Automatic Selection of GA Parameters for Fragile Watermarking

This copy represents the peer reviewed and accepted version of paper:

Marco Botta, Davide Cavagnino, Victor Pomponiu, "Automatic Selection of GA Parameters for Fragile Watermarking", Lecture Notes in Computer Science 8602 (2014), pp. 526-537

Automatic Selection of GA Parameters for Fragile Watermarking

Marco Botta[1(✉)], Davide Cavagnino[1], and Victor Pomponiu[2]

[1] Dipartimento di Informatica, Università degli Studi di Torino
Corso Svizzera 185, 10149 Torino, Italy
{marco.botta,davide.cavagnino}@unito.it
[2] Department of Radiology, University of Pittsburgh
3362 Fifth Avenue, Pittsburgh, 15213, PA, USA
vpomponiu@acm.org

Abstract. Genetic Algorithms (GAs) are known to be valuable tools for optimization purposes. In general, GAs can find good solutions by setting their configuration parameters, such as mutation and crossover rates, population size, etc., to standard (i.e., widely used) values. In some application domains, changing the values of these parameters does not improve the quality of the solution, but might influence the ability of the algorithm to find such solution. In other application domains, fine tuning these parameters could result into a significant improvement of the solution quality. In this paper we present an experimental study aimed at finding how fine tuning the parameters of a GA used for the insertion of a fragile watermark into a bitmap image influences the quality of the resulting digital object. However, when proposing a GA based new tool to non-expert users, selecting the best parameter setting is not an easy task. Therefore, we will suggest how to automatically set the GA parameters in order to meet the quality and/or running time performances requested by the user.

Keywords: Information hiding · Fragile watermarking · Genetic algorithms · Karhunen-Loève Transform

1 Introduction

The digital revolution of the last decade, in which every piece of information is represented, manipulated, stored and reproduced in digital form, brought about new opportunities along with new challenges. One important issue, that is deserving a lot of attention in the literature, is how to determine if a digital object is genuine, i.e., it has not been altered with respect to its original version.

A possible solution to ensure the protection of the media content is digital watermarking [5]. In general, the watermarking process consists of two phases: the embedding phase and the verification phase. During embedding, the digital host object is modified to carry a watermark signal. Then the watermarked object is released into an environment that may alter it. The aim of the verification is to look for the presence of the watermark into the (possibly) altered object. To improve the security, in case the host object is an image the watermark can be embedded in transform domains like the discrete cosine transform (DCT) or the Karhunen-Loève transform (KLT).

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2014, LNCS 8602, pp. 526–537, 2014.
DOI: 10.1007/978-3-662-45523-4_43

The watermark embedded into the host object could be *robust* against manipulations or *fragile*. Fragile watermarking is a particular class of schemes that uses the watermark to alert for any alterations induced to the host signal. There are three main properties required for a fragile watermarking algorithm: 1) the ability of the fragile watermark to detect alterations, 2) the capacity of the fragile watermark to localize the tampered areas and 3) the preservation of the quality of the host signal.

In this paper we evaluate the performance of a Genetic Algorithm (GA) used for embedding a watermark into a bitmap image. The watermark is inserted into a secret space of features extracted from the image, and the GA is used to modify the pixels of the image in such a way that these features contain the intended watermark. In this way, only the entities having knowledge of the secret space will be able to detect modifications to the image by extracting the watermark and comparing it with the one that was inserted, providing a tool to check the authenticity of the image. Moreover, without the knowledge of the secret space it is highly unlikely to successfully tamper (parts of) the image without altering the watermark.

The performance of the GA will be evaluated by varying some of its parameters and measuring the time required for the watermark insertion and the quality of the resulting images. It should be pointed out that in the presented application, the GA is run multiple times, depending on the size of the image to be watermarked, and we must use the best settings that results in lower running times and highest quality. Therefore, after observing the relation among quality, time and GA parameters, we extended the algorithm with the ability to automatically select the appropriate GA parameter settings in order to fulfill the user requirements in terms of quality.

