ISSN 1870-4069

Phase Unwrapping for 3D Object Reconstruction by means of Population-based Metaheuristics

Daniel Duarte-Carrera, Alfonso Rojas-Domínguez, Luis Carlos Padierna

Tecnológico Nacional de México - Instituto Tecnológico de León, León Gto, Mexico
{daducarme,alfonso.rojas,padiernacarlos}@gmail.com

Abstract. Metaheuristics are employed for the solution of the phase unwrapping problem (for 3D object reconstruction) by the *branch cuts* method, posed as an analogous of the traveling salesman problem, which is an NP-hard decision problem. The metaheuristic algorithms carry out a global search for the optimal configuration of the so-called branch cuts which corresponds to a pairing of discontinuities with opposed sign in the wrapped phase map. Three representative algorithms of different metaheuristic families are compared: discrete Particle Swarm Optimization (from bioinspired algorithms), Genetic Algorithms (from evolutionary algorithms) and a novel Estimation of Distribution Algorithm presented in this work that follows a Multinomial distribution. These metaheuristics are comparatively evaluated according to the quality of the solutions achieved, execution time and computational cost, with the aim of building a robust and automated algorithm competitive against traditional methods.

Keywords: Phase unwrapping, Optimization, Estimation of distribution.

1 Introduction

The phase of a signal is often defined within its principal values only, either in $(-\pi, \pi]$ or $(0, 2\pi]$, and it is called true or wrapped phase [1]. In practical applications such as 3D object reconstruction, it is necessary to obtain the phase as a continuous function through a process known as phase unwrapping, which is a technique used to remove the embedded discontinuities in wrapped phase maps [1][2]. The process must detect the 2π discontinuities in the phase and add or subtract 2π an integer number of times to compensate for each discontinuity in subsequent points [3]–[8].

Phase unwrapping algorithms in 2D are most typically divided into two categories: path following or branch cuts methods, and minimum norm methods [2]. The branch cuts method isolates those regions of a phase map that are affected by discontinuities. This is done by the use of barriers or branch cuts, that connect two discontinuity locations, thus achieving path independence [7]. Since its introduction in Goldstein's work in 1988 [9], the branch cuts method has been improved by the incorporation of artificial intelligence techniques (particularly soft computing) [4],[10]. In this work, the branch cuts problem is posed as a computational optimization problem and a comparative evaluation between three types of metaheuristic algorithms is carried out in order to determine their advantages in the solution of said problem. A discrete version of a very popular bio-inspired algorithm, known as Particle Swarm Optimization (d-PSO), is compared against an evolutionary algorithm (Genetic Algorithm) and against a novel

estimation of distribution algorithm: the Multinomial Estimation of Distribution Algorithm (MEDAL). These algorithms are conceptually very different, but whether or not their differences may represent an intrinsic advantage for any of them, is yet to be determined. The goal of this paper is to answer that question.

The rest of this paper is organized as follows: Section 2 provides the required material to understand the formulation of the branch cuts method as an optimization problem. Section 3 briefly presents the different algorithms that are compared. Our experimental methodology and results are reported in Section 4. Finally, Section 5 presents our conclusions and directions for future work.

2 Phase Unwrapping as Optimization Problem

Ghiglia and Pritt explain that there are relatively few inconsistencies along a closed path within a 2D wrapped phase map [2]. These inconsistencies are identified at points where: $\sum_i^M \Delta\psi(p_i) = \pm 2\pi$, where $\Delta\psi(p_i)$ represents the wrapped phase gradient at point $p_i \, \epsilon \, \{P\}$ and M is the total amount of points along the path P [2][11][12]. It follows that there are inconsistencies with positive polarity (2π) and with negative polarity (-2π). In 1988 Goldstein used the term *residue* to describe such inconsistencies and described a method where the charge (sign) of each residue must be balanced out by connecting pairs of residues with opposing polarities; this method is known as the branch cuts method [9]. In practice, the residues are computed as the sum of the gradients along a 2×2 path (counterclockwise) given in Eqns. (1) to (4). Whenever said sum gives a positive result, a positive residue exists at position (r, c); if the sum is negative, a negative residue is present; if the sum is zero then there is no residue:

