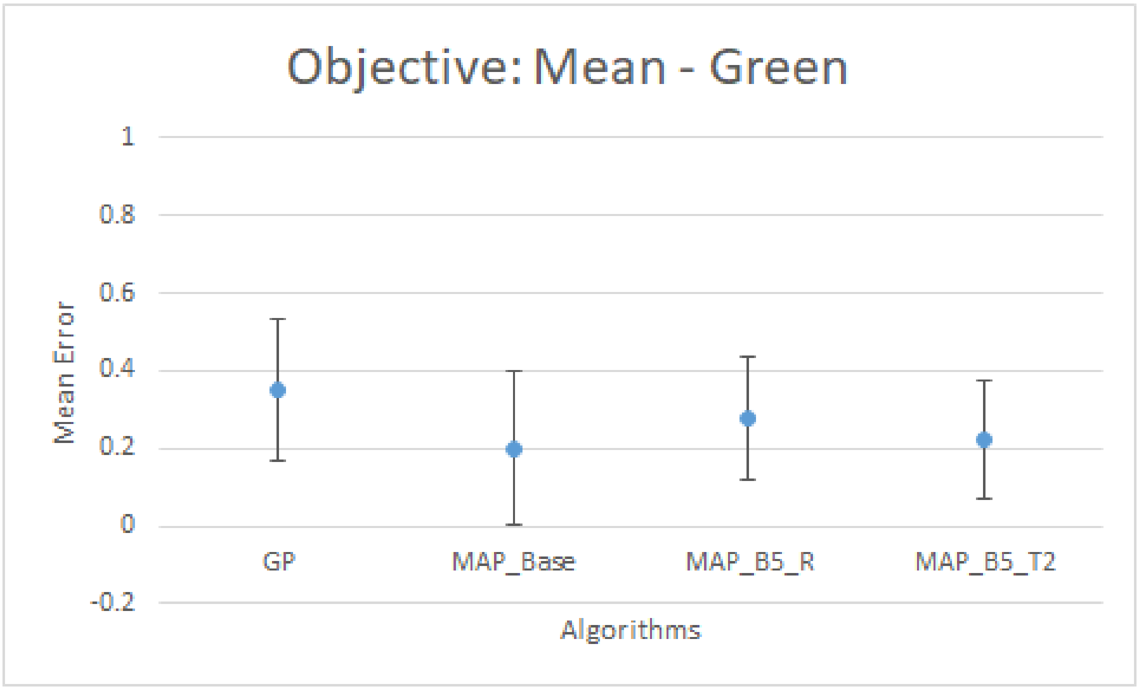


Figure 6.49: Error plot of mean mean pixel value difference of green channel for different algorithms

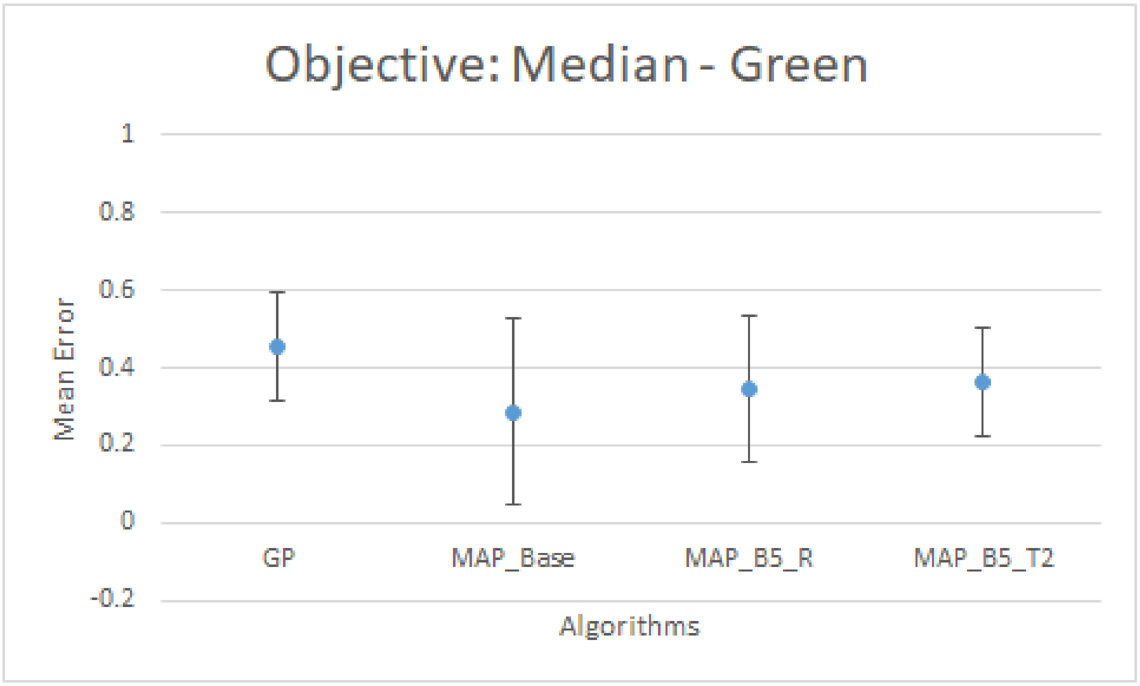


Figure 6.50: Error plot of mean median pixel value difference of green channel for different algorithms

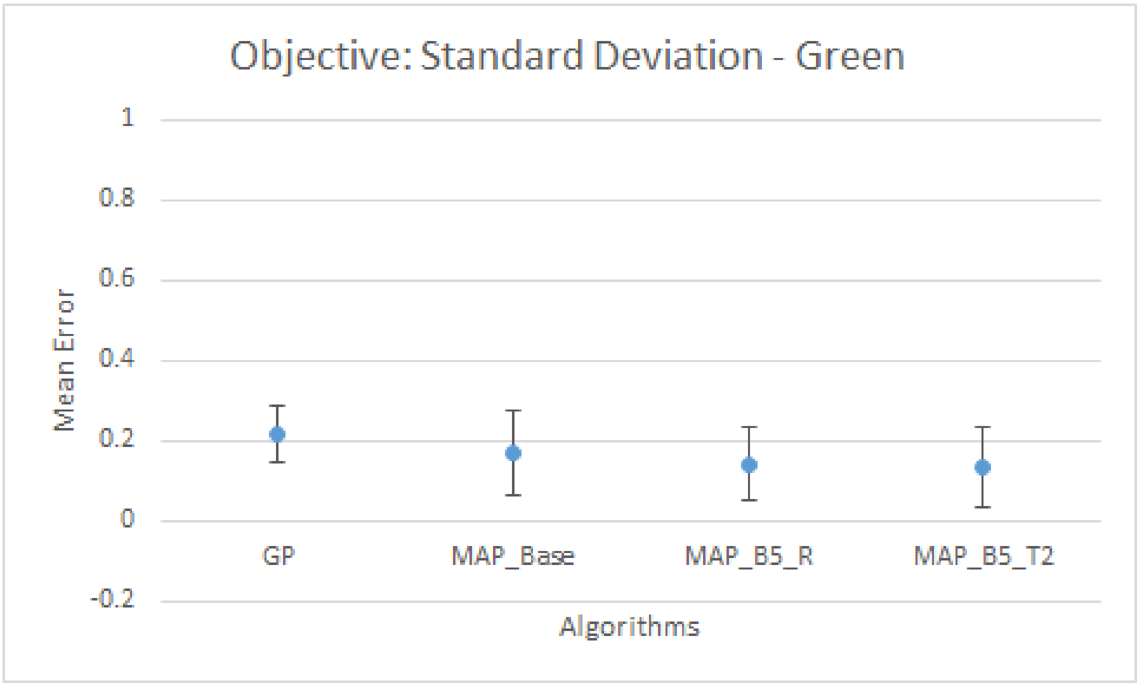


Figure 6.51: Error plot of mean standard deviation value difference of green channel for different algorithms