The rest of the paper is organized as follows: in the next section some previous works related to the use of GA for watermarking will be analyzed, while in section 3 the watermarking algorithm will be briefly presented. Section 4 will discuss the experimental results and an analysis of the GA parameters tuning. The final section will draw some conclusions.

2 GA-Based Watermarking Schemes

There are two main techniques to improve the performances of a watermarking system. The first is to use statistical properties during the watermark verification (e.g. in detection). The latter is to employ genetic optimization to find the values of the embedding features that generate almost optimal performances, in terms of imperceptibility and efficacy. This is the most common approach, due to the simplicity of the technique (does not imply mathematical analysis) and the ease in adapting it to many different types of watermarking systems.

Wang et al. [16] optimized a Least Significant Bit (LSB) substitution watermarking method with the use of GA. The insertion works in the spatial domain, and is based on a mapping function that is optimized by a GA in order to find one that achieves both robustness and imperceptibility. The fitness function takes into consideration the distortion induced by the watermark insertion. A similar approach is adopted by Wu et al. [17] who generates optimal mapping functions for finer regions (blocks) of the host image in order to increase the quality of the watermarked image.

Shieh et al. [12] introduce a GA-based watermarking in DCT domain. The watermark insertion is performed via the manipulation of the polarity between the watermark and the DCT coefficients. The GA is employed to find the DCT coefficients that give the optimal trade-off between robustness and image quality. The same strategy was used by Lu et al. [11] for the DWT coefficients to embed the watermark in a color image while Huang et al. [8] adopt a slight variation of [12].

In order to optimize a DCT-based watermarking method Díaz and Romay [6] use a Pareto-based Multi-Objective Genetic Algorithm. The parameters considered for the optimization are the DCT coefficients. The fidelity and robustness are measured directly on the selected coefficients.

Usman et al. [15] present an algorithm that uses the DCT domain for embedding a fragile watermark with the purpose of content integrity. The host image is divided into blocks of size 8×8, and a GA is used to select five DCT coefficients per block for storing the watermark; at the same time, to deal with possible attacks, the non-selected coefficients of the block and of its neighbours are also involved in the watermark embedding. The GA selects the coefficients using a fitness function that considers the distortion w.r.t. the host image.

Aslantas et al. [1] compare some optimization methods by applying them to an algorithm that inserts a fragile watermark into the DCT coefficients of an image. Given that the inverse transformation in the pixel domain may alter some of the inserted bits due to the integer rounding of the pixels, an optimization step is required to restore the correct watermark values. Differential evolution, clonal selection, particle swarm optimization and genetic algorithms are evaluated varying the parameters of each algorithm and comparing the resulting fitness values, computation times and watermark imperceptibility.

Lee and Ho [9] describe the insertion of a fragile watermark in the LSBs of the pixels of an image. The image is divided into blocks that are classified according to the type of edge content (using the discrete cosine transform coefficients). The edge block classification is used in the fitness function of a genetic algorithm employed for the insertion of the watermark bits.

Shih and Wu [13, 14] applied a GA in order to cope with rounding errors that can occur in the DCT coefficients during the embedding stage. The basic issue is that in DCT embedding, integer pixel values are transformed into real value DCT coefficients followed by the insertion of the watermark. Afterwards, the watermarked coefficients are transformed back to integer pixel values by an inverse DCT. However, information might be lost due to rounding errors. The fitness function applied by the scheme is based on the normalized correlation (NC) between embedded and detected watermark and the distortion between host and watermarked images.

3 The Watermarking Algorithm

Some of the previously cited papers compute a linear transform[1] from pixels to a coefficient space (e.g., DCT), also called frequency space or frequency domain, insert the watermark into the coefficients, inverse transform the coefficients into the pixel domain, and then use a GA to improve the quality of the watermarked image by either

[1] For a presentation of linear transforms refer to [7].

selecting the best coefficients or optimizing the pixel values in order to solve rounding problems.

