Hybridization of the Univariate Marginal Distribution Algorithm with Simulated Annealing for Parametric Parabola Detection

S. Ivvan Valdez, Susana Espinoza-Perez, Fernando Cervantes-Sanchez and Ivan Cruz-Aceves*

Abstract This chapter presents a new hybrid optimization method based on the univariate marginal distribution algorithm for continuous domain, and the heuristic of simulated annealing for the parabola detection problem. The hybrid proposed method is applied on the DRIVE database of retinal fundus images to approximate the retinal vessels as a parabolic shape. The hybrid method is applied separately using two different objective functions. Firstly, the objective function only considers the superposition of pixels between the target pixels in the input image and the virtual parabola, and secondly, the objective function implements a weighted restriction on the pixels close to the parabola vertex. Both objective functions in the hybrid method obtain suitable results to approximate a parabolic form on the retinal vessels present in the retinal images. The experimental results present that the parabola detection results obtained from the proposed method are more robust than those obtained by the comparative method. Additionally, the average execution time achieved from the proposed hybrid method (1.57 seconds) is lower than the computational time obtained by the comparative method on the database of 20 retinal images, which is of interest for computer-aided diagnosis in clinical practice.

S. Ivvan Valdez and Fernando Cervantes-Sanchez
Centro de Investigación en Matemáticas (CIMAT), A.C., Jalisco S/N, Col. Valenciana, C.P. 36000, Guanajuato, Gto, México.

Susana Espinoza-Perez
Universidad del Papaloapan, Ingeniería en Computación, Av. Ferrocarril s/n, CD. Universitaria, C.P. 68400, Loma Bonita, Oaxaca, México.

Ivan Cruz-Aceves (Corresponding author)
CONACYT - Centro de Investigación en Matemáticas (CIMAT), A.C., Jalisco S/N, Col. Valenciana, C.P. 36000, Guanajuato, Gto, México. e-mail: ivan.cruz@cimat.mx

1 Introduction

The automatic detection of parametric objects represents an essential task in different research areas such as medical and natural image analysis. In general, this problem can be addressed in two steps; definition of the parametric equation of the target object (which is not an easy task) and the definition of a search strategy. In the first step, the boundaries of the target object have to be clearly detected in order to perform the matching process with the parametric equation. This task is commonly performed by using edge detection techniques such as Sobel or Canny operators. To perform the second step, an exhaustive or heuristic strategy can be applied.

In literature, the Hough transform (HT) is the most highlighted method for parametric object detection [1, 2, 3]. The HT is a standard technique for parametric shape recognition useful for image analysis and computer vision. HT was first applied for straight line detection [4], and later used to detect circles [5, 6, 7], ellipses [8], and parabolas [9, 10]. The main advantages of the HT method are that it is insensitive to noise and it is easy of implementation, however, the main disadvantage is the execution time, since the computational complexity is $\mathscr{O}(n^s)$, where s is the number of unknown variables in the parametric equation defining the target object.

An important application of the Hough transform for parabola detection is the approximation of the blood-vessels shape in fundus images of the retina. It is known that some phatologies, such as diabetic retonopaty, affect the shape of the vessels in the retina and because of this the shape itself can be used to aid the diagnosis of those diseases [9, 10]. In medical practice, the approximation of the blood-vessels shape in retinal images is an exhaustive manual task which requires the visual detection of representative features on the retinal vasculature. This task can be automated using the HT transform embedded within a computerized aided system in order to reduce the time and intensive labour taken to a specialist during the diagnose of diseases.

On the other hand, to solve the drawback of computational complexity of the HT method, several strategies have been introduced involving least squares [11], pruning-and-voting strategy [12], randomized techniques [13, 14, 15, 16] and population-based methods based on particle swarm optimization [17], electro-magnetism optimization [18], genetic algorithms [19], artificial immune system [20], differential evolution [21], and estimation of distribution algorithms (EDAs) [22, 23].

In general, the performance of the population-based methods is very suitable in terms of computational time and detection accuracy, since they are useful to avoid the local minima problem. In the present chapter, a new hybrid optimization method based on an Estimation of Distribution Algorithm for continuous domain, and Simulated Annealing is introduced to detect parabolic shapes on retinal fundus images. Since the proposed method, is an stochastic optimization technique, the introduced fitness function is based on the shape maximization between a virtual parabola and the target object of the input image. In addition, the proposed method is compared with different state-of-the-art parametric detection methods in terms of computational time, robustness, and detection accuracy.

The present chapter is organized as follows. In section 2, the parabola detection problem is introduced along with the techniques of Simulated Annealing and Estimation of Distribution Algorithms, which are explained in detail. Section 3, the proposed hybrid method for parabola detection is introduced and analyzed. Computational results are presented and discussed in section 4, and conclusions are given in section 5.

2 Background

This section introduces the fundamentals of the parabola detection problem and two Estimation of Distribution Algorithms of the state-of-the-art along with the local search method known as Simulated Annealing.

2.1 Parabola detection problem

The parabola detection problem can be defined as a matching problem. Hence, we draw a parabola in a binary image, then we compute the number of white pixels which match with the white pixels in the target image. In order to draw the parabola we require a set of four parameters $\mathbf{x} = [x_1 = a, x_2 = b, x_3 = c, x_4 = \theta]$ used as shown in Equation (1).

$$\begin{aligned}
\hat{y} &= a\hat{x}^2 + b\hat{x} + c, \\
x_v &= -\frac{b}{2a}, \\
y_v &= ax_v^2 + bx_v + c, \\
x &= integer(\cos(\theta)(\hat{x} - x_v) - \sin(\theta)(\hat{y} - y_v) + x_v), \\
y &= integer(\sin(\theta)(\hat{x} - x_v) + \cos(\theta)(\hat{y} - y_v) + y_v),
\end{aligned} \tag{1}$$

where$[a,b,c]$ are used to defined a parabola aligned to the $y - axis$, then it is rotated a θ angle, with respect to the vertex $((x_v, y_v)$. Consider a target image with height n_{row} and width n_{col}, we draw a parabola with a set of coordinates $[x_i, y_i]$ for $i = 1,2,...,n_{coord}$, where $x_i \in \{1..n_{row}\}$ and $y_i \in \{1..n_{col}\}$. Then, we count the number of pixels p that are different from 0 in the parabola and the target images according to Equation (2). This equation is the *objective function*, notice that $[x_i, y_i]$ depend on $\mathbf{x} = [x_1 = a, x_2 = b, x_3 = c, x_4 = \theta]$.

$$f(x_i, y_i) = \sum_{i=1}^{n_{coord}} p_{par}(x_i, y_i) p_{target}(x_i, y_i), \tag{2}$$

where $p_{par}(x_i, y_i)$ and $p_{target}(x_i, y_i)$ are pixel values in the parabola and target image, respectively.

2.1.1 Hough transform

In image analysis, the Hough transform (HT) is the most commonly used strategy to detect parametric forms such as lines or circles [1]. On the other hand, the HT can also be extended for detecting parabolic shapes. In general, HT requires the parametric equation of the object to be detected in the Cartesian or polar coordinate system. For instance, to detect lines, circles or parabolas in Cartesian coordinates, the following equations can be evaluated, respectively:

$$y = mx + b \tag{3}$$

$$r^2 = (x-a)^2 + (y-b)^2 \tag{4}$$

$$(y-y_0)^2 = 4a(x-x_0) \tag{5}$$

In order to detect parametric objects in images by using the HT algorithm, the input image must be binary. The pixels with intensity different to zero represent the potential object to be detected, while all the pixels with intensity zero represent the background image, which are irrelevant for the process.

