Newcastle University

Bio-computing Project Report

Muran Tang

200638292

Abstract

The genetic algorithm is popular in computing. The bio-computing project focuses on understanding and using the evolution algorithm. This project report describes the strategies used in the image evolution algorithm based on nature evolution. This report represents the results achieved, give critically evaluates the outcome and provides future focus of the project.

1. Project Development

In this part, the focus will be on what strategies are used and how they are implemented during developing the complicated vector art project. The description will be divided into three parts, design of the reference algorithm, design of variants and design of the parameters' experiments.

1.1 Design of Reference Algorithm

In this project, the mu plus lambda algorithm (DEAP Library, 2022) is used as the reference algorithm. The algorithm contains two problems. One problem is that, as generations increase, the amount of computation required per generation will grow exponentially. Another problem is that the algorithm only converges to a certain point, with accuracy fluctuating around 0.965. The design of the reference algorithm (see Algorithm 1) is to tackle the problems. To solve the problem of computing power surges, a restriction is added to the offspring creation method in the algorithm. After mating, the number of polygons in an individual should not be more than the given individual size. If the individual contains over a certain number of polygons, keep deleting a polygon randomly chosen in the individual until the number of polygons is under the restriction. The insufficient convergence accuracy problem of the evolution can be fixed to a certain degree by implementing the evolution struggle detective strategy (ESDS), parameter dynamic change strategy (PDCS) and change stage strategy (CSS) (see Figure 1). When the average fitness of a population reaches a certain value, which is the fitness value that the evolution exceeds and does not go up anymore (see Figure 2), ESDS will be used (see Algorithm 1 [18]). If the standard deviation of 5 consecutive generations is lower than 1E-15, the PDCS will be to change the mutation possibility and crossover possibility in certain ranges and the CSS will be activated which will change the evolution stage from the natural evolution stage (NES) to the artificial evolution stage (AES).

1.2 Design of Variants

There are three variants included in the evolution which are solution representation (generation and storage of a polygon), offspring generation (crossover, mutation, reproduction), and selection function.

1.2.1 Solution Representation

Solution representation focuses on polygon production and storage. During creating a polygon, a redundancy check is applied. Based on the given edge (see Table 1), the same number of points will be generated. In the point generating process, after generating a new point, the point will be checked whether it is already in the polygon and if it is, the point will be discarded and keep generating a point until it is not in the polygon. A colour range restriction strategy is used in generating a new polygon. Before starting the evolution, the target image will be split into different analysis zones (see Figure 4) and the colour range of each zone will be analyzed and stored. The

program will calculate the zones that the points are in and based on that it randomly generates RGBA values for each point. Then, one of the RGBA values will be randomly chosen to be the colour of the polygon. The colour and points of a polygon will be stored in a list. The list size of a polygon is not fixed, and the structure contains tuples of RGBA and points, such as [(RGBA), (point), …, (point)].

1.2.2 Offspring Generation

Evolution is dependent on offspring generation. There are three ways to generate offspring generation, crossover, mutation and representation.

One way uses crossover to generate the offspring. Mating function and individual size restriction of offspring individual is applied. The design of crossover is randomly choosing two individuals from the current population and applying a uniform crossover to crossover the polygons from the parents. The reason for using uniform crossover is that it has an overall 0.2% better performance than others (see Figure 3).

Second way to create the offspring is using mutation. The design of mutation is more complex and has three mutation stages, choosing sub-mutation, choosing a target individual and mutating.

Before activating any sub-mutation method, an activated possibility of each sub-method will be generated and if the possibility is less than the given then activated the method. A stage detected strategy is applied in some sub-mutations. Some sub-mutations have two ways to do the mutation. Before choosing a mutation target (individual, polygon, point), the current stage will be detected and based on the stage choosing the right mutation way (see Figure 5). There are six sub-mutation methods applied, point mutation, colour changing, point addition, point deletion, polygon addition, polygon deletion and reorder the polygons. The directed evolution strategy (DES) and sub-mutation dynamical control (SDC) are implemented in some of the methods. The DES is that when the evolution in AES randomly picks a polygon and either gets the point in the shortest edge of the polygon to mutate or re-generate a colour for the polygon in a specific colour range (see Figure 6,7). The SDC is to change the possibility of sub-mutation (SMPB) based on the current individual fitness. There are two cases of SDC. Case one (SDCI) is that the SMPB will increase as the fitness increasing. Another one (SDCD) is the SMPB will decrease as the fitness increasing. By applying those strategies, change-limited evolution is achieved while preserving randomness. Firstly, the sub-mutation method of mutating a point is implemented. Point mutation only happen in NES. It is randomly selecting a point from a randomly chosen polygon from the given individual and swapping the x, y coordinates. Second, the sub-mutation of mutating colour is applied. In NES, the Gaussian mutation will be used to mutate the colour of a randomly chosen polygon. When artificial evolution begins, using DES, all the colour ranges will be extracted from the zones that the polygon points in and the colour will be generated from a randomly chosen colour range of the colour ranges. Third, sub-mutation is a point addition with using the redundancy check and SDC. The possibility of executing this mutation is in case SDCI. A new point will be directly generated with given coordinates bounds (see Table 1) in NES. During the AES, the DES will be used in generating a new coordinate bound for randomly creating a new point, which effectively

controls the change of the area of the polygon. Fourth, sub-mutation of point deletion is implemented. Because of the nature of a polygon, a polygon must contain at least three points. Therefore, the sub-mutation only is executed when the number of points in the polygon is greater than three. A point will be randomly deleted without any restriction in NES. In contrast, the DES will be used to select a point to delete in AES. Fifth, sub-mutation of polygon addition and deletion are implemented together. The SMPB of these mutations are SDCD. The reason for applying this is that, after mating, the number of polygons in an individual might be decreased. In this project, polygon addition has higher priority than polygon deletion since having more polygons is fundamental to improving the fineness of the image. A polygon will be added only when the number of polygons in the given individual is less than the given individual size (see Table 1) and the polygon making will also follow the rule of making a polygon which has been mentioned previously. The polygon deletion is executed only and only if adding a polygon is not true and the number of polygons is greater than 0.85 times the given individual size and the polygon will be randomly chosen to delete. The final sub-mutation is reordering the polygons in the individual by shuffling the index of the polygons in the list of the individual. The SMPB of this mutation is SDCD because it has the most effective of changing the individual fitness.

Another way to produce the offspring generation is reproduction. An individual from the current population will be randomly chosen to be a new individual in the offspring generation.

1.2.3 Selection Function

The selection function affects the performance of the evolution. In this project, four selection functions were experienced by implementing the same parameters and the evolution algorithm (see Figure 2). With the showing result, the select toursize function has the best convergence, reached 0,96. Therefore, the function is used as the selection function of the evolution.

1.3 Design of the Parameters' Experiments

There are eighteen parameters used in this project (see Table 1). The method used to design the experiment of finding the optimal parameters is the control variate method. The standard of choosing an optimal parameter value contains two aspects. The most important one is that the parameter should get the highest average fitness at the final generation than another group. Second, it should use less generation time to reach 0.95 fitness. The experimental principle is not to increase the computational pressure. The experiment takes group number two of parameters as a control group. For getting an optimal value for each parameter, only one parameter field is changed in an experiment group (see Table 2). After experimenting a parameter, the results from the experiment group and the control group will be analyzed together by visualizing the experiment's data (see Figure 8a, 8b). The experiment starts from the parameters which will not increase computation pressure. Therefore, the experiments on population size, individual size and lambda were the last to experiment with because those will change computation pressure.

2. Results and Evaluation

In the parameter optimisation experiments, a total of 11 sets of experiments were carried out, each varying one or two variables (10 sets of experiments varying only one parameter and the other varying two variables) (see Table 2, Figure 8 - 17). In each set of experiments, two to four different values were tested for one parameter. The experiments used the No. 2 parameters group as a control group. The results of the experiments were divided into three categories, those with an increase in fitness, those with essentially no change in fitness and those with a large change in fitness. First, the group with an increase in fitness was ind_size, which increased by 0.0789% when its value was changed from 100 to 200. Second, the experimental groups with essentially unchanged fitness were indpb, a_max, lambda, seed and sigma. By changing the values of the test variables, the effect on the final fitness was less than 0.5%. Third, the experimental groups with large changes in fitness were pop_size, toursize, edge, mu, cxpb, mutpb and chance. In each group there was a difference in fitness greater than 0.5% from the control group. The difference in fitness was greater than 0.5%. The analysis revealed that changes in pop_size and mu had the greatest effect on fitness, with the maximum effect being close to 4%. In particular, when pop_size and mu were changed from 10 to 20, fitness decreased by 3.78%. The resulting optimal parameter values are shown in the group No. 2. Although fitness increases for an ind_size of 200, the range of increase is very small. In order not to increase the amount of computation needed to evolve each generation, an ind_size of 100 was used as the optimal solution.

The evolution performance has been improved by making improvements to the referencing algorithm. Firstly, by applying the offspring restriction, the code can run for more than 1000 generations. Secondly, by using ESDS, the number of struggle generations dropped significantly (see Figure 21). Before using ESDS, the individuals of generations in range R7 to R11 were very similar, showing almost no evolutionary trend. After appling ESDS, the percentage of the standard division, less than 1E-15, was significantly reduced. The impact was most pronounced in the evolution of range R7, where the percentage of struggle generation decreased by 30%. Although the problem of evolutionary struggle is alleviated, more than half of the individuals in the generations from R4 to R14 were still very similar, suggesting that the population is still not evolving efficiently.

In this project, three images of varying complexity were used as target images. For the easy level of the target image (see Figure 18a), the average fitness is around 0.97(see Figure19a, 20a and Table 3) and the best fitness is 0.9728505882352941, which took 1,500,000 generations. With the same generation time, the median level of the target image (see Figure 18b) only has the average fitness around 0.96 (see Table 4) and the best fitness is 0.9628423529411765. The average fitness of the advance level target image (see Figure 18c) is around 0.95 (see Figure19b,20b and Table 5) and the best result is 0.9512427450980392. By comparing the results with the target image, it was found that the results are still missing a lot of detail. For instance, the face detail is missing in the easy level image (see Figure 18a). This algorithm has better performance of images with large colour blocks when compared to images of different complexity.

3. Conclusions

In this project, the aim is to gain a general understanding of the evolution algorithm, have basic experience in image machine learning and practice the skills of data processing and analysis. During the development of the evolution algorithm, awareness of target image preprocessing is contained, which will analyze the evolutionary complexity of each region of the target and apply a DES during the evolution. Data processing is used in visualizing the data of parameters experiment, which denoise the original data for each experiment's graphs.

