Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 868434, 17 pages
http://dx.doi.org/10.1155/2013/868434

Research Article

Automatic Circle Detection on Images Based on an Evolutionary Algorithm That Reduces the Number of Function Evaluations

Erik Cuevas,[1] Eduardo L. Santuario,[1] Daniel Zaldívar,[1] and Marco Perez-Cisneros[2]

[1] *Departamento de Electrónica, Universidad de Guadalajara, CUCEI, Avenue Revolución 1500, CP 44430, Guadalajara, JAL, Mexico*
[2] *Departamento de Ingenierías, Universidad de Guadalajara, CUTONALA, Morelos 180, CP 45400, Tonalá, JAL, Mexico*

Correspondence should be addressed to Erik Cuevas; erik.cuevas@cucei.udg.mx

Received 15 July 2013; Accepted 2 September 2013

Academic Editor: Raul Rojas

This paper presents an algorithm for the automatic detection of circular shapes from complicated and noisy images with no consideration of the conventional Hough transform principles. The proposed algorithm is based on a newly developed evolutionary algorithm called the Adaptive Population with Reduced Evaluations (APRE). Our proposed algorithm reduces the number of function evaluations through the use of two mechanisms: (1) adapting dynamically the size of the population and (2) incorporating a fitness calculation strategy, which decides whether the calculation or estimation of the new generated individuals is feasible. As a result, the approach can substantially reduce the number of function evaluations, yet preserving the good search capabilities of an evolutionary approach. Experimental results over several synthetic and natural images, with a varying range of complexity, validate the efficiency of the proposed technique with regard to accuracy, speed, and robustness.

1. Introduction

The problem of detecting circular features holds paramount importance in several engineering applications such as automatic inspection of manufactured products and components, aided vectorization of drawings, and target detection [1, 2]. Circle detection in digital images has been commonly solved through the Circular Hough Transform (CHT) [3]. Unfortunately, this approach requires a large storage space that augments the computational complexity and yields a low processing speed. In order to overcome this problem, several approaches which modify the original CHT have been proposed. One well-known example is the Randomized Hough Transform (RHT) [4].

As an alternative to Hough Transform-based techniques, the problem of shape recognition has also been handled through evolutionary methods. In general, they have demonstrated to deliver better results than those based on the HT considering accuracy and robustness [5]. Evolutionary methods approach the detection task as an optimization problem whose solution involves the computational expensive evaluation of objective functions. Such fact strongly restricts their use in several image processing applications; despite this, EA methods have produced several robust circle detectors which use different evolutionary algorithms like Genetic algorithms (GA) [5], Harmony Search (HSA) [6], Electromagnetism-Like (EMO) [7], Differential Evolution (DE) [8], and Bacterial Foraging Optimization (BFOA) [9].

However, one particular difficulty in applying any EA to real-world problems, such as image processing, is its demand for a large number of fitness evaluations before reaching a satisfactory result. Fitness evaluations are not always straightforward as either an explicit fitness function does not exist or the fitness evaluation is computationally expensive.

The problem of excessively long fitness function calculations has already been faced in the field of evolutionary algorithms (EA) and is better known as evolution control or as fitness estimation [10]. For such an approach, the idea is to replace the costly objective function evaluation for some individuals by alternative models which are based on an approximate model of the fitness landscape. The individuals to be evaluated and those to be estimated are determined

following some fixed criteria which depend on the specific properties of the approximate model [11]. The models involved at the estimation can be dynamically built during the actual EA execution, since EA repeatedly sample the search space at different points [12]. There are several alternative models which have been used in combination with popular EAs. Some examples include polynomial functions [13], kriging schemas [14], multilayer perceptrons [15], and radial basis-function networks [16]. In the practice, the construction of successful models which can globally deal with the high dimensionality, ill distribution, and limited number of training samples is very difficult. Experimental studies [17] have demonstrated that if an alternative model is used for fitness evaluation, it is very likely that the evolutionary algorithm will converge to a false optimum. A false optimum is an optimum of the alternative model, which does not coincide with the optimum of the original fitness function. Under such conditions, the use of the alternative fitness models degrade the search effectiveness of the original EAs, producing frequently inaccurate solutions [18].

In an EA, the population size has a direct influence on the solution quality and its computational cost [19]. Traditionally, population size is set in advance to a prespecified value and remains fixed through the entire execution of the algorithm. If the population size is too small, then the EA may converge too quickly affecting severely the solution quality [20]. On the other hand, if it is too large, then the EA may present a prohibitive computational cost [19]. Therefore, an appropriate population size allows maintaining a trade-off between computational cost and effectiveness of the algorithm. In order to solve such a problem, several approaches have been proposed for dynamically adapting the population size. These methods are grouped into three categories [21]: (i) methods that increment or decrement the number of individuals according to a fixed function; (ii) methods in which the number of individuals is modified according to the performance of the average fitness value, and (iii) algorithms based on the population diversity.

Since most of the EAs have been primarily designed to completely evaluate all involved individuals, techniques for reducing the evaluation number are usually incorporated into the original EAs in order to estimate fitness values or to reduce the number of individuals being evaluated [22]. However, the use of alternative fitness models degrades the search effectiveness of the original EAs, producing frequently inaccurate solutions [23].

This paper presents an algorithm for the automatic detection of circular shapes from complicated and noisy images without considering the conventional Hough transform principles. The proposed algorithm is based on a newly developed evolutionary algorithm called the Adaptive Population with Reduced Evaluations (APRE). The proposed algorithm reduces the number of function evaluations through the use of two mechanisms: (1) adapting dynamically the size of the population and (2) incorporating a fitness calculation strategy which decides when it is feasible to calculate or only to estimate new generated individuals.

The APRE method begins with an initial population which is to be considered as a memory during the evolution process. To each memory element, a normalized fitness value, called quality factor is assigned to indicate the solution capacity that is provided by the element. Only a variable subset of memory elements is considered to be evolved. Like all EA-based methods, the proposed algorithm generates new individuals considering two operators: exploration and exploitation. Both operations are applied to improve the quality of the solutions by: (1) searching through the unexplored solution space to identify promising areas that contain better solutions than those found so far and (2) successive refinement of the best found solutions. Once the new individuals are generated, the memory is accordingly updated. At such stage, the new individuals compete against the memory elements to build the final memory configuration. In order to save computational time, the approach incorporates a fitness estimation strategy that decides which individuals can be estimated or actually evaluated. The proposed fitness calculation strategy estimates the fitness value of new individuals using memory elements located in neighboring positions which have been visited during the evolution process. In the strategy, new individuals, that are located near the memory element whose quality factor is high, have a great probability to be evaluated by using the true objective function. Similarly, evaluated those new particles lying in regions of the search space with no previous evaluations are also evaluated. The remaining search positions are only estimated by assigning the same fitness value that is the nearest location element on the memory. The use of such a fitness estimation method contributes to saving computational time, since the fitness value of only very few individuals is actually evaluated whereas the rest is just estimated.

Different to other approaches that use an already existent EA as framework, the APRE method has been completely designed to substantially reduce the computational cost, yet preserving good search effectiveness.

In order to detect circular shapes, the detector is implemented by encoding three pixels as candidate circles over the edge image. An objective function evaluates if such candidate circles are actually present in the edge image. Guided by the values of this objective function, the set of encoded candidate circles are evolved using the operators defined by APRE so that they can fit into the actual circles on the edge map of the image. Comparisons to several state-of-the-art evolutionary-based methods and the Randomized Hough Transform (RHT) approach on multiple images demonstrate a better performance of the proposed method in terms of accuracy, speed, and robustness.

The paper is organized as follows. In Section 2, the APRE algorithm and its characteristics are both described. Section 3 formulates the implementation of the circle detector. Section 4 shows the experimental results of applying our method to the recognition of circles in different image conditions. Finally, Section 5 discusses several conclusions.

2. The Adaptive Population with Reduced Evaluations (APRE) Algorithm

In the proposed algorithm, a population of candidate solutions to an optimization problem is evolved toward better

solutions. The algorithm begins with an initial population which will be used as a memory during the evolution process. To each memory element, it is assigned a normalized fitness value called quality factor that indicates the solution capacity provided by the element.

As a search strategy, the proposed algorithm implements two operations: "exploration" and "exploitation." Both necessary in all EAs [24]. Exploration is the operation of visiting entirely new points of a search space, whilst exploitation is the process of refining those points of a search space within the neighborhood of previously visited locations in order to improve their solution quality. Pure exploration degrades the precision of the evolutionary process but increases its capacity to find new potential solutions [25]. On the other hand, pure exploitation allows refining existent solutions but adversely drives the process to fall in local optimal solutions [26]. Therefore, the ability of an EA to find a global optimal solution depends on its capacity to find a good trade-off between the exploitation of so far found elements and the exploration of the search space.

The APRE algorithm is an iterative process in which several actions are executed. First, the number of memory elements to be evolved is computed. Such number is automatically modified at each iteration. Then, a set of new individuals is generated as a consequence of the execution of the exploration operation. For each new individual, its fitness value is estimated or evaluated according to a decision taken by a fitness estimation strategy. Afterwards, the memory is updated. In this stage, the new individuals produced by the exploration operation compete against the memory elements to build the final memory configuration. Finally, a sample of the best elements contained in the final memory configuration is undergone to the exploitation operation. Thus, the complete process can be divided in six phases: initialization, selecting the population to be evolved, exploration, fitness estimation strategy, memory updating, and exploitation.

2.1. Initialization. Like in EA, the APRE algorithm is an iterative method whose first step is to randomly initialize the population $\mathbf{M}(k)$ which will be used as a memory during the evolution process. The algorithm begins by initializing ($k = 0$) a set of N_p elements ($\mathbf{M}(k) = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_{N_p}\}$). Each element $\mathbf{m}_i$ is an n-dimensional vector containing the parameter values to be optimized. Such values are randomly and uniformly distributed between the prespecified lower initial parameter bound p_j^{low} and the upper initial parameter bound p_j^{high}, just as it described by the following expression:

$$m_{i,j} = p_j^{\text{low}} + \text{rand}(0,1) \cdot \left(p_j^{\text{high}} - p_j^{\text{low}}\right), \quad \text{for } j = 1, 2, \ldots, n, \; i = 1, 2, \ldots, N_p, \tag{1}$$

where j and i are the parameter and element indexes, respectively. Hence, $m_{i,j}$ is the jth parameter of the ith element.