$$\Delta\varphi(1) = \text{sign}\{\psi(r+1, c) - \psi(r, c)\}, \tag{1}$$

$$\Delta\varphi(2) = \text{sign}\{\psi(r+1, c+1) - \psi(r+1, c)\}, \tag{2}$$

$$\Delta\varphi(3) = \text{sign}\{\psi(r, c+1) - \psi(r+1, c+1)\}, \tag{3}$$

$$\Delta\varphi(4) = \text{sign}\{\psi(r, c) - \psi(r, c+1)\}. \tag{4}$$

Once the residues have been identified these are connected in pairs of opposing polarity, forming barriers called branch cuts [9]. Then the phase can be unwrapped along any path without touching these barriers. Many different branch cuts configurations can be formed, affecting the complexity of the phase unwrapping process differently. Thus, the phase unwrapping problem is converted into a problem of finding the pairing of residues that produce the optimal branch cuts configuration.

Two branch cuts configurations are shown in Fig. 1. As can be seen, the pairing of residues in Fig. 1b produced four branch cuts that will make phase unwrapping difficult; the barriers are long and badly arranged (crossing each other). In contrast, the pairing in Fig. 1c also produces four branch cuts but their configuration is much more favorable for the phase unwrapping; the barriers are shorter and better distributed. Notice that one of the residues in both configurations has been joined with the border of the phase map. This is acceptable since there is not always an equal amount of positive

and negative residues [7]. An efficient algorithm for this problem must find the pairing of residues that produces the branch cuts configuration with minimum total length. Then the data can be unwrapped by the flood fill algorithm [11].

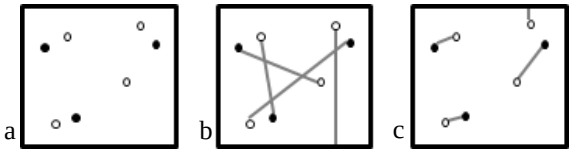

Fig. 1. Two different branch cuts configurations. a.- A set of residues. b.- Unfavorable pairing produces a bad configuration. c.- A different pairing produces an optimal configuration.

The optimization of the branch cuts problem is analogous to a combinatorial problem known as the Traveling Salesman Problem (TSP), which can be summarized as follows [4, 7]: a salesman must visit n cities by means of the shortest possible path; he must visit each city only once and return to the initial city in the end. When the number of cities increases, the TSP cannot be solved in polynomial time since its complexity grows exponentially (it becomes an NP-hard problem). The branch cuts problem is formulated as a TSP problem if the residues are the cities and the sum of the lengths of the branch cuts are the path that the salesman travels [8]. Metaheuristics are effective optimization methods that can be used to tackle this sort of problems.

3 Optimization Metaheuristics for Phase Unwrapping

The different metaheuristic techniques that are compared in this work all share a common codification of the candidate solutions. An individual solution consists in a pairing of residues of opposing polarities (or one residue and one border position). This can always be reorganized as a vector of positive residues and a vector of corresponding negative residues paired with the positive ones. Starting from one solution, new solutions can be generated by keeping the vector of positive residues fixed and changing the position of the negative residues [4]. This is illustrated in Fig. 2 with a small number of residues and wherein some border points, represented by 'B's are included.

All of the metaheuristics discussed herein maintain a so-called population of solutions that they employ to perform their search in the solution space. An initial solution can be found, for example, by application of a simple local search method known as the nearest neighbor method [7]. From this, a population of new solutions is generated automatically by randomly selecting and reordering of k elements in the vector of negative residues. Repeated application of this process creates an initial population.