The resolution of the search space plays an important role, since it is used for testing all possible parameter combinations (exhaustive search strategy). Consequently, there is a trade-off between detection precision and computational time. High precision obtains good detection results involving high execution time, and low precision may not found the target object.

The generic procedure to perform the Hough transform, can be described as follows:

1. Determine the equation of the parametric object to be evaluated, and the number of unknown parameters n.
2. Initialize the dimension of the accumulator array according to n.
3. Compute for each interest pixel (x,y) the parametric equation.
4. Increment the value of the accumulator array for the set of unknown parameters.
5. Find the sets of parameters with the highest values.
6. Determine the best set of parameters for the parametric object (typically by applying a threshold or a local maxima strategy).

The HT strategy presents two main disadvantages. Firstly, the high computational time of the exhaustive search, and secondly, the method to determine the optimal set of values, where the most widely strategy used to find it is the local maxima method.

2.2 Simulated Annealing

The Simulated Annealing (SA) is a widely known algorithm firstly proposed for discrete domains by Kirkpatrick et al. [24], then other researchers have presented different modifications and versions. One the best performed in continuous domains is the one proposed by Corana et al. [25]. We used this version as local optimizer. Nevertheless, simulated annealing is a global optimizer for multimodal functions, we use it as a local optimizer, considering that it search in a vicinity of the best solution at each iteration of the optimization process. SA is shown in Algorithm 1.

Algorithm 1: Simulated annealing.

```
   Input: $T_0$ = A high temperature.
   $x^{inf}, x^{sup}$ = Inferior and superior limits respectively.
1  $t = 0$;
2  Initial solution $\mathbf{x}_{best} = \mathbf{x}_0 = [x_1, x_2, ..., x_n]$;
3  Evaluate and initialize the best objective function value $f_{best} = f(\mathbf{x}_t)$;
4  $\hat{\mathbf{x}} = \mathbf{x}_t$;
5  while Stopping criterion is not met do
6      for $T = 1...N_T$ do
7          for $n_s = 1..N_s$ do
8              for $i = 1..n$ do
9                  $e_i \sim U(-v_i, v_i)$ ;
10                 while $(\hat{x}_i + e_i) < x_i^{inf} \vee (\hat{x}_i + e_i) > x_i^{sup}$ do
11                     $e_i \sim U(x^{inf}, x^{sup})$;
12                 Evaluate $\hat{\mathbf{x}}$;
13                 if $f(\hat{\mathbf{x}}) < f_{best}$ then
14                     $f_{best} = f(\hat{\mathbf{x}})$;
15                     $\mathbf{x}_{best} = f(\hat{\mathbf{x}})$;
16                 Accept the solution $\hat{\mathbf{x}}$ according to the Metropolis criterion;
17         Adjust the step size $v$;
18     $T = \alpha_T T$;
19     Update $\mathbf{x}_t = \mathbf{x}_{best}$ and $f(\mathbf{x}_t) = f_{best}$;
```

The initial temperature is set as: $T_0 = -\hat{f}/(2log(0.2))$ where $\hat{f}$ is the average of the objective function value of the current population. Notice that the initial temperature changes every iteration. The step size is computed as suggested by Corana et al. [25]. Finally, the well known Metropolis criterion is as follows:

1. Let $\mathbf{x}_t$ be the current solution and $\hat{\mathbf{x}}$ the perturbed solution.
2. $\Delta f = f(\mathbf{x}_t) - f(\hat{\mathbf{x}})$.

3. **If** ($\Delta f(\mathbf{x}) \leq 0$) **then** $\mathbf{x}_{t+1} = \hat{\mathbf{x}}$.
4. **else** The perturbed solution is accepted with probability $\exp(\frac{-\Delta f}{T})$, it is to say the perturbed solution replaces the current.

The Metropolis criterion permits to accept wrong solutions than the current with the aim of avoiding local minima problem. The algorithm stops if the objective function is not improved more than $\varepsilon = 1$ in 5 consecutive temperature reductions or it reaches 30 function evaluations. Furthermore, the Simulated Annealing algorithm is presented in the appendix A in C programming language.

2.3 Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDAs) are population-based methods used to solve optimization problems. The main difference with Evolutionary computation techniques, is the fact that the crossover and mutation operators are not required, since the new potential solutions are generated by building probabilistic models based on statistical information of promising solutions [26, 27, 28]. In this work, we focus on the Univariate Marginal Distribution Algorithm in both, discrete and continuous domain.

2.3.1 Univariate Marginal Distribution Algorithm

The Univariate Marginal Distribution Algorithm (UMDA) is a search strategy derived from EDAs to solve linear optimization problems with not many significant dependencies [29]. This stochastic method is a population-based strategy where a marginal probability is computed at each generation [30]. The probabilistic model is calculated from a subset of individuals (potential solutions) in order to generate the next population of the iterative process. Similar to Evolutionary Computation techniques, UMDA uses a fitness function and binary encoding to represent the individuals. Consequently, the genes of the individuals are randomly initialized between $\{0,1\}$ with a uniform probability.

The selection step is applied to select a subset of individuals. This operator uses the truncation strategy, which order the solutions according to their fitness. Subsequently, the estimation of the univariate marginal probabilities P are computed using the subset of individuals. The marginal probability model for independent variables can be defined as follows:

$$P(x) = \prod_{i=1}^{n} P(X_i = x_i) \ , \tag{6}$$

where $x = (x_1, x_2, \ldots, x_n)^T$ is the binary value of the ith bit in the solution, and X_i is the ith uniform random value of the vector X.

Finally, UMDA generates a new population from the estimated marginal probability model. The process is iteratively performed until a convergence criterion is satisfied, and the best individual (elite) is such with the best fitness along generations.

According to the above description, UMDA can be implemented as follows:

1. Initialize number of individuals n.
2. Initialize number of generations t.
3. Initialize selection rate $[0,1]$.
4. Initialize the individuals into the predefined search space.
5. Select a subset of individuals S of $m \leq n$ according to the selection rate.
6. Compute the marginal probabilities $p_i^s(x_i,t)$ of S.
7. Generate n new individuals by computing $p(x,t+1) = \prod_{i=1}^{n} p_i^s(x_i,t)$.
8. Stop if convergence criterion is satisfied, otherwise, repeat steps (4)-(7).

2.3.2 Univariate Marginal Distribution Algorithm for Continuous Domains (UMDA$_c$)

The Univariate Marginal Distribution Algorithm for Continuous Domains (UMDA$_c$) is an evolutionary algorithm from the family of Estimation of Distribution Algorithms (EDA). It was proposed by Larrañaga et al. [26] In this case, we use a particular version which a priori defines the normal probability function, this version is named as UMDA$_{Gc}$. The UMDA$_{Gc}$ considers that variables are independent from each other, then the joint probability function can be written as follows:

$$f_{\mathcal{N}}(x,\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_i}} e^{-\frac{1}{2}\left(\frac{x_i-\mu_i}{\sigma_i}\right)^2} \tag{7}$$

Thus, the parameters μ_i and σ_i must be estimated from the selected set for each dimension i. The UMDA$_{Gc}$ is shown in Algorithm 2. At line 2, the initial population is sampled from a uniform distribution defined inside given limits. At line 3, the population is evaluated, then the half of the best solutions are selected in the S set, and the elite individual, x_{best}, f_{best}, is stored. Then, for n_{gen} generations, the vectors of means μ and standard deviations σ are estimated via maximum likelihood estimators over S. At line 7, the new population is formed by the combination of $n_{pop} - 1$ sampled candidate solutions and the elite individual, and so on. Figure 1 illustrates the flow of the algorithm 2 for the UMDA$_{Gc}$ strategy.

