Cellular Automata Edge Detector Using Genetic Algorithmic approach

Ram Gopal Srikar, Katakam (Student ID:40106010)

Abstract— Edge Detection has been a critical task to be performed in many fields such as medical, autonomous vechiles, biological processing, However due to its limitation to run in parallel makes the implementation of most of the algorithm, slow in real time.In this Paper we propose a concept called Cellular automata which could help parallize the edge detection task of the image making it efficient in run to use multicore processor. Generally, Cellular automata involves building a global behivour based on local pattern hence, to find this pattern and form a generalized rule table we use Genetic Algorithms. In this paper, we will discuss details of cellular automata, genetic algorithms and how these algorithms could be configured to implement celluar automata for Edge Detection.

Index Terms— Cellular automata, Genetic algorithm (GA), Edge Detection

1 Introduction

THe ability to parallize edge detection has been a prominent task in today's work since, Many edge detectors algorithms operate serially making execution time slow. Due to its inate importance to extract valuable information from image, generally by detecting sharp changes in bright image helps to capture important events and changes in real world, this is due to the fact that mostly these discontinuities in image generally can provide information about depth, surface orientation, material properties and scene illumination.

Considering the amount of information could be extracted from the image, Edge detector algortihms are heavily used in medical field for finding edges in X-RAY's, MRI etc., Autonomous cars to understand the environment, biological field etc., Considerding its wide range of use, we try to parallize the edge detection task using The concept of Cellular Automata.

Cellular automata have been a fascinating model, due to its ability to generate complex structure based on simple local rules. This concept application can be found in nature, for example ants building its colonies although may seem complex, but at ant level they follow simple rule to generate it. Other example could be complex fractal structures which are primarily originated from a simple structural element and its simple rule of reputation in certain symmetry generates quite a complex structure.

Hence, it finds its application in many fields such as cryptography, Mathematical and theoretical biology etc., In this paper We use this technique to perform edge detection task in parallel using a simple rule table which is primarily based neigbouring pixels in a region.To generate this rule table, we implement a genetic algorithm and train it which result it in an generalized table which is bound to perform Edge detection task on any image.

For the purpose of the paper we use canny edge detection as a reference point for our algorithm to meet its requirement. The paper is structured as follows,section 2 Bref Description of the problem, Section 3 deals Details Of GA Design, Section 4 includes Results of GA, Section 5 application of generated table on other images,Section 6 includes Details of related Work and Section 7 is Conclustion.

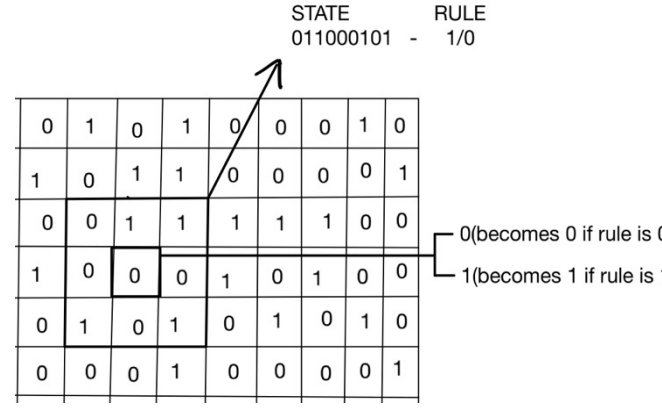

Fig.1 Cellular Automata rule table application using 3x3 window

2 Breif Description

For The purpose of Edge Detection, we find a rules table which change the state of the pixel based on its neighbouring pixels, we consider a 3x3 window for changing that state of each individual pixel either to 1 or 0.

As shown in Fig.1 when we run A 3x3 widow over the image, at any instanace the 3x3 window generates 9-bit binary image which is capable of producing 512 possibilites since 2^9=512 states, each state is assign a particular rule either 0 or 1. 0 means change the center pixel to 0 and 1 means change center pixel in the window to 1.

To genenerate such binary image, we choose a threshold value in gray image and change pixel intensities to either 1 or 0 based on intensities falling above or below threshold.

The Movement of the 3x3 window is wrap around,

where once you reach the right end of the image, the window wraps around and starts from left, similarly if it reaches bottom, it starts from top.

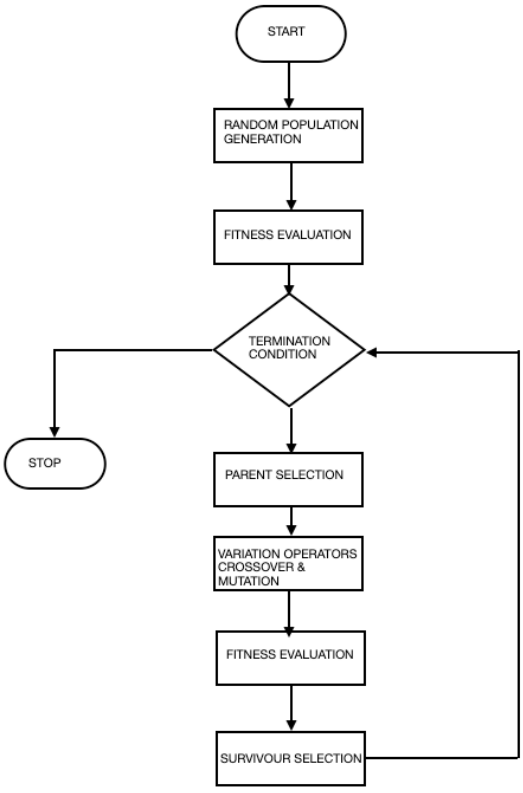

Fig.2 Flow chart of a generic genetic algorithm

3 GENETIC ALGORITHMS

Genetic Algorithm is a search heuristic algorithm inspired by Charles Darwin theory of Evolution. The algorithm behviour generally reflects the natural processes Evolution where the fitness individual survies. As shown in Fig.2 any GA to work in generally involves a population initialization stage where a random number of individual are initialiazed randomnly, these induviduals are tested by their fitness greater the fitness higher is the importance of the individual which makes it more likely for parent selection, Through variation operators such as cross-over and mutation off-spring is generated whose fitness are evaluated which is used for survival selection,based on the idea of survival of the fittest. The process is repeated until a satisfactory fitness of the induvidal is reached which is the termination condition, the design details of these stages are discussed in subsections below and the pseudo code of the GA is shown in Fig.3.

