Human evolutionary model: A new approach to optimization

Oscar Montiel a, Oscar Castillo b,*, Patricia Melin b, Antonio Rodríguez Díaz c, Roberto Sepúlveda a

a CITEDI-IPN, Avenida del Parque #1310, Mesa de Otay Tijuana, BC, Mexico
b Department of Computer Science, Tijuana Institute of Technology, P.O. Box 4207, Chula Vista, CA 91909, USA
c FCQI-UABC Tijuana, BC, Mexico

Received 15 September 2006; accepted 21 September 2006

Abstract

The aim of this paper is to propose the Human Evolutionary Model (HEM) as a novel computational method for solving search and optimization problems with single or multiple objectives. HEM is an intelligent evolutionary optimization method that uses consensus knowledge from experts with the aim of inferring the most suitable parameters to achieve the evolution in an intelligent way. HEM is able to handle experts' knowledge disagreements by the use of a novel concept called Mediative Fuzzy Logic (MFL). The effectiveness of this computational method is demonstrated through several experiments that were performed using classical test functions as well as composite test functions. We are comparing our results against the results obtained with the Genetic Algorithm of the Matlab’s Toolbox, Evolution Strategy with Covariance Matrix Adaptation (CMA-ES), Particle Swarm Optimizer (PSO), Cooperative PSO (CPSO), G3 model with PCX crossover (G3-PCX), Differential Evolution (DE), and Comprehensive Learning PSO (CLPSO). The results obtained using HEM outperforms the results obtained using the abovementioned optimization methods.

© 2006 Elsevier Inc. All rights reserved.

Keywords: HEM; Intelligent evolutionary algorithm; Optimization; Fuzzy adaptation

1. Introduction

Estimating optimal solutions for real-world problems such as finding the maximal possible reliability in the design of a product comprised with reducing its cost, modeling a physical system, or others, has become a necessity since there is no standard and infallible technique. Recently, optimization has received enormous attention as a direct consequence of the rapid progress in computer technology including the availability of powerful software [5].

Although, physical systems are inherently nonlinear [30], they can be modeled using a linear or a nonlinear approach [28,38]; the decision of which approach to use is done depending on the nature of the problem being
solved. Since linear models are simpler it is very common to consider them first, but if they do not yield satisfactory performance, it is necessary to consider a nonlinear one [25].

Flexibility is a key concept for obtaining good models, it is necessary to choose an appropriate model structure as well as the optimization method [18]. Most nonlinear optimization problems are solved using numerical methods; these are iterative methods; which implicitly means that it is necessary to perform several tries before the optimal solution can be reached [36]. Numerical methods for solving search problems have the fundamental task of dividing the search space in two classes: solutions and no solutions. From a big diversity of searching methods, we have to select and implement algorithms that provide us with valid solutions in the searched space for a specific problem. For being consistent, different algorithms applied to the same search space should find the same solutions. Although, from practice we know that some algorithms are more or less efficient [12].

Search problems can be classified into single objective (SO) and multiple objective (MO) optimization problems commonly refereed as SOOP and MOOP, respectively. Basically, this classification is based on the number of objective functions that should be minimized or maximized. In a broad sense, an optimization problem can be stated as follows:

\[
\begin{align*}
\text{Minimize} & \quad f_m(X) & m = 1, 2, 3, \ldots, M; \\
\text{subject to} & \quad g_j(X) \geq 0 & j = 1, 2, \ldots, J; \\
& \quad h_k(X) = 0 & k = 1, 2, \ldots, K; \\
& \quad x_i^L \leq x_i \leq x_i^U & i = 1, 2, \ldots, D.
\end{align*}
\]

where \( X \) is a vector of \( n \) decision variables given by \( X = (x_1, x_2, \ldots, x_n)^T \); \( g_j \) and \( h_k \) are constrained functions with associated \( J \) inequality constraints, and \( K \) equality constraints. The last set of constraints are called variable bounds that restrict each decision variable \( x_i \) to take a value within a lower \( x_i^L \) and upper \( x_i^U \) bounds respectively. These bounds define a decision variable space \( D \), or simply the decision space. Here, we have \( M \) objective functions \( f(X) = (f_1(X), f_2(X), \ldots, f_M(X)) \) that should be minimized or maximized.