Algorithm 2: The Univariate Marginal Distribution for Continuous Domains using a Normal Distribution

```
Data: D=Number of dimensions.
      $n_{pop}$=Population size.
      $n_{gen}$=Number of generations.
      $x_{inf}$ =Inferior search limits.
      $x_{sup}$ =Superior search limits.
Result: $x_{best}$ and $f_{best}$. Optimum aproximation and its objective function value.
1  $t = 0$;
2  $X^t \sim U(x_{inf}, x_{sup})$;
3  $F^t = Evaluate(X^t)$;
4  $[S^t, x^t_{best}, f^t_{best}] = Selection(X^t, F^t)$;
5  for $1..n_{gen}$ do
6      $[\mu^t, \sigma^t] = ParameterEstimation(S^t)$;
7      $X^{t+1} = [x^t_{best}, Sampling(\mu^t, \sigma^t, n_{pop} - 1)]$;
8      $t = t + 1$;
9      $F^t = Evaluate(X^t)$;
10     $[S, x^t_{best}, f^t_{best}] = Selection(X^t, F^t)$;
```

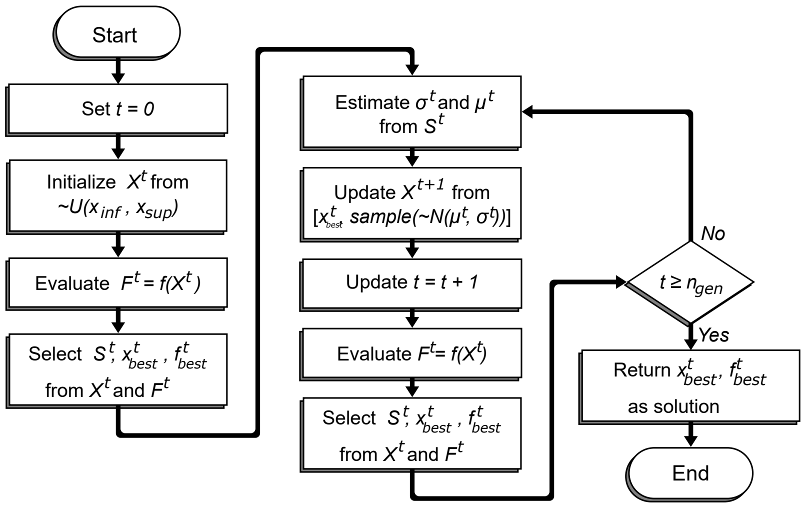

Start
Set $t = 0$
Initialize X^t from ~$U(x_{inf}, x_{sup})$
Evaluate $F^t = f(X^t)$
Select $S^t, x^t_{best}, f^t_{best}$ from X^t and F^t
Estimate σ^t and μ^t from S^t
Update X^{t+1} from $[x^t_{best}$ sample(~$N(\mu^t, \sigma^t)$)]
Update $t = t + 1$
Evaluate $F^t = f(X^t)$
Select $S^t, x^t_{best}, f^t_{best}$ from X^t and F^t
$t \geq n_{gen}$
No
Yes
Return x^t_{best}, f^t_{best} as solution
End

Fig. 1 Flowchart of the algorithm of the Univariate Marginal Distribution Algorithm for continuous domains using a Normal Distribution.

2.3.3 Applications

The Estimation of Distribution Algorithms have proven to be very effective for solving high-dimensional optimization problems. In literature, different mathematical functions have been introduced for testing optimization algorithms. In this section, to illustrate the implementation of UMDA$_{Gc}$ in an optimization task, the *2-D Goldstein-Price* function is introduced.

Commonly, the range for each variable of the test function is $X_1 \in$ [-2, 2], $X_2 \in$ [-2, 2], and the optimal value is located on $f(0,-1) = 3$. The *2-D Goldstein-Price* test function is defined below in Equation 8, and illustrated in Fig. 2.

$$f(x_1,x_2) = \left[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\right] \times \\ \left[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right]. \quad (8)$$

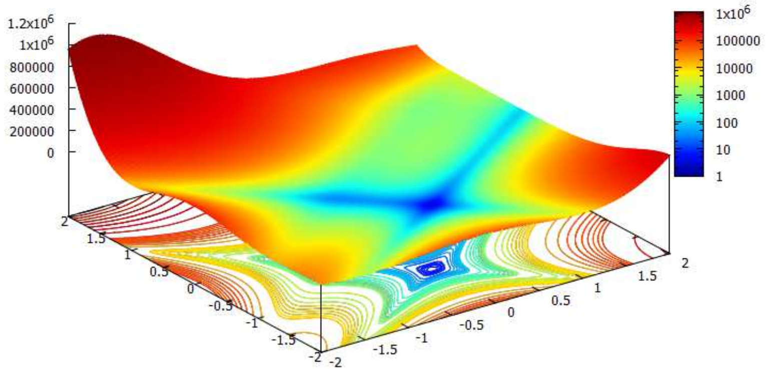
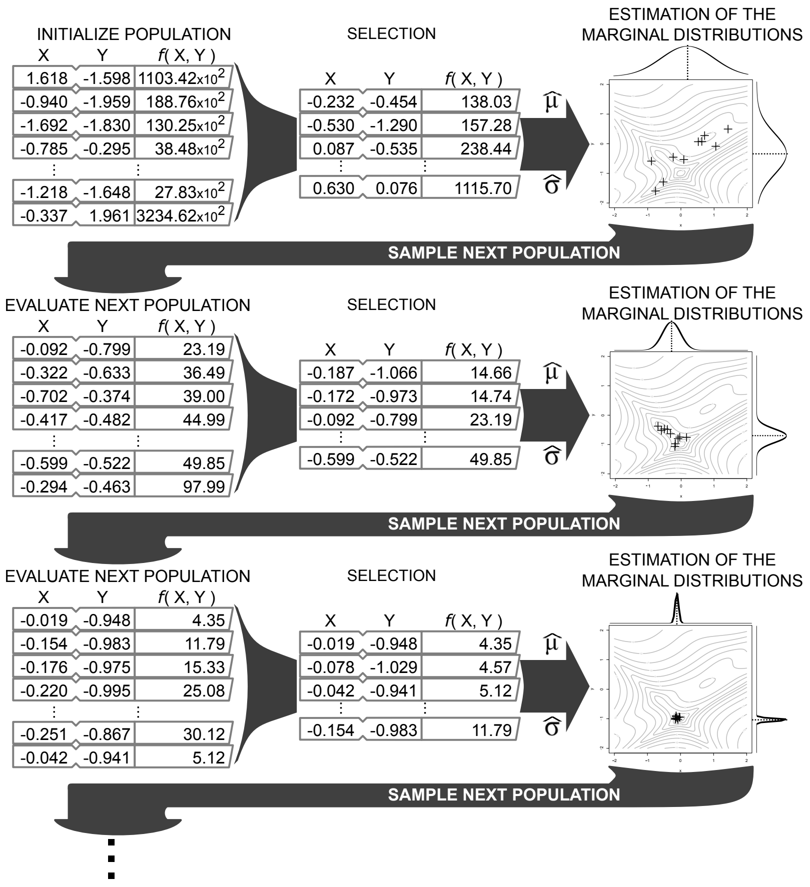


Fig. 2 Goldstein-Price function in two dimensions including level plot in the range $[-2,\ 2]$ for each independent variable.