3.1 Representation

Each individual in the population can be expressed wither in genotype space or phenotype space which is their representation,generally these phenotype and genotype space representation can be different i.e for example a phenotype space could be Integer where as a genotype space could be a binary representation of the integer phenotype space since in certain situation exploring genotype space could be more diverse and find ideal solution. Generally based on the problem we have different representation such as binary, Integer, floating point, Permuation, Tree etc., For the problem statement at hand, we have only two rules either 0 or 1 which can be represented using integer or binary, Since in future implementation more rules can be added we choose Integer representation which is sufficient for problem to be solved.

```
BEGIN
        INITIALISE population with random candidates( random rule table
generation)
        GENERATE final image using rule table initialised above.
        EVALUATE each candidate (FITNESS EVALUATION)
        REPEAT UNTIL( TERMINATION CONDITION is satisfied) DO
                1.SELECT Parents;
                2.RECOMBINE pairs of parents;
                3.MUTATE the resulting Offspring;
                4.GENERATE final image using rule table of the Offspring.
                4.EVALUATE new candidates;(FITNESS EVALUATION)
                5.SELECT individuals for next generation;
        OD
END
```

Fig.3 Pseudo-code for GA(Genetic Algorithm)

3.2 Population and Population Size

Population size determines maximum diversity that could be contained in the population, hence greater the population better would be the results, But at the cost of high computation time, hencewe need to find a maximum population size, beyond that size there is minimal or no growth in fitness values. In our implementation 50 population size would be enough to approximate the start image to goal image.

```
BEGIN
        GOAL IMAGE (image obtained using canny edge detection algorithm)
        FINAL IMAGE(image obtained by applying rule table)
        INTIALISE COUNT to zero
        REPEAT UNTIL( ALL PIXELS IN IMAGE ARE COVERED) DO
                1.COMPARE pixel location of GOAL IMAGE and FINAL IMAGE
                2.IF EQUAL, CONTINUE(skip loop)
                3.IF NOT EQUAL, INCREMENT COUNT by one
        OD
END
```

Fig.4 Pseudo-code for Fitness Evaluation function

3.3 Fitness Evaluation

For performing fitness evaluation between final image (image computed after passing 3x3 window and applying corresponding rules table) and goal image is by using hamming distance. Hamming distance is measured by number of unequal pixel value i.e if the pixel in goal image and final image are dissimilar, then fitness is increased by 1. Fig.4 shows pseudo-code of the fitness function.

3.4 Parent Selection

For selecting parents in the population pool, we use fitness as the basis for selection. Since if in the population any one individual has high fitness it can easily dominate other individual resulting in loss of diversity since, the same individual is choosed many times leading to loss of

diversity. To avoid this problem, we can perform tourna Ment-based selection, linear ranking and exponential

```
#TOURNAMENT-BASED PARENT SELECTION

BEGIN
        POPULATION (pool of individuals from which parents should be
selected)
        POPULATION SIZE(size of the population considered)
        GROUP SIZE( size of sub group that needs to be considered)
        GENERATE a two random sub group of size group size from the
population.
        FIND maximum FITNESS in each sub group.
        PARENTS selected are fittest individuals in two subgroups.
END
```

Fig.5 Pseudo-code for Tournament based Section

```
#EXPONENTIAL RANKING

BEGIN
        POPULATION (pool of individuals from which parents should be
selected)
        POPULATION SIZE(size of the population considered)
        ASSIGN ranks to population based on fitness values.
        FIND new probabilities using the Exponential ranking equation.
        GENERATE two random values.
        FIND where these to values lie in cumulative probability distribution.
        RETURN those corresponding individual which are parents.
END
```

Fig.6 Pseudo-code for Exponential ranking

ranking. We explore different methods and its corresponding performance to finalize which Method performs the best in the given problem constraint. As shown in Fig.[?] we check the performace of tournament-based method, exponential ranking since these approaches outperform linear ranking and random parent selection, we avoid its analysis for this problem. Fig.5 and Fig.6 shows pseudo code for tournament-based selection and exponential ranking.