Figure 6.48, 6.49, 6.50, and 6.51, show the error plots with standard deviation for four objectives (max, mean, median and standard deviation) of green channel for the four algorithms used in this experiment. Error plot for the minimum objective of the red channel is not shown here as it does not have any statistical significance. When we compare these four figures mentioned earlier with Table 6.6 we can see which algorithm performs better in which objective.

For green max objective (see Figure 6.48), MAP_B5_T2 is statistically significantly better (threshold : $p<0.05$) than the MAP_B5_R. For green mean objective (see Figure 6.49), base MAP-Elites performs significantly better (threshold : $p<0.05$) than GP. For green median objectives (see Figure 6.50), MAP_B5_T2 significantly outperforms GP. Finally, for the green standard deviation the two proposed version of MAP-Elites (see Figure 6.51), performs significantly better than GP.

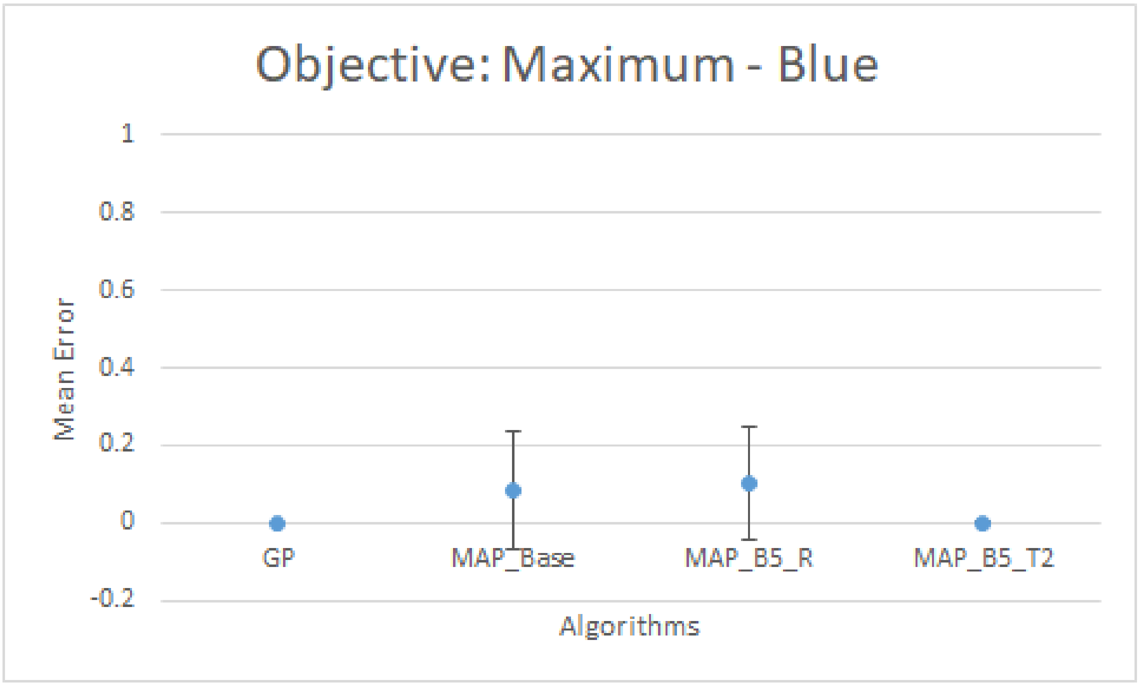


Figure 6.52: Error plot of mean maximum pixel value difference of blue channel for different algorithms

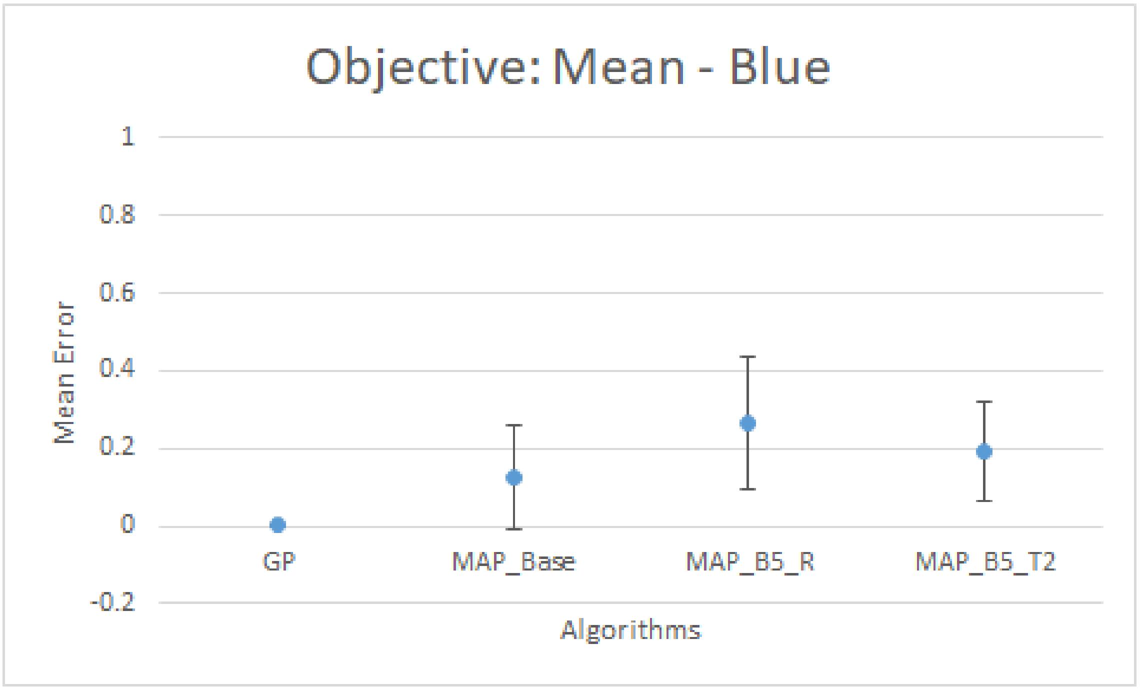


Figure 6.53: Error plot of mean mean pixel value difference of blue channel for different algorithms