The insertion into the coefficients and their inverse transformation are actually not needed when using a GA. Indeed, the GA can be used to optimize pixel values in such a way that when transformed to the coefficient space, the selected coefficients already carry the watermark bits. In the following, we give a description of the algorithm that is an improvement of the one presented in [3], and implements this new idea of not inserting the watermark bits into the KLT coefficients but let the GA free to modify the pixel values in every block in such a way that the watermark extracted from the KLT coefficients is the intended one. Figure 1 shows a block diagram of the watermark embedding scheme.

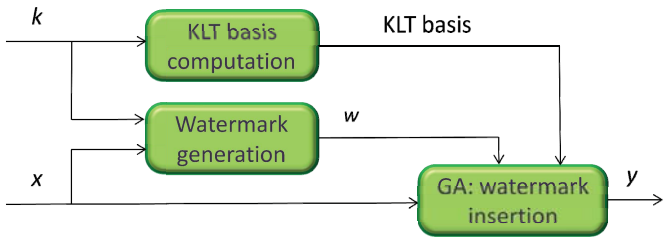


Fig. 1. Block diagram of the watermark embedding

Firstly, a KLT basis is computed from a key image k: its result may be used for watermarking many host images, and is performed only once per key image. The key image, which must be kept secret, is divided into blocks of size $n \times n$ and from the blocks, a Karhunen-Loève basis is derived. This basis defines a linear transformation from the space of pixels to a frequency domain [7]. When a block of $n \times n$ pixels is transformed with this basis, it will produce n^2 coefficients.

In the watermark generation, a cryptographic hash function is applied to a set of pixels of the key image k and of the host image x, to generate a pseudo-random binary sequence that will be used as fragile watermark w for the host image, in order to prevent cut-and-paste attacks, birthday attacks and transplantation attacks [2].

Then, the host image is split into sub-images of size $n \times n$, the watermark is divided into chunks of s bits and each chunk is assigned to one sub-image (for example in raster scan order). For every sub-image, the GA is run and evolves a population of individuals that define a modification of the sub-image pixels, in such a way that when the KLT basis is applied to the sub-image, a secret set of n^2 coefficients is generated, and the s selected coefficients of the KL transformation store the s watermark bits. Thus, an individual of the GA population is composed of n^2 integers (typically in the range [−3, 3]) that added to the pixels produce a modified sub-image. A watermark bit b is extracted from a coefficient c according to the following rule:

$$b = \mathrm{round}(2^{-p}c) \bmod 2 \tag{1}$$

where p is the position (in the binary representation of c) where the watermark bits are stored.

The GA usually runs for a maximum number of generations, but it can be terminated as soon as a viable solution is found. The GA fitness function may take into account many parameters; typically, the distortion of the modified sub-image w.r.t. the original one is a considered factor, and the other is the fact that the modified sub-image should store the s watermark bits. The fitness function F we used for our GA takes into account these two terms: the Bit Error Rate (*BER*), i.e., the correct extraction of the watermark bits from the KLT coefficients, and the Mean Square Error (*MSE*), i.e., the distortion w.r.t. the host image:

$$F = \alpha \cdot BER + \min(\beta, MSE) \tag{2}$$

where $\alpha > \beta$ are chosen so that if $F \leq \beta$ then $BER = 0$; this allows to verify if the GA found a viable solution. Moreover, we adopted the convention the smaller F is, the better the individual.

The result of running the GA optimization on all the host sub-images is the watermark stored in a secret space, so it cannot be easily extracted by an attacker; moreover, any modification to the watermarked image y will result in the modification of one or more watermark bits stored into the coefficients.

