Hybridizing Genetic Algorithms and Path Relinking for Steganography

André Luiz Brazil[1], Angel Sanchez[2], Aura Conci[1], Narcis Behlilovic[3]
[1] Instituto de Computação (Universidade Federal Fluminense), 24210-240 Niterói, Brazil
[2] Departamento de Ciencias de la Computación (Universidad Rey Juan Carlos), 28933 Móstoles (Madrid), Spain
[3] Faculty of Electrical Engineering (University of Sarajevo), 71000 Sarajevo, Bosnia and Herzegovina
aconci@ic.uff.br

Abstract - Each day the Internet connects more people, increasing the need for transmitting secure information. One way to protect the data sent over the web is to conceal the relevant information inside a typical image, hiding the data from intruders. This paper proposes a hybrid heuristic, combining a genetic algorithm and the path relinking metaheuristic to efficiently solve this problem. Computational results show that the proposed algorithm outperforms the LSB (least significant bits) substitution technique, concerning the quality of solutions. In this way, the inclusion of a path relinking procedure can significantly improve the performance of a genetic algorithm for the problem considered.

Keywords – Steganography; LSB substitution; information hiding; image encryption; genetic algorithm; path relinking.

I. INTRODUCTION

Nowadays lot of effort has already been taken on behalf of data protection. The easier and most common way to do this is protect the data using passwords or data cryptography [1-4]. These ways are somehow secure in denying an intruder from getting important information, and many other forms of protection can also be used, but they generally present a significant drawback: the simple evidence of concealment, may be sufficient for the invaders to start trying to get the information. The image hiding process does not present this disadvantage [5, 6]. Great number of people sends pictures inside electronic mail. If important data can be hidden inside a picture without degrading its quality in a perceivable level, crackers may not even notice there is relevant information inside the message. Then steganography protects not only the data, by encoding it inside the image [6], but also avoids the risk of hackers catching the message and trying to unlock the protections.

There are many schemes to hide data inside an image. This work focuses in the use of genetic algorithm combined with a path relinking refinement to improve the results [7-11]. It is organized as follows. Section 2 explains how to hide data inside an image by using the LSB (least significant bits) substitution and applying a GA (genetic algorithm). This GA speeds up the process of finding and choosing the best solutions among the vast number of possibilities and improves the final image quality of the hiding process. In section 3, we detail the new refinement introduced by path relinking that seeks to improve yet more the final quality of the resulting image. Section 4 presents the experimental results and shows a comparison with other existing image hiding methods. Finally, in Section 5, the conclusions to this study and some future ideas are commented.

II. LSB SUBSTITUTION AND GENETIC ALGORITHM

Digital images are commonly described as two-dimensional arrays of pixels [12-15]. Each value of this array is formed by a number of bits, usually 8 bits for a grayscale or 24 bits for a color image. The LSB substitution alone itself proposes to hide information inside the rightmost bits of each image pixel. The resulting image degradation level is directly related to how many bits of the original image are used to hide the data. The LSB substitution can be improved by the use of a substitution matrix. The main idea of the substitution matrix is to convert some colors into others to reduce the resulting image degradation after the LSB substitution. The substitution matrix (see Table 1) is a simple color conversion structure with size $n \times n$, where n is the number of least significant bits being used to hide data. The rows represent the initial colors, while the columns represent the converted colors. The matrix positions can be filled either with the value 0 or the value 1, where '0' indicates nothing will happen and '1' indicates that the row color will be converted into the column color. It is possible to generate several different substitution matrices of the same size that can be applied after the LSB substitution, but among them, it is important to determine the best one to be used. The number of possible substitution matrices is $2^n!$ where n is the number of least significant bits used to hide data. When n=3 or more bits are used, the number of possibilities increase exponentially and it becomes very hard to find the best substitution matrix. For these cases, it becomes necessary to apply an approximate heuristic to speed up the search. The scheme proposed in [7] used a genetic algorithm (GA) to improve the speed of search for a near optimal substitution matrix, in order to enhance the quality of the resulting image. A diagram of the method proposed in our work is presented in Figure 1. To use a genetic algorithm it is necessary to convert the solutions (substitution matrices) into a format that can be handled by the algorithm: an individual that is a gene vector.

