Genetic algorithm based image approximation with geometrical shapes

Gergely Vakulya
Alba Regia Technical Faculty
Óbuda University
Székesfehérvár, Hungary
vakulya.gergely@amk.uni-obuda.hu

***Abstract*—In this paper the design parameters of a genetic algorithm (GA) based image approximation / compression algorithm are analyzed. The efficiency using different approximation shapes and with different GA parameters (population size, mutation rate) are evaluated using test runs. Recommended parameter sets for different scenarios are provided.**

***Keywords*—genetic algorithm, image approximation**

I. Introduction

The genetic algorithm [1] concept can be efficiently used in many optimization problems to provide an alternative solution. It can be used is several areas from data mining [2] to neural networks and deep learning [3]. GA does not guarantee the optimum, and the quality of the approximation highly depends on the implementation and on the carefully chosen design parameters.

One large area, where GA can efficiently be used, is image processing [4], e.g. image approximation.

In [5] an image approximation algorithm was presented using arbitrary triangles to approximate the back and white areas of images. Using different shapes, however, can lead to different results. In this paper additional shapes (rectangles and circles) will be used for the approximation, among the triangles.

The efficiency of the algorithm is highly affected by the design parameters. In this paper the effect of the used shapes, the maximum number of shapes, the mutation and crossover ratio and the distribution and the parameters of the random numbers used in generation of the new shapes and during the mutations are evaluated.

II. Related work

Genetic algorithms [1] are widely used [3], primarily for solving problems, where the number of parameters is too high and no quick solution is known. For most of those problems finding the optimal solution is not critical, and an approximation (or maybe any possible solution) can be acceptable.

An approximate solution can be given to classic NP-complete problems, Vehicle Routing Problem (VRP) [6] and Travelling Salesman Problem (TSP) [7], with genetic algorithms. Another two typical applications are transporting [8] and scheduling [9].

In mechanical engineering genetic algorithms can be used to optimize mechanical components, e.g. to make support elements with minimum weight (and from minimum amount of material) and with maximal strength [10].

Different areas, like fuzzy logic [11], data mining [2] or scheduling [12] can efficiently combined with genetic algorithms.

Genetic algorithms can be effectively used for economic applications [13], e.g. for automatized trading [14] or portfolio analysis [15].

A promising application is using genetic algorithm to optimize the weights of a neural network instead of using back propagation [3], [16]. Two interesting applications of neuro-evolution are an automated player for the classic Nintendo Super Mario Brothers video game (MarIO) [17] and an application, where a virtual robot battle player is controlled by a neuro-evolutionary algorithm [18].

A good application of genetic algorithms is image processing [19], e.g. image segmentation [4] or image enhancement [20]. This paper will focus on a similar application: image approximation [5].

III. The GA-based image compression

A. Problem statement

The algorithm work with 8-bit grayscale images with, and represent them as an $(n \times n)$ matrix:

$$IMG = (a_{i,j} \in 0, 1, \dots 255^{n \times n}), \tag{1}$$

where $a_{i,j}$ is the pixel in the (i, j) coordinates of the input image.

An approximation B of the image consist of the series of k shapes:

$$B = (s_1, s_2, \dots, s_k) \tag{2}$$

where each shape is given with a series of parameters. A shape s is defined as follows:

$$s = (t, p_1, p_2, \dots, p_{pn(t)}, c), \tag{3}$$

where t is the type of the shape and $p_1, \dots, p_{pn(t)}$ are the parameters of the shape, given that $pn(t)$ gives the number

of parameters for the t^{th} shape. The $c \in -255, \ldots, 255$ parameter gives the color of the shape.

Each approximation has a graphical representation R, similarly to the input image:

$$R = (b_{i,j} \in 0, 1, \ldots 255^{n \times n}), \tag{4}$$

where $b_{i,j}$ is the pixel in the (i, j) coordinates of the approximation.

B. Graphical representation

The genetic algorithm handles the following shapes:

- $t = 1$ represents a triangle with $np(1) = 6$ parameters, where (p_1, p_2), (p_3, p_4) and (p_5, p_6) belong the 3 vertices.
- $t = 2$ defines a rectangle with $np(2) = 4$ and (p_1, p_2) and (p_2, p_3) give the opposite vertices.
- Finally, $t = 3$ determines a circle with $np(3) = 3$ parameters, where (p_1, p_2) is the center of the circle and p_3 is the radius.