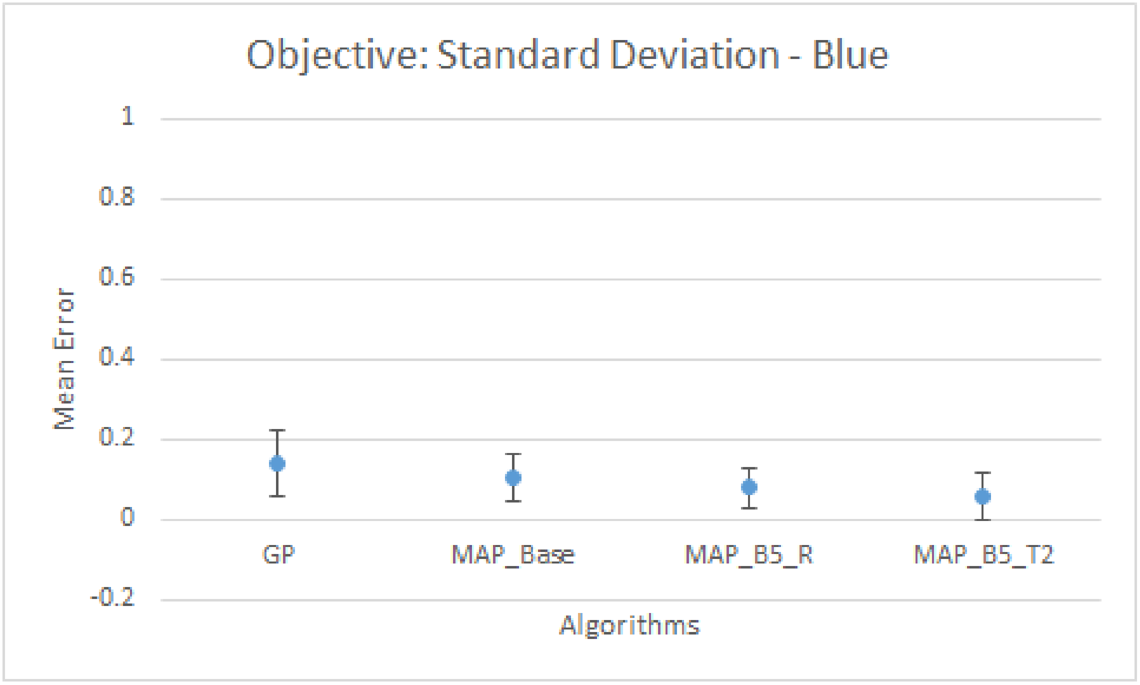


Figure 6.54: Error plot of mean standard deviation value difference of blue channel for different algorithms

Figure 6.52, 6.53, and 6.54, show the error plots with standard deviation for three objectives (max, mean and standard deviation) of green channel for the four algorithms used in this experiment. Error plot for the minimum and median objective of the blue channel is not shown here as it does not have any statistical significance. When we compare these four figures mentioned earlier with Table 6.6 we can see which algorithm performs better in which objective.

For green max objective (see Figure 6.48), GP is statistically significantly better (threshold : $p<0.05$) than the base MAP-Elites and MAP_B5_R. For blue mean objective (see Figure 6.49), GP performs significantly better (threshold : $p<0.05$) than all three MAP-Elites algorithm. Finally, for the blue standard deviation the two proposed version of MAP-Elites (see Figure 6.51), performs significantly better than GP and MAP_B5_T2 significantly outperforms MAP_B5_R.

Chapter 7

Discussion

7.1 Objective Behaviours

Table 7.1: Objectives count with significant difference with at least one other algorithm

	Objectives	**GP**	**MAP_Base**	**MAP_B5_R**	**MAP_B5_T2**
EXP - 1	5	3	5	1	2
EXP - 2	5	3	0	1	2
EXP - 3	6	4	10	2	2
EXP - 4	15	12	5	5	2
EXP - 5	15	7	3	2	7

Table 7.1 shows the number of objectives in each experiments and objective performance count. Numbers in the table represents in how many objectives one algorithm was able to significantly outperform (P-value threshold: 0.05) at least one other algorithm for that same objective. Each time an objective performed better for a algorithm compared with at least another algorithm it is given a point. Shaded cells on the table shows the best performing algorithm for that experiment in terms of maximum number of objective performance count. From this table we can see GP has the most objective performance count in two experiments, tied for best position with MAP_B5_T2 for one. Base MAP-Elites outperformed others in terms of objective performance count in two experiments and MAP_B5_R failed to get best objective performance count in any of the experiments. Keep in mind that these values does not necessarily signifies which algorithm performed the best overall. For more details on the values of Table 7.1 see Tables 5.7, 5.8, 5.9, 6.5 and 6.6 from Chapter 5 and 6.

Table 7.2: Average sum-of-ranks for all solutions from all the experiments

	Objectives/ No. of runs	Norm. average sum-of-ranks				Norm. avg. sum-of-ranks (Re-ranked avg.)				Final re-ranking			
		GP	MAP_Base	MAP_B5_R	MAP_B5_T2	GP	MAP_Base	MAP_B5_R	MAP_B5_T2	GP	MAP_Base	MAP_B5_R	MAP_B5_T2
EXP - 1	5 / 20	2.38	2.32	2.79	2.66	35.30	34.70	47.90	44.10	2.00	1.00	4.00	3.00
EXP - 2	5 / 10	2.64	2.30	2.79	2.52	20.55	18.45	23.50	19.50	3.00	1.00	4.00	2.00
EXP - 3	6 / 10	3.82	2.81	3.87	3.92	25.20	6.10	24.95	25.75	3.00	1.00	2.00	4.00
EXP - 4	15 / 10	7.19	7.92	8.29	7.77	15.80	22.30	24.30	19.60	1.00	3.00	4.00	2.00
EXP - 5	15 / 10	8.37	7.78	8.30	7.48	23.70	19.80	21.30	17.20	4.00	2.00	3.00	1.00
Grayscale experiments average sum-of-rank ranking (EXP - 1, 2, and 3) =										*2.67*	*1.00*	*3.33*	*3.00*
RGB experiments average sum-of-ranks ranking (EXP - 4 and 5) =										*2.50*	*2.50*	*3.50*	*1.50*
Overall average sum-of-ranks raking (EXP - 1 to 5) =										*2.60*	*1.60*	*3.40*	*2.40*