The substitution matrix can be converted into a vector *G*, where each position of the vector corresponds to the same row in the matrix (i.e. vector position 0 = row 0, vector position 1 = row 1, and so on) and the values contained inside these vector positions (genes) will be the column indexes used in *S* where the value 1 appears only once for each given row. This conversion is better shown by the G column of Table I.

TABLE I. 2X2 COLOR SUBSTITUTON MATRIX AND ITS CONVERSION INTO *G* INDIVUDUALS

Original color	Resulting color				
	00	01	10	11	G
00	0	0	0	1	3
01	0	1	0	0	1
10	0	0	1	0	2
11	1	0	0	0	0

Note : It can have only one value 1 for each row/column, in order to maintain all color equivalences

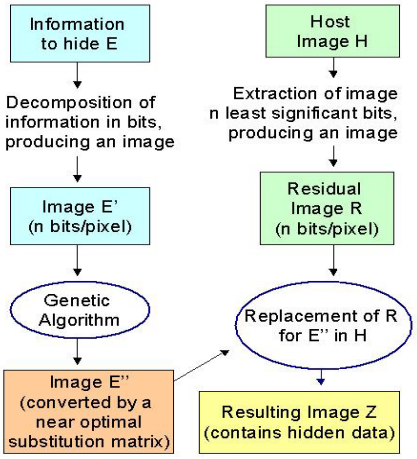

Figure 1. LSB Substitution and Genetic Algorithm for Steganography

The genetic algorithm can be split into the following steps: **Step 1:** Generation of a initial population; **Step 2:** Selection and combination of fragments of existing individuals to produce new ones (crossover); **Step 3:** Mutation of existing individuals, generating new ones; and **Step 4:** Choice of the individuals having best fitness values (between all obtained in this generation) to form the population for the next generation until some stopping criterion apply (i.e. maximum number of generations, individuals or elapsed time). The initial population to start the algorithm (**step 1**) can be obtained by several ways, like a heuristic algorithm, or generating random individuals and so on. The genetic algorithm in [7] generates a set of 10 random individuals to its initial population. The crossover (**step 2**) is a well known method of combining individuals, where some criterion splits two or more individuals into parts and one or more parts of each individual are combined together to form new individuals. In [7], the crossover adopted was a random choice of two individuals (parents) *G1* and *G2* to be combined. Each individual is divided in two equal parts and the first part of *G1* is joined with the second part of *G2* and vice-versa, composing two new individuals (offsprings), *G1'* and *G2'*. After the crossover, the offsprings individuals must be validated, because they may contain some repeated genes and be missing others. In these cases, the repetitions must be fixed by replacing the repeated genes by the missing ones. This validation/fix process occurs in the following two steps: (1) Check each gene of an offspring and form a list of found genes. If a repeated gene is found (appears a second time in the individual) this value is substituted by -1 to mark a vacant position in the individual; (2) Analyze the list of found genes, verifying which ones are missing and then put these missing values in order in each vacant position in the individual (marked by value -1). This crossover process was iterated 10 times, by combining the 10 random chosen pairs of individuals to form the offsprings. Note that the same individual can participate more than once in the whole process (it can be chosen several times).

The mutation itself (**step 3**) helps the genetic algorithm to avoid falling into the local minima problem. It becomes more important when the crossover process combines very similar individuals, thus generating offsprings almost equal to their parents and hanging the evolution of the genetic algorithm. However, the mutation should be applied in a small rate to not compromise what was found until then. In [7], the mutation process is concluded with other steps. Also an elitist criterion was used by selecting only the 10 best individuals to proceed into the next generation, discarding the other ones. The fitness of each individual can be measured by quantitative criterions like the PSNR (Peak Signal-to-Noise Ratio).

III. PATH RELINKING REFINEMENT

The path relinking approach presented in [8] is a very interesting method of local search, and can be applied to a wide range of applications. Other metaheuristic procedures that work with an elite candidate list or a selection of best solutions, like GRASP (Greedy Randomized Adaptive Search Procedure), Tabu Search, VNS (Variable Neighborhood Search), could be applied in this refinement step. Path relinking starts by choosing two among the solutions presented in the best solutions list, electing one of them as the starting point (*S1*) and the other one as the guiding solution (*S2*). Note the ideal condition is to choose two very different solutions from the list, allowing a larger degree of diversity in the solutions generated by the path relinking process and providing a bigger chance to escape from local minima. The next step is to verify the differences between *S1* and *S2*, so it can gradually start to alter parts of *S1* so it becomes each time more similar to *S2*, and map these newly found solutions as intermediary solutions, keeping the ones that transform *S1* into *S2*. The main idea of the method is shown in Figure 2.