In order to guide their search for a global optimum, the algorithms evaluate the quality of the individual solutions that are explored. Said quality is quantitatively captured by a so-called fitness function, f. In the case of the branch cuts problem the optimal solution is the branch cuts configuration of minimal total length. Thus, the fitness function is simply the sum of the lengths (distance between each pair of residues in a solution) of branch cuts ((x, y) is a residue location), as seen in Eq. (5):

$$f = \sum_i^N [(x_i^+ - x_i^-)^2 + (y_i^+ - y_i^-)^2]^{1/2}. \quad (5)$$

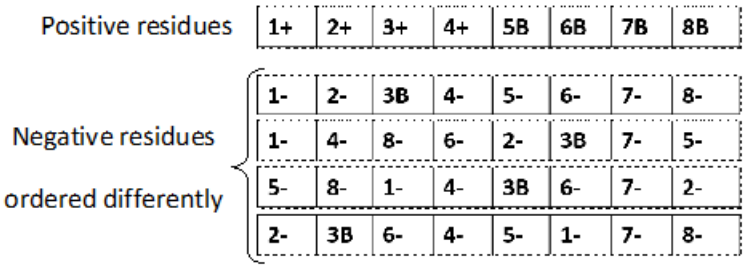

Fig. 2. Codification of solutions. The positive residues are kept in fixed positions; reordering of the negative residues (paired with the positive residues) produces other solutions.

3.1 Bioinspired Algorithm

The particle swarm optimization algorithm (PSO) was developed by Kennedy and Eberhart [12] as a population-based optimization method, inspired on the social behavior of bird flocks. Its objective is to generate increasingly better candidate solutions in an iterative way to reach the optimum [13]. Due to the small number of its parameters, its rapid convergence and its simple implementation, PSO shows better performance than some evolutionary algorithms [13]. PSO is based on the idea of D-dimensional particles moving inside a swarm [4]. The information that the j-th particle uses to move through the search space is the current value of its position U_j, its velocity V_j, its best past position P_j, and the best global position of the swarm P_g.

Table 1. Pseudocode of the PSO algorithm.

	PSO algorithm
1:	Initialize population of D-dimensional particles with random positions and velocities
2:	**Start the loop**
3:	Evaluate the quality of each particle (solution), according to the fitness function.
4:	Compare each solution with its P_j: if the current value is better, then update P_j and U_j with the current solution values.
5:	Identify the best solution and assign it to P_g.
6:	Update the velocity and the position of each particle: $V_j^{t+1} = (w \times V_j^t) + [C_1 \times Rand \times (P_j^t - U_j^t)] + [C_2 \times Rand \times (P_g^t - U_j^t)]$ $U_j^{t+1} = U_j^t + V_j^{t+1}$
7:	If the stopping criterion is met, stop the cycle.
8:	**End the loop**

The original process to implement PSO is described in Table 1 [14], where t denotes the iteration number, C_1 and C_2 are non-negative learning factors, the function $Rand$ generates a random number in (0,1) and w is called the inertia factor. A more detailed explanation of these variables is found in [14]. In this work, a discrete variant of the algorithm, termed d-PSO is used because it better fits the branch cuts problem. The modifications are detailed in [4]. In d-PSO the velocity represents a set of permutations

[4]; the permutations modify the position vector, rearranging its values. In this work the adjustment operator is followed in the same way as in [14].

3.2 Evolutionary Algorithm

The genetic algorithm is also a population-based technique inspired by the mechanisms of natural selection, genetics and evolution of living beings that has proved effective, quick and robust in many optimization problems [15]. Members of the population are called chromosomes [16], and they are formed by a set of D genes in D-dimensional space. Chromosomes perform their own adaptation strategies to evolve through the search space and reach the global optimum [7]. A new generation occurs when three operators that update the population are applied: selection, crossover and mutation [16]. The basic steps of the GA are described in Table 2.

Table 2. Pseudocode of the Genetic Algorithm.