A measure of the objective image quality is the Peak Signal-to-Noise Ratio (*PSNR*) computed as $PSNR = 10 \log_{10} \frac{255^2}{m.s.e.}$ (for 256 grey levels images) where *m.s.e.* is the mean square error between the host image pixels and the watermarked image pixels. The higher this value the better is the resulting watermarked image.

To verify the integrity of a previously watermarked image the secret KLT basis is used to compute the coefficients: from these the watermark bits may be extracted as specified in (1), and compared to the original ones; possibly differing bits reveal a tampering.

4 Experimental Results

Almost all previously cited works did not perform a thorough analysis and tuning of GA parameters, but just reported the used settings. In a previous study [4], we also focused on the analysis of the watermarking algorithm properties, by setting the GA parameters to default values (population size=100, mutation probability p_m=0.05, crossover probability p_c=0.8, terminate if best individual does not change in the last 10 generations or 2000 generations reached) and obtaining good quality results (*PSNR* between 53 and 54 dB).

Here we evaluate the performances of the proposed algorithm and further investigate whether a wise selection of parameters for the GA may further improve the quality of the resulting watermarked images.

We ran a large number of experiments, for a total of more than 90000 watermarked images, on about 100 different combinations of the four GA parameters we studied in this paper, namely population size, number of stable generations (i.e. the minimum number of generations the best solution found so far by the GA does not improve), mutation and crossover probabilities, and collected quality and running times.

For every parameter combination, we report average values computed by inserting a watermark of 8 bits per block (of size 8×8) into 1000 images taken from OPTIMOL [10]: the images are 256 gray levels bitmaps of 256×256 pixels. All experiments have been performed on a set of workstations, each equipped with 4GB RAM and an Intel(R) Xeon(R) E5410 2.33GHz processor. As a multi-dimensional plot would not be easily readable, we will project some of the results obtained along the dimensions we investigated and show simpler plots.

4.1 Convergence Ability

In this application we faced two aspects of convergence: for some sub-images there is a large number of easy reachable solutions and the GA might premature converge to a local minima, while for other sub-images a solution might not even exist, and the GA does not converge at all. To address the premature convergence issue, we varied the number of stable generations as we report in the following. For what concern the second issue, Figure 2 reports the number of images for which the GA failed to find a solution: with population sizes smaller than 50, even setting 70 stable generations, there are a few images on which the GA fails. We also noted that with mutation probability smaller than 0.03 the algorithm does not find a solution in all the cases, even setting population size = 100 and stable generations = 70.

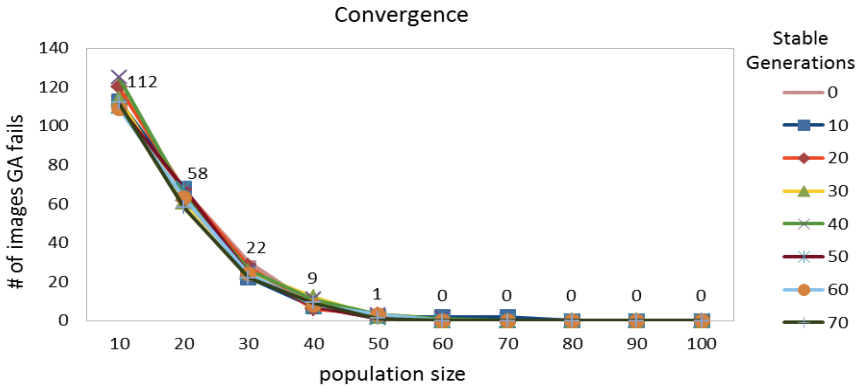


Fig. 2. The covergence of the algorithm vs the size of the population, varying the number of stable generations required to terminate. The labels report the number of images the GA failed for 70 stable generations.

As we would not allow for GA failures (i.e., we want that the watermark is always carried by the image after running the algorithm), we performed experiments and report results for population sizes larger than 50 individuals and $p_m \geq 0.03$.