Probabilities for exponential ranking is calculated using the mathematical equation given in equation(1).

$$P_{exp-rank}(i) = \frac{1-e^{-i}}{c} \qquad (1)$$

```
#UNIFORM CROSSOVER

BEGIN

        PARENTS selected for Off-spring Generation.
        DEFINE  Crossover Rate.
        REPEAT UNTIL( All the Elements in the individual parents are covered) DO
                1.INITIALISE a random value between (0,1)
                2. IF Value initialised is less than crossover rate
                        1.SWAP the rules of the corresponding states
                3.ELSE Continue
        OD

END
```

Fig.7 Pseudo-code for Uniform Crossover

3.5 Surviour Selection

For the purpose of selecting candidates for the next generation, we test the performance of the algorithm between mu+lambda) selection stratergy which involved selecting the top candidates of both parents and offsprings. Since (mu+lambda) is not good at preserving diversity in the algorithm we test the performance of the algorithm while implementing crowding and compare its performance as

```
#UNIFORM MUTATE

BEGIN

        OUTPUT offering from CrossOver.
        DEFINE  Mutation rate.
        REPEAT UNTIL( All the Elements in the individual parents are covered) DO
                1.INITIALISE a random value between (0,1)
                2. IF Value initialised is less than Mutation Rate
                        1.FLIP the Rule bit to 0 if 1 and vice-versa
                3.ELSE Continue
        OD

END
```

Fig.8 Pseudo-code for Uniform Mutate

3.5 Crossover and Mutation

During the mating stage to produce offprings, the crossover and mutation are generally explorative and exploitive methods to move in the solution search space. Where Mutation is used to pass parents characteristics to children were as mutation is used to add values which doesn't exist in either parents. Fig.7

For this problem statement we test Uniform crossover with crossover rate at 0.4 i.e means 40 percent of information is exchanged between parents to form corresponding offsprings, one point crossover and n-point crossover. For mutation we use Uniform mutation with mutation rate 0.1. Fig.8 shows the pseudo code of uniform mutation.

3.6 Diversity measurement

In order to preserve the diversity we need to measure the diversity. Based on Rao (1982), champely and chassel introduced a function using Eucledian distance between spieces defined as

$$\Gamma = \frac{1}{2}\sum_{i=1}^{ns}\sum_{j=1}^{ns} P_i P_j [d(S_i, S_j)]^2 \qquad (2)$$

3.6 Algortihm selection and parameter testing

During the parent selection procedure, we are testing the algorithm on two different parent selection schemes which is tournament-based and expoenential based as shown in Table 1 and check the performance of the algorithm with different parameter setting and find better selection procedure and its corresponding parameter settings.

TABLE 1
Two different parent selection techniques comparision

	ALG-1	ALG-2
Representation	Bit-String	Bit-String
Parent Selection	Tournament-Based	Exponential Ranking
Survivor Selection	(mu+lambda)	(mu+lambda)
CrossOver	Uniform	Uniform
Mutation	Uniform	Uniform

With these two version of ALG-1, ALG-2 we test the performance under different parameter constraints such as different population size,crossover rate and mutation step size and find better combination of of these parameter for which the algorithm performs the best. We judge the performance of the algorithm using max fitness value and avg fitness value, since we are solving a minimization problem we will be concentrating on lower values of maximum fitness and average values. Table 2 demonsatres the performance of ALG-1 under different parameter settings.

TABLE 2
ALG-1 parameter tuning.

GENERATION	10	30	50	50
Population Size	10	30	50	50
Tournament Size	2	6	20	20
Mutation Size	0.1	0.1	0.1	0.2
CrossOver	0.4	0.4	0.4	0.5
Avg fitness	2876.4	2320.73333333333	2015.46	1868.44
Best fitness	2411	1815	1422	1585

TABLE 3
ALG-2 paramter tunning

Generations	10	30	50	50
Population Size	10	30	50	50
Mutation Size	0.1	0.1	0.1	0.2
CrossOver	0.4	0.4	0.4	0.5
Avg fitness	3002.2	2649.33333333333	2301.5	2185.34
Best fitness	2679	1824	1628	1848

From the table and comparision we could conclude that the performance of the algorithm with tournament-based approach with population size 50 and mutation rate 0.1 has the least fitness value among all different instances of the implementation. Hence, we further proceed with this combination to evaluate its consistency using mean and standard deviation of the algorithm. In Section 4 we discuss the behviour of the selected algorithm in more details using results generated from running the algorithm twice.

4. Results

After Runing the algorithm for two times with 50 generation and a population size of 50, the performance of the algorithm can be seen from the Fig.9,Fig.10,Fig.11,Fig.14.As can be observed from the graph in fig.9, the maximum fitness reduction can be achieved within 3 generations.Standard deviation of maximum fitness gives detail fo how the maximum fitness is varied over different generation, greater is the standard deviation at a particular generation greater is the uncertainity to reach to the solution at that particular generation. Hence from the graph in fig.10, it can be concluded that there is a sharp decline in standard deviation within first 3 generations, which makes it more likely to run the algorithm for less than 10 generation, since reaching the solution is highly likely.

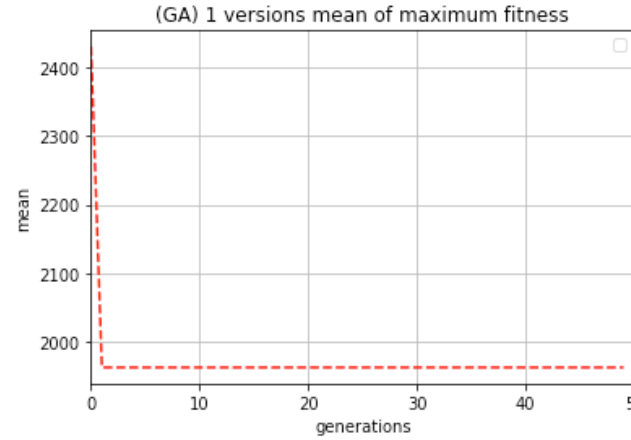

Fig.9 mean of best fitness after running twice.