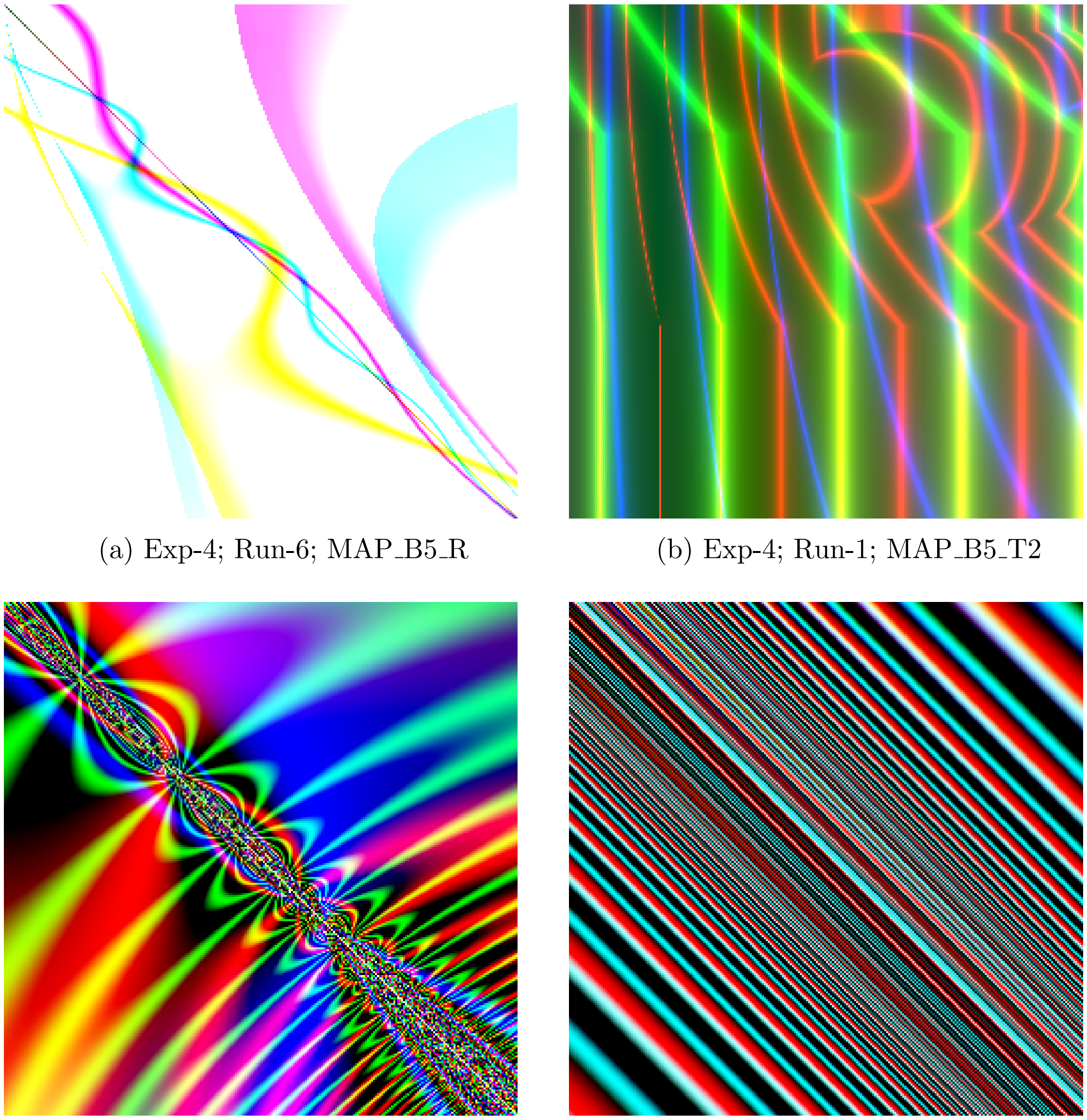
7.2 Overall Performance

Here we compare algorithms based on how well they generate optimized solutions for all objectives. To get an overall idea of the performances of the best solutions per run for each algorithms, we can look at Table 7.2. The same objective difference values of the target image and evolved images that were used to calculate the five P-value tables for five experiments, is used here to calculate the average sum-of-ranks and its variation here. Shaded cell represents the best algorithm for that experiment and its average sum-of-ranks. The second column of Table 7.2 represents the number of objectives and number of runs performed for each algorithm for that experiment.

The third column for the four algorithms shows the sum-of-ranks averaged over the mentioned number of runs showed in column two. The fourth column represents the same thing in a more friendly way (bigger difference). To get these values all the sum-of-ranks were ranked based on their values and then were averaged based on the algorithms to get the value for that experiment. For example experiment-1 has 20 runs so it produces 20 best solutions for each algorithm, making total number of solutions 80, that needs to be ranked. So, from those 80 solutions we get 80 sum-of-ranks values. These values are then re-ranked. Then these ranks are grouped and averaged based on which algorithm produced the solution. And the values in fourth set of columns represent those values. The last set of column just re-ranks the values from the previous set of columns between 1 and 4 as there are four different algorithms.

By looking at the last set of columns (last four columns) we can see that MAP elites shows best overall results in terms of best solutions in all three grayscale experiments. GP and MAP_B5_T2 each shows best overall results in terms of best solutions in one RGB experiment. If we look at the last three rows it shows the average final ranking score between the four algorithms. For the three grayscale experiment base MAP-Elites has the best average rank. For RGB MAP_B5_T2 has the best average rank. And overall base MAP-Elites shows the best average ranks and MAP_B5_T2 shows the second best. In all cases we can see that the MAP_B5_R did not perform very well statistically.

7.3 Some of Our Preferred Solutions


(a) Exp-4; Run-6; MAP_B5_R

(b) Exp-4; Run-1; MAP_B5_T2

(c) Exp-5; Run-8; MAP_B5_R

(d) Exp-5; Run-3; MAP_B5_T2

Figure 7.1: Four of our favourite evolved images from RGB experiments.

Figure 7.1 shows four of our favourite images from RGB experiments. Our MAP-Elites algorithms produced hundreds of solution images. Showing all of them in this paper is not practical. So, we only choose four images, two from each RGB experiments with the new proposed algorithms.

7.4 Challenges of the Proposed Algorithms

Some of the challenges of the proposed algorithms are listed below:

- From the discussion we can see that the random selection approach for the multiple-individual bin MAP-Elites does not perform very well.
- Though the proposed algorithms are faster than than GP, they are little slower than the original MAP-Elites algorithm. The reason is that more operations are needed to be performed for storing, removing and selecting individuals from bins.
- Having a lot of choices for higher dimension of MAP may lead to four or more dimensions of the MAP. Although with two dimensions the algorithm performs well, with higher dimensions there will be exponentially increasing number of bins. With the number of features increasing, it may not find multiple solutions for a single bin.
- A very large map with smaller intervals may also result in few individuals in single bins, making the algorithms perform similarly to the original MAP-Elites.
- It may take a lot of trial and error to find the optimum bin size for a problem. A small bin size may not help to diversify the solution compared with base MAP-Elites. Having a very large bin size will likely contribute to many lower quality solutions being produced.
- As with other many-objective optimization algorithms, there is no guarantee that adequate solutions can be found for difficult problems with many objectives.

Chapter 8

Conclusion

8.1 Conclusion

In this research we have proposed new algorithms for many-objective quality diversity search. We tested three different versions of MAP-Elites, including the original MAP-Elites, to evolve procedural texture images. We have run 5 different experiments and compared their results with each other, as well as with GP. Experiments were divided into two groups: grayscale and RGB. For the grayscale experiments we performed three different experiments, which showed that having unique gray value and entropy as MAP features gives us more diverse solution compared with mean gray value and standard deviation difference.