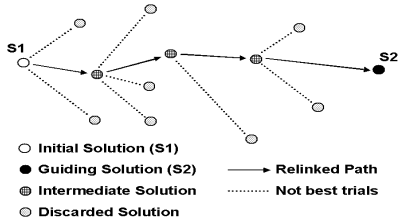

Figure 2 – Path relinking process.

In image hiding, the path relinking process can be applied in the following way. After each execution of the genetic algorithm, it returns a selection of the 10 best individuals. These individuals are then grouped randomly in pairs with the corresponding individuals of the residual image (obtained from the host image as shown in Figure 1). Inside each pair of individuals, one of them is designated as a starting solution *S1* and the other as a guiding solution *S2*. The representation **row → column** is used to indicate the color conversions present in each individual and facilitate the understanding of the path relinking process. To illustrate the process, let us suppose the solutions *S1* and *S2* below have being chosen, both 2×2 substitution matrices (k=2) are:

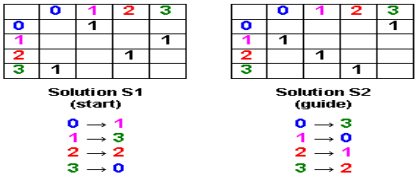

Figure 3 – Path relinking example: initial and guiding solutions.

Now let us focus on how a sample start solution *S1* can be transformed into a guiding solution *S2* (see Figure 3). The first step is to look at the guiding solution and identify which parts in *S2* are different from *S1* solution. In this case, none of the conversions in *S2* (**0** → **3**, **1** → **0**, **2** → **1** and **3** → **2**) exist in *S1*, so it is possible to start transforming *S1* into *S2* by changing any of these parts. In fact, all alternatives must be tried to see which one is the best. Starting with **0** → **3**, for example: in order to transform *S1* so it incorporates the **0** → **3** conversion, it is necessary to find inside *S1* the following conversions: (1) The conversion whose original color is **0**: there is **0** → **1** in *S1*; (2) The conversion whose transformed color is **3**: there is **1** → **3** in *S1*. To achieve the conversion **0** → **3** in *S1*, the colors present in the conversions **0** → **1** and **1** → **3** must be swapped. This is illustrated by Figure 4.

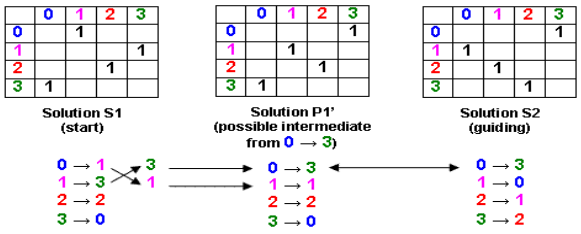

Figure 4 – Path relinking example (continuation): transformation between initial S1 and intermediate P1' solutions.

Now, there is a possible intermediate solution *P1'*. This process must be repeated to every other possible first step to turn the solution *S1* into solution **S2**. The other possible first steps are: **1** → **0**, **2** → **1** and **3** → **2**. By trying the three other possible first steps, a total of four possible intermediate solutions *P1'*, *P2'*, *P3'* and *P4'* will be available. Now, it is necessary to choose one of them to be the definitive intermediate solution *S1'*. This can be done by calculating the fitness value (PSNR) for each one of the four possible solutions and then choosing the one presenting the highest PSNR value as definitive intermediate solution *S1'*, so we can continue the process [16]. The next step is to identify the possibilities available into turning the intermediate solution *S1'* into *S2*. For this next step, the options to be tried are: **1** → **0**, **2** → **1** and **3** → **2**. By repeating the same process on the possible intermediate solutions the *S2* guiding solution is reached. If one of these intermediate solutions through path relinking process presents fitness value (i.e. its PSNR) higher than the best solution found until now, this new solution will be added to the best solutions list. The whole process ends when there are no more solutions on the list of best solutions to be combined. Interesting results also were found by applying the path relinking process in a reverse way (reverse path relinking). Some experiments applying path relinking starting from solution *S2* to the guiding solution *S1* were also tried, thus obtaining more intermediate solutions, and some times a better final result than only when applying path relinking from *S1* to *S2* [16].