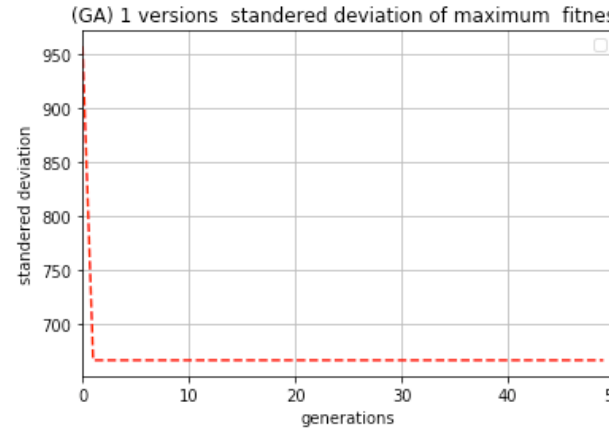

Fig.10 standard deviation of best fitness after running twice.

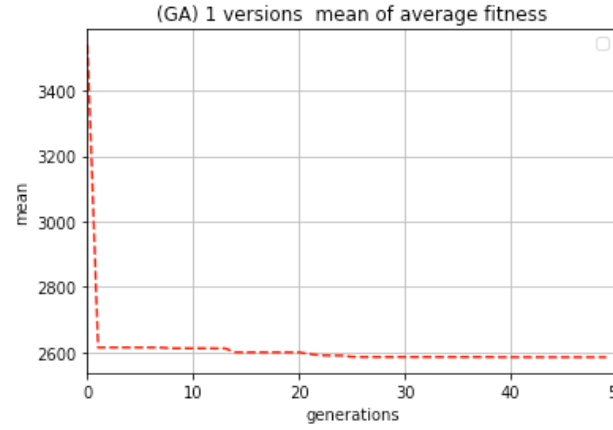

Fig.11 mean of average fitness after running twice.

To understand the diversity present in the population it is required to calculate average fitness across the population. From fig.11 it can be shown that mean of average fitness is falling sharply, this could indicate in reduced diversity in the whole population. Similarly from fig.14 standard deviation of average fitness across the population gives certainity in the prediction.