Each element $\mathbf{m}_i$ has two associated characteristics: a fitness value $J(\mathbf{m}_i)$ and a quality factor $Q(\mathbf{m}_i)$. The fitness value $J(\mathbf{m}_i)$ assigned to each element $\mathbf{m}_i$ can be calculated by using the true objective function $f(\mathbf{m}_i)$ or only estimated by using the proposed fitness strategy $F(\mathbf{m}_i)$. In addition to the fitness value, it is also assigned to $\mathbf{m}_i$, a normalized fitness value called quality factor $Q(\mathbf{m}_i)$ ($Q(\cdot) \in [0,1]$), which is computed as follows:

$$Q(\mathbf{m}_i) = \frac{J(\mathbf{m}_i) - \text{worst}_{\mathbf{M}}}{\text{best}_{\mathbf{M}} - \text{worst}_{\mathbf{M}}}, \tag{2}$$

where $J(\mathbf{m}_i)$ is the fitness value obtained by evaluation $f(\cdot)$ or by estimation $F(\cdot)$ of the memory element $\mathbf{m}_i$. The values $\text{worst}_{\mathbf{M}}$ and $\text{best}_{\mathbf{M}}$ are defined as follows (considering a maximization problem):

$$\text{best}_{\mathbf{M}} = \max_{k \in \{1,2,\ldots,N_p\}} \left(J(\mathbf{m}_k)\right), \qquad \text{worst}_{\mathbf{M}} = \min_{k \in \{1,2,\ldots,N_p\}} \left(J(\mathbf{m}_k)\right). \tag{3}$$

Since the mechanism by which an EA accumulates information regarding the objective function is an exact evaluation of the quality of each potential solution, initially, all the elements of $\mathbf{M}(k)$ are evaluated without considering the fitness estimation strategy proposed in this paper. This fact is only allowed at this initial stage.

2.2. Selecting the Population to Be Evolved. At each k iteration, it must be selected which and how many elements from $\mathbf{M}(k)$ will be considered to build the population $\mathbf{P}^k$ in order to be evolved. Such selected elements will be undergone by the exploration and exploitation operators in order to generate a set of new individuals. Therefore, two things need to be defined: the number of elements N_e^k to be selected and the strategy of selection.

2.2.1. The Number of Elements N_e^k to Be Selected. One of the mechanisms used by the APRE algorithm for reducing the number of function evaluations is to modify dynamically the size of the population to be evolved. The idea is to operate with the minimal number of individuals that guarantee the correct efficiency of the algorithm. Hence, the method aims to vary the population size in an adaptive way during the execution of each iteration. At the beginning of the process, a predetermined number N_e^0 of elements are considered to build the first population; then, it will be incremented or decremented depending on the algorithm's performance. The adaptation mechanism is based on the lifetime of the individuals and on their solution quality.

In order to compute the lifetime of each individual, it is assigned a counter c_i ($i \in (1, 2, 3, \ldots, N_p)$) to each element $\mathbf{m}_i$ of $\mathbf{M}(k)$. When the initial population $\mathbf{M}(k)$ is created, all the counters are set to zero. Since the memory $\mathbf{M}(k)$ is updated at each generation, some elements prevail and others will be substituted by new individuals. Therefore, the counter of the surviving elements is incremented by one whereas the counter of new added elements is set to zero.

Another important requirement to calculate the number of elements to be evolved is the solution quality provided by each individual. The idea is to identify two classes of elements, those that provide good solutions and those that can be

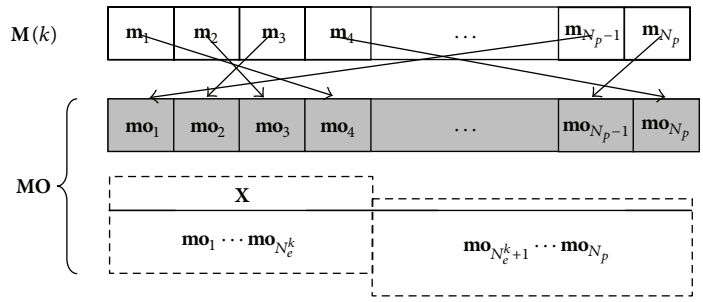

Figure 1: Necessary postprocessing implemented by the selection strategy.

(1) **Input:** Current population $\mathbf{M}(k)$, counters $c_1, \ldots, c_{N_p}$, the past number of individuals N_e^{k-1} and the constant factor s.
(2) $J_A \leftarrow (1/N_p) \sum_{i=1}^{N_p} J(\mathbf{m}_i)$
(3) $\mathbf{G} \leftarrow$ FindIndividualsOverJA($\mathbf{M}(k), J_A$)
(4) $\mathbf{B} \leftarrow$ FindIndividualsUnderJA($\mathbf{M}(k), J_A$)
(5) $c_l \leftarrow$ FindCountersOfG($\mathbf{G}$) (Where $l \in \mathbf{G}$)
(6) $c_q \leftarrow$ FindCountersOfB($\mathbf{B}$) (Where $q \in \mathbf{B}$)
(7) $A \leftarrow \text{floor}\left(\left(|\mathbf{G}| \cdot \sum_{l \in \mathbf{G}} c_l - |\mathbf{B}| \cdot \sum_{q \in \mathbf{B}} c_q\right)/s\right)$
(8) $N_e^k \leftarrow N_e^{k-1} + A$
(9) **Output:** The number N_e^k

Algorithm 1: Selection of the number of individuals N_e^k to be evolved.

considered as bad solutions. In order to classify each element, the average fitness value J_A produced by all the elements of $\mathbf{M}(k)$ is calculated as

$$J_A = \frac{1}{N_p} \sum_{i=1}^{N_p} J(\mathbf{m}_i), \tag{4}$$

where $J(\cdot)$ represents the fitness value corresponding to $\mathbf{m}_i$. These values are evaluated either by the true objective function $f(\mathbf{m}_i)$ or by the fitness estimation strategy $F(\mathbf{m}_i)$. Considering the average fitness value, two groups are built: the set $\mathbf{G}$ constituted by the elements of $\mathbf{M}(k)$ whose fitness values are greater than J_A and the set $\mathbf{B}$ which groups the elements of $\mathbf{M}(k)$ whose fitness values are equal or lower than J_A.

Therefore, the number of individuals of the current population that will be incremented or decremented at each generation is calculated by the following model:

$$A = \text{floor}\left(\frac{|\mathbf{G}| \cdot \sum_{l \in \mathbf{G}} c_l - |\mathbf{B}| \cdot \sum_{q \in \mathbf{B}} c_q}{s}\right), \tag{5}$$

where the floor $(\cdot)$ function maps a real number to the previous integer. $|\mathbf{G}|$ and $|\mathbf{B}|$ represent the number of elements of $\mathbf{G}$ and $\mathbf{B}$, respectively, whereas $\sum_{l \in \mathbf{G}} c_l$ and $\sum_{q \in \mathbf{B}} c_q$ indicate the sum of the counters that correspond to the elements of $\mathbf{G}$ and $\mathbf{B}$, respectively. The factor s is a term used for fine tuning. A small value of s implies a better algorithm's performance at the price of an increment in the computational cost. On the other hand, a big value of s involves a low computational cost at the price of a decrement in the performance algorithm. Therefore, the s value must reflex a compromise between performance and computational cost. In our experiments such compromise has been found with $s = 10$.

Therefore, the number the elements that define the population to be evolved is computed according to the following model:

$$N_e^k = N_e^{k-1} + A. \tag{6}$$

Since the value of A can be positive or negative, the size of the population $\mathbf{P}^k$ may be higher or lesser than $\mathbf{P}^{k-1}$. The computational procedure that implements this method is presented in Algorithm 1, in form of pseudocode.

2.2.2. Selection Strategy for Building $\mathbf{P}^k$. Once the number of individuals has been defined, the next step is the selection of N_e^k elements from $\mathbf{M}(k)$ for building $\mathbf{P}^k$. A new population $\mathbf{MO}$ which contains the same elements that $\mathbf{M}(k)$ is generated but sorted according to their fitness values. Thus, $\mathbf{MO}$ presents in its first positions the elements whose fitness values are better than those located in the last positions. Then, $\mathbf{MO}$ is divided in two parts: $\mathbf{X}$ and $\mathbf{Y}$. The section $\mathbf{X}$ corresponds to the first N_e^k elements of $\mathbf{MO}$ whereas the rest of the elements constitute the part $\mathbf{Y}$. Figure 1 shows this process.

In order to promote diversity, in the selection strategy, the 80% of the N_e^k individuals of $\mathbf{P}^k$ are taken from the first elements of $\mathbf{X}$ and named as Fe as shown in Figure 2, where Fe = floor $(0.8 * N_e^k)$. The remaining 20% of the individuals are randomly selected from section $\mathbf{Y}$. Hence, the last set of Se elements (where Se $= N_e^k -$ Fe) is chosen considering that

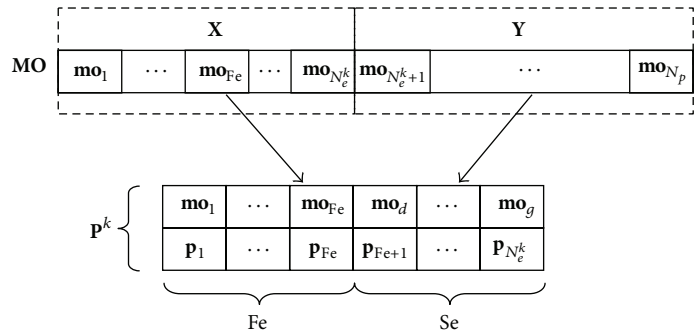

Figure 2: Employed selection strategy to build the population $\mathbf{P}^k$, where $d, g \in \mathbf{Y}$.

(1) **Input:** Current population $\mathbf{M}(k)$ and the number of individuals N_e^k.
(2) $\mathbf{MO} \leftarrow \mathrm{SortElementsFitness}(\mathbf{M}(k))$
(3) $[\mathbf{X}, \mathbf{Y}] \leftarrow \mathrm{DivideMO}\left(\mathbf{MO}, N_e^k\right)$
(4) $\mathrm{Fe} \leftarrow \mathrm{floor}\left(0.8 * N_e^k\right)$
(5) $\mathrm{Se} \leftarrow N_e^k - \mathrm{Fe}$
(6) $\left(\mathbf{p}_1^k, \ldots, \mathbf{p}_{\mathrm{Fe}}^k\right) \leftarrow \mathrm{SelectElementsOfX}(\mathbf{X}, \mathrm{Fe})$
(7) $\left(\mathbf{p}_{\mathrm{Fe}+1}^k, \ldots, \mathbf{p}_{\mathrm{Fe}+\mathrm{Se}}^k\right) \leftarrow \mathrm{SelectRandomElementsOfY}(\mathbf{Y}, \mathrm{Se})$
(8) **Output:** Population $\mathbf{P}^k$ to be evolve

Algorithm 2: Selection strategy for building $\mathbf{P}^k$.

all elements of $\mathbf{Y}$ have the same possibility of being selected. Figure 2 shows a description of the selection strategy. The computational procedure that implements this method is presented in Algorithm 2, in form of pseudocode.