	Genetic Algorithm
1:	Initialize chromosome population, probability of reproduction P_r and mutation P_m.
2:	**Start the loop**
3:	Evaluate the quality of each solution, according to the fitness function.
4:	Use a selection operator to choose two parent chromosomes.
5:	**If** P_r, Apply the crossover operator. **End if**
6:	**If** P_m, Apply the mutation operator. **End if**
7:	Accept the new solution if its quality is better.
8:	If the stopping criterion is met, stop the cycle.
9:	**End the loop**

The selection operator chooses a pair of chromosomes for crossover, allowing their genes to pass to the next generation [16]. The crossover operator combines some of the genes of each of the parent chromosomes; both chromosomes are split in the middle and the four parts are combined to form two offspring [7]. Subsequently, the quality of the new chromosomes is measured and only the best is passed on to the next generation [17]. The mutation operator forms a new chromosome through alterations of a chromosome [7]. The heuristic *twors* is used for this purpose [18].

3.3 Multinomial Estimation of Distribution Algorithm

An estimation of distribution algorithm (EDA) is a population-based optimization technique that tracks the statistics of a population of candidate solutions [19]. The search for the global optimum is carried out by creating new and better solutions through these statistics, *i.e.*, recreating the population iteratively and updating the statistics based on the best individuals in each generation [19]. This concept was taken as guidance to design a multinomial distribution EDA for discrete values, since this distribution models the probability of k categories in n trials, which fits well the branch

cuts problem. The steps followed by this Multinomial Estimation of Distribution Algorithm (MEDAL) are shown in Table 3.

Table 3. Pseudocode for the Multinomial Estimation of Distribution Algorithm.

	MEDAL algorithm
1:	Initialize a population of random candidate solutions $\{X_i\}^0, i \in [1, N]$.
2:	**Start the loop**
3:	Evaluate each solution according to the fitness function.
4:	Select the best M individuals from the population $\{X_i\}^t$, $(M < N)$.
5:	Estimate a multinomial distribution from the selected M individuals.
6:	Generate a new population $\{X_i\}^{t+1}$ sampling from the multinomial distribution.
7:	If the stopping criterion is met, stop the cycle.
8:	**End the loop**

During execution of the method, new populations are generated with new probabilities of occurrence per position. The objective is that all individuals converge to one, i.e., the probability of occurrence of a value in a position becomes equal to one (or as high as possible). When this happens, the algorithm ends. Due to the almost null existence of parameters to be tuned, the EDAs are considered agile and efficient algorithms, achieving convergence in a short time despite their susceptibility to get stuck in local optima.

4 Experimental Results and Discussion

The algorithms discussed above were implemented to solve the branch cuts problem and obtain an effective 3D reconstruction of test objects. Each of three test objects is a 512×512 pixels image generated through the MATLAB *peaks* function. For each of these, a wrapped phase map was obtained through the *arctangent* function. The residues in each map were identified by application of Eqs. (1)-(4) as explained in Section 2. Different residue sets were obtained for each map: 1511 residues in the first image (757 positives, 754 negatives); 995 residues in the second image (493 positives, 502 negatives); 1542 residues in the third image (772 positives, 770 negatives). These residues were given to each metaheuristic, together with an initial population of random solutions. This process was repeated 35 times per test image to provide statistical support to our conclusions. Identical initial populations were given to all algorithms in each trial to ensure a fair comparison. The initial populations included 500 solutions each, and the generations per trial were 1000. Tests were performed on an Intel Core-i7 2.40 GHz processor with 8GB of RAM. The test objects (wrapped phase maps), residues, and reconstructions (continuous phase) are shown in Fig. 3. Notice that at this resolution no difference is noticeable between different methods. Therefore, these images are representative of any of the three algorithms tested.