IV. Experimental Results

Experimental results of the implemented schemes are presented in this section. The code was written in C++ language. All test images used are 8-bits grayscale [16]. The embedded (hidden) ones have 256×512 pixels, while the host images are always composed of 512×512 pixels. This is to ensure the use of four least significant bits, generate a large number of possible solutions and use half of the image size to store the hidden data. All test images are very common and were the same used in [7, 12, 16]. The embedded (hidden) images in the experiments were: 'Jet', 'Scene', 'Tiffany', 'Text' and 'Boat', respectively. Similarly, the host images used were: 'Lena', 'Baboon', 'Text', 'Peppers' and 'Barbara', respectively. Two works were selected for comparison of results [7, 12], each one showing a different strategy for image hiding. The comparisons were most similar as possible, utilizing images of the same size and aspect than the ones presented by those papers. The results presented by the method are average values of three executions for each instance of embedded image and host image. In [7], the main idea is to apply a genetic algorithm to speed up the search for a near optimal solution, instead using simple LSB substitution. This has many benefits, allowing the work with larger numbers of least significant bits ($k \geq 3$), otherwise it would take too long to find an optimal solution among all possibilities. In [12] the strategy is using dynamic programming to optimize the computations of the MSE values from all possible solutions and then find an optimal solution for LSB substitution. This work is also presented as a speed and quality optimization over [7], replacing the genetic algorithm. We also found some other papers with outstanding quality results, like [13], that use compression of information to hide by creation of a dictionary of terms. That strategy allows the hiding of much more data, even more than one entire image of the same size as the host image, but their results could not compared with ours, since [13] was hiding only full size images inside the host images. This could only be achieved using compression strategy. Other papers like [14, 15] were also revised and compared to our results but due to

space considerations it is not possible to show more comparative results. Table II shows our PSNR quality results and average time results, comparing them to the obtained and reported PSNR values for LSB and GEN with simple LSB substitution and the genetic algorithm approaches [7]. These authors were contacted and sent us updated correct data for these results. Table III presents the MSE quality results of [12] and compares with our results, where GEN [12] are the results shown by [12] with genetic algorithm and DYN [12] uses the optimal LSB substitution with dynamic programming strategy. GA+PR is our new proposal, an implementation of the genetic algorithm with the path relinking approach. After the normal path relinking process, a reverse path relinking also is applied, considering solution S1 as the guiding one and S2 as initial. During this process, all intermediate solutions found in the path relinking process are immediately evaluated by the fitness function (PSNR) and the ones presenting equal or better values than the best solution are added to the elite group so they can be combined with the others not yet chosen solutions from the elite group too. Each solution is selected for combining only once. GA+PR' is a modified version of GA+PR, using a simpler and faster path relinking approach, by changing always the first different part between solutions *S1* and *S2*, instead to testing all the possible changes to see which one is the best. GA+PR' is also less selective, since it accepts any intermediate solution found that is better than the worst solution present inside the population for the next generation (10 best individuals) until a maximum of 30 new individuals.

TABLE II. PSNR VALUES OF OUR RESULTS AND THOSE IN [7]

Images Secret → Host	Method (average PSNR result)			
	LSB[7]	GEN[7]	GA+PR	GA+PR'
Jet → Lena	32.04	32.71	32.79	32.71
Scene → Lena	32.10	32.55	32.58	32.57
Tiffany → Lena	31.21	32.90	32.90	32.87
Text → Lena	29.51	34.27	34.61	34.36
Jet → Baboon	32.11	32.79	32.89	32.83
Scene → Baboon	32.13	32.50	32.68	32.60
Tiffany → Baboon	31.31	32.95	33.02	33.01
Text → Baboon	29.60	34.38	34.83	34.76
Jet → Text	30.51	30.87	30.77	30.66
Scene → Text	29.07	30.35	30.49	30.42
Tiffany → Text	30.75	30.90	30.95	30.74
Text → Text	30.81	33.43	33.91	33.85
Average quality	**30.93**	**32.55**	**32.70**	**32.62**
Average Time in seconds			**577.2**	**81.76**

TABLE III - MSE VALUES OF OUR RESULTS AND THOSE IN [12]