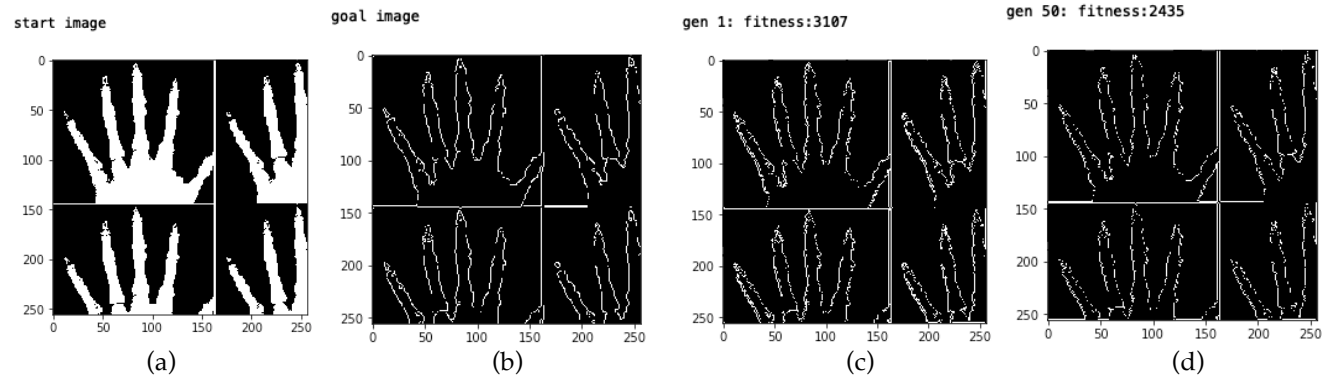

Fig.12 Result of training the GA for 50 generations

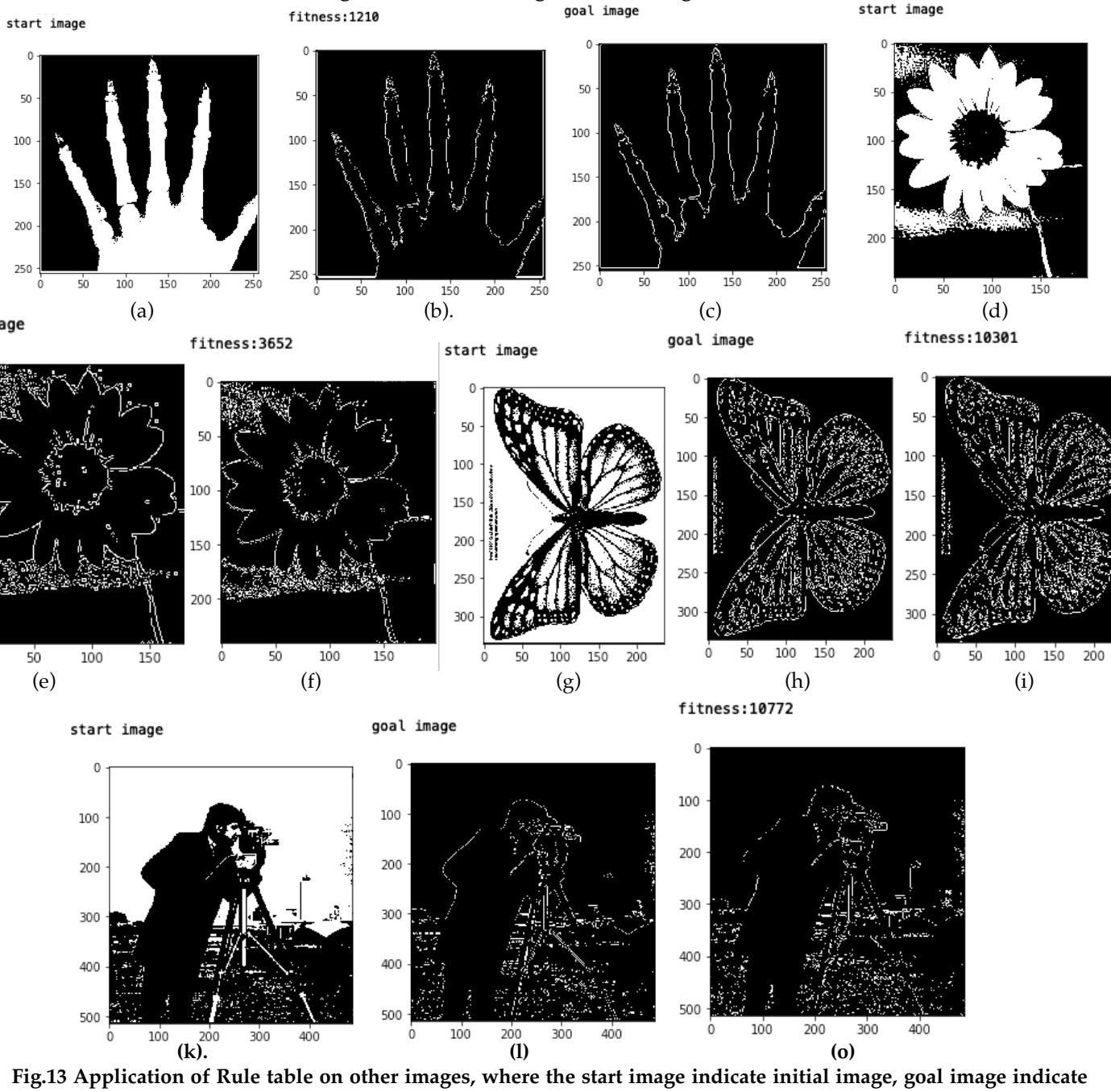

Fig.13 Application of Rule table on other images, where the start image indicate initial image, goal image indicate result after running canny and fitness indicate final image after application of rule table and the numberical value beside fitness indicate hamming distance between final and goal image.

lets check the input and output to the algorithm and how well is it performing th e task of edge detection, below are few images showcasing the performance of the algorithm.

Fig.12(a) gives the details about input data to GA, where the Fig.12(b) is the output of canny edge-detection algorithm. Fig.12(c) and Fig.12(d) shows learning of the from generation 1 and generation 50.

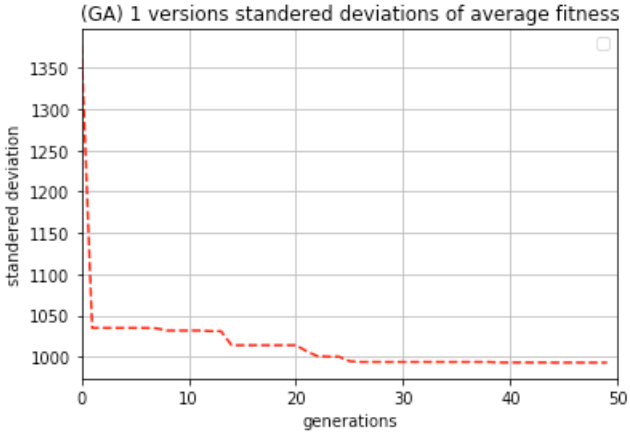

Fig.14 mean of average fitness after running twice.

6 Application of Built Rule Table

Once the GA produces rule table for a run of 50 generation with a size of 50 population, the table is applied to other images to check its performance in edge detection before and after the applying the rule table which has been generated from the GA. As can be notted from the Fig.13 95% of image reconstruction for edge detection is performed, however with more population and generation more accurate image reconstruction could be formed but the execution time of the algorithm grows exponential with increase in population size and generation. For a minmal resources it could be observed maximal reproduction of edges of the image could be observed, hence any algorithm which is not parallelizable could be done by using this approach, since rule table running on 3x3 image patch can be able to run simultaneously because the patches are not inter-dependent.

7 Further Development

From the performance figues of tournmant-based selection it could be noticed at after first 10 generations there is no learning curve, but since for the constraint of 50 generation and 50 population the performance of tournmanent-based selection outscales expoenential ranking. If we further increase the population size and generation, we wouldn't find any increments in the performance of the algorithm, hence this is not the ideal solution for long runs.

To further establish that exponential ranking is superior over tournmant-based over long runs, we check the performance of the exponential-ranking as seen in Fig.15, Fig.16, Fig.17and Fig.18. As can be observed from the graph it can be noticed that the algorithm is actively learning, with noticeable changes till 50th generation which makes it better choice for choosing generation and population more than 50.

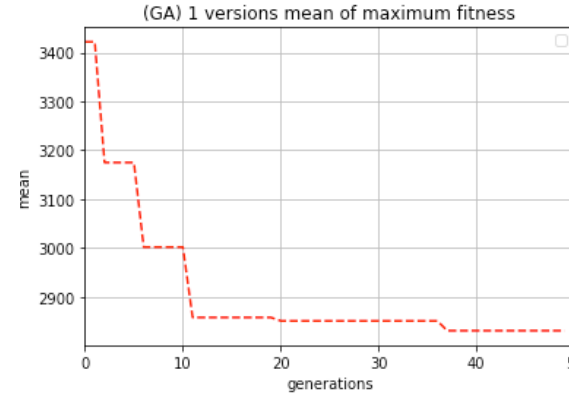

Fig.15 mean of maximum fitness after running twice.