There are some highlights of the project. One is that many strategies are used in this algorithm to improve the performance of the evolution. The ESDS, PDCS and CSS improve the standard division of each generation. A special function is included in this project, which is breakpoint renewal capability. The function will store the final population into a csv file and the population will be able to reload to do another evolution. Another is splitting the target image, which is the key to effectively improving the evolution. However, some defects are contained. In the parameters experiment, too few subjects are contained for each experiment group, which made a large experimental interval since the optimum parameters might be the local best instead of a global one. The reason that how CXPB and MUTPB affect fitness has not researched (see Figure 23). In the meantime, the pair relationship is not studied. For example, in this case, population size and individual size might be banded to study how their effect the evolution performance.

Although the fitness of the evolution reaches 0.95 in around 4000 generations, the rate of increasing fitness is continuously decreasing after that, and the evolution contains a problem of losing the detail of the target. Therefore, some improvements can be applied, and some directions can be researched in the future. First, in the image analysis, using edge detection to get the major features of the target (see Figure 22) and based on that, split the target into two aspects, the main feature and background, which might prove an opportunity of getting a more accurate complexity analyzation later. Find the large and less complex area as the target to generate first because Steven and Brian (2011) mentioned in the article that 'larger areas contribute more to fitness than smaller areas'. For evolution the main feature, the mask processing (Steven and Brian, 2011) can be used to keep details. Second, a model can be made for predicting the future fitness because it is unrealistic to run the program with 1,000,000 times generation in each experiment and the evolution performance got in a limited generations can be inaccurate because of the volatility of the evolution (see Figure 2). Third, the gene cloud model (Tao, 2018) can be implemented to store a group of polygons of a population.

References

Bergen, S. & Ross, B.J. (2011) Automatic and interactive evolution of vector graphics images with genetic algorithms. The Visual computer. 28 (1), 35–45.

Wu, T. (2018) Image-Guided Rendering with an Evolutionary Algorithm Based on Cloud Model. Computational intelligence and neuroscience. 20184518265–19.

DEAP Library (2022, Jan). Complete Algorithms. Available at https://deap.readthedocs.io/en/master/api/algo.html#complete-algorithms (Accessed: 8th May, 2022)

Appendices

Input The parameters
Output The final population, evolution information log and rendering images
[1] Initial the header of evolution information.
[2] Evaluate the individuals with an individual fitness.
[3] Evolution information recoding start.
[4] Initial struggle time count
[5] **repeat generational process**
[6] **for** gen = 1 -> N_{gen} **do**
[7] Vary the population
[8] Evaluate the individuals with an invalid fitness
[9] Update the fame hall with the generate individuals
[10] Combine the population and the offspring individuals
[11] **if** (current stage == artificial evolution stage) **then**
[12] Select the best individual
[13] Select the next generation using initial select function
[14] **if** (current stage == artificial evolution stage) **then**
[15] Put the best individual in to the population [0]
[16] Set evolution stage to nature evolution stage
[17] Update the statistics with the new population
[18] **if** (population standard < 1e - 15) **then**
[19] Add 1 into struggle time count
[20] **else**
[21] Set struggle time count back to 0
[23] Set evolution stage to nature evolution stage
[24] Reset mutpb and cxpb
[25] **if** (straggle time count >= 5) **then**
[26] Set evolution stage to artificial evolution stage
[27] Add mutpb with random number in range (0.25, 0.30)
[28] The cxpb is assigned with 1 - mutpb
[29] **if** (gen % 100 == 0) **then**
[30] Save the image of the best individual
[31] **endfor**

Algorithm 1：The design of the reference algorithm.

```
# start orthogenesis
if struggle_time_count >= 5:
    zones['artificial_evolution'] = True
    mutpb = mutpb + random.uniform(0.25, 0.30)
    cxpb = 1 - mutpb
```

Figure 1 : The example of dynamic changing parameter in design the reference algorithm

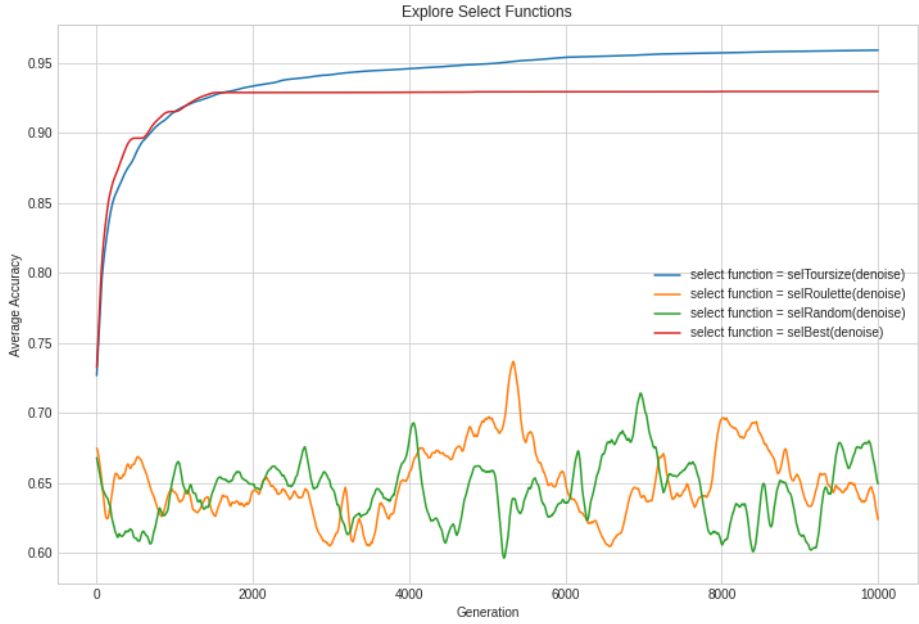


Figure 2 : The example of the performance of using different select functions. The graph of select best function is the example of unexpected convergence. It shows the volatility of the evolution.

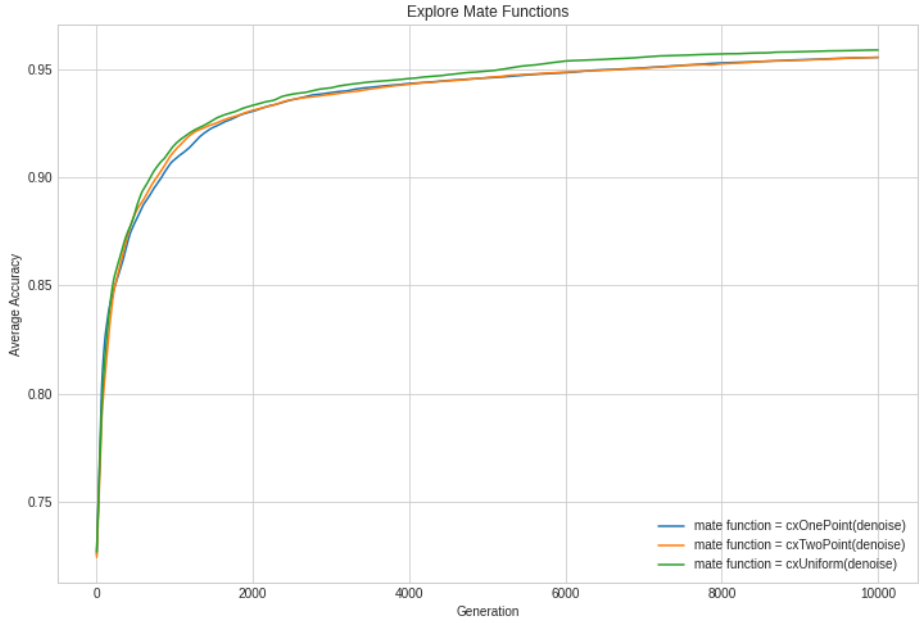


Figure 3 : The example of the performance of the three differences mating function. The final average fitness of one-point and two-point crossover is around 0.957 and the uniform crossover has around 0.959 fitness.

Type	Parameter	Value	Desciption
Indivudual stage	ind_size	100	The maximum size of polygon that each individual can contain.
	indpb	0.5	The possibility of change each polygon in mutGaussian function.
	a_min	10	The lower bound of transparency of Alpha in RGBA.
	a_max	80	The upper bound of transparency of Alpha in RGBA.
	co_min	0	The lower bound of the coordinate can generate for each point in a polygon.
	co_max	199	The upper bound of the coordinate can generate for each point in a polygon.
	edge	3	Default edge for each polygon in an individual.
	sigma	10	The amplitude of distribution in mutGaussian function.
	chance	0.5	The chance to execute the different mutation script in mutation function.
Evolution stage	pop_size	10	The original population size.
	toursize	5	The number of individuals participating in each tournament.
	mu	10	The number of individuals to select for the next generation.
	lambda	50	The number of children to produce at each generation.
	cxpb	0.4	The probability that an offspring is produced by crossover.
	mutpb	0.5	The probability that an offspring is produced by mutation.
	ngen	10000	The number of generation.
	seed	31	The seed value for random function.
Result storagy satge	k	1	The number of select the best individuals in the final generation to store the images.

Table 1 : The parameter configuration in the evolution algorithm. All the parameters are optimized

.

（a）

Zone 1 3210	Zone 2 3912	Zone 3 3454
Zone 4 2722	Zone 5 3967	Zone 6 2377
Zone 7 2588	Zone 8 2430	Zone 9 1804

(b)

Figure 4 : The example of splitting target image into nine smaller pieces. For each piece which is shown as a cell on the image, it has its given name (a). Show the color richness in different zones(b). Basing on the color in each range, color range will be generated for each zone.

```
elif op_choice < cxpb + mutpb:  # Apply mutation
    from complicated_vector_art import zones
    if zones['artificial_evolution']:
        ind = toolbox.clone(random.choice(tools.selBest(population, k=3)))
    else:
        ind = toolbox.clone(random.choice(population))
    ind, = toolbox.mutate(ind)
    del ind.fitness.values
    offspring.append(ind)
```

Figure 5 ： The example of stage detected strategy. The artificial evolution field in zones dictionary is default as false and it will be set to true until the evolution reaches a certain value.

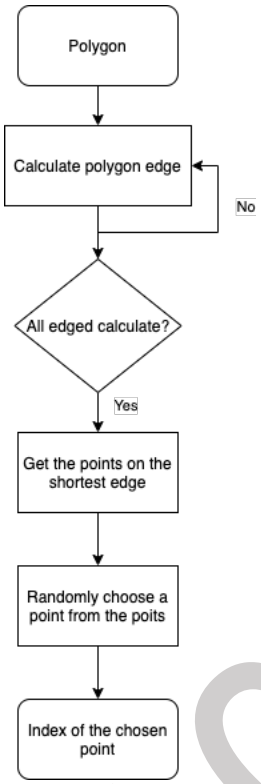


Figure 6 : The example of select the point which might affect the polygon less in the directed evolution strategy.