4.2 Image Quality

The first experiment was aimed at analyzing how smaller populations than the default (100) influence the performances of the algorithm, in order to find an optimal trade-off between quality and running time. Moreover, for every population size, we varied the number of stable generations, from 0 (stops as soon as a solution is found) to 70 (stops when the best solution does not change for 70 consecutive generations), in steps of 10. Figure 3(a) reports the average quality of the watermarked images as a function of population size (different lines) and number of stable generations, by setting $p_c = 0.8$ and $p_m = 0.05$ (similar behaviours were obtained with other combinations of p_c and p_m). As expected, increasing the GA population size slightly improves the quality of the resulting images, but the gain becomes less meaningful for increasingly larger populations. This is probably due to the fact that the exploration ability of the GA in this context does not depend so much on the number of individuals, but rather on other GA parameters. Indeed, as it can be seen from Figure 3(a), the *PSNR* increases by 6 dB for an increasing number of stable generations in which the GA was left running after having reached a solution. This behavior was not anticipated even though it is not really surprising: the fitness function we used has a lot of local minima, so by letting the GA run for longer, a better solution can be found. Anyway, by analyzing the running times (see Figure 3(b)) larger populations and larger number of stable generations result in longer running times. It should be pointed out that there is a linear correlation between *PSNR* and time, on one side, and population size and stable generations on the other side. We will exploit these correlations later on.

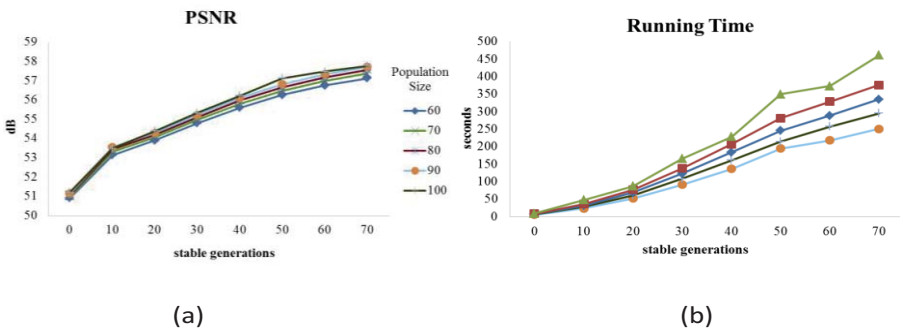
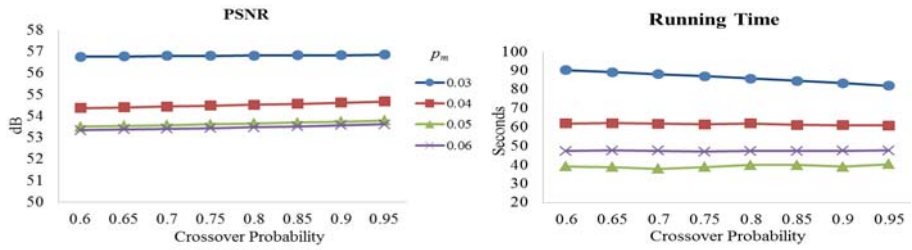


Fig. 3. The *PSNR* (a) and Running Time (b) vs the number of stable generations, varying the size of the population, $p_c = 0.8$, $p_m = 0.05$

4.3 Crossover Probability

Figure 4(a) shows the influence of the crossover probability on the image quality. Again, the population size has been set to 100, stable generations to 10 and the mutation probability varies between 0.03 and 0.06. We note that large crossover probabilities do not significantly increase the *PSNR* of the watermarked images, as crossing good individuals better explores solutions that may be quasi optimal. Running times are not very affected by crossover probability (Figure 4(b)), even though p_c=0.9 is consistently faster than p_c=0.8.