Moreover, to solve the Goldstein-Price function, the population of the UMDA$_{Gc}$ strategy is randomly distributed in the search space and also it is encoded using the two dimensions of the problem for each individual. The numerical example of the optimization process using the real-coded UMDA$_{Gc}$ is illustrated in Fig. 3. In the illustration, the ESTIMATION OF THE MARGINAL DISTRIBUTIONS block marks each individual in S^t with a cross on the level curves of the Goldstein-Price function. The convergence to the optimal solution is exemplified as the reduction of the dispersion of the individuals within the search space throughout three generations.

The reduction of the dispersion is represented as the narrowing of the distribution curves present in the margin of the block.



INITIALIZE POPULATION

X	Y	f(X, Y)
1.618	-1.598	1103.42×10^2
-0.940	-1.959	188.76×10^2
-1.692	-1.830	130.25×10^2
-0.785	-0.295	38.48×10^2
⋮	⋮	
-1.218	-1.648	27.83×10^2
-0.337	1.961	3234.62×10^2

SELECTION

X	Y	f(X, Y)
-0.232	-0.454	138.03
-0.530	-1.290	157.28
0.087	-0.535	238.44
⋮	⋮	
0.630	0.076	1115.70

EVALUATE NEXT POPULATION

X	Y	f(X, Y)
-0.092	-0.799	23.19
-0.322	-0.633	36.49
-0.702	-0.374	39.00
-0.417	-0.482	44.99
⋮	⋮	
-0.599	-0.522	49.85
-0.294	-0.463	97.99

SELECTION

X	Y	f(X, Y)
-0.187	-1.066	14.66
-0.172	-0.973	14.74
-0.092	-0.799	23.19
⋮	⋮	
-0.599	-0.522	49.85

EVALUATE NEXT POPULATION

X	Y	f(X, Y)
-0.019	-0.948	4.35
-0.154	-0.983	11.79
-0.176	-0.975	15.33
-0.220	-0.995	25.08
⋮	⋮	
-0.251	-0.867	30.12
-0.042	-0.941	5.12

SELECTION

X	Y	f(X, Y)
-0.019	-0.948	4.35
-0.078	-1.029	4.57
-0.042	-0.941	5.12
⋮	⋮	
-0.154	-0.983	11.79

Fig. 3 Numerical example for solving the 2-D Goldstein-Price function using UMDA$_{GC}$ as optimization strategy.

3 Proposed Hybrid Method

In this section, the proposed hybrid method based on the Univariate Marginal Distribution Algorithm and Simulated Annealing is presented. In the first stage, the hybrid scheme which is the main contribution of the present work is explained in detail. Finally, the process to adapt the parametric parabola problem to be solved by the proposed method is introduced.

3.1 Hybridization of UMDA$_c$ with Simulated Annealing

The main proposal is a hybridization of UMDA$_{Gc}$ with simulated annealing (SA) as follows: at each generation the best solution from UMDA$_{Gc}$ (the elite) is used as starting point for the SA.

In the case the elite individual is improved, this solution is always inserted in the selected set and stored. In addition, for the annealing schedule, the initial temperature is computed by using the objective function values in the last population in UMDA$_{Gc}$. In the same vein, the inferior and superior search limits for all variables of the SA are computed by using the standard deviation vectors, computed in the last UMDA$_{Gc}$ iteration, as follows: $x^t_{inf} = x^t_{best} - 0.05sd^t$ and $x^t_{sup} = x^t_{best} + 0.05sd^t$. According to these limits, the search performed by the SA is performed in a small vicinity of the current best solution. In addition, the smaller the standard deviation of the current population is, the smaller the vicinity in the SA algorithm. Thus, both algorithms take advantage of each other, the UMDA$_{Gc}$ is used to provide an adequate temperature, initial point and search limits of the SA, while the SA often improves the best solution in the UMDA population, an bias the search to most promising regions.

Algorithm 3 describes the general hybrid proposal. Lines 5 and 12 shows where the SA step is, as a result of this step the selected set $\hat{S}^t$ and the elite x^{best} could be updated, if the best solution from $UMDA_{Gc}$ is improved, in such a case this new elite is inserted to the selected set $\hat{S}^t$ and, as a consequence, used to compute the UMDA$_{Gc}$ parameters, as it is shown in Line 7.

This hybrid algorithm could be applied to any optimization problem, and it requires as input the population size which is set to 30 in all our experiments reported, a maximum number of generations which is set to 30, and search limits.

For the particular case of using this algorithm for parabola detection, the search limits are computed for each target image as it is shown in Line 2, this procedure is detailed in in Section 3.2.

The flowchart of the UMDA$_{Gc}$/SA strategy is presented in Fig. 4, where the process of improve the UMDA$_{Gc}$ best solution each iteration using Simulated Annealing is highlighted. Computationally, the hybridization of the UMDA$_{Gc}$ can expect an average time overhead of up to 48.91 % and a mean memory overhead of up to 0.51 % due to the improvement of the solution using Simulated Annealing.

Algorithm 3: Hybrid UMDA$_{Gc}$/SA

```
Data: D=Number of dimensions.
      $n_{pop}$=Population size.
      $n_{gen}$=Number of generations.
      $x_{inf}$ and $x_{sup}$=Inferior and superior limits.
Result: $x_{best}$ and $f_{best}$. Optimum approximation and its objective function
        value.
1  $t = 0$;
2  $X^t \sim U(x_{inf}, x_{sup})$;
3  $F^t = Evaluate(X^t)$;
4  $[S^t, x^t_{best}, f^t_{best}] = Selection(X^t, F^t, T)$;
5  $[\hat{S}^t, x^t_{best}, f^t_{best}] = SimulatedAnnealing(x^t_{best}, f^t_{best})$;
6  for $1..n_{gen}$ do
7      $[\mu^t, \sigma^t] = ParameterEstimation(\hat{S}^t)$;
8      $X^{t+1} = [x^t_{best}, Sampling(\mu^t, \sigma^t, n_{pop} - 1)]$;
9      $t = t + 1$;
10     $F^t = Evaluate(X^t)$;
11     $[S, x^t_{best}, f^t_{best}] = Selection(X^t, F^t)$;
12     $[\hat{S}^t, x^t_{best}, f^t_{best}] = SimulatedAnnealing(x^t_{best}, f^t_{best}, T)$;
```

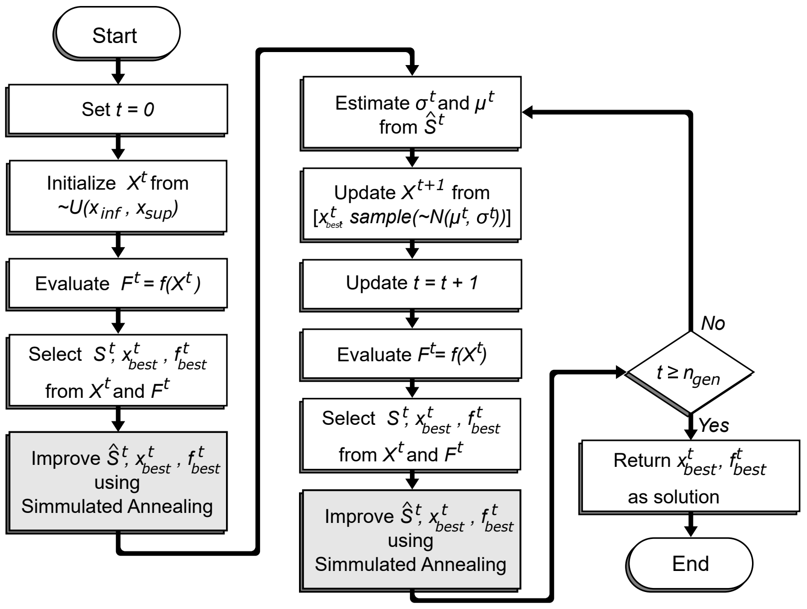

Start
Set $t = 0$
Initialize X^t from $\sim U(x_{inf}, x_{sup})$
Evaluate $F^t = f(X^t)$
Select S^t, x^t_{best}, f^t_{best} from X^t and F^t
Improve $\hat{S}^t$, x^t_{best}, f^t_{best} using Simmulated Annealing
Estimate σ^t and μ^t from $\hat{S}^t$
Update X^{t+1} from $[x^t_{best}$, sample($\sim N(\mu^t, \sigma^t)$)]
Update $t = t + 1$
Evaluate $F^t = f(X^t)$
Select S^t, x^t_{best}, f^t_{best} from X^t and F^t
Improve $\hat{S}^t$, x^t_{best}, f^t_{best} using Simmulated Annealing
$t \geq n_{gen}$
No
Yes
Return x^t_{best}, f^t_{best} as solution
End