2.3. Exploration Operation. The first main operation applied to the population $\mathbf{P}^k$ is the exploration operation. Considering $\mathbf{P}^k$ as the input population, APRE mutates $\mathbf{P}^k$ to produce a temporal population $\mathbf{T}^k$ of N_e^k vectors. In the exploration operation two different mutation models are used: the mutation employed by the Differential Evolution algorithm (DE) [27] and the trigonometric mutation operator [28].

2.3.1. DE Mutation Operator. In this mutation, three distinct individuals $r1$, $r2$, and $r3$ are randomly selected from the current population $\mathbf{P}^k$. Then, a new value $h_{i,j}$ considering the following model is created:

$$h_{i,j} = p_{r1,j} + F\left(p_{r2,j} - p_{r3,j}\right), \tag{7}$$

where $r1$, $r2$, and $r3$ are randomly selected individuals such that they satisfy $r1 \neq r2 \neq r3 \neq i, i = 1$ to N_e^k (population size), and $j = 1$ to n (number of decision variable). Hence, $p_{i,j}$ is the jth parameter of the ith individual of $\mathbf{P}^k$. The scale factor, F(0,1+), is a positive real number that controls the rate at which the population evolves.

2.3.2. Trigonometric Mutation Operator. The trigonometric mutation operation is performed according to the following formulation:

$$\begin{aligned}
h_{i,j} &= p_{\mathrm{Av}}(j) + F_1\left(p_{r1,j} - p_{r2,j}\right) \\
&\quad + F_2\left(p_{r2,j} - p_{r3,j}\right) + F_3\left(p_{r3,j} - p_{r1,j}\right), \\
p_{\mathrm{Av}}(j) &= \frac{p_{r1,j} + p_{r2,j} + p_{r3,j}}{3}, \\
F_1 &= \left(d_{r2} - d_{r1}\right), \qquad F_2 = \left(d_{r3} - d_{r2}\right), \\
F_3 &= \left(d_{r1} - d_{r3}\right), \\
d_{r1} &= \frac{\left|J\left(\mathbf{p}_{r1}\right)\right|}{d_T}, \qquad d_{r2} = \frac{\left|J\left(\mathbf{p}_{r2}\right)\right|}{d_T}, \qquad d_{r3} = \frac{\left|J\left(\mathbf{p}_{r3}\right)\right|}{d_T}, \\
d_T &= \left|J\left(\mathbf{p}_{r1}\right)\right| + \left|J\left(\mathbf{p}_{r2}\right)\right| + \left|J\left(\mathbf{p}_{r3}\right)\right|,
\end{aligned} \tag{8}$$

where $\mathbf{p}_{r1}$, $\mathbf{p}_{r2}$, and $\mathbf{p}_{r3}$ represent the individuals $r1$, $r2$, and $r3$ randomly selected from the current population $\mathbf{P}^k$ whereas $J(\cdot)$ represents the fitness value (calculated or estimated) corresponding to $\mathbf{p}_i$. Under this formulation, the individual $p_{\mathrm{Av}}(j)$ to be perturbed is the average value of three randomly selected vectors ($r1$, $r2$, and $r3$). The perturbation to be imposed over such individual is implemented by the sum of three weighted vector differentials. F_1, F_2, and F_3 are the weights applied to these vector differentials. Notice that the trigonometric mutation is a greedy operator since it biases the

```
(1)  Input: Current population P^k
(2)  for i = 1 to N_e^k do
(3)  (p_r1, p_r2, p_r3) ← SelectElements()      % Considering that r1 ≠ r2 ≠ r3 ≠ i
(4)     for j = 1 to n do
(5)        if (rand (0, 1) <= MR) then
(6)           h_{i,j} ← DEMutation(p_r1, p_r2, p_r3) % (7)
(7)        else
(8)           h_{i,j} ← TrigonometricMutation(p_r1, p_r2, p_r3) % (8)
(9)        end if
(10)       if (rand(0, 1) <= CH) then
(11)          t_{i,j} ← h_{i,j}
(12)       else
(13)          t_{i,j} ← p_{i,j}
(14)       end if
(15)    end for
(16) end for
(17) Output: Population T^k
```

Algorithm 3: Exploration operation of APRE algorithm.

$p_{\mathrm{Av}}(j)$ strongly in the direction where the best one of three individuals is lying.

Computational Procedure. Considering $\mathbf{P}^k$ as the input population, all its N_e^k individuals are sequentially processed in cycles beginning by the first individual $\mathbf{p}_1$. Therefore, in the cycle i (where it is processed the individual i), three distinct individuals $r1$, $r2$, and $r3$ are randomly selected from the current population considering that they satisfy the following conditions $r1 \neq r2 \neq r3 \neq i$. Then, it is processed each dimension of $\mathbf{p}_i$ beginning by the first parameter 1 until the last dimension n has been reached. At each processing cycle, the parameter $p_{i,j}$ considered as a parent, creates an offspring $t_{i,j}$ in two steps. In the first step, from the selected individuals $r1$, $r2$, and $r3$, a donor vector $h_{i,j}$ is created by means of two different mutation models. In order to select which mutation model is applied, a uniform random number is generated within the range $[0, 1]$. If such number is less than a threshold MR, the donor vector $h_{i,j}$ is generated by the DE mutation operator; otherwise, it is produced by the trigonometric mutation operator. Such process can be modeled as follows:

$$h_{i,j} = \begin{cases} \text{By using (7)} & \text{with probability MR} \\ \text{By using (8)} & \text{with probability } (1 - \mathrm{MR}). \end{cases} \tag{9}$$

In the second step, the final value of the offspring $t_{i,j}$ is determined. Such decision is stochastic; hence, a second uniform random number is generated within the range $[0, 1]$. If this random number is less than CH, $t_{i,j} = h_{i,j}$; otherwise, $t_{i,j} = p_{i,j}$. This operation can be formulated as follows:

$$t_{i,j} = \begin{cases} h_{i,j} & \text{with probability CH} \\ p_{i,j} & \text{with probability } (1 - \mathrm{CH}). \end{cases} \tag{10}$$

The complete computational procedure is presented in Algorithm 3, in form of pseudocode.

2.4. Fitness Estimation Strategy. Once the population $\mathbf{T}^k$ has been generated by the exploration operation, it is necessary to calculate the fitness value provided by each individual. In order to reduce the number of function evaluations, a fitness estimation strategy that decides which individuals can be estimated or actually evaluated is introduced. The idea of such a strategy is to find the global optimum of a given function considering only very few number of function evaluations.

In this paper, we explore a local approximation scheme that estimates the fitness values based on previously evaluated neighboring individuals, stored in the memory $\mathbf{M}(k)$ during the evolution process. The strategy decides if an individual $\mathbf{t}_i$ is calculated or estimated based on two criteria. The first one considers the distance between $\mathbf{t}_i$ and the nearest element $\mathbf{m}^{\mathrm{ne}}$ contained in $\mathbf{M}(k)$ (where $\mathbf{m}^n \in (\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_{N_p})$) whereas the second one examines the quality factor provided by the nearest element $\mathbf{m}^{\mathrm{ne}}(Q(\mathbf{m}^{\mathrm{ne}}))$.

In the model, individuals of $\mathbf{T}^k$ that are near the elements of $\mathbf{M}(k)$ holding the best quality values have a high probability to be evaluated. Such individuals are important, since they will have a stronger influence on the evolution process than other individuals. In contrast, individuals of $\mathbf{T}^k$ that are also near the elements of $\mathbf{M}(k)$ but with a bad quality value maintain a very low probability to be evaluated. Thus, most of such individuals will only be estimated, assigning it the same fitness value that the nearest element of $\mathbf{M}(k)$. On the other hand, those individuals in regions of the search space with few previous evaluations (individuals of $\mathbf{T}^k$ located farther than a distance D) are also evaluated. The fitness values of these individuals are uncertain since there is no close reference (close points contained in $\mathbf{M}(k)$).

Therefore, the fitness estimation strategy follows two rules in order to evaluate or estimate the fitness values.

(1) If the new individual $\mathbf{t}_i$ is located closer than a distance D with respect to the nearest element $\mathbf{m}^{\mathrm{ne}}$ stored in $\mathbf{M}$, then a uniform random number is generated

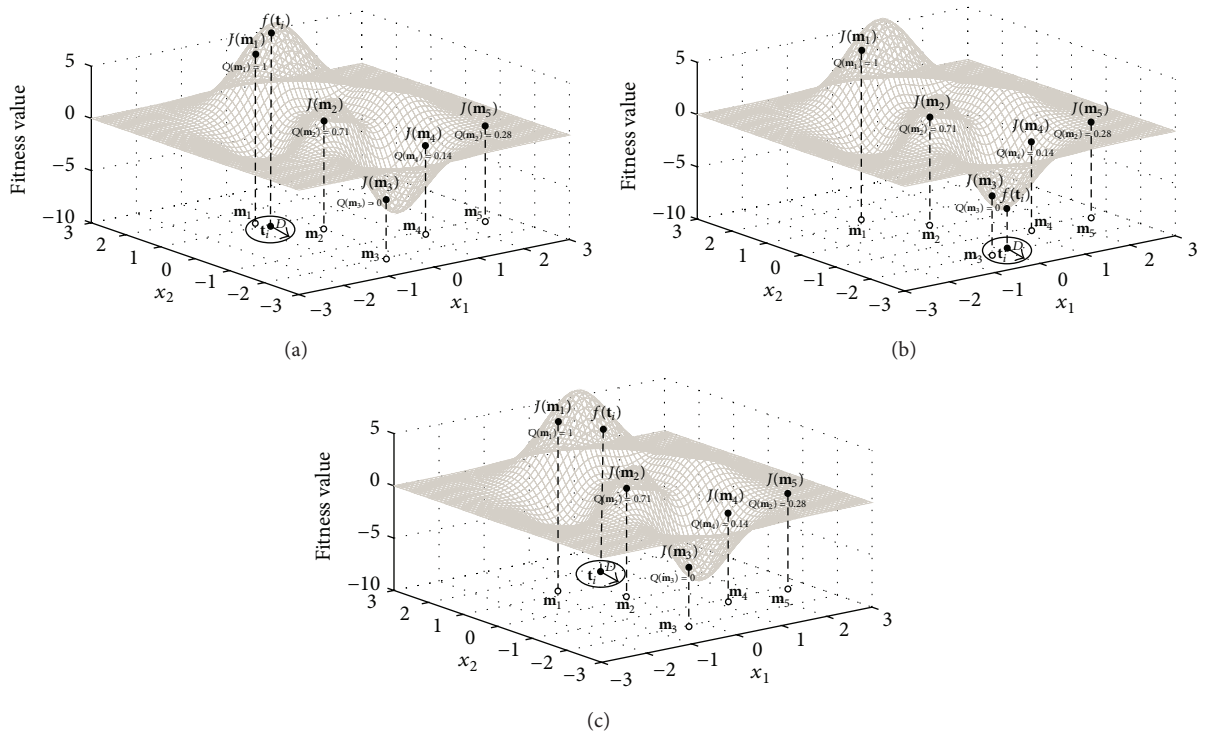

Figure 3: The fitness estimation strategy. (a) According to the rule 1, the individual $\mathbf{t}_i$ has a high probability to be evaluated $f(\mathbf{t}_1)$, since it is located closer than a distance D with respect to the nearest element $\mathbf{m}^{\text{ne}} = \mathbf{m}_1$ whose quality factor $Q(\mathbf{m}_1)$ corresponds to the best value. (b) According to the rule 1, the individual $\mathbf{t}_i$ has a high probability to be estimated $F(\mathbf{t}_1)$ (assigning it the same fitness value as $\mathbf{m}^{\text{ne}}$ ($F(\mathbf{t}_i) = J(\mathbf{m}_3)$)), since it is located closer than a distance D with respect to the nearest element $\mathbf{m}^{\text{ne}} = \mathbf{m}_3$ whose quality factor $Q(\mathbf{m}_3)$ corresponds to the worst value. (c) According to the rule 2, the individual $\mathbf{t}_i$ is evaluated, as there is no close reference in its neighborhood.