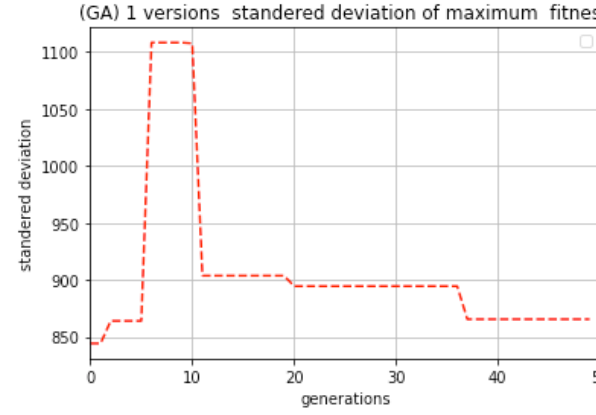

Fig.16 standard deviation of maximum fitness after running twice.

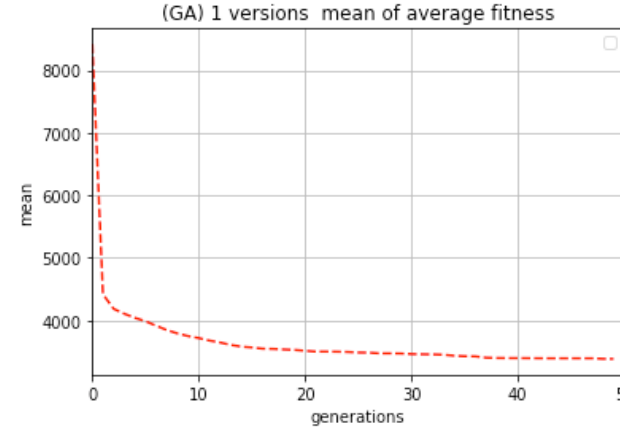

Fig.17 mean of average fitness after running twice.

7 Related Work

Among all the Edge detection task, we are concentrating specifically on task which enable the edge detection task to be parallelized. One of the most similar paper was Deepak Ranjan Nayak, et.al-"Cellular Automata Based Optimal Edge Detection Technique Using Twenty-Five

neighbourhood model"[1], However in this approach TFNCA model where are the rules are computed dynamically based on pixel neighborhood which is a 5x5 window in this case. Due to its Dynamic computation, the execution speed of the algorithm is considerably slow compared to GA appraoch.

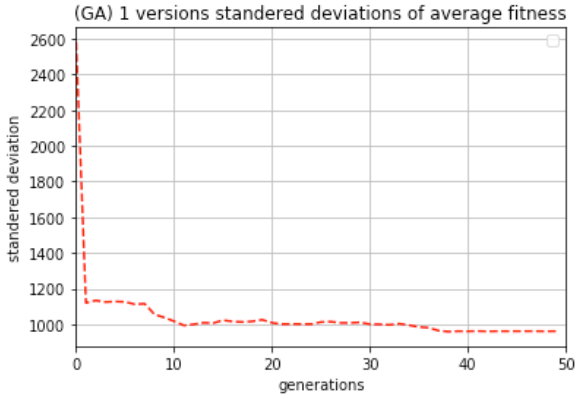

Fig.18 standard deviation of average fitness after running twice.

7 Conclusion

In this paper, we have discussed about Cellular automata Edge detector Using GA approach, we have discussed possible two best implementation for the problem statement and tried to analyse the algorithm behavior under different parameter constraints. From these experiments we have choosen a better GA implementation and its corresponding parameter values and have generated a rule table which is tested on few images and compared its performance with edge detector.

Hence, with this implementation we could parallelize the edge detector problem to run faster and more efficiently.

References

[1] Nayak, Deepak Ranjan, Sumit Kumar Sahu, and Jahangir Mohammed. "A cellular automata based optimal edge detection technique using twenty-five neighborhood model." *arXiv preprint arXiv:1402.1348* (2014).

RULE TABLE FOR EDGE DETECTION THAT IS GENERATED FROM RUNNING THE GA ALGORITHM.

Where "000000000" corresponds to state and 0 or 1 corresponds to Rule.