Grayscale experiments showed how adding just one more additional feature can have a large impact on the output (see experiment 1 and 3). So, this gives us an insight that adding useful objectives can help improve results. Compared with GP, MAP-Elites generated more diverse solutions, and MAP_B5_R and MAP_B5_T2 produced more diverse results compared with base MAP-Elites. In terms of the objective difference values of the best individuals in each run, base MAP-Elites outperformed the other three algorithms in grayscale experiments.

RGB experiments used the same set of features as objectives. But from the MAP-Elites MAP (see Figure 6.6 and 6.37), we can see that most solutions are clustered in the middle, as the solutions do not always reach the maximum possible number of unique RGB colours, keeping the right most section of the MAP mostly empty. In terms of diversity, we found a similar behaviour as was observed with grayscale experiments. But in terms of the objective values, MAP_B5_T2 outperformed the other three algorithms.

By distributing the range non-uniformly, we can spread out the middle section of the range more with a smaller range intervals, and have a large interval for the left most and right most values. This will help us get better quality and more diverse solutions for the MAP. As for the solution quality between GP and three version of MAP elites, we cannot reach any definite conclusion as the algorithms are very different in behaviour. As mentioned earlier, in terms of art, art is subjective. MAP-Elites gives us more diverse solutions making it ideal for Evo Art, as the user will have a wide range of images to choose from. The two new proposed versions of MAP-Elites bring out the benefits of evolution, where less fit individuals sometimes get a chance to be part of reproduction making the solution more diverse.

The GP behaviour we observed was expected, as GP converges over multiple generations and the results tend to have similar features. Our expectation was that the two new versions of MAP-Elites will outperform GP and base MAP-Elites in terms of diversity and best solutions. In terms of the statistical analysis, MAP_B5_T2 did well in the RGB experiments only. MAP_B5_R failed to show better statistical performance compared with other three algorithms. Which procedural texture image a user will prefer is subjective. In that regard, MAP-Elites and the proposed version of MAP-Elites can be considered to have an advantage over GP as they provide a user with more choices to select from.

Some other things did not behave as we have hoped. For example, some features are difficult to see on the MAP, as they are too subtle for human perception. The light weight features from Lombardi *et. al* [64], although efficient to compute, are not sophisticated and may have limited application to Evo Art. But the overall results show promise, and can be improved with further research.

8.2 Comparison to related work

This section compares this research with similar work elsewhere.

- **Ross and Zhu** used multi-objective optimization in their GP based procedural texture generation [78]. For fitness evaluation they used Pareto ranking.
- **Salami** used GP in her research [81]. She also used the feature sets used by Lombardi et al. [64]. She used both RGB and HSV feature sets proposed by Lombardi, and also made some custom sets from the features from those two sets. Sum-of-ranks was used for fitness evaluation.
- The new system will prove to be a better choice for handling many-objective problems and generating diverse solution as the experiments have shown. And it can be used in other multi-objective or many-objective problems as well, where a diverse set of high quality solutions are required.

8.3 Future Work

The following ideas can be considered for future work:

- We can experiment with more images with different feature values to see the resulting images.
- Add more image features as objectives. For example, add edge as one of the objectives along with the current ones.
- We can use the HSV feature set proposed by Lombardi *et al.* [64] as objectives.
- Perlin noise [10, 75] can be added to the GP language, which is a pseudo noise that aids the creation of natural appearing textures.
- Using island model GP as evolution algorithm [90].
- Choosing new features sets for the MAP-Elites MAP behaviour, and adding the current MAP features as objectives.
- Have a higher dimension MAP with more features.

- Using tournament selection instead of random selection in original MAP-Elites algorithm.
- Use different MAP sizes (ex: a larger MAP size can be used to find more diverse solutions)
- Use custom range for the current features instead of the uniform range to let the more crowded part of the MAP spread out to hold more solutions
- Weighted normalized sum-of-ranks can be used to give more weight to mean, median and standard deviation to make them more favourable as Min and Max showed more improvement than those three sets naturally.
- A human survey can be conducted to see if the human preference match with the evolved images.
- Applying these algorithms to solve other many-objective optimization and search problems.

Bibliography

[1] Breast cancer wisconsin (diagnostic) data set. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)? [Online; accessed 11-April-2021].

[2] Edge detection. https://en.wikipedia.org/wiki/Edge_detection. [Online; accessed 10-April-2021].

[3] Edge detection. https://www.mathworks.com/discovery/edge-detection.html. [Online; accessed 11-April-2021].

[4] Entropy. https://en.wikipedia.org/wiki/Entropy. [Online; accessed 11-April-2021].

[5] Entropy. https://johnloomis.org/ece563/notes/basics/entropy/entropy.html. [Online; accessed 10-April-2021].

[6] Hsl and hsv. https://en.wikipedia.org/wiki/HSL_and_HSV. [Online; accessed 09-April-2021].

[7] Image segmentation. https://www.mathworks.com/discovery/image-segmentation.html. [Online; accessed 11-April-2021].

[8] Image segmentation. https://en.wikipedia.org/wiki/Image_segmentation. [Online; accessed 11-April-2021].

[9] Les parapluies de viborg. https://commons.wikimedia.org/wiki/File:Les_Parapluies_de_Viborg.jpg. [Online; accessed 15-May-2021].

[10] Perlin noise. https://en.wikipedia.org/wiki/Perlin_noise. [Online; accessed 10-April-2021].

[11] Phoenicopterus ruber in são paulo zoo. `https://commons.wikimedia.org/wiki/File:Phoenicopterus_ruber_in_S\%C3\%A3o_Paulo_Zoo.jpg`. [Online; accessed 15-May-2021].

[12] Procedural texture. `https://en.wikipedia.org/wiki/Procedural_texture`. [Online; accessed 11-April-2021].

[13] Rgb color model. `https://en.wikipedia.org/wiki/RGB_color_model`. [Online; accessed 09-April-2021].

[14] Sobel operator. `https://en.wikipedia.org/wiki/Sobel_operator`. [Online; accessed 10-April-2021].

[15] Texture synthesis. `https://en.wikipedia.org/wiki/Texture_synthesis`. [Online; accessed 11-April-2021].

[16] F. Alamdar and M. R. Keyvanpour. A new color feature extraction method based on dynamic color distribution entropy of neighborhoods. *arXiv preprint arXiv:1201.3337*, 2012.

[17] A. Alvarez, S. Dahlskog, J. Font, and J. Togelius. Empowering quality diversity in dungeon design with interactive constrained amp-elites. In *2019 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2019.

[18] A. Alvarez, J. M. M. F. Fernandez, S. Dahlskog, and J. Togelius. Interactive constrained map-elites: Analysis and evaluation of the expressiveness of the feature dimensions. *IEEE Transactions on Games*, 2020.

[19] J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. C. Jain, and C. F. Shu. Virage image search engine: an open framework for image management. In *Storage and Retrieval for Still Image and Video Databases IV*, volume 2670, pages 76–87. International Society for Optics and Photonics, 1996.

[20] M. Baniasadi. Genetic programming for non-photorealistic rendering. Master's thesis, Brock University, St. Catharines, ON, Canada, 2013.

[21] P. J. Bentley and D. W. Corne. *Creative Evolutionary Systems*. Morgan Kaufmann, 2002.