The average total length obtained by each metaheuristic is reported in Table 4, as well as the average elapsed time, function calls (until convergence or completion of the allocated function calls) and mean squared error (MSE) of the reconstructions. The

best results are shown in bold typeface. The stopping criterion was to observe a standard deviation $\sigma = 0.7$ computed over the top 20% of solutions in any generation.

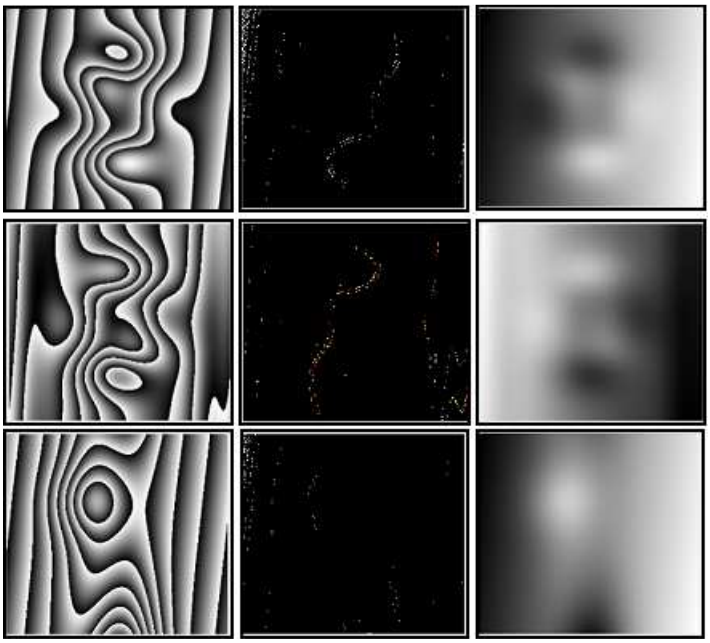

Fig. 3. Wrapped phase maps (left column, 512×512 pixels). Corresponding residues (center column, contrast exaggerated for clarity). Continuous phase obtained by the metaheuristics (right column, at this resolution no difference is noticeable between different methods).

Table 4. Comparison between d-PSO, GA and EDA. Average over 35 trials.

Algorithm	Time (m)	*Branch cuts* total length	Generations	MSE
First test object				
d-PSO	**1.77**	**2.57 E+03**	1000	**0.5028**
Genetic	2.85	2.58 E+03	423.31	0.8363
EDA	7.11	2.58 E+03	**157.11**	0.6117
Second test object				
d-PSO	**1.32**	**2.20 E+03**	1000	**0.3204**
Genetic	1.39	2.25 E+03	317.08	0.4573
EDA	2.50	2.28 E+03	**132.6**	0.5218
Third test object				
d-PSO	1.81	**2.02 E+03**	1000	**0.0665**
Genetic	**0.68**	2.14 E+03	130.08	0.1397
EDA	3.02	2.07 E+03	**89.62**	0.0434

In Table 4 it can be seen that the three different metaheuristics obtained similar results in terms of average total distance of branch cuts. The lowest average total length was obtained by d-PSO on the three test images. This algorithm also required the lowest execution time in two of the three test objects (first and second). However, the

stopping criterion was never reached by d-PSO, and it consumed the total of allocated generations in every case. In contrast, the GA converged at less than half of the total generations available, and the EDA, at less than a quarter. In other words, the EDA converges much more quickly, but each of its iterations requires more execution time. The behavior of the PSO algorithm is the opposite, and the GA is in the middle of these two extremes.

In order to confirm the statistical significance of the differences observed between the results of the algorithms, and provided that our experiments fit the conditions of a randomized complete block design, the Friedman test (a two-way analysis of variance on ranks) [20] was performed on the results from the 35 experimental trials. In this test, the null hypothesis (h_0) is that the difference in the performance of the algorithms is not statistically significant. Table 5 shows the results of the statistic, the corresponding p-values and the test conclusion at a significance level of $\propto = 0.05$.

Table 5. Friedman test over 35 trials per test object.