within the range $[0, 1]$. If such number is less than $Q(\mathbf{m}^{\text{ne}})$, $\mathbf{t}_i$ is evaluated by the true objective function ($f(\mathbf{t}_i)$). Otherwise, its fitness value is estimated assigning it the same fitness value that $\mathbf{m}^{\text{ne}}$ ($F(\mathbf{t}_i) = J(\mathbf{m}^{\text{ne}})$). Figures 3(a) and 3(b) draw the rule procedure.

(2) If the new individual $\mathbf{t}_i$ is located longer than a distance D with respect to the nearest individual location $\mathbf{m}^{\text{ne}}$ stored in $\mathbf{M}$, then the fitness value of $\mathbf{t}_i$ is evaluated using the true objective function ($f(\mathbf{t}_i)$). Figure 3(c) outlines the rule procedure.

From the rules, the distance D controls the trade off between the evaluation and estimation of new individuals. Unsuitable values of D result in a lower convergence rate, longer computation time, larger function evaluation number, convergence to a local maximum, or unreliability of solutions. Therefore, the D value is computed considering the following equation:

$$D = \frac{\sum_{j=1}^{n} \left(p_j^{\text{high}} - p_j^{\text{low}}\right)}{50 \cdot n}, \tag{11}$$

where p_j^{low} and p_j^{high} represent the prespecified lower bound and the upper bound of the j-parameter, respectively, within an n-dimensional space. Both rules show that the fitness estimation strategy is simple and straightforward. Figure 3 illustrates the procedure of fitness computation for a new candidate solution $\mathbf{t}_i$ considering the two different rules. In the problem the objective function f is maximized with respect to two parameters (x_1, x_2). In all figures (Figures 3(a), 3(b), and 3(c)) the memory $\mathbf{M}(k)$ contains five different elements ($\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3, \mathbf{m}_4$, and $\mathbf{m}_5$) with their corresponding fitness values ($J(\mathbf{m}_1), J(\mathbf{m}_2), J(\mathbf{m}_3), J(\mathbf{m}_4)$, and $J(\mathbf{m}_5)$) and quality factors ($Q(\mathbf{m}_1), Q(\mathbf{m}_2), Q(\mathbf{m}_3), Q(\mathbf{m}_4)$, and $Q(\mathbf{m}_5)$). Figures 3(a) and 3(b) show the fitness evaluation ($f(x_1, x_2)$) or estimation ($F(x_1, x_2)$) of the new individual $\mathbf{t}_i$ following the rule 1. Figure 3(a) represents the case when $\mathbf{m}^{\text{ne}}$ holds a good quality factor whereas Figure 3(b) when $\mathbf{m}^{\text{ne}}$ maintains a bad quality factor. Finally, Figure 3(c) presents the fitness evaluation of $\mathbf{t}_i$ considering the conditions of rule 2. The procedure that implements the fitness estimation strategy is presented in Algorithm 4, in form of pseudocode.

2.5. Memory Updating. Once the operations of exploration and fitness estimation have been applied, it is necessary to update the memory $\mathbf{M}(k)$. In the APRE algorithm, the memory $\mathbf{M}(k)$ is updated considering the following procedure.

(1) The elements of $\mathbf{M}(k)$ and $\mathbf{T}^k$ are merged into $\mathbf{M}_U$ ($\mathbf{M}_U = \mathbf{M}(k) \cup \mathbf{T}^k$).

(2) From the resulting elements of $\mathbf{M}_U$, it is selected the N_p best elements according to their fitness values to build the new memory $\mathbf{M}(k + 1)$.

```
(1)  Input: Population T^k and memory M(k)
(2)  for i = 1 to N_e^k do
(3)  m^ne ← FindNearestElementOfM(t_i)
(4)  distance ← FindTheDistance(t_i, m^ne)
(5)      if (distance < D) then
(6)          if (rand(0, 1) <= Q(m^ne)) then
(7)              J(t_i) ← f(t_i)   % Evaluation
(8)          else                                  (Rule 1)
(9)              J(t_i) ← J(m^ne)   % Estimation
(10)         end if
(11)     else
(12)         J(t_i) ← f(t_i)   % Evaluation        (Rule 2)
(13)     end if
(14) end for
(15) Output: fitness values of T^k
```

Algorithm 4: Fitness estimation strategy.

(3) The counters $c_1, c_2, \ldots, c_{N_p}$ must be updated. Thus, the counter of the surviving elements is incremented by 1 whereas the counter of modified elements is set to zero.

2.6. Exploitation Operation. The second main operation applied by the APRE algorithm is the exploitation operation. Exploitation, in the context of EA, is the process of refining the solution quality of existent promising solutions within a small neighborhood. In order to implement such a process, a new memory **ME** is generated, which contains the same elements that $\mathbf{M}(k+1)$ but sorted according to their fitness values. Thus, **ME** presents in its first positions the elements whose fitness values are better than those located in the last positions. Then, the 10% of the N_p (N_e) individuals are taken from the first elements of **ME** to build the set **E** ($\mathbf{E} = \{\mathbf{me}_1, \mathbf{me}_2, \ldots, \mathbf{me}_{N_e}\}$, where $N_e = \text{ceil}\ (0.1 \cdot N_p)$).

To each element $\mathbf{me}_i$ of **E** a probability p_i which express the likelihood of the element $\mathbf{me}_i$ to be exploited is assigned. Such a probability is computed as follows:

$$p_i = \frac{N_e + 1 - i}{N_e}. \tag{12}$$

Therefore, the first elements of **E** have a better probability to be exploited than the last ones. In order to decide if the element $\mathbf{me}_i$ must be exploited, a uniform random number is generated within the range $[0, 1]$. If such a number is less than p_i, then the element $\mathbf{me}_i$ will be modified by the exploitation operation. Otherwise, it remains without changes.

If the exploitation operation over $\mathbf{me}_i$ is verified, the position of $\mathbf{me}_i$ is perturbed considering a small neighborhood. The idea is to test if it is possible to refine the solution provided by $\mathbf{me}_i$ modifying slightly its position. In order to improve the exploitation process, the proposed algorithm starts perturbing the original position within the interval $[-D, D]$ (where D is the distance defined in (11)) and then gradually is reduced as the process evolves. Thus, the perturbation over a generic element $\mathbf{me}_i$ is modeled as follows:

$$me_{i,j}^{\text{new}} = me_{i,j} + \left[D \frac{ng - k}{ng} \right] (2 \cdot \text{rand}\,(0, 1) - 1), \tag{13}$$

where k is the current iteration and ng is the total number of iterations from which consists the evolution process. Once $\mathbf{me}_i^{\text{new}}$ has been calculated, its fitness value is computed by using the true objective function ($J(\mathbf{me}_i^{\text{new}}) = f(\mathbf{me}_i^{\text{new}})$). If $\mathbf{me}_i^{\text{new}}$ is better than $\mathbf{me}_i$ according to their fitness values, the value of $\mathbf{me}_i$ in the original memory $\mathbf{M}(k+1)$ is updated with $\mathbf{me}_i^{\text{new}}$; otherwise the memory $\mathbf{M}(k+1)$ remains without changes. The procedure that implements the exploitation operation is presented in Algorithm 5, in form of pseudocode.

In order to demonstrate the exploitation operation, Figure 4(a) illustrates a simple example. A memory $\mathbf{M}(k+1)$ of ten different 2-dimensional elements is assumed ($N_p = 10$). Figure 4(b) shows the previous configuration of the proposed example before the exploitation operation takes place. Since only the 10% of the best elements of $\mathbf{M}(k+1)$ will build the set **E**, $\mathbf{m}_5$ is the single element that constitutes **E** ($\mathbf{me}_1 = \mathbf{m}_5$). Therefore, according to (12), the probability p_1 assigned to $\mathbf{me}_1$ is 1. Under such circumstances, the element $\mathbf{me}_1$ is perturbed considering (13), generating the new position $\mathbf{me}_1^{\text{new}}$. As $\mathbf{me}_1^{\text{new}}$ is better than $\mathbf{me}_1$ according to their fitness values, the value of $\mathbf{m}_5$ in the original memory $\mathbf{M}(k+1)$ is updated with $\mathbf{me}_i^{\text{new}}$. Figure 4(c) shows the final configuration of $\mathbf{M}(k+1)$ after the exploitation operation has been achieved.