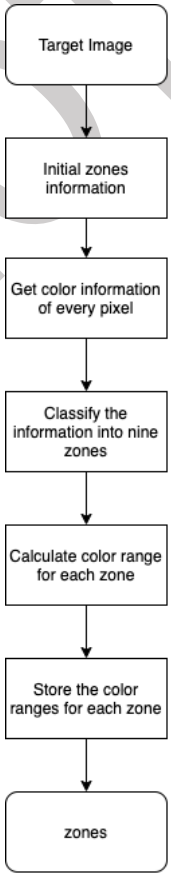
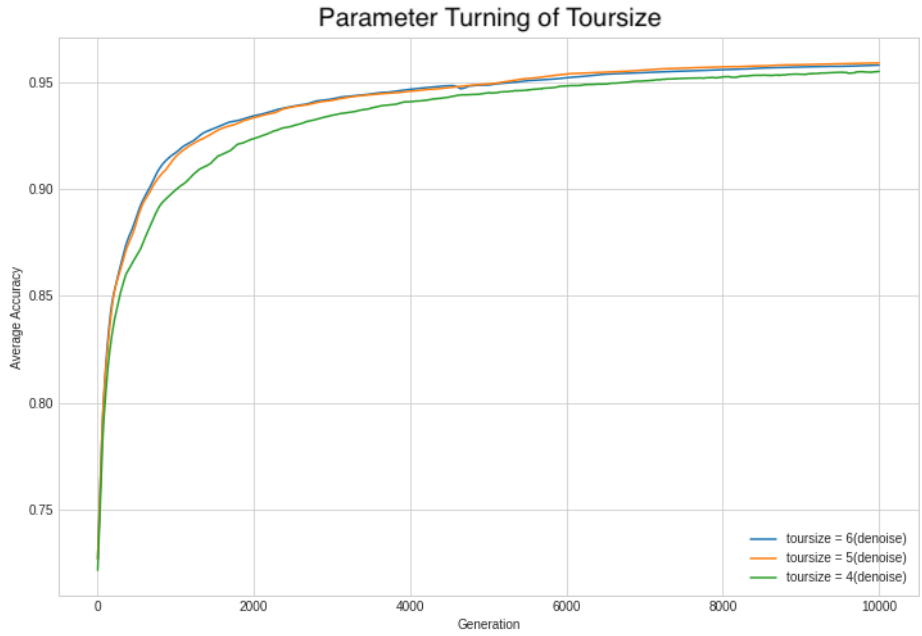


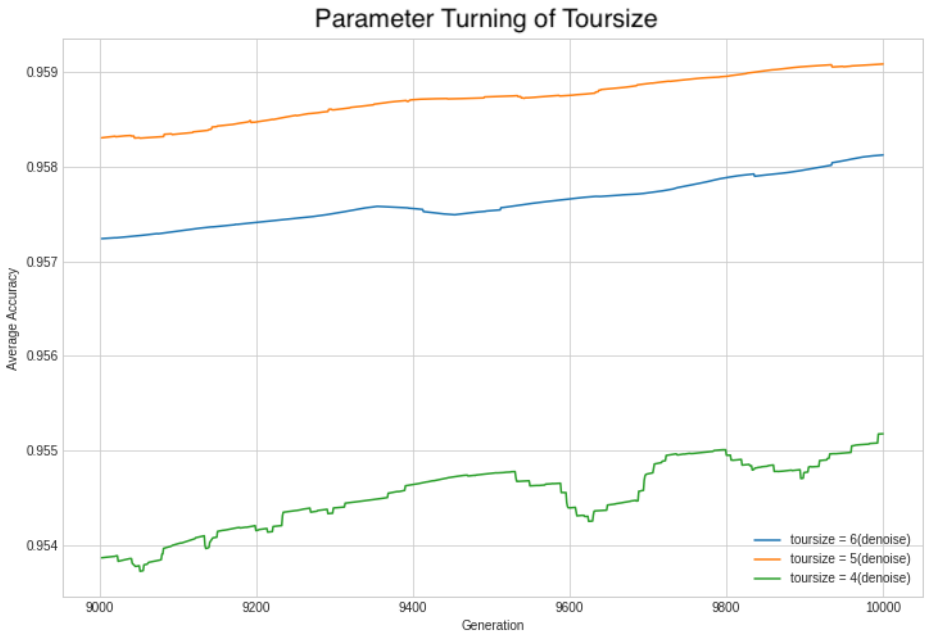
Figure 7 : The example of analysis of target image before evolution begin.

No.	ind_size	pop_size	indpb	toursize	a_min	a_max	co_min	co_max	edge	mu	lambda	cxpb	mutpb	ngen	k	seed	sigma	chance	avg	accuracy difference
1	100	10	0.5	6	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.5	95.8143%	-0.0975%
2	100	10	0.5	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.5	95.9118%	0.0000%
3	100	10	0.5	4	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.5	95.5196%	-0.3923%
4	100	10	0.5	3	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.5	95.0463%	-0.8656%
5	100	10	0.5	5	10	70	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.5	95.7030%	-0.2088%
6	100	10	0.5	5	10	100	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.5	95.6609%	-0.2509%
7	100	10	0.5	5	10	120	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.5	95.7573%	-0.1545%
8	100	10	0.5	5	10	140	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.5	95.7547%	-0.1571%
9	100	10	0.5	5	10	80	0	199	4	10	50	0.4	0.5	10000	1	31	10	0.5	95.6456%	-0.2662%
10	100	10	0.5	5	10	80	0	199	5	10	50	0.4	0.5	10000	1	31	10	0.5	95.4197%	-0.4921%
11	100	10	0.5	5	10	80	0	199	6	10	50	0.4	0.5	10000	1	31	10	0.5	95.3513%	-0.5605%
12	100	10	0.5	5	10	80	0	199	3	10	50	0.5	0.5	10000	1	31	10	0.5	95.5315%	-0.3803%
13	100	10	0.5	5	10	80	0	199	3	10	50	0.3	0.5	10000	1	31	10	0.5	95.3108%	-0.6010%
14	100	10	0.5	5	10	80	0	199	3	10	50	0.2	0.5	10000	1	31	10	0.5	93.5792%	-2.3326%
15	100	10	0.5	5	10	80	0	199	3	10	50	0.4	0.4	10000	1	31	10	0.5	94.8683%	-1.0435%
16	100	10	0.5	5	10	80	0	199	3	10	50	0.4	0.6	10000	1	31	10	0.5	95.7904%	-0.1214%
17	100	10	0.4	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.5	95.7092%	-0.2026%
18	100	10	0.6	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.5	95.8893%	-0.0225%
19	100	10	0.7	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.5	95.6887%	-0.2231%
20	100	10	0.5	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	5	0.5	95.4441%	-0.4677%
21	100	10	0.5	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	15	0.5	95.5078%	-0.4040%
22	100	10	0.5	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.4	95.8202%	-0.0916%
23	100	10	0.5	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.6	95.2780%	-0.6338%
24	100	10	0.5	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.7	94.8510%	-1.0608%
25	150	10	0.5	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.5	95.8271%	-0.0847%
26	200	10	0.5	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.5	95.8700%	-0.0418%
27	250	10	0.5	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	31	10	0.5	95.9907%	0.0789%
28	100	10	0.5	5	10	80	0	199	3	10	100	0.4	0.5	10000	1	31	10	0.5	95.8861%	-0.0257%
29	100	10	0.5	5	10	80	0	199	3	10	150	0.4	0.5	10000	1	31	10	0.5	95.7247%	-0.1871%
30	100	10	0.5	5	10	80	0	199	3	10	200	0.4	0.5	10000	1	31	10	0.5	95.4770%	-0.4348%
31	100	20	0.5	5	10	80	0	199	3	20	50	0.4	0.5	10000	1	31	10	0.5	92.1229%	-3.7889%
32	100	30	0.5	5	10	80	0	199	3	30	50	0.4	0.5	10000	1	31	10	0.5	95.1367%	-0.7751%
33	100	40	0.5	5	10	80	0	199	3	40	50	0.4	0.5	10000	1	31	10	0.5	94.3334%	-1.5784%
34	100	10	0.5	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	21	10	0.5	95.4199%	-0.4920%
35	100	10	0.5	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	11	10	0.5	95.4718%	-0.4400%
36	100	10	0.5	5	10	80	0	199	3	10	50	0.4	0.5	10000	1	41	10	0.5	95.6548%	-0.00257

Table 2 : The table of the detail of experience of finding the optimal parameters. The control group is No.2. The cells marked with the same color is an experiment group of the parameter in the column. The annalistic is based on analysis the performance of the experiment group and control group.