Fig. 4 Flowchart of the algorithm of the Hybrid UMDA$_{Gc}$/SA. The highlighted process in the chart represents the use of Simulated Annealing to improve the best solution found by UMDA$_{Gc}$.

3.2 Parametric parabola detection using the proposed method

For the purpose of using this proposal for parabola detection, a candidate solution (chromosome) is represented by four variables as previously mentioned in Section 2.1, $\mathbf{x} = [x_1 = a, x_2 = b, x_3 = c, x_4 = \theta]$, and the search limits are necessary.

Algorithm 3 requires the superior and inferior search limits, our proposal considers to compute these limits for each target image automatically, the procedure assumes that the parabola is always vertically aligned, but it could be upwards or downwards, this direction is, also, automatically detected by using the following procedure:

- From the target image we randomly select 10% of the non-zero pixels.
- Using these pixels we compute a least squares parabola fitting, hence we obtain a, b and c quotients.
- Repeat the two steps above 10 times.
- Then, we get a set of 10 $[a,b,c]$ quotients. we compute the means μ_a, μ_b and μ_c and standard deviations σ_a, σ_b and σ_c. Notice that the sign of the a parameter in the equation $ax^2 + bx + c$, determines whether the parabola opens upwards or downwards. Thus, we define two cases:

  – if (μ_a <0)
    $a_{inf} = \mu_a - \sigma_a$,
    $b_{inf} = \mu_b - \sigma_b$,
    $c_{inf} = \mu_c - \sigma_c$,
    $a_{sup} = \mu_a + 4\sigma_a$,
    $b_{sup} = \mu_b + 4\sigma_b$,
    $c_{sup} = \mu_c + 4\sigma_c$,
    $x_v = -b_{sup}/(2a_{sup})$, notice that this is the $x-$coordinate of the parabola vertex using the superior limits.
    $y_v = a_{sup}x_v^2 + b_{sup}x_v + (\mu_c + 4\sigma_c)$, this is the $y-$coordinate of the parabola vertex.
    $c_{sup} = c_{inf} + (n_{row} - y_v) - 1$

  – else
    $a_{inf} = \mu_a - 4\sigma_a$,
    $b_{inf} = \mu_b - 4\sigma_b$,
    $x_v = -b_{inf}/(2a_{inf})$, notice that this is the $x-$coordinate of the parabola vertex using the inferior limits.
    $y_v = a_{inf}x_v^2 + b_{inf}x_v + (\mu_c - 4\sigma_c)$, this is the $y-$coordinate of the parabola vertex, using the inferior limits.
    $c_{inf} = (\mu_c - 4\sigma_c) - y_v + 1$,
    $a_{sup} = \mu_a + \sigma_a$,
    $b_{sup} = \mu_b + \sigma_b$,
    $c_{sup} = \mu_c + \sigma_c$.

The purpose of this procedure, is to find appropriate limits. Hence we known that a least squares fitting is suitable, nevertheless there could be pixels that are not representative of the target image, thus, that is the reason of randomly selecting 10% of them with the aim of reducing the bias given by outliers. Thus, the procedure uses the parameters from ten parabolas to determine correct limits using its mean and standard deviation. In addition, by observation we determine that, if the parabola opens downwards we must extend the superior limits and vice versa, thats the reason why set the limit 4 standard deviations in one direction and only 1 in the other one.

4 Computational experiments

In this section, the proposed hybrid method based on the Univariate Marginal Distribution Algorithm and Simulated Annealing is applied on the 20 retinal fundus images of the publicly available DRIVE database [1]. The computational experiments are performed using the Matlab software version 2016, on a computer with an Intel Core i5, 4GB of RAM, and 2.4GHz processor. To evaluate the proposed method, it is directly compared with the evolutionary technique proposed by Guerrero-Turrubiates et al. [22] based on Estimation of Distribution Algorithms in terms of execution time. Moreover, in order to apply the parabola detection methods, an automatic vessel segmentation method has to be applied. In the experiments, the method based in Gaussian matched filters proposed by Cruz-Aceves et al. [28] has been introduced.

On the other hand, the method proposed by Guerrero-Turrubiates et al. [22], uses the Univariate Marginal Distribution Algorithm as optimization strategy to solve the parabola detection problem. According to this method, the average execution time is 4.5838 seconds for the set of retinal images, and it was performed applying the following set of parameters:

Table 1 UMDA parameters for the computational experiments using the method proposed in [22].

Parameter	Value
Number of individuals	10
Selection rate	0.6
Maximum number of generations	30

Figure 5, illustrates the obtained parabola detection results applying the previously mentioned method.

The main contribution of the present chapter, is the implementation of a hybrid strategy involving the UMDA method and the heuristic of Simulated Annealing. The

[1] J. J. Staal, M. D. Abramo, M. Niemeijer, M. A. Viergever and B. van Ginneken. Ridge based vessel segmentation in color images of the retina. IEEE Transactions on Medical Imaging.23(4):501-509, 2004.

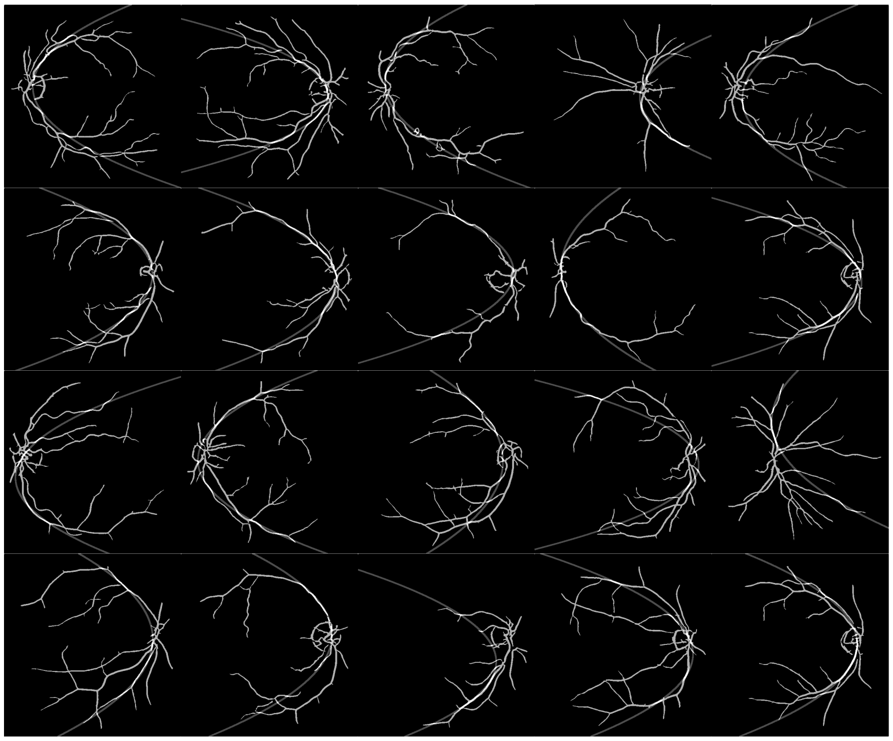


Fig. 5 Parabola detection using the method in [22] on the 20 retinal fundus images of the DRIVE Database.