2.7. Computational Procedure. The computational procedure for the proposed algorithm can be summarized in Algorithm 6.

The APRE algorithm is an iterative process in which several actions are executed. After initialization (lines 2-3), the number of memory elements to be evolved are computed. Such number is automatically modified at each iteration

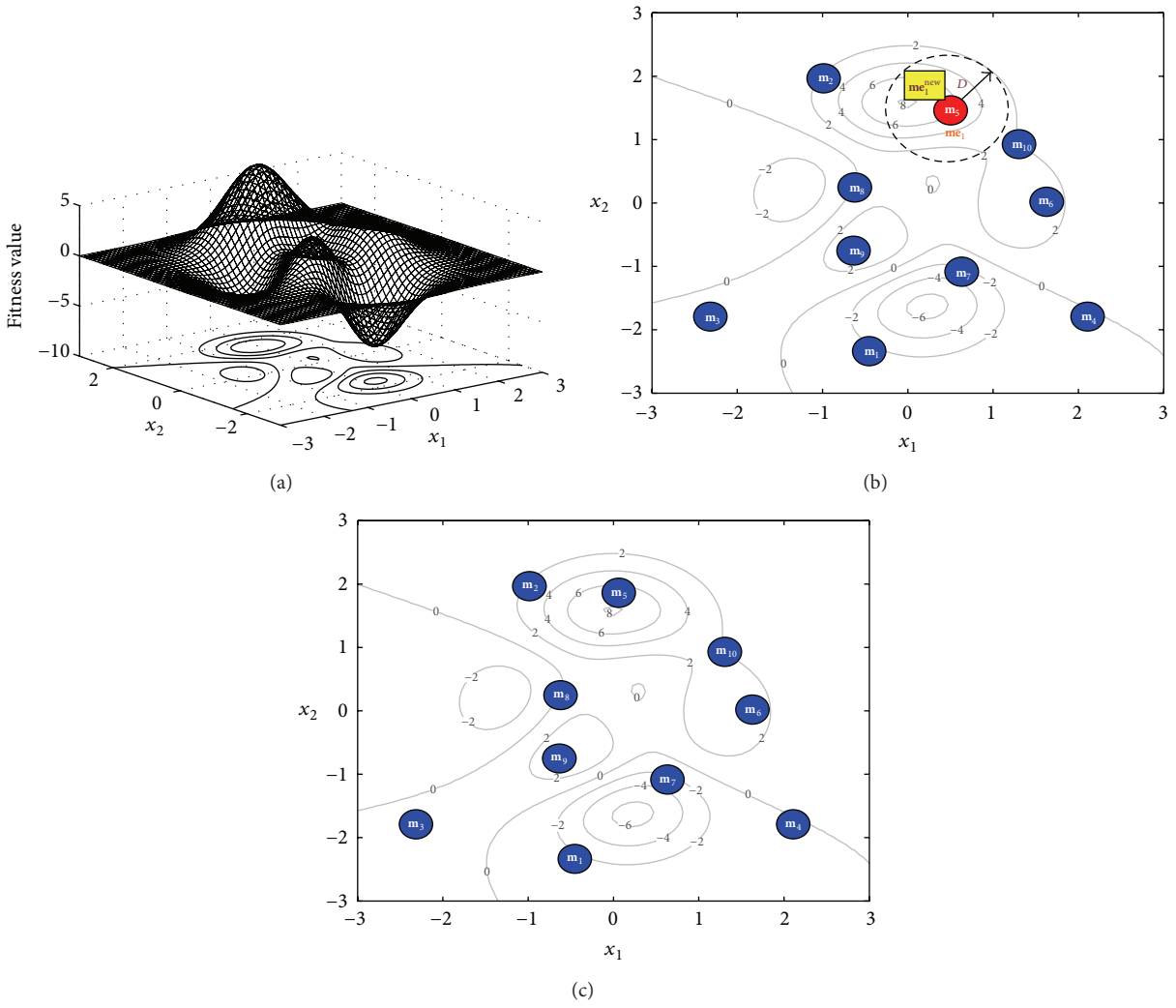

FIGURE 4: Example of the mating operation: (a) function example, (b) initial configuration before the exploitation operation, and (c) configuration after the operation.

(1) **Input:** New memory $\mathbf{M}(k+1)$, current iteration k
(2) $\mathbf{ME} \leftarrow \text{SortElementsFitness}(\mathbf{M}(k+1))$
(3) $N_e \leftarrow \text{ceil}(0.1 \cdot N_p)$
(4) $\mathbf{E} \leftarrow \text{SelectTheFirstElements}(\mathbf{ME}, N_e)$
(5) **for** $i = 1$ to N_e **do**
(6) $p_i \leftarrow (N_e + 1 - i)/N_e$
(7) **if** (rand(0, 1) <= p_i) **then**
(8) **for** $j = 1$ to n **do**
(9) $me_{i,j}^{\text{new}} \leftarrow me_{i,j} + \left[D\left((ng - k)/ng\right)\right](2 \cdot \text{rand}(0,1) - 1)$
(10) **end for**
(11) $J(\mathbf{me}_i^{\text{new}}) \leftarrow f(\mathbf{me}_i^{\text{new}})$
(12) **if** ($J(\mathbf{me}_i^{\text{new}}) > J(\mathbf{me}_i)$) **then**
(13) $\mathbf{M}(k+1) \leftarrow \text{MemoryIsUpdated}(\mathbf{me}_i^{\text{new}})$
(14) **end if**
(15) **end if**
(16) **end for**
(17) **Output:** Memory $\mathbf{M}(k+1)$

ALGORITHM 5: Exploitation operation of APRE.

(1) **Input:** N_p, N_e^0, MR, CH and max k (where max k is the maximum number of iterations).
(2) $\mathbf{M}(1) \leftarrow$ InitializeM(N_p)
(3) $c_1, \ldots, c_{N_p} \leftarrow$ ClearCounters()
(4) **for** $k = 1$ to max k **do**
(5) **Algorithm** 1
(6) **Algorithm** 2
(7) **Algorithm** 3
(8) **Algorithm** 4
(9) $\mathbf{M}(k+1) \leftarrow$ UpdateM($\mathbf{M}(k)$)
(10) $c_1, \ldots, c_{N_p} \leftarrow$ UpdateCounters($c_1, \ldots, c_{N_p}$)
(11) $\mathbf{Q} \leftarrow$ CalculateQualityFactor($\mathbf{M}(k)$)
(12) **Algorithm** 5
(13) **end for**
(14) *Solution* $\leftarrow$ FindBestElement($\mathbf{M}(k)$)
(15) **Output:** *Solution*

Algorithm 6: Computational procedure of APRE.

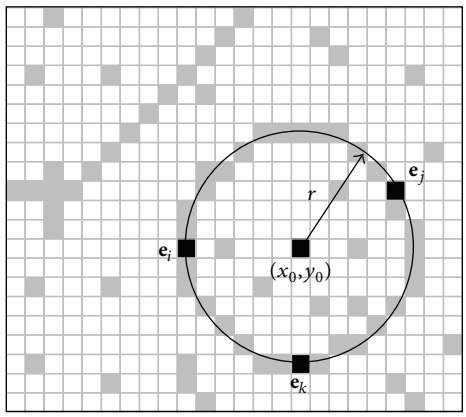

Figure 5: Circle candidate (individual) built from the combination of points $\mathbf{e}_i$, $\mathbf{e}_j$, and $\mathbf{e}_k$.

(lines 5-6). Then, a set of new individuals is generated as a consequence of the execution of the exploration operation (line 7). For each new individual, its fitness value is estimated or evaluated according to a decision taken by a fitness estimation strategy (line 8). Afterwards, the memory is updated. In this stage, the new individuals produced by the exploration operation compete against the memory elements to build the final memory configuration (lines 9–11). Finally, a sample of the best elements contained in the final memory configuration is undergone to the exploitation operation (line 12). This cycle is repeated until the maximum number of the iterations Max$k$ has been reached.

3. Implementation of APRE-Based Circle Detector

3.1. Individual Representation. In order to detect circle shapes, candidate images must be preprocessed first by the well-known Canny algorithm which yields a single-pixel edge-only image. Then, the (x_i, y_i) coordinates for each edge pixel $\mathbf{e}_i$ are stored inside the edge vector $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{zn}\}$, with zn being the total number of edge pixels. Each circle C uses three edge points as individuals in the optimization algorithm. In order to construct such individuals, three indexes $\mathbf{e}_i$, $\mathbf{e}_j$, and $\mathbf{e}_k$, are selected from vector $\mathbf{E}$, considering the circle's contour that connects them. Therefore, the circle $C = \{\mathbf{e}_i, \mathbf{e}_j, \mathbf{e}_k\}$ that crosses over such points may be considered as a potential solution for the detection problem. Considering the configuration of the edge points shown by Figure 5, the circle center (x_0, y_0) and the radius r of C can be computed as follows:

$$(x - x_0)^2 + (y - y_0)^2 = r^2. \tag{14}$$

Considering

$$\mathbf{A} = \begin{bmatrix} x_j^2 + y_j^2 - \left(x_i^2 + y_i^2\right) & 2 \cdot \left(y_j - y_i\right) \\ x_k^2 + y_k^2 - \left(x_i^2 + y_i^2\right) & 2 \cdot \left(y_k - y_i\right) \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} 2 \cdot \left(x_j - x_i\right) & x_j^2 + y_j^2 - \left(x_i^2 + y_i^2\right) \\ 2 \cdot \left(x_k - x_i\right) & x_k^2 + y_k^2 - \left(x_i^2 + y_i^2\right) \end{bmatrix},$$

$$x_0 = \frac{\det(\mathbf{A})}{4\left(\left(x_j - x_i\right)\left(y_k - y_i\right) - \left(x_k - x_i\right)\left(y_j - y_i\right)\right)}, \tag{15}$$

$$y_0 = \frac{\det(\mathbf{B})}{4\left(\left(x_j - x_i\right)\left(y_k - y_i\right) - \left(x_k - x_i\right)\left(y_j - y_i\right)\right)},$$

$$r = \sqrt{\left(x_0 - x_d\right)^2 + \left(y_0 - y_d\right)^2},$$

where $\det(\cdot)$ is the determinant and $d \in \{i, j, k\}$. Figure 5 illustrates the parameters defined by (14) to (17).

3.2. Objective Function. In order to calculate the error produced by a candidate solution C, a set of test points is calculated as a virtual shape which, in turn, must be validated, that is, if it really exists in the edge image. The test set is represented by $\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{sn}\}$, where sn is the number of points over which the existence of an edge point, corresponding to C, should be validated. In our approach, the set $\mathbf{A}$ is generated by the Midpoint Circle Algorithm

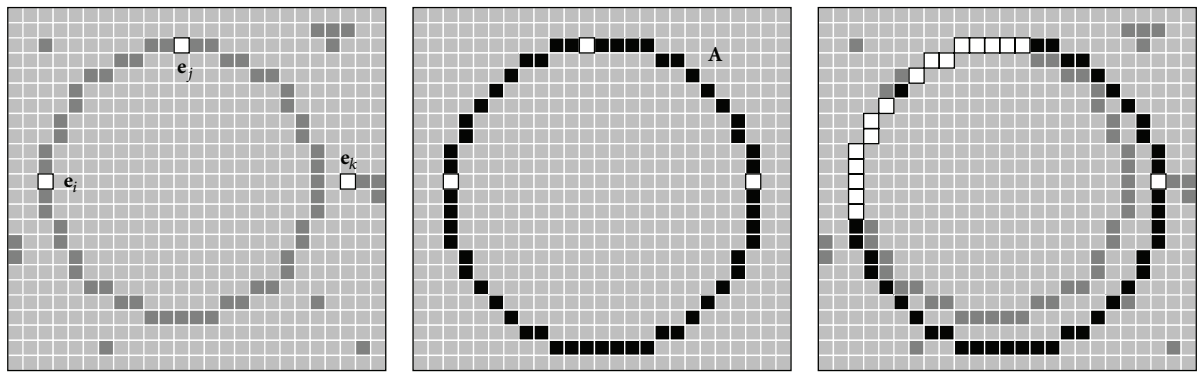

Figure 6: Procedure to evaluate the objective function $J(C)$. The image (a) presents the original edge map while (b) present the virtual shape **A** corresponding to C. The image (c) shows the coincidences between both images by means of white pixels whereas the virtual shape is depicted in black.