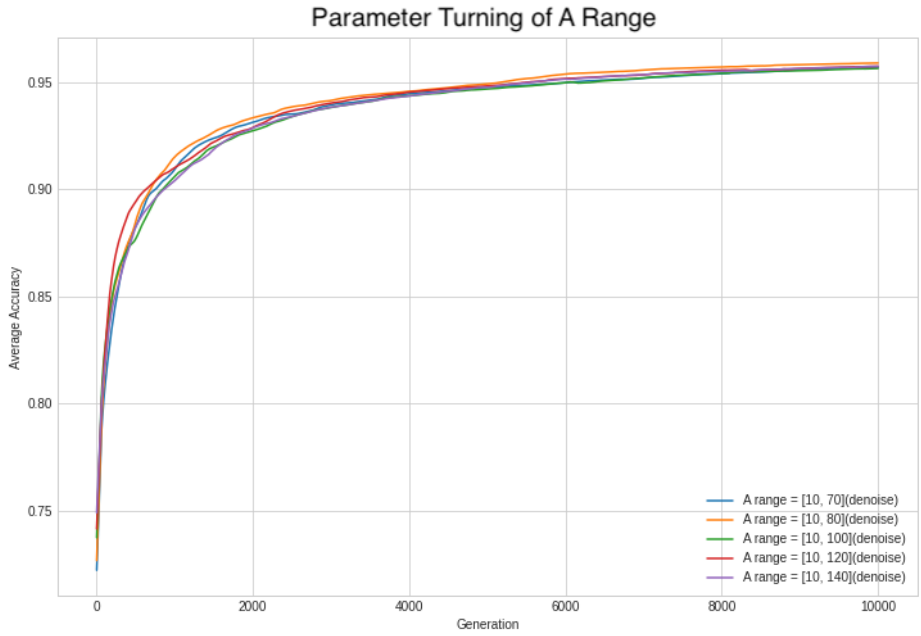

Parameter Turning of Toursize
Average Accuracy
Generation
0.95
0.90
0.85
0.80
0.75
0
2000
4000
6000
8000
10000
toursize = 6(denoise)
toursize = 5(denoise)
toursize = 4(denoise)

(a)

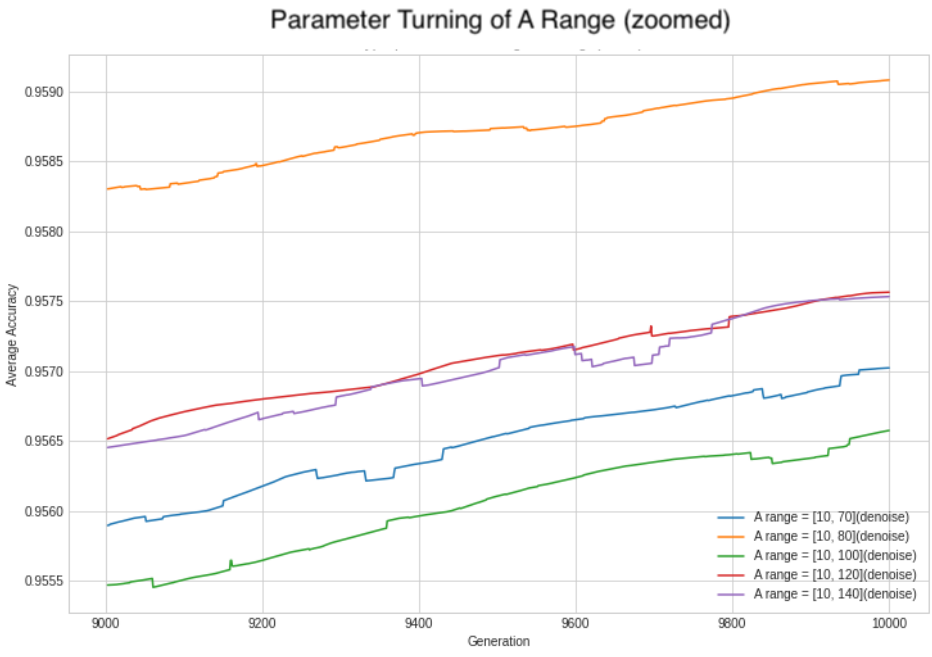

Parameter Turning of Toursize
Average Accuracy
Generation
0.959
0.958
0.957
0.956
0.955
0.954
9000
9200
9400
9600
9800
10000
toursize = 6(denoise)
toursize = 5(denoise)
toursize = 4(denoise)

(b)

Figure 8

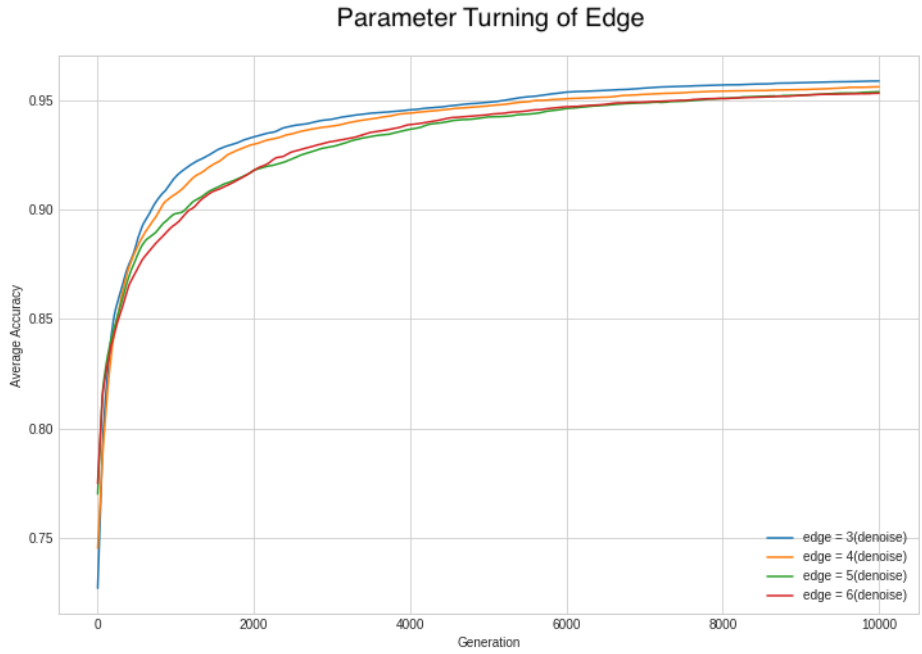

Parameter Turning of A Range
Average Accuracy
Generation
A range = [10, 70](denoise)
A range = [10, 80](denoise)
A range = [10, 100](denoise)
A range = [10, 120](denoise)
A range = [10, 140](denoise)

(a)

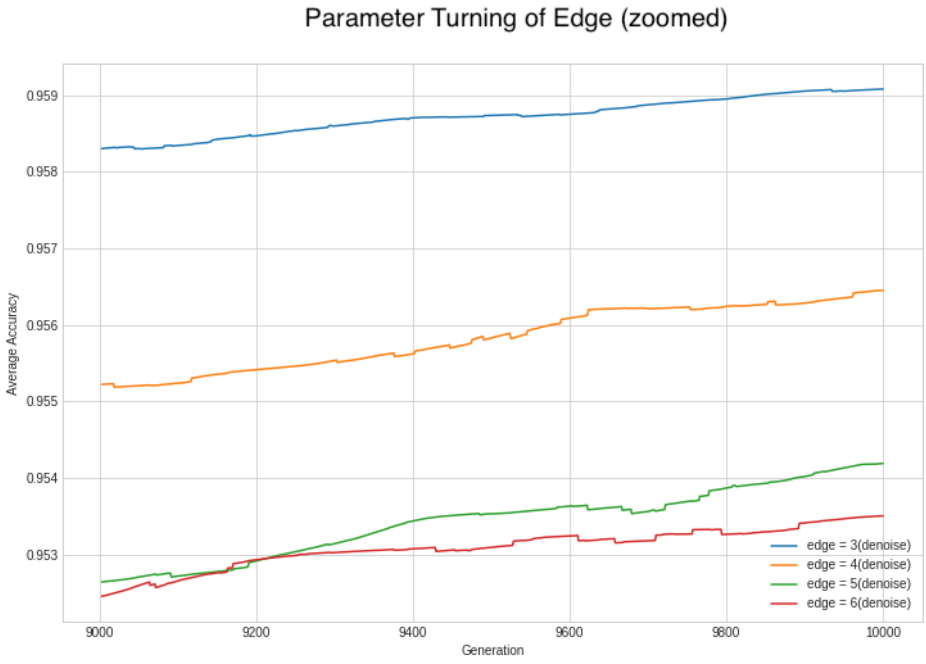

Parameter Turning of A Range (zoomed)
Average Accuracy
Generation
A range = [10, 70](denoise)
A range = [10, 80](denoise)
A range = [10, 100](denoise)
A range = [10, 120](denoise)
A range = [10, 140](denoise)

(b)

Figure 9


Parameter Turning of Edge
Average Accuracy
Generation
0.95
0.90
0.85
0.80
0.75
0
2000
4000
6000
8000
10000
edge = 3(denoise)
edge = 4(denoise)
edge = 5(denoise)
edge = 6(denoise)

(a)


Parameter Turning of Edge (zoomed)
Average Accuracy
Generation
0.959
0.958
0.957
0.956
0.955
0.954
0.953
9000
9200
9400
9600
9800
10000
edge = 3(denoise)
edge = 4(denoise)
edge = 5(denoise)
edge = 6(denoise)

(b)