During calculation of R the shapes are rendered overlaying each other with adding their color to the actual image, stating from a white image. After adding a shape, the value of each pixel is kept between 0 and 255.

The aim of the algorithm is to find an approximation B with minimum error $e(B)$, i.e. with minimum difference between the original image IMG and the graphical representation R of the approximation B:

$$e(B) = \left| \sum_{i,j} (IMG_{i,j} - R_{i,j}) \right| \tag{5}$$

The conventional term related to genetic algorithms is *fitness function*, which can be defined as the opposite of the error:

$$f(B) = -e(B). \tag{6}$$

C. The genetic algorithm

The genetic algorithm uses a P *population* of p approximations:

$$P = B_1, \ldots, B_p. \tag{7}$$

At first each approximation of the population is initiated with an approximation contains a number of randomly generated shapes. Then every population is generated as follows.

- The graphical representation is rendered for each approximation.
- The fitness function is calculated using the graphical representations and the original image.
- A fixed number of approximations are selected from the population with the highest fitness, which are left intact.
- A fixed number of approximations are overwritten by new ones generated using crossover.
- The rest of the approximations are randomly modified in terms of the coordinates and the color with predefined probabilities.

For sake of simplicity the details of the genetic operators are not discussed here in detail. They can be found in [5].

D. Software architecture

The image approximation algorithm is implemented in C language with as few external dependencies as possible to be portable to different operating systems. The different parameters (i.e. population size, resolution, maximum number of shapes, shape types) can be set at compile time.

The software excepts the input image in BMP format in grayscale with a fixed resolution. The graphical representation of the approximation with the smallest error is saved in every tenth iteration in raw format and converted to JPEG using an external program.

The implementation uses a single thread only, but several instances with different settings can be run in parallel.

IV. Test results

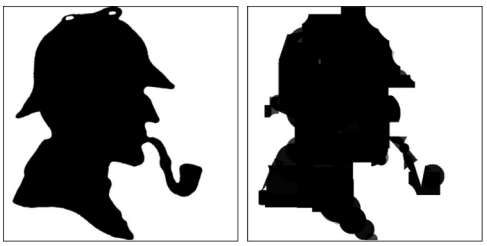
To test the performance of the method with different settings and to analyze the effect of different parameters the genetic algorithm was run with several different settings. The parameters can be summarized as follows:

- Maximum number of shapes: $20 \leq k_{max} \leq 100$
- Population size: $p = 4096$
- Intact approximations: 256
- Overwritten approximations: 1024
- Shapes: triangles only, rectangles only, circles only and all shapes mixed.
- Random shape generation and mutation with random numbers with uniform and Gaussian distribution.

During generating and modifying the shapes two different strategies are used. The *uniform* strategy uses uniform distribution for all random numbers, i.g. the vertices of the triangles and the rectangles, and the center and the radius of the circles are generated with uniform random distribution. The same is true for the modification of the vertices and the radii as well.

The *Gaussian* strategy uses random numbers with Gaussian distribution. During generation of triangles and rectangles first a center point is generated with uniform distribution and the vertices are shifted with Gaussian offsets. The circles are generated with uniform centers and Gaussian radii.

The test image can be seen in Fig. 1(a). It is a moderately detailed, well distinguishable silhouette of Sherlock Holmes. Fig. 1(b) contains the best approximation during the experiments with the number of shapes limited to 80 ($k_{max} = 80$).



(a) (b)

Fig. 1. The test image (a) and the best matching image after 5000 generations with the best performing settings.

Fig. 2 shows the best approximation of the test image after 50, 200 and 1000 generations using mixed shapes (upper row) and circles (lower row). The maximum number of shapes were $k_{max} = 80$.