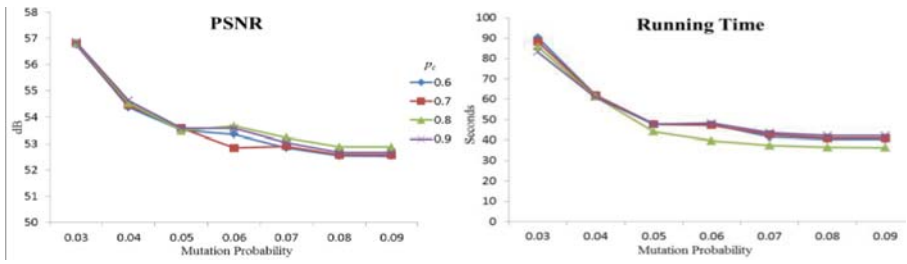


(a) (b)

Fig. 4. (a) The watermarked image quality (measured in dB) vs the crossover probability and (b) the average running time for one image vs the crossover probability

4.4 Mutation Probability

Figure 5 reports the quality and running times of the watermarked images when varying the mutation probability used by the GA, and by setting the population size to 100, stable generations to 10 and p_c from 0.6 to 0.9 (in steps of 0.1). As pointed out above for mutation probabilities lower than 0.03, the algorithm does not always find a solution. Both *PSNR* and running times decrease with larger mutation probabilities, so the algorithm is faster but with lower quality. This is due to the fact that once a solution is found, it is likely that it remains the best for the requested stable generations, as new offsprings will undergo mutation at a higher rate and explore far from optimal solutions. There is no significant difference in terms of PSNR, even though $p_c = 0.8$ is faster than the other settings for increasing values of p_m; nonetheless a value of p_m larger than 0.04 worsen the achievable results.



(a) (b)

Fig. 5. (a) Quality of the watermarked images (in terms of *PSNR*) vs the probability of mutation. (b) Running time (for a single image) of the insertion algorithm vs the probability of mutation.

4.5 Sensitivity to Modifications

Given that the objective of fragile image watermarking is the detection of unauthorized pixel modifications, a set of tests aimed at verifying the ability of the algorithm to detect modifications to a single pixel in an image block were performed. In particular, we wanted to verify if the images with increased quality (i.e. *PSNR*) obtained from a fine tuning of the GA parameters, were still bearing a watermark able to detect the modification of a single pixel by a single gray level. Figure 6 reports the percentage of image blocks from which an alteration of a single pixel (by ±1 gray level, Figure 6(a), and ±2 gray levels, Figure 6(b)) was detected by the verification algorithm.

As it can be seen the sensitivity does not significantly increase for increasing populations for small number of stable generations, while for larger values of stable generations the sensitivity of the detection algorithm raises of 6-8% for ±1 and 3% for ±2. We can conclude that a larger number of stable generations not only results in higher quality watermarked images, but also increases the probability of detecting alterations.

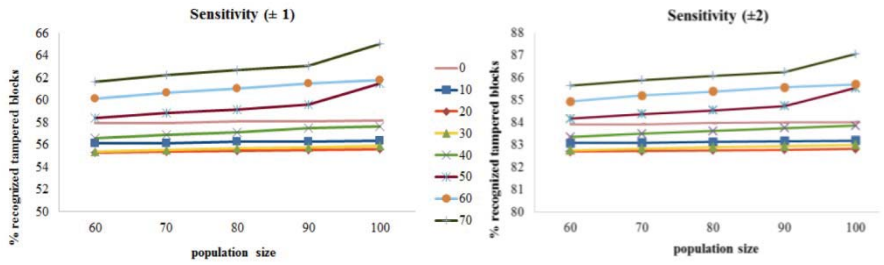


Fig. 6. The percentage of recognized tampered blocks (changing the gray level by ±1 (a) and by ±2 (b)) vs the size of the population, varying the number of generations required for a stable fitness value, $p_c = 0.8$, $p_m = 0.05$

5 Discussion and Conclusions

The GA-based optimization techniques have been efficiently applied in many different watermarking scenarios. A base line watermarking system, due to its modular nature, can be easily extended to incorporate a GA.