Figure 10

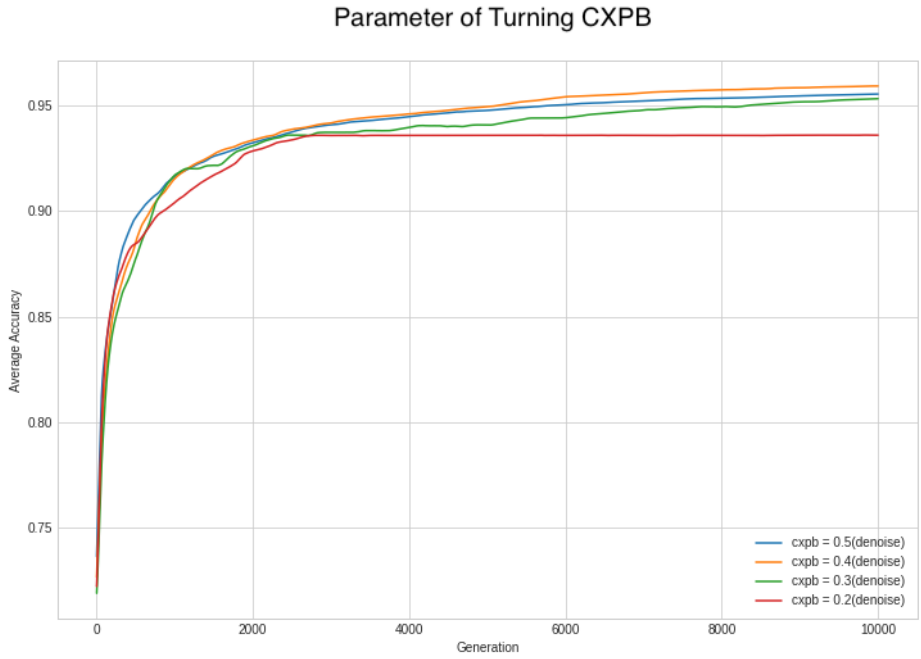
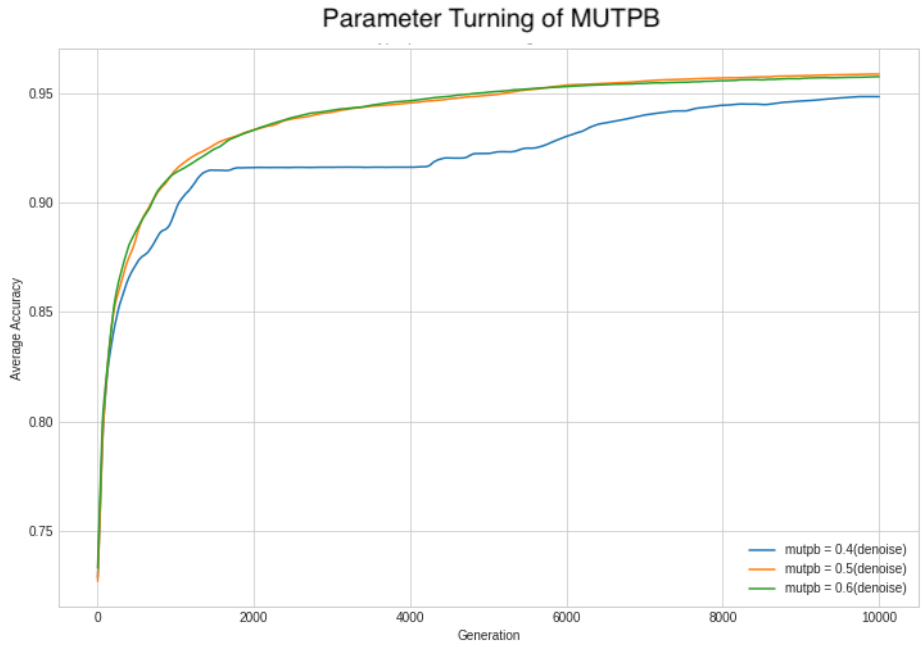
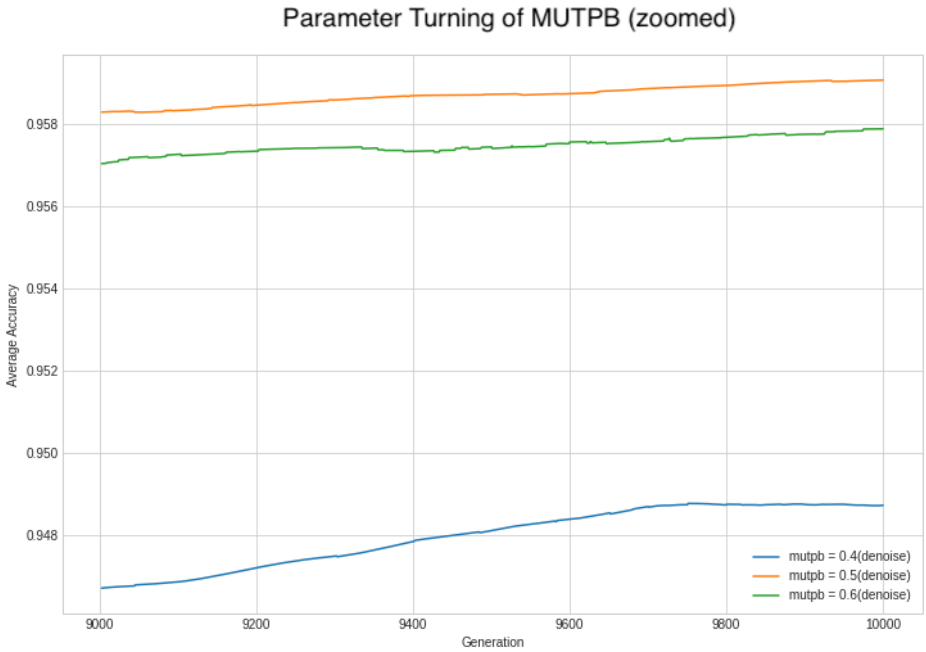


Figure 11

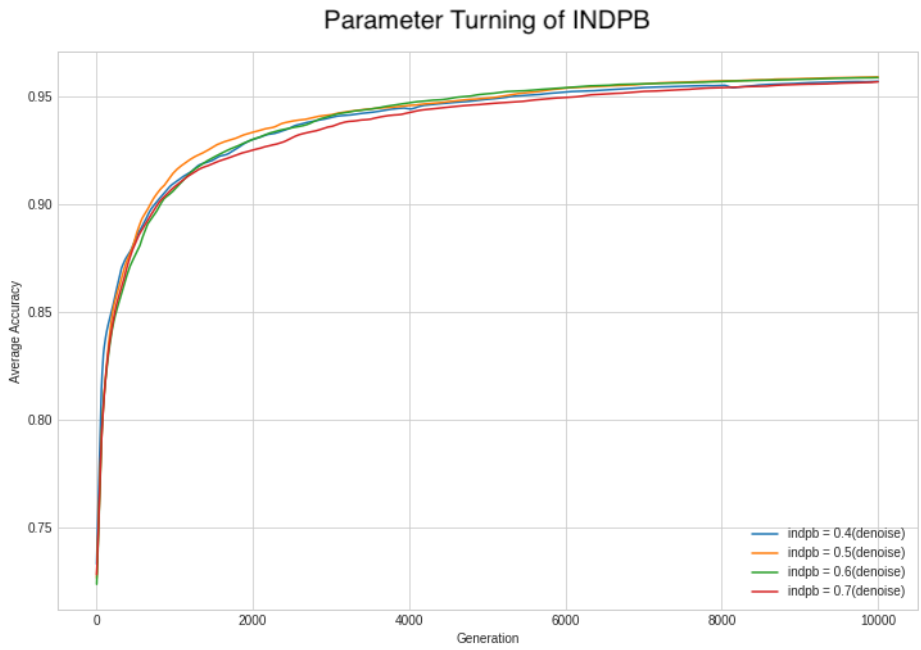

Parameter Turning of MUTPB
Average Accuracy
Generation
0.95
0.90
0.85
0.80
0.75
0
2000
4000
6000
8000
10000
mutpb = 0.4(denoise)
mutpb = 0.5(denoise)
mutpb = 0.6(denoise)

(a)

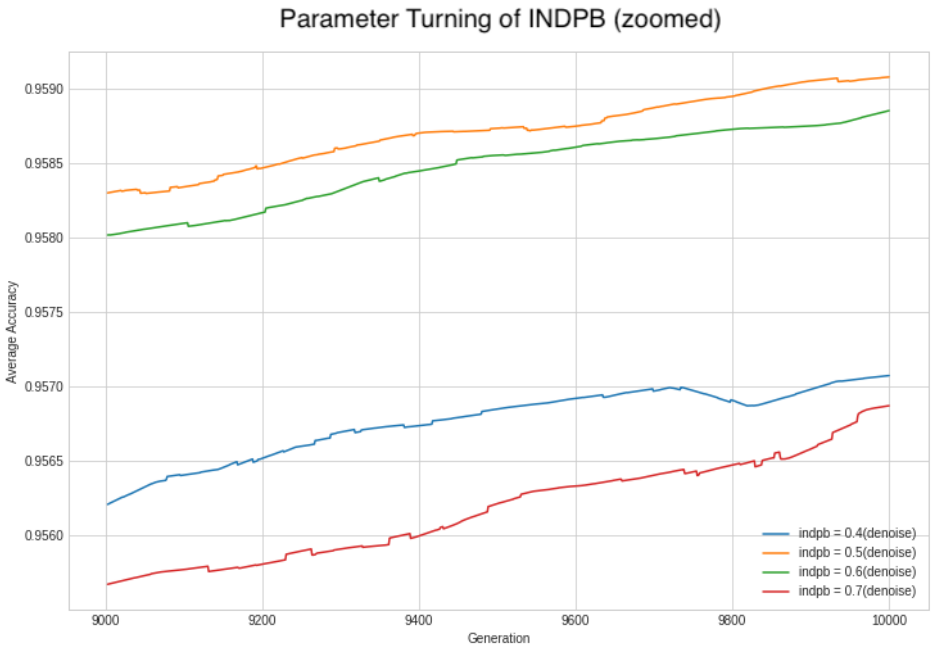

Parameter Turning of MUTPB (zoomed)
Average Accuracy
Generation
0.958
0.956
0.954
0.952
0.950
0.948
9000
9200
9400
9600
9800
10000
mutpb = 0.4(denoise)
mutpb = 0.5(denoise)
mutpb = 0.6(denoise)

(b)

Figure 12


Parameter Turning of INDPB
Average Accuracy
Generation
0.95
0.90
0.85
0.80
0.75
0
2000
4000
6000
8000
10000
indpb = 0.4(denoise)
indpb = 0.5(denoise)
indpb = 0.6(denoise)
indpb = 0.7(denoise)

(a)


Parameter Turning of INDPB (zoomed)
Average Accuracy
Generation
0.9590
0.9585
0.9580
0.9575
0.9570
0.9565
0.9560
9000
9200
9400
9600
9800
10000
indpb = 0.4(denoise)
indpb = 0.5(denoise)
indpb = 0.6(denoise)
indpb = 0.7(denoise)

(b)