Test object	Statistic	P-value	Result ($\propto = 0.05$)
Fitness (total length of branch cuts)			
1	1.6	0.4496	Accept h_0
2	10.34	0.0057	Reject h_0
3	27.83	9.05E-07	Reject h_0
Execution time			
1	55.6	8.44E-13	Reject h_0
2	46.69	7.28E-11	Reject h_0
3	51.6	6.24E-12	Reject h_0
Generations (function calls)			
1	62.91	2.17E-14	Reject h_0
2	70	6.30E-16	Reject h_0
3	52.63	3.73E-12	Reject h_0

The Friedman test was applied on our 35 observations of fitness (total branch cuts length); execution time; and generations (function calls), for each of the three test images. As can be seen, the Friedman test rejects the null hypothesis in the majority of the cases. In other words, in the great majority of cases, there is statistical evidence to accept the observed differences between the algorithms reported in Table 4.

Combining the results in Table 4 with the statistical tests in Table 5, important conclusions can be formulated. First, regarding the average fitness measures (third column in Table 4), we consider that the differences are relatively small (and for one test object not statistically significant). Thus, it can be concluded that there is no clear advantage of any method over the rest with respect to the fitness produced; the algorithms are equally effective. In contrast, the differences in the average execution times (second column in Table 4) are quite substantial. As said before, the d-PSO algorithm is the fastest, the EDA the slowest, and the GA is somewhat in the middle. Finally, substantial and significant differences are also observed in the number of Generations required by the algorithms. Here d-PSO is the least efficient, the EDA is the most efficient and the GA is again in the middle. The three algorithms produced good quality solutions (small reconstruction MSE), but it is important to highlight that these measure is not

directly optimized by the algorithms, since their fitness function was formulated in terms of the total branch cuts length only.

5 Conclusions

The problem of branch cuts was formulated in terms of the combinatorial TSP, in order to obtain the benefits gained from years of research on the subject and the use of heuristic techniques to solve it. The success of this novel formulation was demonstrated through the application of three different types of metaheuristics for optimization. The compared techniques are representative of different metaheuristic families: bioinspired, genetic and estimation of distributions algorithms. The Multinomial Estimation of Distribution Algorithm (MEDAL) is a novel formulation of an EDA, created to work with discrete values.

These metaheuristics were applied to solve the branch cuts phase unwrapping problem and these were compared based on their efficiency. The algorithms were tested on three simulated images, demonstrating a fast and efficient reduction of the total branch cuts length and offering better unwrapping results than the initial solutions. Fewer pixels were used as barriers, and smooth continuous phase maps with minor deformities were obtained. All three techniques proved to be efficient, finding better and almost equivalent solutions. Therefore, we consider that they are equally effective in solving this problem.

The fastest algorithm was d-PSO, followed by GA and then MEDAL, with clear and significant differences. On the other hand, MEDAL required substantially fewer generations to converge; on average, it required almost 10 times fewer generations than d-PSO, which did not satisfy the stopping criterion in any of the tests. Thus, with respect to the efficiency of the algorithms, we are faced with contradicting evidence.

Nevertheless, considering that the execution time is dependent on computer characteristics, programming skills, etc., but the number of generations is an objective measure, we conclude in favor of the MEDAL as the most efficient. This conclusion is also influenced by the fact that the MEDAL has no control parameters to be adjusted and therefore it is the most practical for a user to employ.

In future work, the methods studied herein will be employed on real data, combining structured light techniques (to model the objects) with phase shifting to demodulate the phase. Also, the possibility of employing different local techniques to generate different initial solutions will be explored. Based on the results reported, the MEDAL metaheuristic will be considered, because it proved the most efficient on this problem. The experiments presented provide us with the knowledge to make this informed decision, which was not possible prior to the realization of this work. However, extensive testing on real data must be performed before we can conclusively recommend a particular method. The MEDAL metaheuristic will be tested in other applications such as discrete hyper-parameter tuning of classifiers.