UMDA method has been adopted because of the good results obtained in previous works for the parabola problem. Here, two different methods are proposed, where the main difference is the objective function to be evaluated. The detection results of both methods are presented below.

4.1 Parabola detection on retinal fundus images

In the first method we use binary images, for maximizing objective function in Equation (2). Notice that for binary images any characteristic of the parabola is as good as any other, that is to say the algorithm looks for matching any segment in the parabola independent of curvature to the target image. We perform 30 trials for the set of 20 retinal fundus images. In average, each execution lasts 1.57 seconds with an implementation in C language, using the GNU gcc compiler version 6.4. The best detection result for each set of the trials is reported in Figure 6.

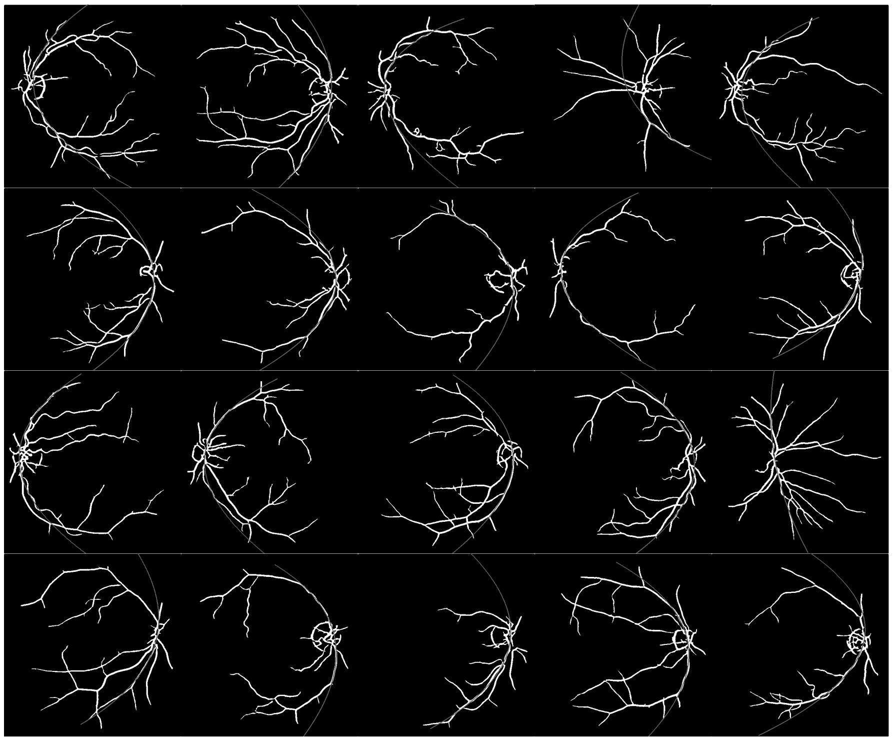


Fig. 6 Parabola detection using the Hybrid UMDA$_{Gc}$/SA on retinal fundus images of the DRIVE Database.

4.2 Weighting the parabola pixels to improve detection

The second proposed method is a manner of weighting pixels of the parabola image, in order to associate a higher importance to the most relevant characteristics. In this sense, it is of interest to find in the target image a similar region than the parabola vertex, we consider that this region close to the change in the derivative sign is the most important. For this purpose use Equation (9), notice that the denominator of the first term is the absolute value of the derivative of the parabola plus 1, we sum 1 to avoid division by zero. Thus, $\Delta x \in [1,255]$ is maximum when the derivative is 0, that is to say, when the x_i is the parabola vertex. We use the very same procedure and objective function, but for this case each parabola pixel increases with a different value the objective function. The results are shown in Figure 7.

$$\Delta x = 254\left(\frac{1.0}{|2.0ax_i + b| + 1} + \frac{1}{254}\right). \tag{9}$$

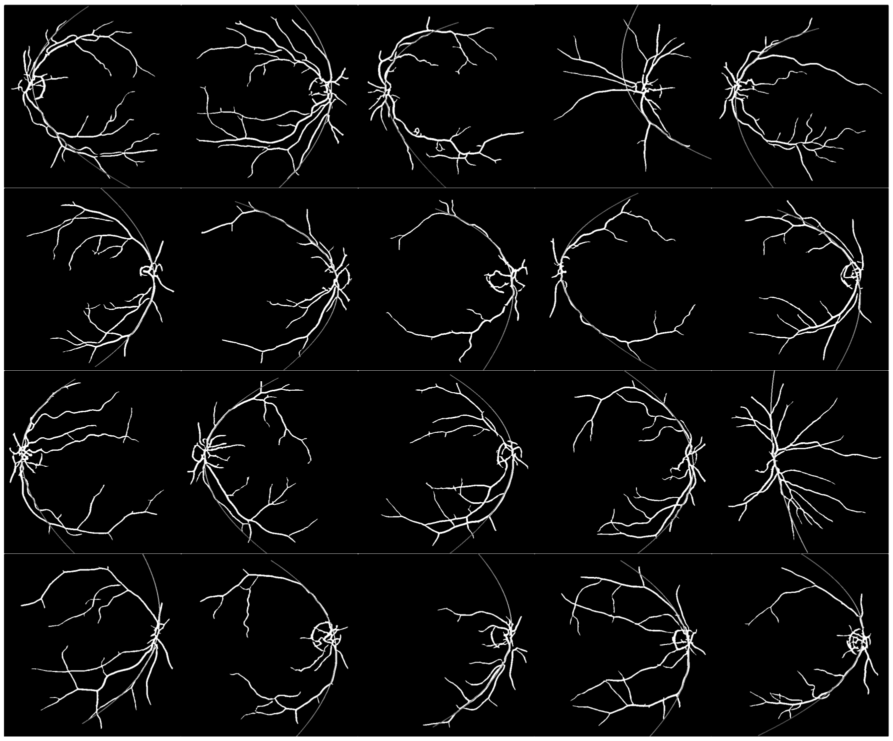


Fig. 7 Parabola detection using the second Hybrid UMDA$_{Gc}$/SA strategy on retinal fundus images of the DRIVE Database.

The quantitative analysis of the two strategies introduced in this work and the method proposed by Guerrero-Turrubiates [22] is presented in Table 2. The accuracy metric was defined as the correspondence ratio of vessel pixels superposed by the parametrized parabola. This metric was selected due to the difficulty to define a parabola ground-truth approved by an expert. According to the comparison analysis, the two hybrid strategies surpass the non-hybrid method. Moreover, the approach of weighting the parabola pixels presents the better performance with the highest correspondence ratio.

Table 2 Comparative analysis of the average accuracy using 20 images of the DRIVE database of retinal fundus images.