Figure 13

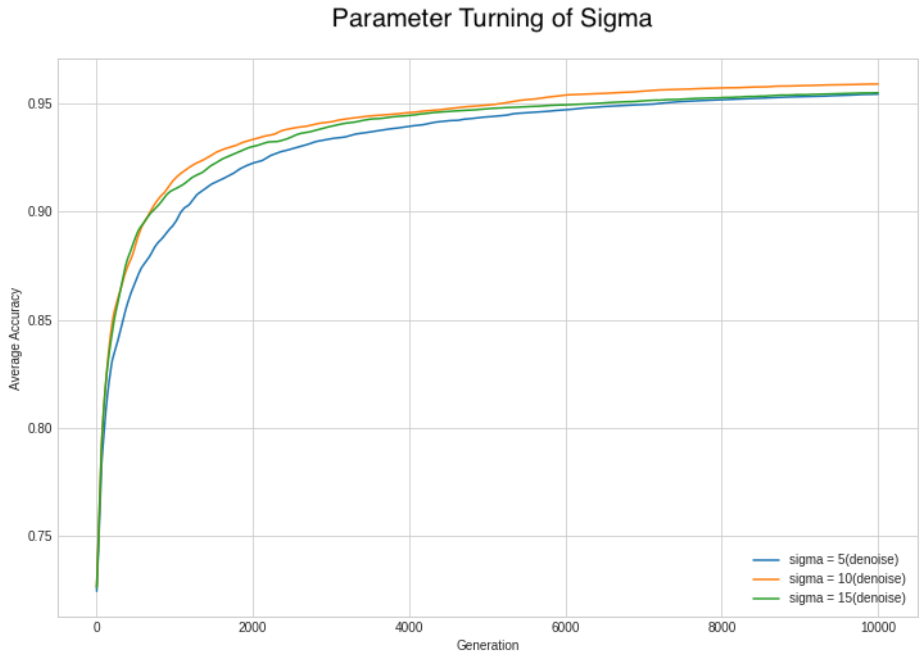
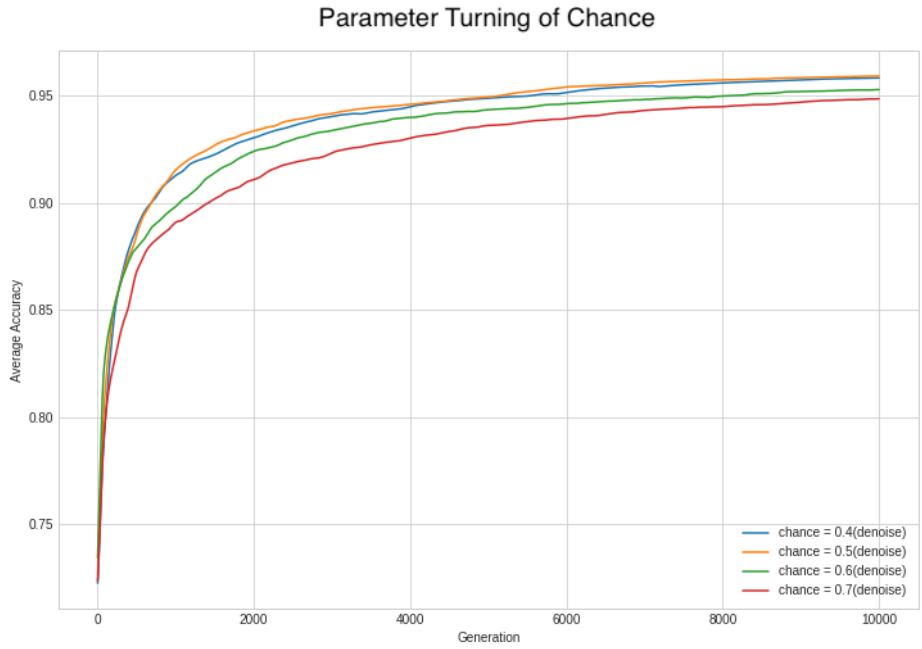
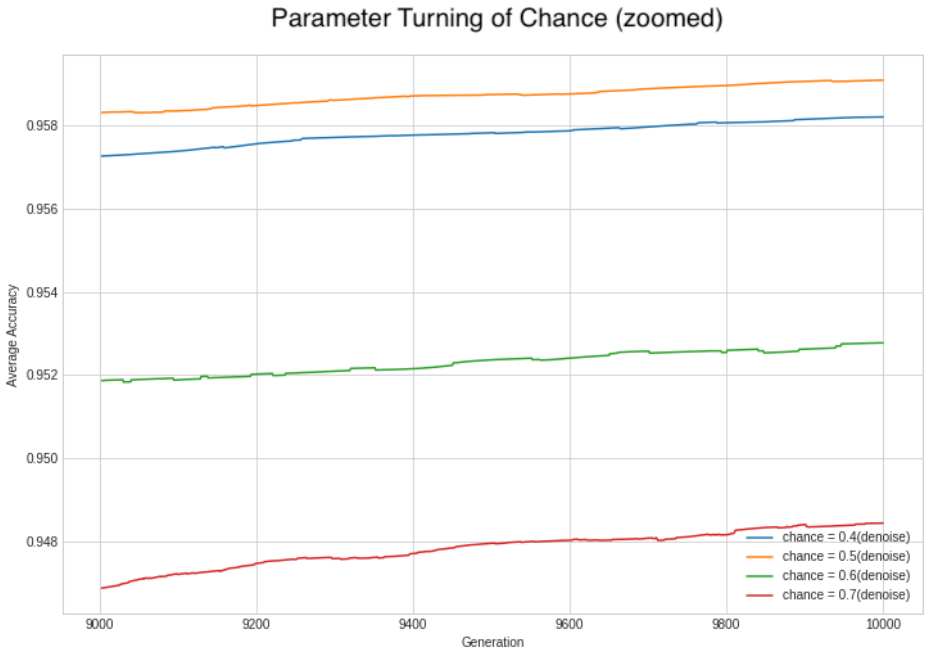


Figure 14

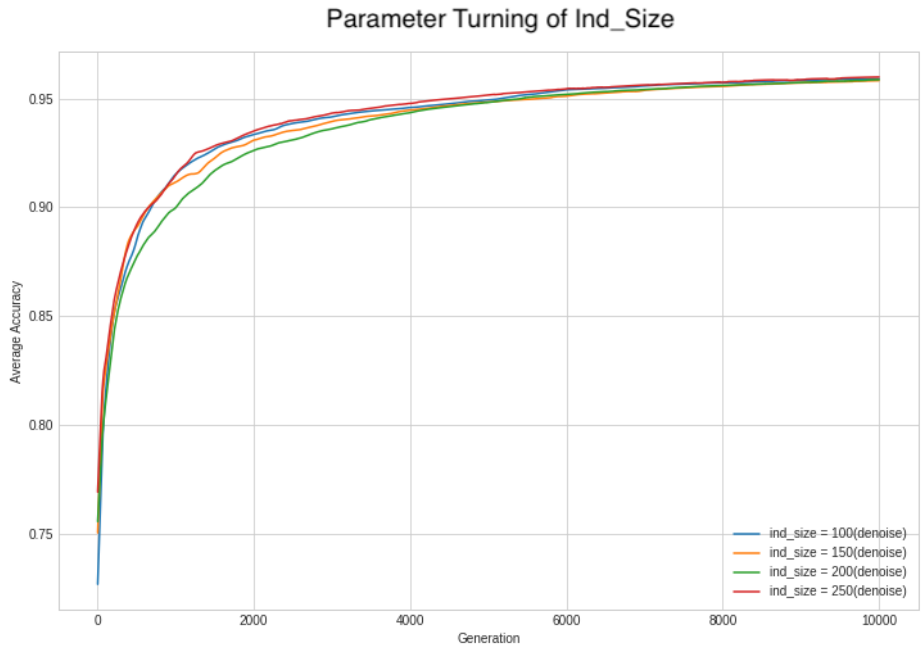

Parameter Turning of Chance
Average Accuracy
Generation
0.95
0.90
0.85
0.80
0.75
0
2000
4000
6000
8000
10000
chance = 0.4(denoise)
chance = 0.5(denoise)
chance = 0.6(denoise)
chance = 0.7(denoise)

(a)


Parameter Turning of Chance (zoomed)
Average Accuracy
Generation
0.958
0.956
0.954
0.952
0.950
0.948
9000
9200
9400
9600
9800
10000
chance = 0.4(denoise)
chance = 0.5(denoise)
chance = 0.6(denoise)
chance = 0.7(denoise)

(b)

Figure 15


Parameter Turning of Ind_Size
Average Accuracy
Generation
0.95
0.90
0.85
0.80
0.75
0
2000
4000
6000
8000
10000
ind_size = 100(denoise)
ind_size = 150(denoise)
ind_size = 200(denoise)
ind_size = 250(denoise)

(a)


Parameter Turing of Ind_Size (zoomed)
Average Accuracy
Generation
0.9600
0.9595
0.9590
0.9585
0.9580
0.9575
9000
9200
9400
9600
9800
10000
ind_size = 100(denoise)
ind_size = 150(denoise)
ind_size = 200(denoise)
ind_size = 250(denoise)

(b)

Figure 16

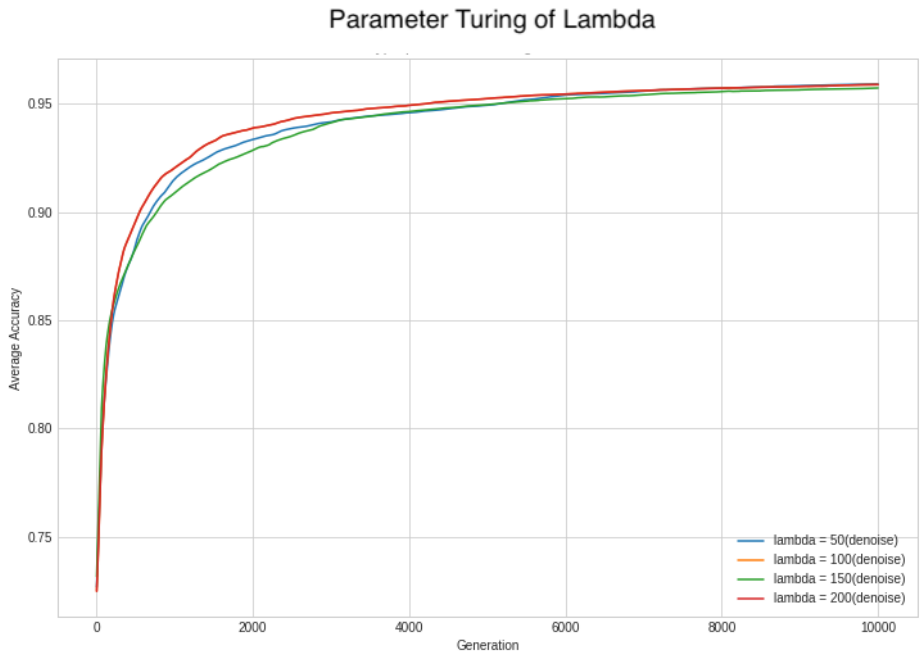
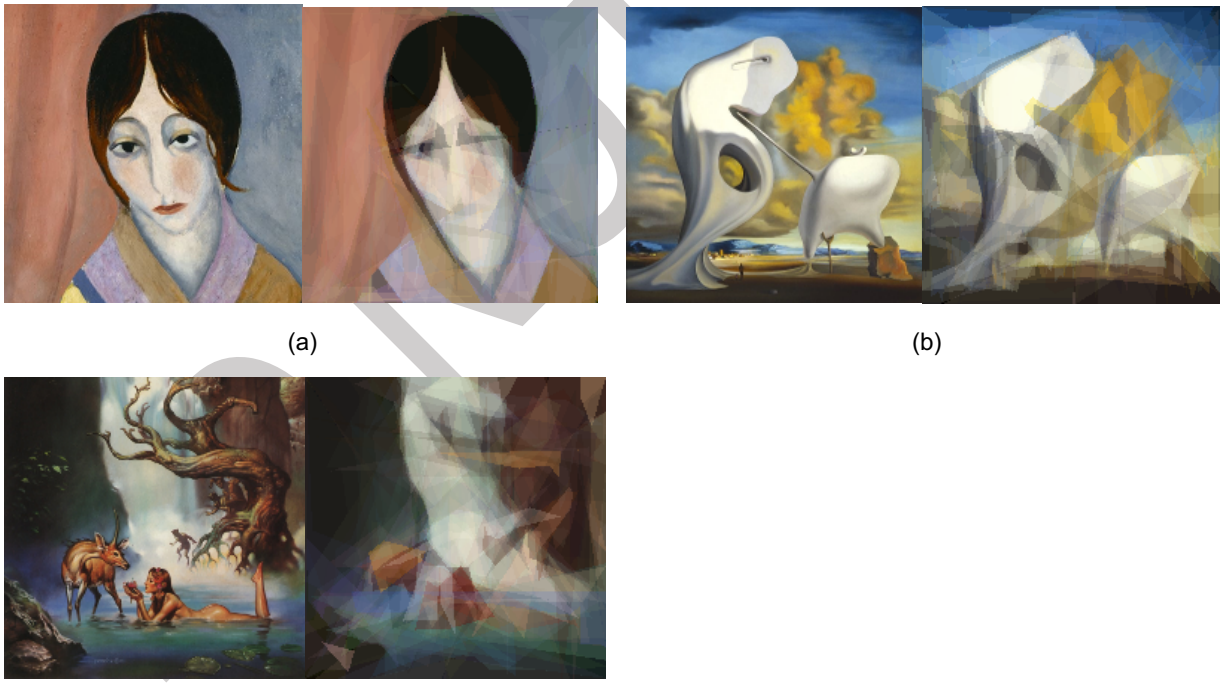