(MCA) [29]. The MCA is a searching method which seeks the required points for drawing a circle digitally. Therefore, MCA calculates the necessary number of test points sn to totally draw the complete circle. Such a method is considered the fastest because MCA avoids computing square-root calculations by comparing the pixel separation distances among them.

The objective function $J(C)$ represents the matching error produced between the pixels **A** of the circle candidate C (individual) and the pixels that actually exist in the edge image, yielding

$$J(C) = \frac{\sum_{v=1}^{sn} G(\mathbf{a}_v)}{sn}, \tag{16}$$

where $G(\mathbf{a}_v)$ is a function that verifies the pixel existence in $\mathbf{a}_v$, with $\mathbf{a}_v \in \mathbf{A}$ and sn being the number of pixels lying on the perimeter corresponding to C currently under testing. Hence, function $G(\mathbf{a}_v)$ is defined as

$$G(\mathbf{a}_v) = \begin{cases} 1, & \text{if the pixel } (\mathbf{a}_v) \text{ is an edge point,} \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

A value of $J(C)$ near to one implies a better response from the "circularity" operator. Figure 6 shows the procedure to evaluate a candidate solution C by using the objective function $J(C)$. Figure 6(a) shows the original edge map **E**, while Figure 6(b) presents the virtual shape **A** representing the particle $C = \{\mathbf{e}_i, \mathbf{e}_j, \mathbf{e}_k\}$. In Figure 6(c), the virtual shape **A** is compared to the edge image, point by point, in order to find coincidences between virtual and edge points. The particle has been built from points $\mathbf{e}_i$, $\mathbf{e}_j$, and $\mathbf{e}_k$ which are shown by Figure 6(a). The virtual shape **A**, obtained by MCA, gathers 56 points ($sn = 56$) with only 17 of them existing in both images (shown as white points in Figure 6(c)) and yielding $\sum_{v=1}^{sn} G(\mathbf{a}_v) = 17$, therefore $J(C) \approx 0.30$.

3.3. The Multiple Circle Detection Procedure. In order to detect multiple circles, the APRE-detector is iteratively applied. At each iteration, two actions are developed. In the first one, a new circle is detected as a consequence of the execution of the APRE algorithm. The detected circle corresponds to the candidate solution C with the best found $J(C)$ value. In the second one, the detected circle is removed from the original edge map. The processed edge map without the removed circle represents the input image for the next iteration. Such process is executed over the sequence of images until the $J(C)$ value would be lower than a determined threshold that is considered as permissible.

Table 1: APRE detector parameters.

N_p	N_e^0	MR	CH	max k
50	20	0.6	0.8	200

4. Results on Multicircle Detection

In order to achieve the performance analysis, the proposed approach is compared to the GA-based algorithm [5], the BFAO detector [9] and the RHT method [4] over an image set.

The GA-based algorithm follows the proposal of Ayala-Ramirez et al. [5], which considers the population size as 70, the crossover probability as 0.55, the mutation probability as 0.10, and the number of elite individuals as 2 and 200 generations. The roulette wheel selection and the 1-point crossover operator are both applied. The parameter setup and the fitness function follow the configuration suggested in [5]. The BFAO algorithm follows the implementation from [9] considering the experimental parameters as $S = 50$, $N_c = 350$, $N_s = 4$, $N_{\text{ed}} = 1$, $P_{\text{ed}} = 0.25$, $d_{\text{attract}} = 0.1$, $w_{\text{attract}} = 0.2$, $w_{\text{repellant}} = 10$, $h_{\text{repellant}} = 0.1$, $\lambda = 400$, and $\psi = 6$. Such values are found to be the best configuration set according to [9]. Both, the GA-based algorithm and the BAFO method use the same objective function that is defined by (16). Likewise, the RHT method has been implemented as it is described in [4]. Finally, Table 1 presents the parameters for the APRE algorithm used in this work. They have been kept for all test images after being experimentally defined.

Images rarely contain perfectly-shaped circles. Therefore, with the purpose of testing accuracy for a single-circle, the

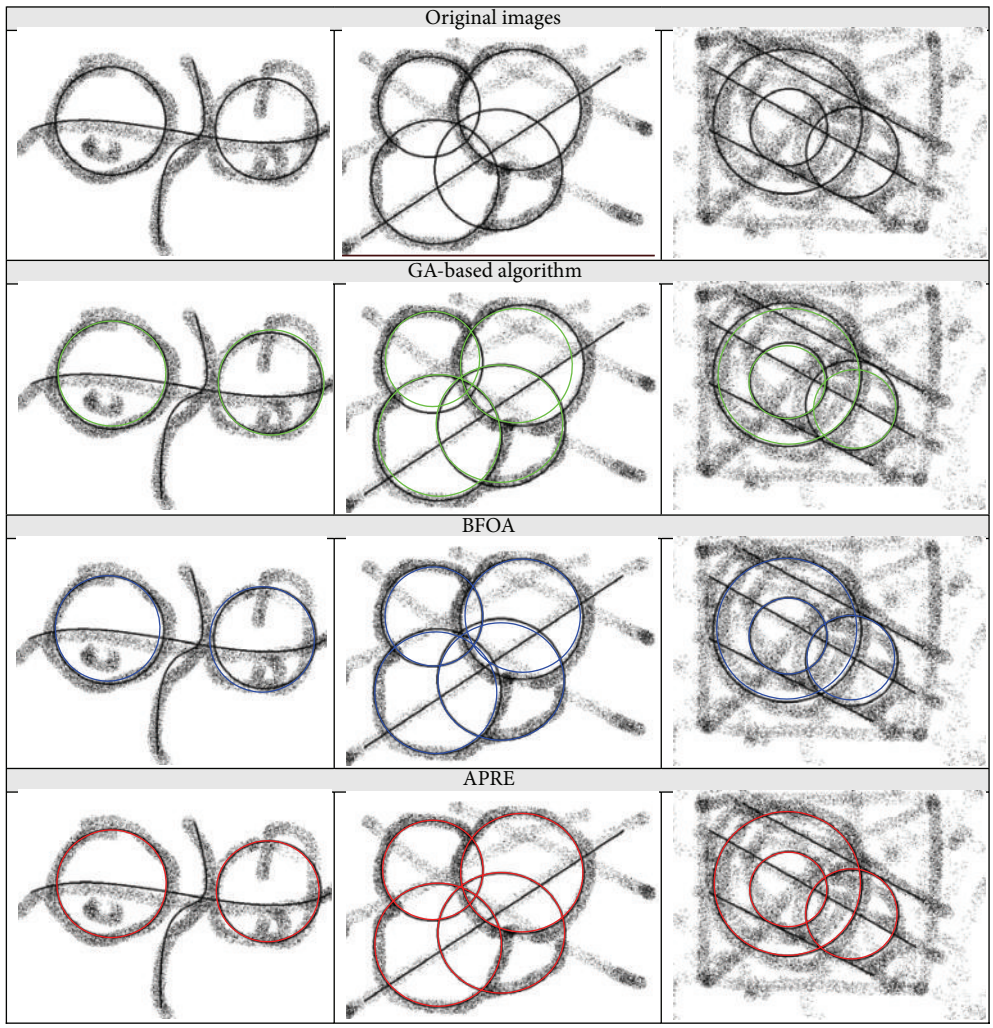

FIGURE 7: Synthetic images and their detected circles for GA-based algorithm, the BFOA method, and the proposed APRE algorithm.

detection is challenged by a ground-truth circle which is determined from the original edge map. The parameters (x_{true}, y_{true}, and r_{true}) representing the testing circle are computed using the (6)–(9) for three circumference points over the manually-drawn circle. Considering the centre and the radius of the detected circle are defined as (x_D, y_D) and r_D, the Error Score (Es) can be accordingly calculated as

$$\text{Es} = \eta \cdot \left(\left|x_{\text{true}} - x_D\right| + \left|y_{\text{true}} - y_D\right|\right) + \mu \cdot \left|r_{\text{true}} - r_D\right|. \quad (18)$$

The central point difference ($|x_{\text{true}} - x_D| + |y_{\text{true}} - y_D|$) represents the centre shift for the detected circle as it is compared to a benchmark circle. The radio mismatch ($|r_{\text{true}} - r_D|$) accounts for the difference between their radii. η and μ represent two weighting parameters which are to be applied separately to the central point difference and to the radio mismatch for the final error Es. At this work, they are chosen as $\eta = 0.05$ and $\mu = 0.1$. Such a choice ensures that the radius difference would be strongly weighted in comparison to the difference of central circular positions between the manually detected and the machine-detected circles. Here, we assume that if Es is found to be less than 1, then the algorithm gets a success; otherwise, we say that it has failed to detect the edge-circle. Note that for $\eta = 0.05$ and $\mu = 0.1$Es < 1 means the maximum difference of radius tolerated is 10, while the maximum mismatch in the location of the center can be 20 (in number of pixels). In order to appropriately compare the detection results, the Detection Rate (DR) is introduced as a performance index. DR is defined as the percentage of reaching detection success after a certain number of trials. For "success" it does mean that the compared algorithm is able to detect all circles contained in the image, under the restriction that each circle must hold the condition Es < 1. Therefore, if at least one circle does not fulfil the condition of Es < 1, the complete detection procedure is considered as a failure.