[{"000000000": 0}, {"000000001": 0}, {"000000010": 0}, {"000000011": 0},
{"000000100": 0}, {"000000101": 0}, {"000000110": 0}, {"000000111": 1},
{"000001000": 0}, {"000001001": 0}, {"000001010": 1}, {"000001011": 1},
{"000001100": 0}, {"000001101": 1}, {"000001110": 1}, {"000001111": 1},
{"000010000": 0}, {"000010001": 0}, {"000010010": 1}, {"000010011": 0},
{"000010100": 1}, {"000010101": 1}, {"000010110": 0}, {"000010111": 0},
{"000011000": 0}, {"000011001": 0}, {"000011010": 0}, {"000011011": 1},
{"000011100": 0}, {"000011101": 1}, {"000011110": 1}, {"000011111": 1},
{"000100000": 1}, {"000100001": 0}, {"000100010": 0}, {"000100011": 0},
{"000100100": 0}, {"000100101": 0}, {"000100110": 0}, {"000100111": 0},
{"000101000": 0}, {"000101001": 0}, {"000101010": 0}, {"000101011": 0},
{"000101100": 1}, {"000101101": 1}, {"000101110": 1}, {"000101111": 0},
{"000110000": 0}, {"000110001": 1}, {"000110010": 0}, {"000110011": 0},
{"000110100": 0}, {"000110101": 0}, {"000110110": 1}, {"000110111": 0},
{"000111000": 0}, {"000111001": 0}, {"000111010": 1}, {"000111011": 0},
{"000111100": 0}, {"000111101": 1}, {"000111110": 1}, {"000111111": 1},
{"001000000": 0}, {"001000001": 1}, {"001000010": 0}, {"001000011": 0},
{"001000100": 1}, {"001000101": 0}, {"001000110": 0}, {"001000111": 1},
{"001001000": 0}, {"001001001": 1}, {"001001010": 1}, {"001001011": 0},
{"001001100": 0}, {"001001101": 1}, {"001001110": 1}, {"001001111": 1},
{"001010000": 0}, {"001010001": 0}, {"001010010": 1}, {"001010011": 0},
{"001010100": 0}, {"001010101": 0}, {"001010110": 1}, {"001010111": 1},
{"001011000": 1}, {"001011001": 0}, {"001011010": 1}, {"001011011": 1},
{"001011100": 1}, {"001011101": 0}, {"001011110": 0}, {"001011111": 0},
{"001100000": 0}, {"001100001": 0}, {"001100010": 1}, {"001100011": 1},
{"001100100": 0}, {"001100101": 0}, {"001100110": 1}, {"001100111": 0},
{"001101000": 0}, {"001101001": 1}, {"001101010": 0}, {"001101011": 1},
{"001101100": 0}, {"001101101": 1}, {"001101110": 1}, {"001101111": 0},
{"001110000": 0}, {"001110001": 1}, {"001110010": 1}, {"001110011": 0},
{"001110100": 1}, {"001110101": 0}, {"001110110": 1}, {"001110111": 0},
{"001111000": 0}, {"001111001": 0}, {"001111010": 0}, {"001111011": 0},
{"001111100": 1}, {"001111101": 0}, {"001111110": 0}, {"001111111": 0},
{"010000000": 1}, {"010000001": 1}, {"010000010": 0}, {"010000011": 0},
{"010000100": 0}, {"010000101": 0}, {"010000110": 1}, {"010000111": 0},
{"010001000": 0}, {"010001001": 1}, {"010001010": 0}, {"010001011": 1},
{"010001100": 0}, {"010001101": 0}, {"010001110": 0}, {"010001111": 0},
{"010010000": 1}, {"010010001": 0}, {"010010010": 0}, {"010010011": 0},
{"010010100": 1}, {"010010101": 0}, {"010010110": 1}, {"010010111": 0},
{"010011000": 0}, {"010011001": 1}, {"010011010": 0}, {"010011011": 0},
{"010011100": 0}, {"010011101": 1}, {"010011110": 1}, {"010011111": 0},
{"010100000": 1}, {"010100001": 0}, {"010100010": 1}, {"010100011": 0},
{"010100100": 0}, {"010100101": 0}, {"010100110": 1}, {"010100111": 0},
{"010101000": 0}, {"010101001": 0}, {"010101010": 1}, {"010101011": 0},
{"010101100": 0}, {"010101101": 0}, {"010101110": 1}, {"010101111": 1},
{"010110000": 0}, {"010110001": 1}, {"010110010": 0}, {"010110011": 0},
{"010110100": 1}, {"010110101": 0}, {"010110110": 1}, {"010110111": 1},
{"010111000": 0}, {"010111001": 1}, {"010111010": 0}, {"010111011": 0},
{"010111100": 1}, {"010111101": 0}, {"010111110": 0}, {"010111111": 1},
{"011000000": 1}, {"011000001": 0}, {"011000010": 0}, {"011000011": 0},
{"011000100": 0}, {"011000101": 0}, {"011000110": 0}, {"011000111": 0},
{"011001000": 0}, {"011001001": 1}, {"011001010": 1}, {"011001011": 0},
{"011001100": 1}, {"011001101": 0}, {"011001110": 1}, {"011001111": 0},
{"011010000": 1}, {"011010001": 1}, {"011010010": 1}, {"011010011": 1},
{"011010100": 0}, {"011010101": 1}, {"011010110": 0}, {"011010111": 0},
{"011011000": 0}, {"011011001": 1}, {"011011010": 1}, {"011011011": 0},
{"011011100": 0}, {"011011101": 0}, {"011011110": 1}, {"011011111": 0},
{"011100000": 0}, {"011100001": 1}, {"011100010": 0}, {"011100011": 1},
{"011100100": 0}, {"011100101": 0}, {"011100110": 0}, {"011100111": 0},
{"011101000": 0}, {"011101001": 0}, {"011101010": 1}, {"011101011": 0},
{"011101100": 1}, {"011101101": 1}, {"011101110": 0}, {"011101111": 1},
{"011110000": 1}, {"011110001": 1}, {"011110010": 0}, {"011110011": 1},
{"011110100": 0}, {"011110101": 1}, {"011110110": 0}, {"011110111": 0},
{"011111000": 1}, {"011111001": 1}, {"011111010": 0}, {"011111011": 0},
{"011111100": 0}, {"011111101": 1}, {"011111110": 1}, {"011111111": 0},
{"100000000": 1}, {"100000001": 0}, {"100000010": 0}, {"100000011": 0},
{"100000100": 0}, {"100000101": 0}, {"100000110": 0}, {"100000111": 0},
{"100001000": 1}, {"100001001": 1}, {"100001010": 1}, {"100001011": 1},
{"100001100": 0}, {"100001101": 1}, {"100001110": 0}, {"100001111": 1},
{"100010000": 1}, {"100010001": 0}, {"100010010": 1}, {"100010011": 0},
{"100010100": 1}, {"100010101": 0}, {"100010110": 0}, {"100010111": 0},
{"100011000": 1}, {"100011001": 0}, {"100011010": 0}, {"100011011": 0},
{"100011100": 1}, {"100011101": 0}, {"100011110": 0}, {"100011111": 0},
{"100100000": 1}, {"100100001": 0}, {"100100010": 1}, {"100100011": 0},
{"100100100": 0}, {"100100101": 1}, {"100100110": 0}, {"100100111": 1},
{"100101000": 0}, {"100101001": 0}, {"100101010": 0}, {"100101011": 0},
{"100101100": 0}, {"100101101": 0}, {"100101110": 0}, {"100101111": 0},
{"100110000": 0}, {"100110001": 1}, {"100110010": 0}, {"100110011": 1},
{"100110100": 1}, {"100110101": 0}, {"100110110": 0}, {"100110111": 1},
{"100111000": 0}, {"100111001": 1}, {"100111010": 1}, {"100111011": 0},
{"100111100": 1}, {"100111101": 0}, {"100111110": 1}, {"100111111": 0},
{"101000000": 1}, {"101000001": 0}, {"101000010": 0}, {"101000011": 1},
{"101000100": 1}, {"101000101": 0}, {"101000110": 0}, {"101000111": 0},
{"101001000": 0}, {"101001001": 0}, {"101001010": 0}, {"101001011": 0},
{"101001100": 1}, {"101001101": 0}, {"101001110": 1}, {"101001111": 0},
{"101010000": 0}, {"101010001": 0}, {"101010010": 0}, {"101010011": 0},
{"101010100": 0}, {"101010101": 0}, {"101010110": 1}, {"101010111": 1},
{"101011000": 0}, {"101011001": 1}, {"101011010": 0}, {"101011011": 1},
{"101011100": 0}, {"101011101": 0}, {"101011110": 1}, {"101011111": 1},
{"101100000": 0}, {"101100001": 1}, {"101100010": 1}, {"101100011": 1},
{"101100100": 0}, {"101100101": 1}, {"101100110": 0}, {"101100111": 0},
{"101101000": 1}, {"101101001": 1}, {"101101010": 0}, {"101101011": 0},
{"101101100": 0}, {"101101101": 1}, {"101101110": 0}, {"101101111": 1},
{"101110000": 1}, {"101110001": 1}, {"101110010": 0}, {"101110011": 1},