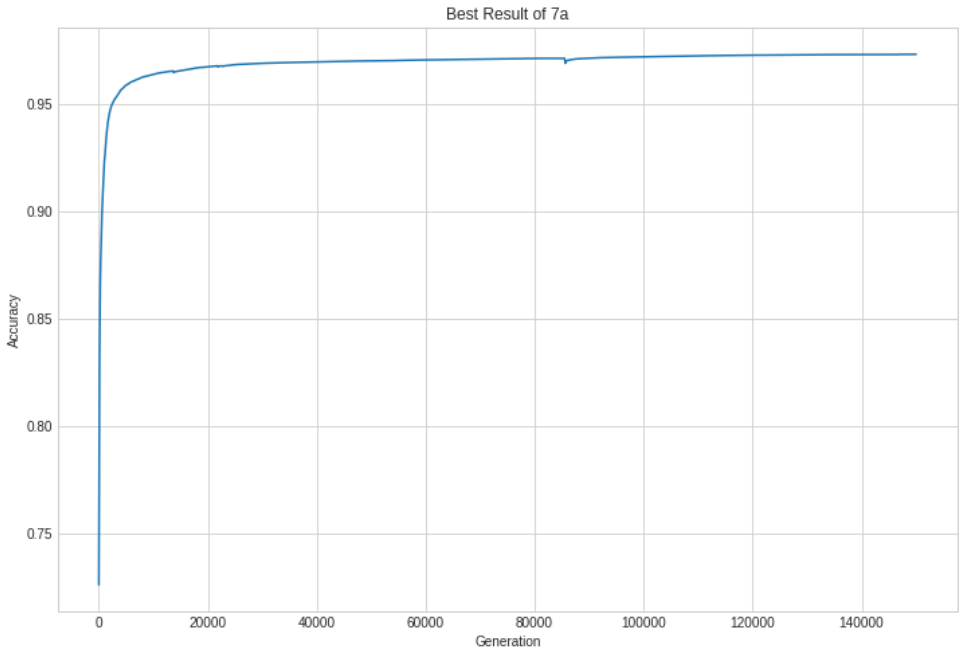
Figure 17



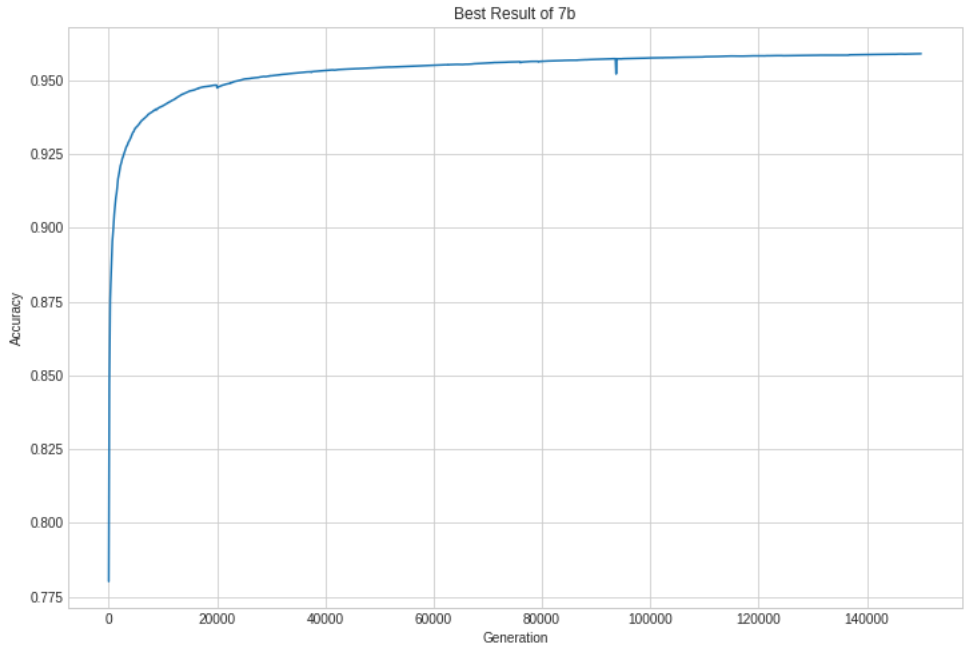
(a) (b)

(c)

Figure 18: The example of the best image in the finial population. (a): The easy level of the target image and the final image. (b): The median level of the target image and the final image. (c): The advance level of the target image and the final image.

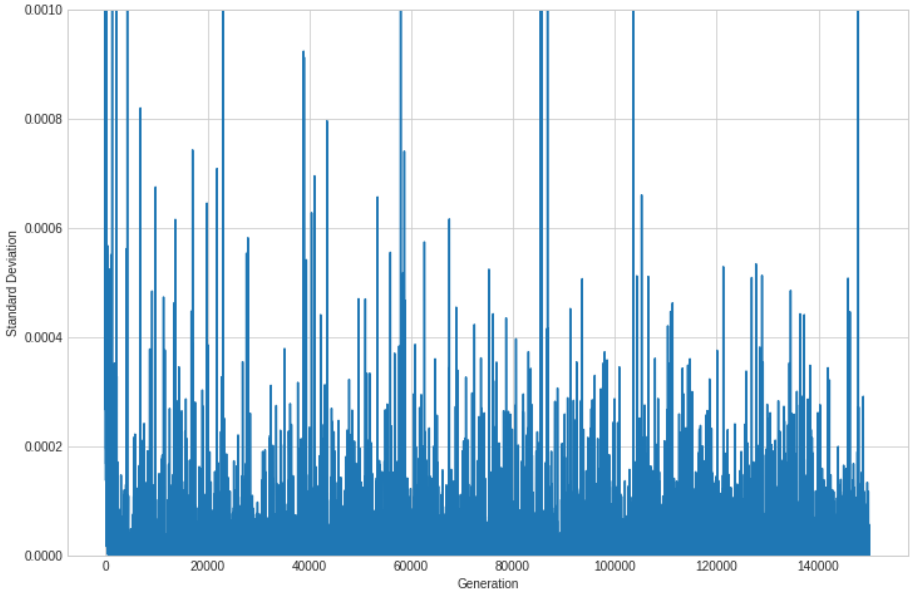


(a)

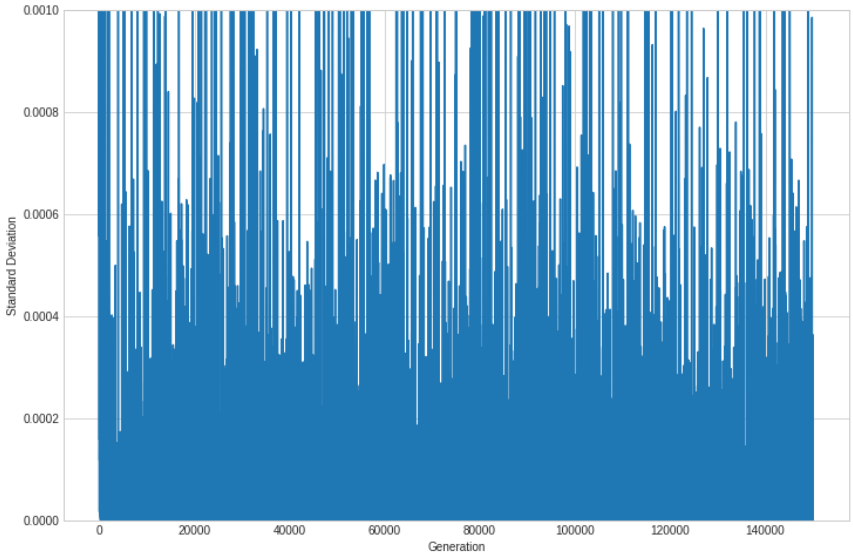


(b)

Figure 19: The graph (a) of the average fitness of each generation in the best result of the easiest target image. The graph (b) of the average fitness of each generation in the best result of the easiest target image. (The data is denoised.)