However, the use of a GA rises several important concerns such as the dimensionality of the parameters to be adjusted in order to achieve an optimum tradeoff between robustness and quality, the time overhead, and the statistic assumption (i.e., Gaussian) of the host signal and noise source. Moreover, for a non-expert user to optimally set GA parameters is even more difficult.

In this paper, by fine tuning the parameters of the GA, we significantly improved (> 5dB) the performances of a watermarking system without affecting the sensitivity of the embedded watermark, but at the expense of larger computation times. It should be pointed out that from a pure statistical point of view, all the reported results are statistically significant (p<0.0001), due to the large number of images used for testing, but from an application point of view, only changes of PSNR greater than 1dB can be considered a real improvement.

From these results, we can provide the user with an easy tool to automatically select the best combination of GA parameters that provides the requested performances.

As shown in the reported graphs, there seems to be a quasi-linear correlation between *PSNR,* on one side, and population size and stable generations on the other side, so we used a linear regression algorithm to come up with the following formula:

$$PSNR = 1.7906*p_c - 87.0244*p_m + 0.0265*\text{popsize} + 0.0757*\text{stableGen} + 53.8123 \quad (3)$$

A similar expression can be derived for the expected running time for watermarking an image:

$$time = 96.3561*p_c - 466.5239*p_m + 1.8742*\text{popsize} + 5.1698*\text{stableGen} - 223.1268 \quad (4)$$

As we obtained slightly better quality results by setting p_c= 0.9 and p_m= 0.03, from (3) and (4) one can derive the minimum values of population size and stable generations that provide a given *PSNR* value in the lowest running times.

Figure 7 shows a knob that is presented in the user interface of the application: it allows the user to set the desired quality of the resulting watermarked image. Depending on this setting, the system automatically selects the best combination of GA parameters, according to the solution of expressions (3) and (4), as outlined above, that results in the fastest running times.

For instance, if the user sets the knob to 57 dB, the system automatically chooses p_m=0.03, p_c=0.9, a population size of 70 and number of stable generations = 30, for a lowest running time of 135.88 seconds. It should be pointed out that the running time is related to a specific hardware configuration, and it might be lower or greater on the user's computer. Here it is just used to find the best values of population size and stable generations and not to predict the running time.

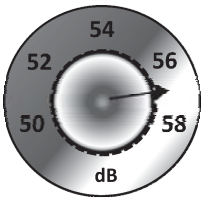


Fig. 7. The knob for selecting the desired quality in dB

As a final consideration, we point out that the overall best performance of our algorithm is 59.78 dB (p_c=0.9, p_m=0.03, popsize=100, stable generations=50) when inserting 8 bits-per-block, compared to a best performance among the papers cited in Section 2 of 57.66 dB, when inserting only 4 bits-per-block. Moreover, it should be noted that in many cases the comparison with other algorithms is not easy due to the different payloads and insertion methods used, so we referred to the data reported in the papers. Anyway, the difference in quality at this level is very significant according to a statistical t-test (p<0.001).

The aim of this study was not only that of finding the GA parameters that deliver the best performances, but also to analyze how they influence the GA performance in optimization problems such the one considered here: in our system, we used a standard GA (SteadyStateGA from GALib) that already implements a number of well-known techniques to control how the population evolves. From our experiments, we

found that changing the common GA parameters, such as mutation and crossover probabilities or population size, does not influence the performances as much as changing the number of stable generations. We think that this is mainly due to the fact that the optimization problem has a large number of solutions, so finding a viable one is pretty simple, but letting the GA running for more generations helps improving the quality of the solution returned. This should be taken into consideration when using GAs in other application domains in which there are a lot of possible solutions.

As future work, we are planning to investigate the implications of automatically choose different GA parameter settings for different sub-images, by predicting the effort necessary to store the watermark bits in each of them, instead of using the same configuration on every sub-image.