In order to use an error metric for multiple-circle detection, the averaged Es produced from each circle in the image is considered. Such criterion, defined as the Multiple Error (ME), is calculated as follows:

$$\text{ME} = \left(\frac{1}{\text{NC}}\right) \cdot \sum_{R=1}^{\text{NC}} \text{Es}_R, \quad (19)$$

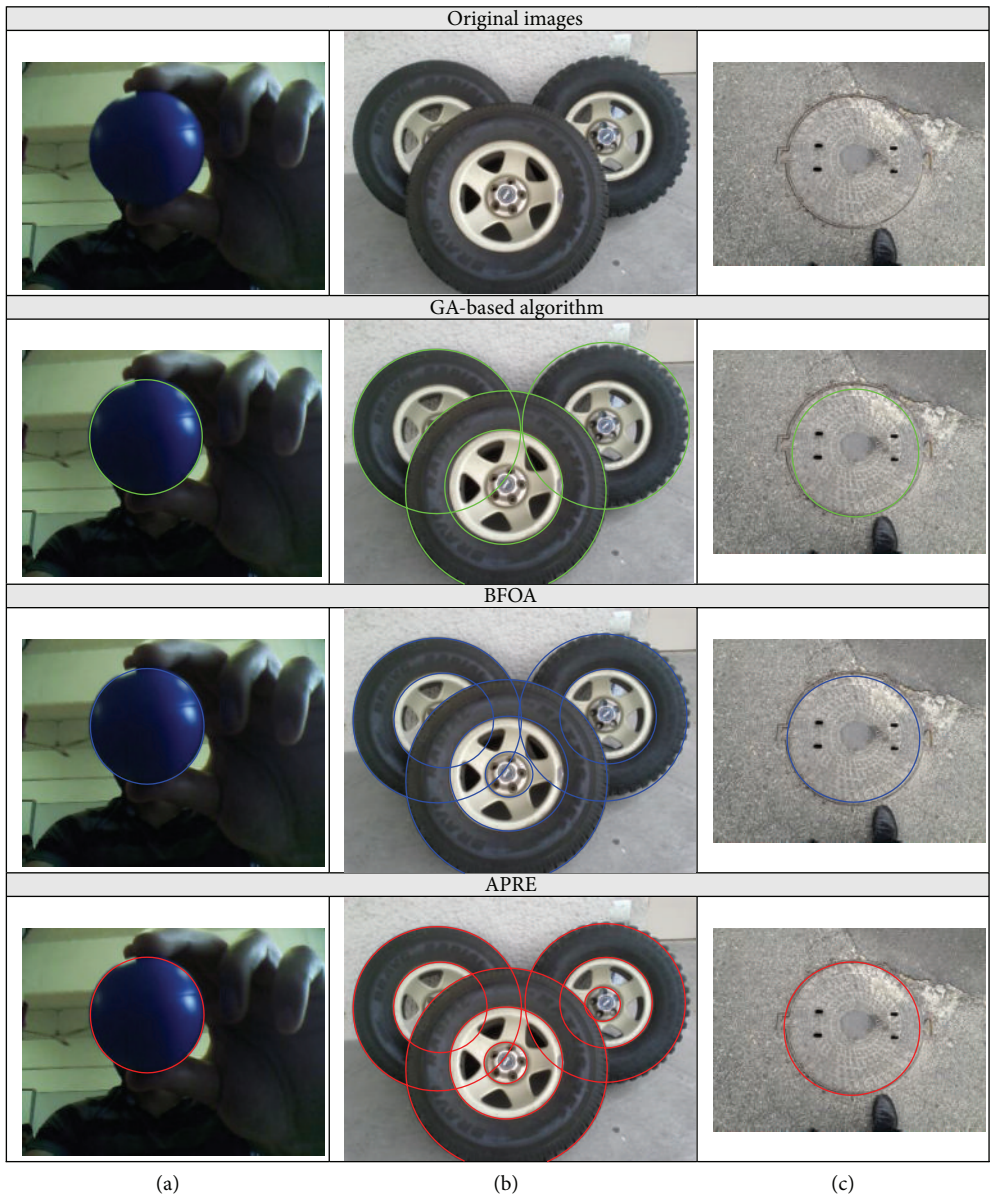

Figure 8: Real-life images and their detected circles for GA-based algorithm, the BFOA method, and the proposed APRE algorithm.

where NC represents the number of circles within the image according to a human expert.

Figure 7 shows three synthetic images and the resulting images after applying the GA-based algorithm [5], the BFOA method [9], and the proposed approach. Figure 8 presents experimental results considering three natural images. The performance is analyzed by considering 35 different executions for each algorithm. Table 2 shows the averaged execution time, the averaged number of function evaluations, the detection rate in percentage, and the averaged multiple error (ME), considering six test images (shown by Figures 7 and 8). Close inspection reveals that the proposed method is able to achieve the highest success rate still keeping the smallest error and demanding less computational time and a lower number of function evaluations for all cases.

In order to statistically analyze the results in Table 2, a nonparametric significance proof known as the Wilcoxon's rank test [30–32] for 35 independent samples has been conducted. Such proof allows assessing result differences among two related methods. The analysis is performed considering a 5% significance level over the number of function evaluations and a multiple error (ME) data. Tables 3 and 4 report the P values produced by Wilcoxon's test for a pairwise comparison of the number of function evaluations and the multiple error (ME), considering two groups gathered as APRE versus GA and APRE versus BFOA. As a null hypothesis, it is assumed that there is no difference between the values of the two algorithms. The alternative hypothesis considers an existent difference between the values of both approaches. All P values reported in Tables 3 and 4 are less than 0.05 (5%

Table 2: The averaged execution-time, detection rate, the averaged multiple error for the GA-based algorithm, the BFOA method, and the proposed APRE algorithm, considering six test images are shown by Figures 8 and 9.

	Performance indexes	Synthetic images			Natural images		
		(a)	(b)	(c)	(a)	(b)	(c)
GA	Averaged execution time	2.23	3.15	4.21	5.11	6.33	7.62
	Averaged number of function evaluations	14,000	14,000	14,000	14,000	14,000	14,000
	Success rate (DR) (%)	88	79	74	90	83	84
	Averaged ME	0.41	0.51	0.48	0.45	0.81	0.92
BFOA	Averaged execution time	1.71	2.80	3.18	3.45	4.11	5.36
	Averaged number of function evaluations	17,500	17,500	17,500	17,500	17,500	17,500
	Success rate (DR) (%)	99	92	88	96	89	92
	Averaged ME	0.33	0.37	0.41	0.41	0.77	0.37
APRE	Averaged execution time	0.21	0.36	0.20	1.10	1.61	1.95
	Averaged number of function evaluations	2,321	2,756	3,191	4,251	3,768	3,834
	Success rate (DR) (%)	100	100	100	100	100	100
	Averaged ME	0.22	0.26	0.15	0.25	0.37	0.41

Table 3: P values produced by Wilcoxon's test comparing APRE to GA and BFOA over the averaged number of function evaluations from Table 2.

Image	P value	
	APRE versus GA	APRE versus BFOA
Synthetic images		
(a)	$3.3124e-006$	$5.7628e-006$
(b)	**$5.3562e-007$**	**$6.8354e-007$**
(c)	$4.1153e-006$	$1.1246e-005$
Natural images		
(a)	$4.5724e-006$	$4.5234e-006$
(b)	**$6.7186e-006$**	**$5.3751e-006$**
(c)	$8.7691e-007$	$6.2876e-006$

Table 4: P values produced by Wilcoxon's test comparing APRE to GA and BFOA over the averaged ME from Table 2.

Image	P value	
	APRE versus GA	APRE versus BFOA
Synthetic images		
(a)	$1.7345e-004$	$1.5294e-004$
(b)	**$1.6721e-004$**	**$1.4832e-004$**
(c)	$1.0463e-004$	$1.9734e-004$
Natural images		
(a)	$1.5563e-004$	$1.6451e-004$
(b)	**$1.2748e-004$**	**$1.5621e-004$**
(c)	$1.0463e-004$	$1.7213e-004$

significance level) which is a strong evidence against the null hypothesis, indicating that the best APRE mean values for the performance are statistically significant which has not occurred by chance.

Figure 9 demonstrates the relative performance of APRE in comparison with the RHT algorithm as it is described in [4]. All images belonging to the test are complicated and contain different noise conditions. The performance analysis is achieved by considering 35 different executions for each algorithm over the three images. The results, exhibited in Figure 9, present the median-run solution (when the runs were ranked according to their final ME value) obtained throughout the 35 runs. On the other hand, Table 5 reports the corresponding averaged execution time, detection rate (in %), and average multiple error (using (10)) for APRE and RHT algorithms over the set of images. Table 5 shows a decrease in performance of the RHT algorithm as noise conditions change. Yet the APRE algorithm holds its performance under the same circumstances.

5. Conclusions

In this paper, a novel evolutionary algorithm called the Adaptive Population with Reduced Evaluations (APRE) is introduced to solve the problem of circle detection. The proposed algorithm reduces the number of function evaluations through the use of two mechanisms: (1) adapting dynamically the size of the population and (2) incorporating a fitness calculation strategy which decides when it is feasible to calculate or only estimate new generated individuals.

The algorithm begins with an initial population which will be used as a memory during the evolution process. To each memory element, it is assigned a normalized fitness value called quality factor that indicates the solution capacity provided by the element. From the memory, only a variable subset of elements is considered to be evolved. Like other population-based methods, the proposed algorithm generates new individuals considering two operators: exploration and exploitation. Such operations are applied to improve the quality of the solutions by (1) searching the unexplored solution space to identify promising areas containing better solutions than those found so far and (2) successive refinement of the best found solutions. Once the new individuals are generated, the memory is updated. In such stage, the new individuals compete against the memory elements to build the final memory configuration.

In order to save computational time, the approach incorporates a fitness estimation strategy that decides which