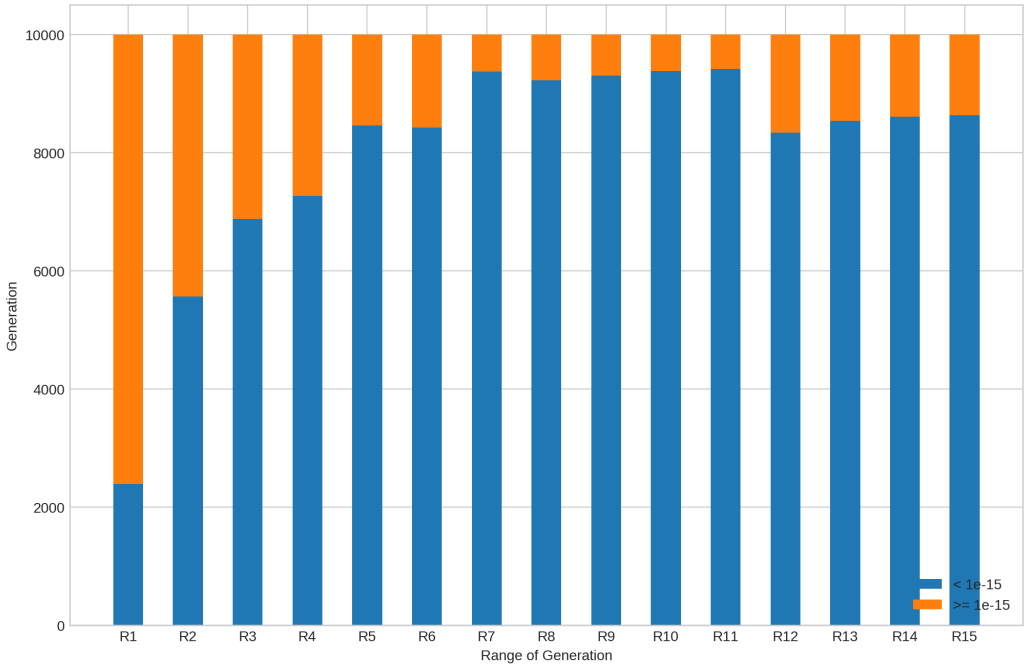


(a)

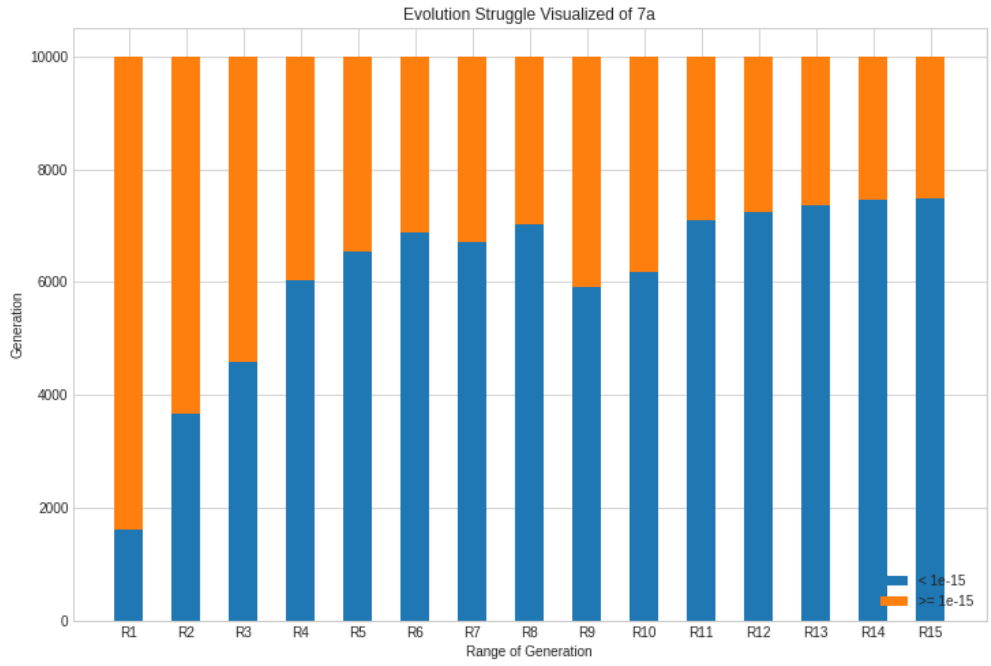


(b)

Figure 20: The overview of the standard division (a) of the best result of Figure 18(a). The overview of the standard division (b) of the best result of Figure 18(b). (a): The standard division of the most generations is around 0.002. (b): The deviation is not change much and average around 0.0006.



(a)



(b)

Figure 21: The evolution struggle of the easy level of image. The evolution struggle before using DSES (a). The evolution struggle before using DSES (b) The visualization of evolution struggle level. Each range is 10,000 generations and the orange bar present the generations NOT struggle and the others stands for struggle generations. The bar containing more orange means that it has higher possibility of evolution.

MulanFirst_0.9728505882352941

gen	nevals	min	max	avg	std
149992	45	0.9728505882352940	0.9728505882352940	0.9728505882352940	0.0
149993	42	0.9728505882352940	0.9728505882352940	0.9728505882352940	0.0
149994	47	0.9728505882352940	0.9728505882352940	0.9728505882352940	0.0
149995	45	0.9728505882352940	0.9728505882352940	0.9728505882352940	0.0
149996	47	0.9728505882352940	0.9728505882352940	0.9728505882352940	0.0
149997	50	0.9728503921568630	0.9728505882352940	0.9728505686274510	5.88235294141093E-08
149998	46	0.972843137254902	0.9728505882352940	0.97284984313725 5	2.23529411763623E-06
149999	48	0.9728467647058820	0.9728505882352940	0.9728502058823530	1.14705882352517E-06
150000	47	0.9728505882352940	0.9728505882352940	0.9728505882352940	0.0

Table 3: The table show the last 9 generation detail of the easiest target image.

7bFirst_0.9628423529411765

gen	nevals	min	max	avg	std
149992	45	0.9628411764705880	0.9628420588235290	0.9628419705882350	2.64705882346838E-07
149993	45	0.9628411764705880	0.9628420588235290	0.9628419705882350	2.64705882346838E-07
149994	47	0.9628420588235290	0.9628420588235290	0.9628420588235290	0.0
149995	44	0.9628420588235290	0.9628420588235290	0.9628420588235290	0.0
149996	46	0.9628420588235290	0.9628420588235290	0.9628420588235290	0.0
149997	45	0.9628399019607840	0.9628420588235290	0.9628418431372550	6.47058823521896E-07
149998	46	0.9628420588235290	0.9628423529411770	0.9628420882352940	8.82352941378173E-08
149999	42	0.9628420588235290	0.9628423529411770	0.9628421176470590	1.17647058850423E-07
150000	46	0.9628420588235290	0.9628423529411770	0.9628421470588230	1.34781638117747E-07

Table 4: The table show the last 9 generation detail of the median level target image.

7cFirst_0.9512427450980392

gen	nevals	min	max	avg	std
149993	40	0.951241568627451	0.951241568627451	0.951241568627451	0.0
149994	46	0.9511591176470590	0.951241568627451	0.9512333235294120	2.47352941176504E-05
149995	43	0.951241568627451	0.951241568627451	0.951241568627451	0.0
149996	49	0.9511914705882350	0.951241568627451	0.9512365588235290	1.50294117647221E-05
149997	46	0.9512210784313730	0.951241568627451	0.9512395196078430	6.14705882352462E-06
149998	44	0.951241568627451	0.9512427450980390	0.9512418039215690	4.70588235268465E-07
149999	46	0.951241568627451	0.9512427450980390	0.9512418039215690	4.70588235268465E-07
150000	45	0.951241568627451	0.9512427450980390	0.951242156862745	5.88235294085582E-07

Table 5: The table show the last 8 generation detail of the advance level target image.

Figure 22: The example of edge detection of major feature.

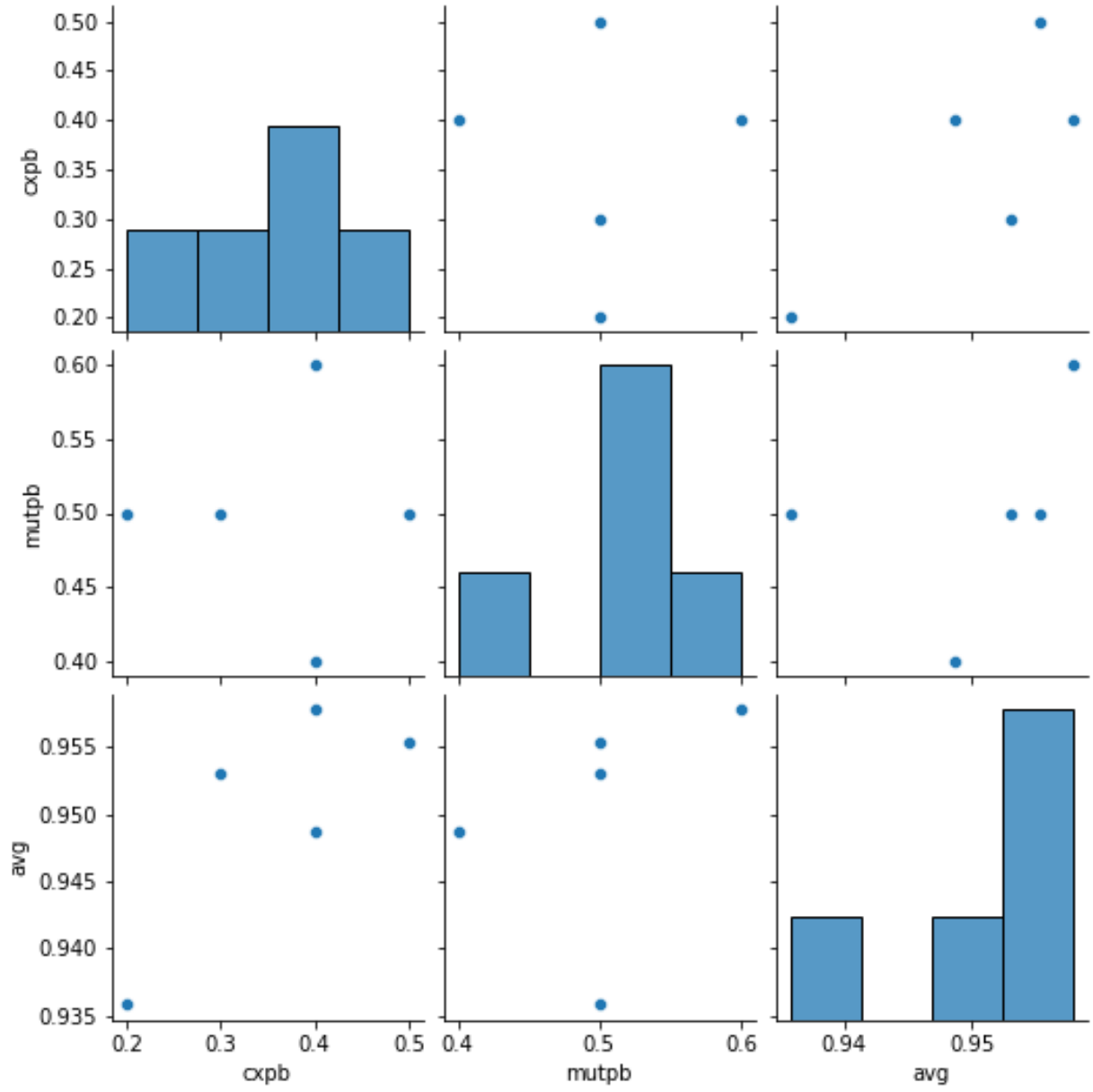


Figure 23: The example of the pair plot of crossover possibility parameter and mutation possibility parameter.