[22] P. J. Bentley and J. P. Wakefield. Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. In *Soft Computing in Engineering Design and Manufacturing*. Springer Verlag, 1997.

[23] S. Bergen and B. J. Ross. Evolutionary art using summed multi-objective ranks. In *Genetic Programming Theory and Practice VIII*, pages 227–244. Springer, 2011.

[24] Y. Bi, B. Xue, and M. Zhang. Genetic programming with a new representation to automatically learn features and evolve ensembles for image classification. *IEEE Transactions on Cybernetics*, 51(4):1769–1783, 2020.

[25] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[26] J. Branke, K. Deb, K. Miettinen, and R. Slowiński. *Multiobjective Optimization: Interactive and Evolutionary Approaches*, volume 5252. Springer, Berlin Heidelberg, 2008.

[27] P. A. Bromiley, N. A. Thacker, and E. Bouhova-Thacker. Shannon entropy, renyi entropy, and information. *Statistics and Inf. Series (2004-004)*, 2004.

[28] S. Brown. Retrievalware from excalibur technologies and finds a toolkit for the creation of information servers. *TIP Applications*, 9:4–5, 1996.

[29] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

[30] W. Chen, Y. Q. Shi, and G. Xuan. Identifying computer graphics using hsv color model and statistical moments of characteristic functions. In *2007 IEEE International Conference on Multimedia and Expo*, pages 1123–1126. IEEE, 2007.

[31] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems.* Kluwer, 2nd edition, 2007.

[32] C. Colas, V. Madhavan, J. Huizinga, and J. Clune. Scaling map-elites to deep meuroevolution. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 67–75, 2020.

[33] D. W. Corne and J. D. Knowles. Techniques for highly multiobjective optimisation: Some nondominated points are better than others. In *Proc. GECCO 2007*, pages 773–780. ACM Press, 2007.

[34] R. Dawkins. *The Blind Watchmaker.* Norton & Company, Inc, 1986.

[35] E. Dolson, A. Lalejini, and C. Ofria. Exploring genetic programming systems with map-elites. In *Genetic Programming Theory and Practice XVI*, pages 1–16. Springer, 2019.

[36] S. Elaoud, T. Loukil, and J. Teghem. The pareto fitness genetic algorithm: Test function study. *European Journal of Operational Research*, 177(3):1703–1719, 2007.

[37] M. Flageat and A. Cully. Fast and stable map-elites in noisy domains using deep grids. In *Artificial Life Conference Proceedings*, pages 273–282. MIT Press, 2020.

[38] P. J. Fleming, R. C. Purshouse, and R. J. Lygoe. Many-objective optimization: An engineering design perspective. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 14–32. Springer, 2005.

[39] M. C. Fontaine, S. Lee, L. B. Soros, F. D. M. Silva, J. Togelius, and A. K. Hoover. Mapping hearthstone deck spaces through map-elites with sliding boundaries. In *Proceedings of The Genetic and Evolutionary Computation Conference*, pages 161–169, 2019.

[40] F. Fortin, F. M. D. Rainville, M. A. Gardner, M. Parizeau, and C. Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

[41] M. Garza-Fabre, G. T. Pulido, and C. A. C. Coello. Ranking methods for many-objective optimization. In *Mexican International Conference on Artificial Intelligence*, pages 633–645. Springer, 2009.

[42] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison Wesley, 1989.

[43] J. Gomes, P. Mariano, and A. L. Christensen. Devising effective novelty search algorithms: A comprehensive empirical study. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 943–950, 2015.

[44] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*, volume 3. Prentice Hall, 2002.

[45] J. Graf and W. Banzhaf. Interactive Evolution of Images. In *Proc. Intl. Conf. on Evolutionary Programming*, pages 53–65, 1995.

[46] P. A. Hajare and P. A. Tijare. Edge detection techniques for image segmentation. *International Journal of Computer Science and Applications*, 4(1), 2011.

[47] R. M. Haralick and L. G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29(1):100–132, 1985.

[48] C. R. Harris, K. J. Millman, S. J. V. D. Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. V. Kerkwijk, M. Brett, A. Haldane, J. F. D. Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K.n Sheppard, T. Reddy, W. Weckesser, H. Abbasi, . Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585:357—-362, 2020.

[49] K. Héberger and K. Kollár-Hunek. Sum of ranking differences for method discrimination and its validation: Comparison of ranks with random numbers. *Journal of Chemometrics*, 25(4):151–158, 2011.

[50] E. D. .Heijer and A. E. Eiben. Comparing aesthetic measures for evolutionary art. In *European Conference on the Applications of Evolutionary Computation*, pages 311–320. Springer, 2010.

[51] J. H. Holland. An introductory analysis with applications to biology, control, and artificial intelligence. *Adaptation in Natural and Artificial Systems. First Edition, The University of Michigan, USA*, 1975.

[52] A. E. M. Ibrahim. Evolutionary techniques for procedural texture automation. In *International Symposium on Visual Computing*, pages 623–632. Springer, 2013.

[53] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization by nsga-ii and moea/d with large populations. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 1758–1763. IEEE, 2009.

[54] H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization: A short review. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 2419–2426. IEEE, 2008.

[55] C. R. Johnson, E. Hendriks, I. J. Berezhnoy, E. Brevdo, S. M. Hughes, I. Daubechies, J. Li, E. Postma, and J. Z. Wang. Image processing for artist identification. *IEEE Signal Processing Magazine*, 25(4):37–48, 2008.

[56] N. Justesen, S. Risi, and J. B. Mouret. Map-elites for noisy domains by adaptive sampling. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 121–122, 2019.

[57] P. Kamavisdar, S. Saluja, and S. Agrawal. A survey on image classification approaches and techniques. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(1):1005–1009, 2013.

[58] A. Khalifa, S. Lee, A. Nealen, and J. Togelius. Talakat: Bullet hell generation through constrained map-elites. In *Proceedings of The Genetic and Evolutionary Computation Conference*, pages 1047–1054, 2018.

[59] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, volume 1. MIT Press, 1992.

[60] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*, volume 5. Springer Science & Business Media, 2006.

[61] J. Lehman and K. O. Stanley. Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. In *Proc. Artificial Life XI*, pages 329–336, 2008.

[62] J. Lehman and K. O. Stanley. Novelty search and the problem with objectives. In *Genetic Programming Theory and Practice IX*, pages 37–56. Springer, 2011.

[63] B. Li, J. Li, K. Tang, and X. Yao. Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 48(1):1–35, 2015.

[64] T. Lombardi, S. Cha, and C. Tappert. A lightweight image retrieval system for paintings. In *Storage and Retrieval Methods and Applications for Multimedia 2005*, volume 5682, pages 236–246. International Society for Optics and Photonics, 2005.

[65] W. Ma and B. S. Manjunath. Netra: A toolbox for navigating large image databases. *Multimedia Systems*, 7(3):184–198, 1999.

[66] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, pages 50–60, 1947.

[67] O. Marques and B. Furht. Content-based visual information retrieval. In *Distributed Multimedia Databases: Techniques and Applications*, pages 37–57. IGI Global, 2002.