{"101110100": 0}, {"101110101": 0}, {"101110110": 1}, {"101110111": 0},
{"101111000": 0}, {"101111001": 1}, {"101111010": 1}, {"101111011": 0},
{"101111100": 1}, {"101111101": 1}, {"101111110": 0}, {"101111111": 1},
{"110000000": 0}, {"110000001": 0}, {"110000010": 1}, {"110000011": 0},
{"110000100": 1}, {"110000101": 1}, {"110000110": 0}, {"110000111": 0},
{"110001000": 0}, {"110001001": 0}, {"110001010": 0}, {"110001011": 1},
{"110001100": 0}, {"110001101": 1}, {"110001110": 0}, {"110001111": 1},
{"110010000": 1}, {"110010001": 1}, {"110010010": 0}, {"110010011": 0},
{"110010100": 1}, {"110010101": 1}, {"110010110": 1}, {"110010111": 0},
{"110011000": 0}, {"110011001": 1}, {"110011010": 0}, {"110011011": 0},
{"110011100": 0}, {"110011101": 0}, {"110011110": 1}, {"110011111": 1},
{"110100000": 0}, {"110100001": 0}, {"110100010": 0}, {"110100011": 0},
{"110100100": 0}, {"110100101": 0}, {"110100110": 0}, {"110100111": 0},
{"110101000": 0}, {"110101001": 1}, {"110101010": 0}, {"110101011": 0},
{"110101100": 0}, {"110101101": 0}, {"110101110": 1}, {"110101111": 1},
{"110110000": 1}, {"110110001": 1}, {"110110010": 0}, {"110110011": 1},
{"110110100": 1}, {"110110101": 0}, {"110110110": 1}, {"110110111": 0},
{"110111000": 0}, {"110111001": 0}, {"110111010": 0}, {"110111011": 1},
{"110111100": 0}, {"110111101": 1}, {"110111110": 0}, {"110111111": 0},
{"111000000": 0}, {"111000001": 1}, {"111000010": 1}, {"111000011": 1},
{"111000100": 0}, {"111000101": 1}, {"111000110": 1}, {"111000111": 0},
{"111001000": 1}, {"111001001": 1}, {"111001010": 0}, {"111001011": 0},
{"111001100": 0}, {"111001101": 0}, {"111001110": 0}, {"111001111": 0},
{"111010000": 0}, {"111010001": 0}, {"111010010": 1}, {"111010011": 0},
{"111010100": 1}, {"111010101": 0}, {"111010110": 1}, {"111010111": 1},
{"111011000": 0}, {"111011001": 0}, {"111011010": 0}, {"111011011": 0},
{"111011100": 0}, {"111011101": 0}, {"111011110": 1}, {"111011111": 0},
{"111100000": 0}, {"111100001": 0}, {"111100010": 0}, {"111100011": 0},
{"111100100": 0}, {"111100101": 0}, {"111100110": 0}, {"111100111": 0},
{"111101000": 1}, {"111101001": 0}, {"111101010": 1}, {"111101011": 0},
{"111101100": 0}, {"111101101": 1}, {"111101110": 1}, {"111101111": 0},
{"111110000": 0}, {"111110001": 1}, {"111110010": 0}, {"111110011": 0},
{"111110100": 0}, {"111110101": 0}, {"111110110": 0}, {"111110111": 0},
{"111111000": 1}, {"111111001": 0}, {"111111010": 1}, {"111111011": 1},
{"111111100": 1}, {"111111101": 0}, {"111111110": 0}, {"111111111": 0}]