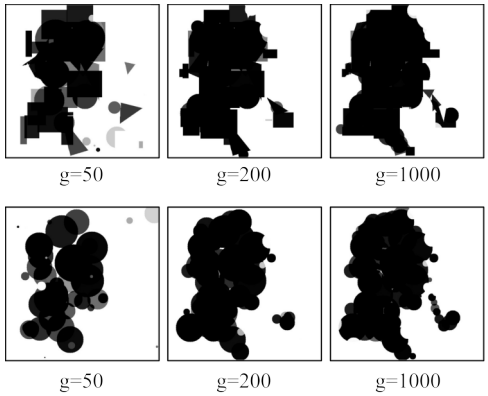


Fig. 2. The best approximation of the test image after 50, 200 and 1000 generations using mixed shapes (upper row) and circles only (lower row).

On Figs. 3 and 5 the effect of using different shapes and different random number strategies are visualized. For this experiment $k_{max} = 80$ was set. Fig. 3 shows the results of the uniform strategy, while Fig. 4 belongs to the Gaussian random number strategy. Gaussian distribution leads to slightly lower errors in each case. During the experiments triangles gave the worst performance, while the other two shapes and the mixed mode performed almost equally, being the mixed mode the best setting.

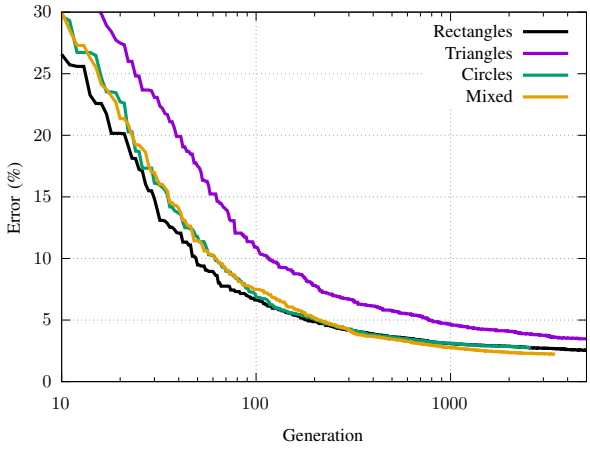


Fig. 3.

To evaluate the effect of k_{max} and the random distribution the same test image was approximated with 9 different k_{max} values between 20 and 100, and the genetic algorithm was run with the Gaussian random number strategy and with mixed shapes. Fig. 5 shows the error of the best approximation in each generation.

According to the tests the achievable minimum error decreases with more and more shapes. Using less than 40 or 50 shapes does not give a recognizable approximation. On the other hand, using more than 80 shapes does not significantly reduce the achievable error.

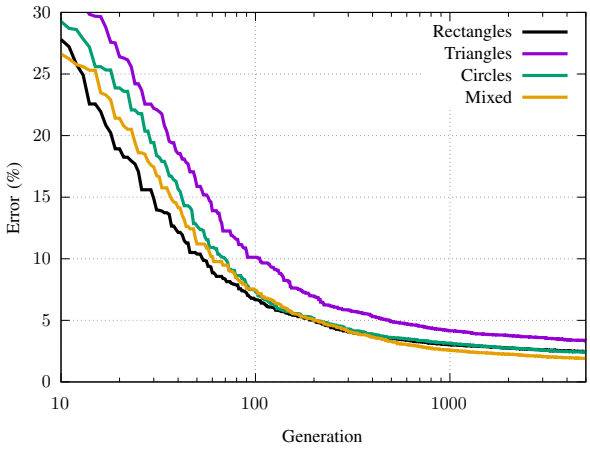


Fig. 4.

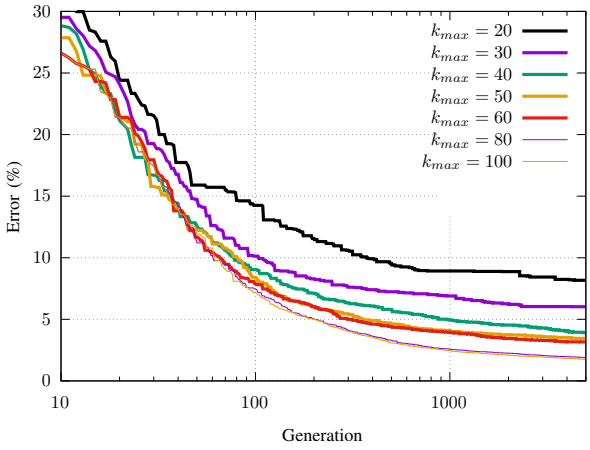


Fig. 5.