Methods for solving SOOP and MOOP are substantially different because besides the number of objectives that each approach needs to handle, there are other differences between them. In an SOOP we have only one goal to satisfy which is to find the optimal solution, although the search space can have several local optimums, solutions, and the goal is always to find the global optimum. In contrast, in MOOP there are two goals, one is to advance toward the Pareto optimal front, the other goal is to maintain a diverse set of solutions in the non-dominated front. Another difference is that in SOOP there exists only one search space – the decision variable space – but in MOOP we have in addition the objective space that is formed by the objective functions [10].

Nowadays, there are several well-known optimization methods that are widely addressed in the literature [5, 25, 27]. The Human Evolutionary Model is an intelligent computational model for solving SO and MO search and nonlinear global optimization problems. At this time, HEM uses floating-point codification; hence we are adopting some well-known operators normally used in this type of coding [10]. HEM should learn from experts using consensus knowledge with the aim of inferring the most suitable parameters to achieve the evolution. Since human experts are part of the system it is very common that they differ about knowledge for different reasons; therefore, information might have considerable degree of disagreement that can be interpreted as a contradiction, and it can be represented by the use of agreement and non-agreement membership functions. HEM is an intelligent method that uses a novel concept called Mediative Fuzzy Logic (MFL) for handling doubtful and contradictory information [20, 21]. It was conceived with the ability of learning and/or dealing with new situations and it has attributes such as reasoning, generalization, discovering, association and abstraction with the goal of predicting and taking decisions for making easier adaptation in complex and changing environments [6, 21]. Moreover, HEM is an intuitive method, this characteristic could seem vague, but intuition is like that, when we recognize a face, nobody knows how intuition works, but it does. Evolutionary algorithms mimic biological evolution, although we know they use fitness functions for finding optimal solutions, there is not a deductive process for finding solutions, since not necessarily previous good solutions will produce new ones that are better. We know in complete detail what evolutionary algorithms do, but we do not understand how they work in the way we understand a rational process. HEM has an Adaptive Intelligent/Intuitive System (AIIS) that allows the algorithm to adapt itself using intuition as a driving force, with this we mean that although we created the necessary mechanism to achieve this adaptation, actu
ally we do not know in a rational sense how is it working. A related work about handling uncertainty in information is [37]. A different method to solve conflicts in information is given in [26].

The idea of HEM is to provide an intelligent optimization method for SOOP and MOOP. HEM has two different intelligent methods: one is for solving SOOP, and finding the ideal, utopian and nadir objective vectors that are used in MOOP [10], the second method is to find the Pareto set for MOOP. In this paper, the focus is on explaining the SOO method.

Several experiments were performed, using classical test functions (benchmark functions), like the De Jong’s test set [8], Rastrigin’s function, and others. Moreover, we used a novel composition test function suite developed by Liang et al. in [16]. This benchmark set is made up by six composed functions named “Composite function 1” (CF1) to “Composite Function 6” (CF6). The composite functions were especially designed to test optimization algorithms providing several challenges with randomly located global optima and several randomly located deep local optima. The use of these composite functions is very important for testing novel optimization algorithms because even algorithms that have demonstrated their excellent performance on standard benchmark functions have failed on all the composite functions CF1 to CF6 for 10 dimensions [16]. Some examples of such algorithms that have failed are: Evolution Strategy with Covariance Matrix Adaptation (CMA-ES) [13], Particle Swarm Optimizer (PSO) [11], Cooperative PSO (CPSO) [34,35], G3 model with PCX crossover (G3-PCX) [9], Differential Evolution (DE) [31,32]. Comprehensive Learning PSO (CLPSO) [17] works only well optimizing CF1 and it fails for CF2 to CF6.