Method	Average accuracy
Guerrero-Turrubiates method [22]	1.3047 %
Hybrid UMDA$_{Gc}$/SA	2.4800 %
Hybrid UMDA$_{Gc}$/SA weighting pixels	**2.5288 %**

Table 3 presents the average execution time of the proposed method and the UMDA [22], MIPAV [31] and RANSAC [32] methods applied for the parametric parabola detection reported by Guerrero-Turrubiates [22]. According to the computational results, the proposed Hybrid UMDA$_{Gc}$/SA execution time is lower than the non-hybrid approaches. The highest accuracy and shortest execution time of the hybrid approaches introduced in this work make them more suitable to aid the detection of diseases in fundus images of the retina in the medical practice.

Table 3 Comparative analysis of the average execution time using the DRIVE database of retinal fundus images.

Method	Average execution time (s)
Proposed method	**1.57**
Guerrero-Turrubiates method [22]	4.5838
MIPAV [31]	43.65
RANSAC [32]	16.66

5 Concluding Remarks

In this chapter a new hybrid optimization method based on the univariate marginal distribution algorithm for continuous domains and simulated annealing has been proposed for the parabola detection problem. The proposed method was applied in the DRIVE database of retinal fundus images. The method was applied with two different objective functions; firstly, only taking into account the superposition of pixels, and secondly, by weighting the detection of the parabola vertex. Both two strategies have obtained suitable results in the approximation of the parabolic shape of the retinal images, obtaining more robust results than the comparative method. In addition, the average computational time (1.57 seconds) of the proposed method outperforms the performance of the comparative method using the database of 20 retinal fundus images, which is useful for systems that perform computer-aided diagnosis in clinical practice.

Acknowledgments

This research has been supported by the National Council of Science and Technology of México under the project: Cátedras-CONACYT 3150-3097.

Appendix A

Simulated Annealing code in C programming language used for the Hybrid UMDA$_{Gc}$/SA algorithm proposed in this work.

```
#include "isimulated_annealing.h"
int RealPerturbation(double *Xorig, double *Xpert,int nvar,
 double *xinf,double *xsup, double *params, Irand *seed)
{
  static int i;
  int j=0;
  for (j=0; j<nvar; j++)
      Xpert[j]=Xorig[j];
  if (i<0 || i>nvar-1)
       i=0;
         Xpert[i]=(unifDou(2.0,seed)-1.0)*params[i];
   if (Xpert[i]<xinf[i])
       Xpert[i]=xinf[i];
   else if (Xpert[i]>xsup[i])
       Xpert[i]=xsup[i];
  i++;
return 0;
}

int realSA_UpdateParams(int neps,int generation,
double *ParamsIn, double *ParamsOut, int nvar,
                        realSA_PERTURBATION perType, int Ns,
                        int nprocess,double cu,
                        double *xinf, double *xsup )
{
 double u;
 int i;
 for (i=0; i<nvar; i++)
 {
         u=(ParamsIn[i])/((double)Ns*(double)nprocess);
     if (u>0.6)
             ParamsOut[i]=ParamsOut[i]*(1.0+cu*(u-0.6)/0.4);
     else if(u<0.4)
             ParamsOut[i]=ParamsOut[i]/(1.0+cu*(0.4- u)/0.4);
       if (ParamsOut[i]>(xsup[i]-xinf[i]))
               ParamsOut[i]=(xsup[i]-xinf[i]);
     }
  return 0;
}
```

```
double realSA(FOBJDATA fobjdata,int nvar, double T0,
                        int Neps, int maxeval,double cu,
                        int Ns, double rT, int NT,
                        double eps, EDARESULT *result,int print,
             realSA_PERTURBATION perType,
             double *xinf, double *xsup, Irand *seed)
{
  double **Xlocal, *Xbest, *xeval;
  int nthreads=1;
  double **Xopt,**XBest;
  double *Fopt,*Fbest,fbest,*Flocal,DF,pDF,T=T0,u;
  int **nu;
  int t,s,i,p=1,accept,j,nepsU,neps=0;
  double *ParamsIn,*ParamsOut;
  ///Parameters for perturbing the solution

  ParamsIn=dvector(nvar+1);
  ///Measured parameters for inside use
  ParamsOut=dvector(nvar+1);
  ///Computed parameters for inside use
  xeval=dvector(nvar);
  XBest=dmatrix(nthreads,nvar);
  Xlocal=dmatrix(nthreads,nvar);
  Xopt=dmatrix(nthreads,nvar);
  Flocal=dvector(nthreads);
  Fbest=dvector(nthreads);
  Fopt=dvector(nthreads);
  Xbest=dvector(nvar);
  nu=imatrix(nthreads,nvar);
  ///step size
  for (i=0; i<nvar; i++)
       ParamsOut[i]=(xsup[i]-xinf[i])/2.0;

     for (i=0; i<nvar; i++)
         Xbest[i]=result->xbest[i];
     fbest=result->fbest;
  for (p=0; p<nthreads; p++){
     for (i=0; i<nvar; i++){
          XBest[p][i]=Xbest[i];
          Xopt[p][i]=Xbest[i];
          nu[p][i]=0;
     }
     Fbest[p]=fbest;
     Fopt[p]=fbest;
  }
```

```
///Depending on the perturbation method
int neval=0;
while (neps<Neps && neval<maxeval)
{
for (t=0; t<NT; t++)
{
 for (s=0; s<Ns; s++)
 {
  for (p=0; p<nthreads; p++){
   for (i=0; i<nvar; i++){
    RealPerturbation(XBest[p], Xlocal[p],nvar, xinf,xsup,
                                     ParamsOut,seed);
                Flocal[p]= Fobj(fobjdata,Xlocal[p],nvar);
                (*result).neval++;
                neval++;
                ///Storing the best overall solutions
                if (Flocal[p]>Fopt[p])
                {
                    for (j=0; j<nvar; j++)
                        Xopt[p][j]=Xlocal[p][j];
                    Fopt[p]=Flocal[p];
                }
                ///Metropolis criterion
                DF=Fbest[p]-Flocal[p];
                accept=0;
                if (DF<=0.0) accept=1;
                else{
                    pDF=exp(-DF/T);
                    u=unifDou(1.0,seed);
                    if(u<pDF)
                        accept=1;
                }
                if (accept)
                {
                        for (j=0; j<nvar; j++)
                            XBest[p][j]=Xlocal[p][j];
                        Fbest[p]=Flocal[p];
                        nu[p][i]++;
                }
                }
            }
            }
            ///Best solution found until now
            for (p=0; p<nthreads;p++)
            {
                if (Fopt[p]>fbest);
                {
                    fbest=Fopt[p];
                    for (i=0; i<nvar; i++)
                        Xbest[i]=Xopt[p][i];
                }
            }
            if (print>=2)
            {
                printf("t=␣%d␣fbest=␣%lf",t,fbest);
                for (i=0; i<nvar; i++)
                    printf("%.6g␣",Xbest[i]);
                printf("\n");
            }
            ///Adjusting parameters for the solution perturbation
            for (p=1; p<nthreads; p++)
                for (i=0; i<nvar; i++)
                    nu[0][i]+=nu[p][i];
            ///Input parametes the number of accepted solutions
            ParamsIn[0]=0.0;
            for (i=0; i<nvar; i++)
                 ParamsIn[0]+=(double)nu[0][i];
                 realSA_UpdateParams(neps,t, ParamsIn, ParamsOut,
                 nvar, perType, Ns,nthreads,cu,xinf, xsup);
            for (p=0; p<nthreads; p++)
                for (i=0; i<nvar; i++)
                    nu[p][i]=0;
        }
        T=rT*T;
        nepsU=0;
        for (p=0; p<nthreads; p++)
             nepsU+=(int)(fabs(Fbest[p]-fbest)<eps);
        neps+=(int)(nepsU==nthreads);
        for (p=0; p<nthreads; p++){
          for (i=0; i<nvar; i++){
                XBest[p][i]=Xbest[i];
                Xopt[p][i]=Xbest[i];

          }
          Fbest[p]=fbest;
          Fopt[p]=fbest;
          if (print>=1)
            printf("fbest=%.6g␣T=%lf␣neps=%d\n",fbest,T,neps);
        }
  }
```

```
  for (i=0; i<nvar;i++)
       result->xbest[i]=Xbest[i];
  result->fbest=fbest;

  On_FinalizeRutine(1);
  free(ParamsIn);
  free(ParamsOut);
  free_dmatrix(XBest,nthreads) ;
  free_dmatrix(Xlocal,nthreads) ;
  free_dmatrix(Xopt,nthreads) ;
  free(Flocal);
  free(Fbest);
  free(Fopt);
  free(Xbest);
  free(xeval);
  free_imatrix(nu,nthreads);
return fbest;
}
```