V. Summary

In this paper a genetic algorithm based image approximation method was presented, which extends a similar, existing solution. The approximation is based on a genetic algorithm and uses a limited number of triangles, rectangles, circles, or all of them. The paper discussed several different design parameters of the algorithm: different limits for the number of shapes, different random number generation strategies and different shapes. In all tests the triangle based approximation was outperformed by the others, and the mixed strategy provided the best results. With using Gaussian distribution during some part of the algorithm the results can slightly be improved.

References

[1] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*. Berlin Heidelberg: Springer-Verlag, 2008.

[2] T. Barman and N. Deb, "State of the art review of speech recognition using genetic algorithm," in *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*. IEEE, 2017, pp. 2944–2946.

[3] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *arXiv preprint arXiv:1712.06567*, 2017.

[4] S. Pare, A. K. Bhandari, A. Kumar, G. K. Singh, and S. Khare, "Satellite image segmentation based on different objective functions using genetic algorithm: a comparative study," in *2015 IEEE international conference on digital signal processing (DSP)*. IEEE, pp. 730–734.

[5] G. Vakulya, "Evaluation of a genetic algorithm based image approximation method," in *AIS 2021-16th International Symposium on Applied Informatics and Related Areas - Proceedings*, 2021, pp. 12–15.

[6] K. Q. Zhu, "A new genetic algorithm for vrptw," in *Proceedings of the international conference on artificial intelligence*, 2000.

[7] Z. H. Ahmed, "Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator," *International Journal of Biometrics & Bioinformatics (IJBB)*, vol. 3.6, p. 96, 2010.

[8] C. Lin, J. Yu, J. Liu, and C. Lee, "Genetic algorithm for shortest driving time in intelligent transportation systems," in *International Conference on Multimedia and Ubiquitous Engineering 2008*, 2008, pp. 402–406.

[9] C. Lim and E. S., "Production planning in manufacturing/remanufacturing environment using genetic algorithm," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, 2005.

[10] M. T. Bhoskar, M. O. K. Kulkarni, M. N. K. Kulkarni, M. S. L. Patekar, G. Kakandikar, and V. Nandedkar, "Genetic algorithm and its applications to mechanical engineering: A review," *Materials Today: Proceedings*, vol. 2, no. 4-5, pp. 2624–2630, 2015.

[11] J. J. Buckley and Y. Hayashi, "Fuzzy genetic algorithm and applications," *Fuzzy sets and systems*, vol. 61, no. 2, pp. 129–136, 1994.

[12] J. F. Gonçalves, J. J. de Magalhães Mendes, and M. G. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," *European journal of operational research*, vol. 167, no. 1, pp. 77–95, 2005.

[13] H. Dawid and M. Kopel, "On economic applications of the genetic algorithm: a model of the cobweb type," *Journal of Evolutionary Economics*, vol. 8, no. 3, pp. 297–315, 1998.

[14] R. Kuo, C. Chen, and Y. Hwang, "An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network," *Fuzzy Sets and Systems*, vol. 118, no. 1, pp. 21–45, 2001.

[15] S. Slimane and M. Benbouziane, "Portfolio selection using genetic algorithm," *Journal of Applied Finance & Banking*, vol. 2, no. 4, pp. 143–154, 2012.

[16] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[17] B. A., S. Y., R. G., and C. J, "Learning levels of mario ai using genetic algorithms," *Advances in Artificial Intelligence. CAEPIA 2015. Lecture Notes in Computer Science*, vol. 9422.

[18] S. Y., Z. E., and S. M., "Using genetic programming to evolve robocode players," vol. 3447.

[19] S. Jayaraman, S. Esakkirajan, , and T. Veerakumar, *Digital image processing*. India: Mc Graw Hill, 2009.

[20] H. Deborah and A. M. Arymurthy, "Image enhancement and image restoration for old document image using genetic algorithm," in *2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies*. IEEE, pp. 108–112.