Images Secret → Host	Method (MSE result)			
	GEN[12]	DYN[12]	GA+PR	GA+PR'
Jet → Lena	34.1974	33.3233	34.4237	34.8153
Tiffany → Lena	34.7052	33.2183	33.1082	33.5905
Boat → Lena	36.1665	34.8507	25.4192	28.0142
Jet → Peepers	34.7647	33.2655	32.9643	33.4315
Tiffany → Peepers	34.5167	33.0889	31.9725	32.5092
Boat → Peepers	36.5978	34.7991	24.4591	28.9842
Jet → Barb	34.6808	33.3242	34.0227	34.6535
Tiffany → Barb	35.1498	33.2026	32.8813	33.5136
Boat → Barb	36.3747	34.8720	25.0162	27.8681
Average MSE	**35.2393**	**33.7716**	**30.4741**	**32.6763**
Average Time in seconds			**409.891**	**75.821**

V. CONCLUSIONS

The use of path relinking as an improvement over the existing steganography method shows itself relatively expensive when we look at the additional time elapsed in execution. Nevertheless, in an image hiding process, the main objective to be achieved is the concealment of the information, making it "invisible" to intruders, so the quality gain was the focus of this work. The best configuration was obtained by executing the genetic algorithm with a total of eight generations, applying the path relinking refinement and the reversal path relinking at the end of each genetic algorithm generation over the best individuals chosen to form the next generation population. After the tests, we got an improvement over the results, so we can see the method shows itself effective when discovering better solutions in the middle of the best ones already existing, especially when dealing with situations producing a great deal of diverse solutions, making it more competitive when compared with others.

ACKNOWLEDGEMENT

The second author thanks the Spanish project TIN2008-06890-C02-02. The third author acknowledges Brazilian agency CAPES.

REFERENCES

[1] J.K. Jan and Y.M. Tseng, "On the security of image encryption method", Inform. Process Lett 60, 1996, pp. 261-265.

[2] N. Bourbakis and C. Alexopoulos, "Picture data encryption using scan patterns", Pattern Recognition 25, 1992, pp. 567-581.

[3] H.J. Highland, "Data encryption: a non-mathematical approach", Comput. Security 16, 1997, pp. 369-386.

[4] M. Rhee, Cryptography & Secure Communication, McGraw-Hill, 1994.

[5] W. Bender, D. Gruhl, N. Morimoto and A. Lu, "Techniques for data hiding", IBM Syst J 35, 1996.

[6] Z. Duric, M. Jacobs and S. Jajoolia, "Information hiding: Steganography and steganoanalysis", Handbook of Statistics 24, 2005, pp. 171-187.

[7] Ran-Zan Wang, Chi-Fang Lin and Ja-Chen Lin, "Image hiding by optimal LSB substitution and genetic algorithm", Pattern Recognition 34 (3), 2001, pp. 671-683.

[8] J. K. Hao, E. Lutton, E. Ronald , M. Schoenauer and D. Snyers, "A Template for Scatter Search and Path Relinking", LNCS 1363, Springer 1997, pp. 13-54.

[9] M. Laguna. "Scatter search and path Relinking: methodology and applications". URL: http://leeds-faculty.colorado.edu/laguna

[10] Y. Rochat and É. D. Taillard, "Probabilistic diversification and intensification in local search for vehicle routing", Journal of Heuristics 1, 1995, pp. 147 – 167.

[11] C. R. Reeves, "Genetic algorithms, path relinking and the flowshop sequencing problem", Evolutionary Computation J 6, 1998, pp. 230-234.

[12] Chin-Chen Chang, Ju-Yuan Hsiao and Chi-Shiang Chan, "Finding optimal least-significant-bit substitution in image hiding by dynamic programming strategy", Pattern Recognition 36, 2003, pp. 1583-1595.

[13] Yu-Chen Hu, "High-capacity image hiding scheme based on vector quantization", Pattern Recognition 39, 2006, pp. 1715-1724.

[14] S. Wang, "Steganography of capacity required using modulo operator for embedding secret image", Applied Mathematics and Computation, 164 (1), 2005, pp. 99-116.

[15] Chih-Ching Thien and Ja-Chen Lin, "A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function", Pattern Recognition 36 (12), 2003, pp. 2875-2881.

[16] A. Brazil, Path Relinking and Aes Cryptography in Color Image Steganography, M.Sc. Dissertation, Computer Institute, UFF, Brazil http://www.ic.uff.br/PosGraduacao/lista_dissertacao.php?ano=2008.