[68] W. McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[69] P. E. McKnight and J. Najab. Mann-whitney u test. *The Corsini Encyclopedia of Psychology*, pages 1–1, 2010.

[70] B. L. Miller and D. E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9(3):193–212, 1995.

[71] J. B. Mouret. Novelty-based multiobjectivization. In *New Horizons in Evolutionary Robotics*, pages 139–154. Springer, 2011.

[72] J. B. Mouret and J. Clune. Illuminating search spaces by mapping elites. *ArXiv*, abs/1504.04909, 2015.

[73] J. B. Mouret and S. Doncieux. Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evolutionary Computation*, 20(1):91–133, 2012.

[74] P. Ngatchou, A. Zarei, and M. A. El-Sharkawi. Pareto multi objective optimization. In *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, pages 84–91. IEEE, 2005.

[75] K. Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

[76] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza. *A Field Guide to Genetic Programming.* Lulu. com, 2008.

[77] C. Poynton. *Digital video and HD: Algorithms and Interfaces.* Elsevier, 2012.

[78] B. J. Ross and H. Zhu. Procedural texture evolution using multi-objective optimization. *New Generation Computing*, 22(3):271–293, 2004.

[79] G. V. Rossum and F. L. Drake. *Python 3 Reference Manual.* CreateSpace, Scotts Valley, CA, 2009.

[80] Y. Rui, T. S. Huang, and S. Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10(1):39–62, 1999.

[81] E. Salimi. Statistical image analysis for image evolution. Master's thesis, Brock University, St. Catharines, ON, Canada, 2016.

[82] E. Samuelsen and K. Glette. Multi-objective analysis of map-elites performance. *arXiv preprint arXiv:1803.05174*, 2018.

[83] M. W. Schwarz, W. B. Cowan, and J. C. Beatty. An experimental comparison of rgb, yiq, lab, hsv, and opponent color models. *ACM Transactions on Graphics (TOG)*, 6(2):123–158, 1987.

[84] M. Sharifi, M. Fathy, and M. T. Mahmoudi. A classified and comparative study of edge detection algorithms. In *Proceedings. International Conference on Information Technology: Coding and Computing*, pages 117–120. IEEE, 2002.

[85] K. Sims. Artificial evolution for computer graphics. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, pages 319–328, 1991.

[86] J. R. Smith and S. Chang. Tools and techniques for color image retrieval. In *Storage and Retrieval for Still Image and Video Databases IV*, volume 2670, pages 426–437. International Society for Optics and Photonics, 1996.

[87] D. G. Stork. Computer vision and computer graphics analysis of paintings and drawings: An introduction to the literature. In *International Conference on Computer Analysis of Images and Patterns*, pages 9–24. Springer, 2009.

[88] M. A. Stricker and M. Orengo. Similarity of color images. In *Storage and Retrieval for Image and Video Databases III*, volume 2420, pages 381–392. International Society for Optics and Photonics, 1995.

[89] D. Tarapore, J. Clune, A. Cully, and J. B. Mouret. How do different encodings influence the performance of the map-elites algorithm? In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 173–180, 2016.

[90] D. Whitley, S. Rana, and R. B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7(1):33–47, 1999.

[91] A. L. Wiens and B. J. Ross. Gentropy: Evolutionary 2d texture generation. *Computers and Graphics Journal*, 26(1):75–88, 2002.

[92] Y. Wu, Y. Zhou, G. Saveriades, S. Agaian, J. P. Noonan, and P. Natarajan. Local shannon entropy measure with statistical tests for image randomness. *Information Sciences*, 222:323–342, 2013.

[93] F. Xue, A. C. Sanderson, and R. J. Graves. Pareto-based multi-objective differential evolution. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, volume 2, pages 862–869. IEEE, 2003.

[94] S. L. Yadav and A. Sohal. Comparative study of different selection techniques in genetic algorithm. *International Journal of Engineering, Science and Mathematics*, 6(3):174–180, 2017.

[95] D. X. M. Zheng, S. T. Ng, and M. M. Kumaraswamy. Applying pareto ranking and niche formation to genetic algorithm-based multiobjective time–cost optimization. *Journal of Construction Engineering and Management*, 131(1):81–91, 2005.

Appendix A

Additional Analysis

A.1 System Testing

Table A.1: GP parameter for breast cancer experiment

Parameter Name	**Value**
Population Size	1000
No. of Generation	50
Tree Initializer	Half and Half
Initial Tree Depth	Min: 2 Max: 6
Max Tree Depth	17
Crossover	100%
Mutation	20%
Selection	Tournament
Tournament Size	3
Training Data Size	30%
Test Data Size	70%
Number of Runs	10

To test our MAP-Elites algorithm implementation, we have run several experiments. For base a benchmark the traditional GP system is been used [40]. Using the same dataset and same GP parameters multiple experiments have been performed with different MAP-Elites parameters. 10 runs have been performed for each experiments. Table A.1 shows the GP parameters used for the experiments.

Table A.2: GP function set for breast cancer experiment

Function Name	**Description**
add	Add two numbers
sub	Subtract two numbers
mul	Multiply two numbers
protectedDiv	Divide one number by another.
and	Logical AND operation between two numbers
or	Logical OR operation between two numbers
not	Logical NOT operation on a numbers
lt	Checks if (number-1) < (number-2)
eq	Checks if two numbers are equal
if_then_else	Conditional statement for decision making

For testing purposes the *breast cancer* data set from Wisconsin (diagnostic) is used [1]. This dataset consists of 32 attributes including 30 floating point features value, the actual diagnostic data and 1 identification number. The value of the identification number does not contribute to the determination of breast cancer and is ignored. All the values of the dataset (except the identification number) are used as terminals of the GP system. Ephemeral constant has also been used as an additional terminal. Table A.2 shows the GP function set and Table A.3 shows the terminal set used in this experiment.

For the MAP-Elites system, the same set of GP parameters are used. Different MAP sizes are used. A 2D MAP of size 5×5, 10×10, and 17×20 are used. An initial batch size (number of new individuals created randomly in the first iteration to initialize the MAP) was 80% of each MAP size. Bin sizes used are 1 and 5 for different experiments.

Table A.3: GP terminal set for breast cancer experiment

Terminal	**Details**
Mean_Radius	Mean radius
Mean_Texture	Mean texture
Mean_Perimeter	Mean perimeter
Mean_Area	Mean area
Mean_Smoothness	Mean smoothness
Mean_Compactness	Mean compactness
Mean_Concavity	Mean concavity
Mean_ConcavePoints	Mean concave points
Mean_Isymmetry	Mean symmetry
Mean_FractalDimension	Mean fractal dimension
STE_Radius	Radius standard error
STE_Texture	Texture standard error
STE_Perimeter	Perimeter standard error
STE_Area	Area standard error
STE_Smoothness	Smoothness standard error
STE_Compactness	Compactness standard error
STE_Concavity	Concavity standard error
STE_ConcavePoints	Concave points standard error
STE_Isymmetry	Symmetry standard error
STE_FractalDimension	Fractal dimension standard error
Largest_Radius	Largest radius
Largest_Texture	Largest texture
Largest_Perimeter	Largest perimeter
Largest_Area	Largest area
Largest_Smoothness	Largest smoothness
Largest_Compactness	Largest compactness
Largest_Concavity	Largest concavity
Largest_ConcavePoints	Largest concave points
Largest_Isymmetry	Largest symmetry
Largest_FractalDimension	Largest fractal dimension
rand1	Ephemeral constant.