References

1. Aslantas, V., Ozer, S., Ozturk, S.: Improving the performance of DCT-based fragile watermarking using intelligent optimization algorithms. Optics Communications **282**(14), 2806–2817 (2009)
2. Barreto, P.S.L.M., Kim, H.Y., Rijmen, V.: Toward secure publickey blockwise fragile authentication watermarking. In: IEE Proceedings - Vision, Image and Signal Processing 2002, vol. 148(2), pp. 57–62 (2002)
3. Botta, M., Cavagnino, D., Pomponiu, V.: KL-F: Karhunen-Loève Based Fragile Watermarking. In: 5th International Conference on Network and System Security NSS 2011, pp. 65–72 (2011)
4. Botta, M., Cavagnino, D., Pomponiu, V.: Fragile watermarking using Karhunen-Loève transform: the KLT-F approach. Accepted for publication in Soft Computing, Springer (2014)
5. Cox, I.J., Miller, M.L., Bloom, J.A., Fridrich, J., Kalker, T.: Digital Watermarking and Steganography, 2nd edn. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2008)
6. Díaz, D.S., Romay, M.G.: Introducing a watermarking with a multi-objective genetic algorithm. In: Proceedings of the 2005 conference on Genetic and evolutionary computation (GECCO), pp. 2219–2220 (2005)
7. Gonzalez, R.C., Wintz, P.: Digital Image Processing, 2nd ed. Addison-Wesley Publishing Company (1987)
8. Huang, H.-C., Chu, C.-M., Pan, J.-S.: The optimized copyright protection system with genetic watermarking. Soft Computing **13**(4), 333–343 (2009)
9. Lee, S.-K., Ho, Y.-S.: Fragile watermarking scheme using a simple genetic algorithm. In: International Conference on Consumer Electronics, pp. 190–191 (2002)
10. Li, L.-J., Wang, G., Fei-Fei, L.: OPTIMOL: automatic Object Picture collecTion via Incremental MOdel Learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2007)
11. Lu, Yinghua, Han, Jialing, Kong, Jun, Yang, Yulong, Hou, Gang: A Novel Color Image Watermarking Method Based on Genetic Algorithm and Hybrid Neural Networks. In: Greco, Salvatore, Hata, Yutaka, Hirano, Shoji, Inuiguchi, Masahiro, Miyamoto, Sadaaki, Nguyen, Hung Son, Słowiński, Roman (eds.) RSCTC 2006. LNCS (LNAI), vol. 4259, pp. 806–814. Springer, Heidelberg (2006)
12. Shieh, C.-S., Huang, H.-C., Wang, F.-H., Pan, J.-S.: Genetic watermarking based on transform-domain techniques. Pattern Recognition **37**(3), 555–565 (2004)

13. Shih, F.Y., Wu, Y.-T.: Robust watermarking and compression for medical images based on genetic algorithms. Information Sciences **175**(3), 200–216 (2005)
14. Shih, F.Y., Wu, Y.-T.: Enhancement of image watermark retrieval based on genetic algorithms. Journal of Visual Communication and Image Representation **16**(2), 115–133 (2005)
15. Usman, I., Khan, A., Chamlawi, R., Majid, A.: Image Authenticity and Perceptual Optimization via Genetic Algorithms and a Dependence Neighborhood. International Journal of Applied Mathematics and Computer Sciences **4**(1), 37–42 (2007)
16. Wang, R.-Z., Lin, C.-F., Lin, J.-C.: Image hiding by optimal LSB substitution and genetic algorithm. Pattern Recognition **34**(3), 671–683 (2001)
17. Wu, Ming-Ni, Lin, Min-Hui, Chang, Chin-Chen: A LSB Substitution Oriented Image Hiding Strategy Using Genetic Algorithms. In: Chi, Chi-Hung, Lam, Kwok-Yan (eds.) AWCC 2004. LNCS, vol. 3309, pp. 219–229. Springer, Heidelberg (2004)