Acknowledgements. This work was supported by the National Council of Science and Technology of Mexico (CONACYT) through scholarships 416913 (D. Duarte) and 375524 (L. C. Padierna), and Research Grant CATEDRAS-2598 (A. Rojas).

References

1. Ying, L.: Phase unwrapping. Wiley Encycl. Biomed. Eng., 24, 1–11 (2006)
2. Ghiglia, D.C., Pritt, M.D.: Two-Dimensional Phase Unwrapping: Theory, Algorithms, and Software. John Wiley & Sons, New Jersey, USA (1998)
3. De Souza, J.C., Oliveira, M.E., dos Santos, P.A.M.: Branch-cut algorithm for optical phase unwrapping. Opt. Lett., 40(15), 3456–3459 (2015)
4. He, W., Cheng, Y., Xia, L., Liu, F.: A new particle swarm optimization-based method for phase unwrapping of MRI data. Comput. Math. Methods Med. (2012)
5. Herszterg, I.H.: 2D-Phase Unwrapping via Minimum Spanning Forest with Balance Constraints. Master's Thesis, Pontificia Universidade Católica do Rio de Janeiro (2015)
6. Huang, Q., Zhou, H., Dong, S., Xu, S.: Parallel Branch-Cut Algorithm Based on Simulated Annealing for Large-Scale Phase Unwrapping. IEEE Trans. Geosci. Remote Sens. 53(7), 3833–3846 (2015)
7. Karout, S.A., Gdeisat, M.A., Burton, D.R., Lalor, M.J.: Two-dimensional phase unwrapping using a hybrid genetic algorithm. Appl. Opt. 46 (5), 730–743 (2007)
8. Karout, S.A., Gdeisat, M.A., Burton, D.R., Lalor, M.J.: Residue Vector, An Approach to Branch-Cut Placement in Phase Unwrapping: Theoretical Study. Appl. Opt. 46(21), 4712–27 (2007)
9. Goldstein, R.M., Zebker, H.A., Werner, C.L.: Satellite radar interferometry: Two-dimensional phase unwrapping. Radio Sci. 23(4) 713–720 (1988)
10. Wei, Z., Xu, F., Jin, Y.: Phase unwrapping for SAR interferometry based on an ant colony optimization algorithm. Int. J. Remote Sens. 293, 711–725(2008)
11. Chen, K., Xi, J., Yu, Y., Chicharo, J.F.: Fast Quality-guided flood-fill phase unwrapping algorithm for three-dimensional Fringe Pattern Profilometry. Optical Metrology and Inspection for Industrial Applications 7855, 1–9 (2010)
12. Kennedy J., Eberhart, R.: Particle Swarm Optimization. Swarm intelligence, 1(1), 33-57 (2007)
13. Shi, X. H., Liang, Y.C., Lee, H.P., Lu, C., Wang, Q.X.: Particle swarm optimization-based algorithms for TSP and generalized TSP. Inf. Process. Lett.103(5), 169–176 (2007)
14. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization An overview. Swarm Intell. 1(1), 33–57 (2007)
15. Holland, J.: Genetic Algorithms. Scientific American 267(1), 66–73 (1986)
16. Goldberg, D.E.: Genetic algorithms in search, optimization and machine learning. Addison Wesley, Reading, MA, USA (1989)
17. Pavez-Lazo B., Soto-Cartes, J.: A deterministic annular crossover genetic algorithm optimisation for the unit commitment problem. Expert Syst. Appl. 38(6), 6523–6529 (2011)
18. Abdoun, O., Abouchabaka, J., Tajani, C.: Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem. International Journal of Emerging Sciences 2(1), 61–77 (2012)
19. Simons D.: Evolutionary optimization algorithms. John Wiley & Sons, New Jersey, USA (2013)
20. Derrac J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. 1(1), 3–18 (2011)