References

1. D. Ballard, “Generalizing the Hough transform to detect arbitrary shapes,” *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.
2. J. Illingworth and J. Kittler, “A survey of the hough transform,” *Computer Vision, Graphics and Image Processing*, vol. 44, no. 1, pp. 87–116, 1988.
3. V. Leavers, “Survey: which hough transform?,” *Computer Vision Graphics and Image Processing: Image Understanding*, vol. 58, pp. 250–264, 1993.
4. R. O. Duda and P. E. Hart, “Use of the hough transformation to detect lines and curves in pictures,” *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.
5. E. Davies, “A modified Hough scheme for general circle location,” *Pattern Recognition Letters*, vol. 7, pp. 37–43, 1987.
6. D. Ioannou, W. Huda, and A. Laine, “Circle recognition through a 2D Hough transform and radius histogramming,” *Image and Vision Computing*, vol. 17, pp. 15–26, 1999.
7. L. Jiang, “Efficient randomized Hough transform for circle detection using novel probability sampling and feature points,” *Optik*, vol. 123, pp. 1834–1840, 2012.
8. R. Yip, P. Tam, and D. Leung, “Modification of Hough transform for circles and ellipses detection using a 2-dimensional array,” *Pattern Recognition*, vol. 25, pp. 1007–1022, 1992.
9. F. Oloumi and R. Rangayyan, “Detection of the temporal arcade in fundus images of the retina using the Hough transform,” *Conference proceedings IEEE Engineering in Medicine and Biology Society*, vol. 1, pp. 3585–3588, 2009.
10. F. Oloumi, R. Rangayyan, and A. L. Ells, “Parabolic modeling of the major temporal arcade in retinal fundus images,” *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 7, pp. 1825–1838, 2012.
11. N. Chernov and C. Lesort, “Least squares fitting of circles,” *J. Math. Imaging Vision*, vol. 23, no. 3, pp. 239–252, 2005.
12. K. Chung and Y. Huang, “A pruning-and-voting strategy to speed up the detection for lines, circles and ellipses,” *J. Inf. Sci. Eng.*, vol. 24, no. 2, pp. 503–520, 2008.

13. 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.
14. X. Zhang, Q. Su, and Y. Zhu, “Fast algorithm for circle detection using randomized Hough transform,” *Comput. Eng. Appl.*, vol. 44, no. 22, pp. 62–64, 2008.
15. T. Chen and K. Chung, “An efficient randomized algorithm for detecting circles,” *Computer Vision and Image Understanding*, vol. 83, no. 2, pp. 172–191, 2001.
16. L. Jiang, “Fast detection of multi-circle with randomized Hough transform,” *Optimization Letters*, vol. 5, no. 5, pp. 397–400, 2009.
17. H. Cheng, Y. Guo, and Y. Zhang, “A novel Hough transform based on eliminating particle swarm optimization and its applications,” *Pattern Recognition*, vol. 42, no. 9, pp. 1959–1969, 2009.
18. E. Cuevas, D. Oliva, D. Zaldivar, M. Perez-Cisneros, and H. Sossa, “Circle detection using electro-magnetism optimization,” *Information Science*, vol. 182, no. 1, pp. 40–55, 2012.
19. 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.
20. E. Cuevas, V. Osuna-Enciso, F. Wario, D. Zaldivar, and M. Perez-Cisneros, “Automatic multiple circle detection based on artificial immune systems,” *Expert Systems with Applications*, vol. 39, pp. 713–722, 2012.
21. E. Cuevas, D. Zaldivar, M. Perez-Cisneros, and M. Ramrez-Ortegon, “Circle detection using discrete differential evolution optimization,” *Pattern Analysis and Applications*, vol. 14, no. 1, pp. 93–107, 2011.
22. J. de Jesus Guerrero-Turrubiates, I. Cruz-Aceves, S. Ledesma, J. M. Sierra-Hernandez, J. Velasco, J. G. Avina-Cervantes, M. S. Avila-Garcia, H. Rostro-Gonzalez, and R. Rojas-Laguna, “Fast Parabola Detection Using Estimation of Distribution Algorithms,” *Computational and Mathematical Methods in Medicine*, vol. 6494390, pp. 1–13, 2017.
23. I. Cruz-Aceves, J. Guerrero-Turrubiates, and J. M. Sierra-Hernandez, “Parametric object detection using estimation of distribution algorithms,” *Hybrid Intelligent Techniques for Pattern Analysis and Understanding*, vol. 1, pp. 69–92, 2017.
24. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, *et al.*, “Optimization by simulated annealing,” *science*, vol. 220, no. 4598, pp. 671–680, 1983.
25. A. Corana, M. Marchesi, C. Martini, and S. Ridella, “Minimizing multimodal functions of continuous variables with the simulated annealing algorithm,” *ACM Trans. Math. Softw.*, vol. 13, pp. 262–280, Sept. 1987.
26. P. Larrañaga and J. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer, Boston, MA, 2002.
27. M. Hauschild and M. Pelikan, “An introduction and survey of estimation of distribution algorithms,” *Swarm and Evolutionary Computation*, vol. 1, no. 3, pp. 111–128, 2011.
28. I. Cruz-Aceves, A. Hernandez-Aguirre, and S. Ivvan-Valdez, “On the performance of nature inspired algorithms for the automatic segmentation of coronary arteries using Gaussian matched filters,” *Applied Soft Computing*, pp. 665–676, 2016.
29. I. Cruz-Aceves, F. Cervantes-Sanchez, A. Hernandez-Aguirre, R. Perez-Rodriguez, and A. Ochoa-Zezzatti, “A novel Gaussian matched filter based on entropy minimization for automatic segmentation of coronary angiograms,” *Computers and Electrical Engineering*, pp. 263–275, 2016.
30. L. Lozada-Chang and R. Santana, “Univariate marginal distribution algorithm dynamics for a class of parametric functions with unitation constraints,” *Information Sciences*, vol. 181, pp. 2340–2355, 2011.
31. M. J. McAuliffe, F. M. Lalonde, D. McGarry, W. Gandler, K. Csaky, and B. L. Trus, “Medical image processing, analysis & visualization in clinical research,” in *Proceedings of the IEEE Symposium on Computer-Based Medical Systems*, pp. 381–388, 2001.
32. P. Niedfeldt and R. Beard, “Recursive ransac: Multiple signal estimation with outliers,” *IFAC Proceedings Volumes*, vol. 46, no. 23, pp. 430 – 435, 2013. 9th IFAC Symposium on Nonlinear Control Systems.

Index