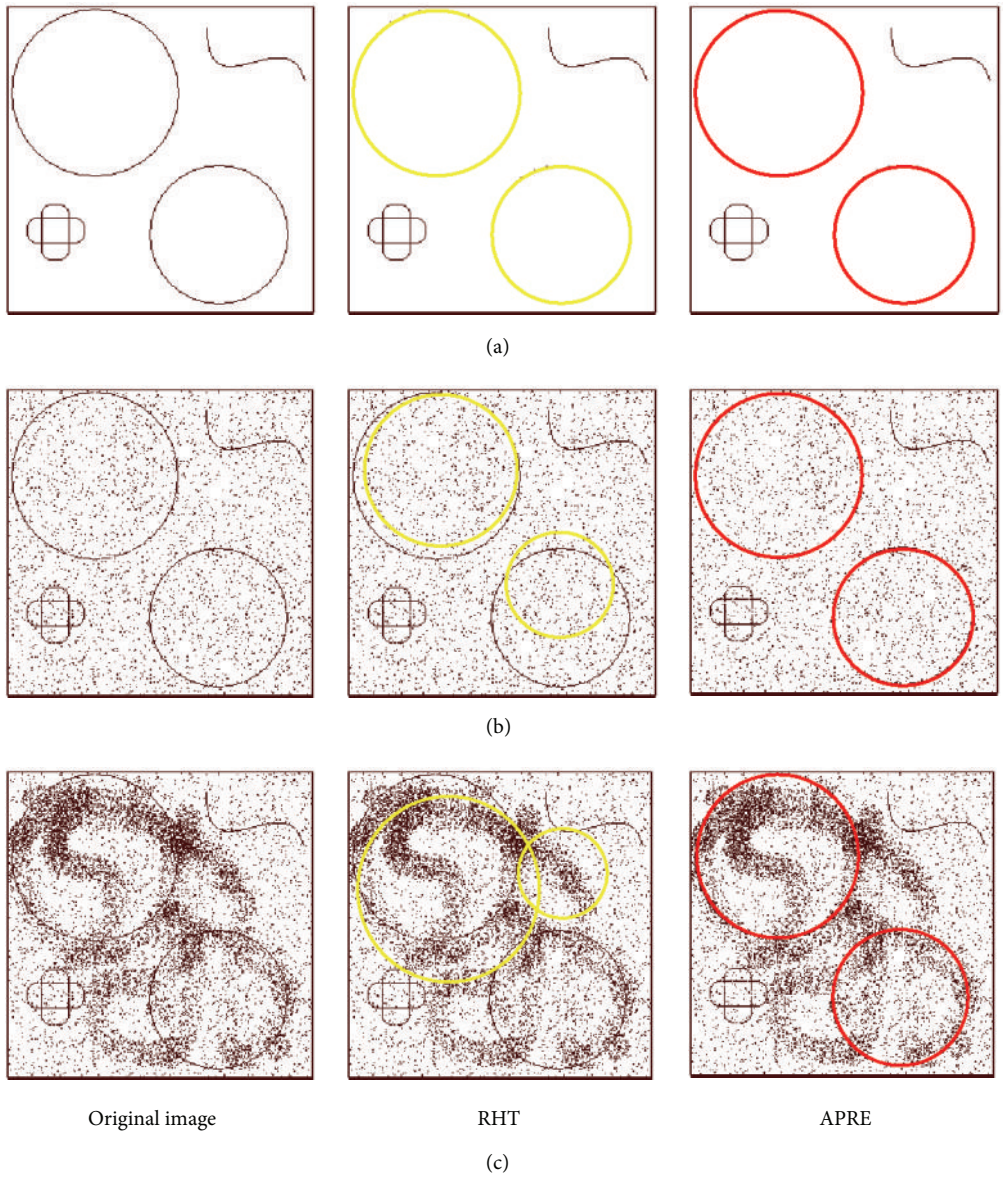

Figure 9: Relative performance of the RHT and the APRE.

Table 5: Average time, detection rate, and averaged error for APRE and RHT, considering three test images.

Image	Average time		Success rate (DR) (%)		Average ME	
	RHT	APRE	RHT	APRE	RHT	APRE
(a)	7.82	0.20	100	100	0.19	0.12
(b)	**8.65**	**0.12**	**64**	**100**	**0.47**	**0.11**
(c)	10.65	0.15	11	100	1.21	0.13

individuals can be estimated or actually evaluated. As a result, the approach can substantially reduce the number of function evaluations, yet preserving its good search capabilities. The proposed fitness calculation strategy estimates the fitness value of new individuals using memory elements located in neighboring positions that have been visited during the evolution process. In the strategy, those new individuals, close to a memory element whose quality factor is high, have a great probability to be evaluated by using the true objective function. Similarly, it is also evaluated those new particles lying in regions of the search space with no previous evaluations. The remaining search positions are estimated assigning them the same fitness value that the nearest location of the memory element. By the use of such fitness estimation method, the fitness value of only very few individuals are actually evaluated whereas the rest is just estimated.

Different to other approaches that use an already existent EA as framework, the APRE method has been completely designed to substantially reduce the computational cost but preserving good search effectiveness.

To detect the circular shapes, the detector is implemented by using the encoding of three pixels as candidate circles over the edge image. An objective function evaluates if such

candidate circles are actually present in the edge image. Guided by the values of this objective function, the set of encoded candidate circles are evolved using the operators defined by APRE so that they can fit to the actual circles on the edge map of the image.

In order to use either a fitness estimation strategy or an adaptive population size approach, it is necessary but not sufficient to tackle the problem of reducing the number of function evaluations. Using a fitness estimation strategy, during the evolution process with no adaptation of the population size to improve the population diversity, makes the algorithm defenseless against the convergence to a false minimum and may result in poor exploratory characteristics of the algorithm [18]. On the other hand, the adaptation of the population size omitting the fitness estimation strategy leads to increase in the computational cost [20]. Therefore, it does seem reasonable to incorporate both approaches into a single algorithm.

In order to test the circle detection accuracy, a score function is used (19). It can objectively evaluate the mismatch between a manually detected circle and a machine-detected shape. We demonstrated that the APRE method outperforms both the evolutionary methods (GA and BFOA) and Hough Transform-based techniques (RHT) in terms of speed and accuracy, considering a statistically significant framework (Wilconxon test). Results show that the APRE algorithm is able to significantly reduce the computational overhead as a consequence of decrementing the number of function evaluations.

Funding

The proposed algorithm is part of the vision system used by a biped robot supported under the Grant CONACYT CB 181053.

References

[1] X. Bai, X. Yang, and L. J. Latecki, "Detection and recognition of contour parts based on shape similarity," *Pattern Recognition*, vol. 41, no. 7, pp. 2189–2199, 2008.

[2] K. Schindler and D. Suter, "Object detection by global contour shape," *Pattern Recognition*, vol. 41, no. 12, pp. 3736–3748, 2008.

[3] T. J. Atherton and D. J. Kerbyson, "Using phase to represent radius in the coherent circle Hough transform," in *Proceedings of the IEE Colloquium on the Hough Transform*, IEE, London, UK, May 1993.

[4] L. Xu, E. Oja, and P. Kultanen, "A new curve detection method: randomized hough transform (RHT)," *Pattern Recognition Letters*, vol. 11, no. 5, pp. 331–338, 1990.

[5] V. Ayala-Ramirez, C. H. Garcia-Capulin, A. Perez-Garcia, and R. E. Sanchez-Yanez, "Circle detection on images using genetic algorithms," *Pattern Recognition Letters*, vol. 27, no. 6, pp. 652–657, 2006.

[6] E. Cuevas, N. Ortega-Sánchez, D. Zaldivar, and M. Pérez-Cisneros, "Circle detection by harmony search optimization," *Journal of Intelligent & Robotic Systems*, vol. 66, no. 3, pp. 359–376, 2011.

[7] E. Cuevas, D. Oliva, D. Zaldivar, M. Pérez-Cisneros, and H. Sossa, "Circle detection using electro-magnetism optimization," *Information Sciences*, vol. 182, no. 1, pp. 40–55, 2012.

[8] E. Cuevas, D. Zaldivar, M. Pérez-Cisneros, and M. Ramírez-Ortegón, "Circle detection using discrete differential evolution optimization," *Pattern Analysis&Applications*, vol. 14, no. 1, pp. 93–107, 2011.

[9] S. Dasgupta, S. Das, A. Biswas, and A. Abraham, "Automatic circle detection on digital images with an adaptive bacterial foraging algorithm," *Soft Computing*, vol. 14, no. 11, pp. 1151–1164, 2010.

[10] Y. Jin, "Comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, vol. 9, no. 1, pp. 3–12, 2005.

[11] Y. Jin, "Surrogate-assisted evolutionary computation: recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[12] J. Branke and C. Schmidt, "Faster convergence by means of fitness estimation," *Soft Computing*, vol. 9, no. 1, pp. 13–20, 2005.

[13] Z. Zhou, Y. Ong, M. Nguyen, and D. Lim, "A Study on polynomial regression and gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, vol. 3, Edinburgh, UK, September 2005.

[14] A. Ratle, "Kriging as a surrogate fitness landscape in evolutionary optimization," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 15, no. 1, pp. 37–49, 2001.

[15] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff, "Generalizing surrogate-assisted evolutionary computation," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 329–355, 2010.

[16] Y. S. Ong, K. Y. Lum, and P. B. Nair, "Hybrid evolutionary algorithm with Hermite radial basis function interpolants for computationally expensive adjoint solvers," *Computational Optimization and Applications*, vol. 39, no. 1, pp. 97–119, 2008.

[17] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 481–494, 2002.

[18] Y. S. Ong, P. B. Nair, and A. J. Keane, "Evolutionary optimization of computationally expensive problems via surrogate modeling," *AIAA Journal*, vol. 41, no. 4, pp. 687–696, 2003.

[19] D. Chen and C. Zhao, "Particle swarm optimization with adaptive population size and its application," *Applied Soft Computing*, vol. 9, no. 1, pp. 39–48, 2009.

[20] W. Zhu, Y. Tang, J. Fang, and W. Zhang, "Adaptive population tuning scheme for differential evolution," *Information Sciences*, vol. 223, pp. 164–191, 2013.

[21] J. Brest and M. Sepesy Maučec, "Population size reduction for the differential evolution algorithm," *Applied Intelligence*, vol. 29, no. 3, pp. 228–247, 2008.

[22] S. Oh, Y. Jin, and M. Jeon, "Approximate models for constraint functions in evolutionary constrained optimization," *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 11, pp. 6585–6603, 2011.

[23] C. Luo, S.-L. Zhang, C. Wang, and Z. Jiang, "A metamodel-assisted evolutionary algorithm for expensive optimization," *Journal of Computational and Applied Mathematics*, vol. 236, no. 5, pp. 759–764, 2011.

[24] K. C. Tan, S. C. Chiam, A. A. Mamun, and C. K. Goh, "Balancing exploration and exploitation with adaptive variation for

evolutionary multi-objective optimization," *European Journal of Operational Research*, vol. 197, no. 2, pp. 701–713, 2009.

[25] E. Alba and B. Dorronsoro, "The exploration/exploitation tradeoff in dynamic cellular genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 2, pp. 126–142, 2005.

[26] B. O. Arani, P. Mirzabeygi, and M. S. Panahi, "An improved PSO algorithm with a territorial diversity-preserving scheme and enhanced exploration-exploitation balance," *Swarm and Evolutionary Computation*, vol. 11, pp. 1–15, 2013.

[27] R. Storn and K. Price, "Differential evolution-a simple and efficient adaptive scheme for global optimisation over continuous spaces," Technical Report, TR-95-012, Institute for Clinical Systems Improvement (ICSI), Berkeley, Calif, USA, 1995.

[28] R. Angira and A. Santosh, "Optimization of dynamic systems: a trigonometric differential evolution approach," *Computers and Chemical Engineering*, vol. 31, no. 9, pp. 1055–1063, 2007.

[29] J. Bresenham, "Linear algorithm for incremental digital display of circular Arcs," *Communications of the ACM*, vol. 20, no. 2, pp. 100–106, 1977.

[30] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[31] S. Garcia, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special session on real parameter optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.

[32] J. Santamaría, O. Cordón, S. Damas, J. M. García-Torres, and A. Quirin, "Performance evaluation of memetic approaches in 3D reconstruction of forensic objects," *Soft Computing*, no. 8-9, pp. 883–904, 2009.