Feature 1 for the X-axis of the MAP is the tree size. So, the range of feature 1 was between 1 and 17. Feature 2 for the Y-axis is the number of nodes of the tree, which we have found in different runs never exceeds above 400. So, we have set feature 2 range between 1 and 400. The fitness domain range was between 0 and infinity. The fitness function measures how many data example it was able to classify correctly. The MAP showed in Figure A.1 is from one of the experiment runs of 17×20 size MAP which has 5 items per bin. The X-axis of the map represents depth of the tree and the Y-axis represents total number of nodes in the tree. From the figure we can see that almost half of the bins in the MAP has never been used, meaning no solution was found for those bins. By looking at the experiment we can see the reason for this. A tree with very low depth can not have very high number of nodes. Similarly, tree with very high depth can not have just a couple of nodes.

Table A.4 shows the result of different runs with different MAP-Elites parameters. We can see from the table what impact changing some of the MAP-Elites parameters can have on results. All of the values are averaged over 10 runs. Filled bins column represents how many final solutions are on the MAP has at the end of the run. MAP-Elites has generated over 160 solutions with different fitness values.

Another thing we can see that when we have 5 items per bin, the fitness value is lower compared to 1 items per bin. This is because those bins can store up to 5 best solutions found for that feature space in the MAP and then takes all those 5 values into account while calculating the average fitness. On the other hand, the single item bins only store the best solution for that feature space, resulting in a higher average fitness value.

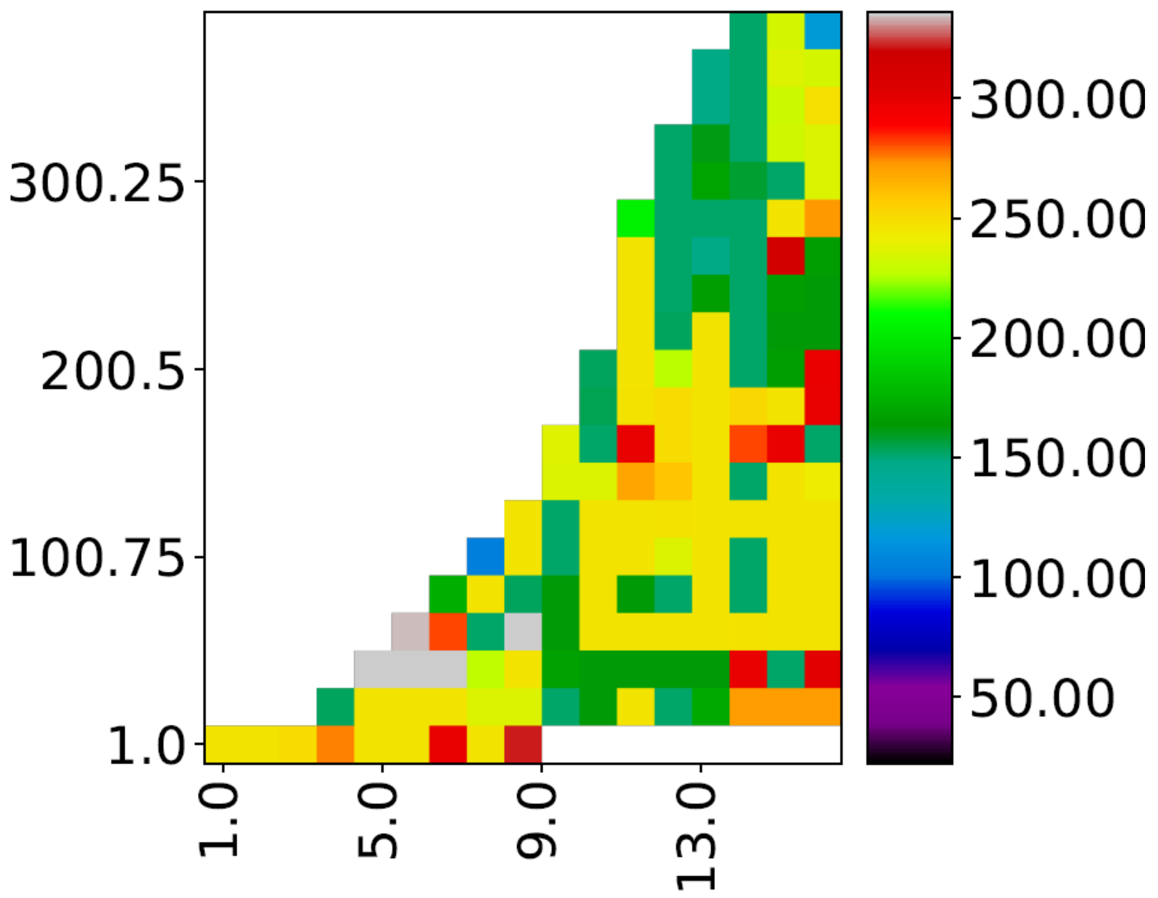


Figure A.1: A single run 17 × 20 MAP of breast cancer experiment [X-axis: tree depth (range: 1 - 17); Y-axis: total number of nodes (range: 1 - 400)].

We can understand the fitness diversity of the solutions generated by MAP-Elites by looking at the average fitness value of the MAP. With a larger size MAP, we get more diverse solutions which reduces the average fitness value. In this experiment the tree size and number of nodes has been used as determining feature on the MAP. Where GP was only able to produce 1 solution per run the MAP-Elites produced many solutions per run.

Table A.4: Result of 10 run average of GP system and MAP-Elites system

Algo.	**MAP Size**	**Init. Batch**	**Bin Size**	**Batch Size**	**Iterations**	**Filled Bins**	**Avg Fitness**
Base GP	N-A	1000	N-A	N-A	50	N-A	366
MAP Elites	5×5	20	1	1	50000	5	380
				10	5000	10	384
			5	1	50000	1	220
				10	5000	4	207
	10×10	80	1	1	50000	16	374
				50	1000	47	378
			5	1	50000	14	204
				50	1000	45	231
	17×20	272	1	1	50000	41	369
				50	1000	160	369
			5	1	50000	39	226
				50	1000	161	216

From Table A.4 we can see how different MAP sizes, bin sizes and batch sizes affects the number of solutions found and the quality of solutions. More solutions means more diverse solutions. To make the comparison fair, the total number of evaluated individuals are kept the same for all the experiments. We can see from the table that the average fitness of MAPs with bin size 5 is lower than the average fitness of MAPs with bin size 1. This is because single-individual bins only stores the best solution for that feature space, where multiple-individual bins here are storing up to five solutions which may not have the same fitness as the best individual in that bin. But during the average fitness calculation, those less fit individuals are also taken into account, resulting a lower fitness average for multiple-individual bin MAPs.