The rest of the paper is organized as follows: Section 2 is devoted to describe the HEM model. In Section 3, a description of the test suite that was used to test this proposal is presented. In Section 4, the experiments that were made are described. In Section 5, a discussion about the results and comparisons is presented. Finally, in Section 6, the conclusions are presented.

2. The human evolutionary model

2.1. Human evolutionary model description

The HEM model can be formally defined as the following structure:

\[ HEM = (H, AIIS, P, O, S, E, L, TL/PS, VRL, POS) \]  

- \( H \)  human
- \( AIIS \) adaptive intelligent intuitive system
- \( P \) population of size \( N \) individuals
- \( O \) single or a multiple objective optimization goals
- \( S \) evolutionary strategy used for reaching the objectives expressed in \( O \)
- \( E \) environment, here we can have predators, etc
- \( L \) landscape, i.e., the scenario where the evolution must be performed
- \( TL/PS \) Tabu List formed by the best solutions found/Pareto Set
- \( VRL \) visited regions list
- \( POS \) Pareto optimal set

In Fig. 1, we have a general description of HEM containing six main blocks. In the first block, we show that the human or group of humans is part of the system. HEM is an intelligent evolutionary algorithm that learns from experts their rational and intuitive procedures that they use to solve optimization problems. In this model, we consider that we have two kinds of humans: real human beings and artificial humans. In the first block of Fig. 1 we show that real human beings form one class. In the second block, the artificial human implemented in the AIIS of HEM is shown. Humans as part of the system are in charge of teaching the artificial human all the knowledge needed for realizing the searching task. The AIIS should learn the rational and intuitive knowledge from the experts; the final purpose is that the artificial human eventually can substitute the human beings most of the times. HEM has a feedback control system formed by blocks three and four; they work coordinately for monitoring and evaluating the evolution of the problem to be solved. In the fifth block,
Fig. 1. This is a sketch of the Human Evolutionary Model (HEM). We show the main components of it. AIIS is controlling all of the parameters of this evolutionary model.

we have a single objective optimization (SOO) method for solving single objective optimization problems (SOOP). In addition, using the SOO method we can find the ideal, utopian and nadir vectors for multiple objective optimization problems (MOOP). In the sixth block, we have a multiple objective optimization (MOO) method, which is dedicated to find the Pareto optimal set (POS) in MOOP.

Emotions are an essential part of human perception, and they are closely related with the success or failure of achieving certain goal. There are several well-known optimization methods, some of them are of deterministic or stochastic nature, and they are guided by objective functions to fulfill their task. However, for all existing methods human intervention is indispensable at several stages, for example in selecting the most suitable method and initial parameters, etc. Moreover, if the selected method does not satisfy the optimization task, the human will need to select a different method. Hence, it is desirable to have an optimization method with certain human abilities to automate the optimization process. HEM, as any model, is a human conceptualization that can be separated from reality, but the model is trying to emulate the process that human experts use to select parameters for solving optimization problems.

In Fig. 2, we show a representation of HEM’s population; here \( P \) represents a population of \( N \) individuals, which size can vary through generations. A particular characteristic of individuals in HEM is that we are not only including the decision variables; also, each individual has associated other variables that are called genetic effects (GE) that will influence the searching process. An individual in HEM is composed of three parts:

1. A genetic representation \( gr \), that can be codified using binary or floating-point representation. Decision variables are codified in this part.
2. A set of genetic effects \( ge \), that are attributes of each individual such as “physical structure”, “gender”, “actual age”, “maximum age allowed”, “pheromone level”, etc. For example, the genetic attribute gender of individual \( n \) at generation \( x \) is defined as \( ge(3)_{n,x} = ge(gender)_{n,x} \), and the actual age as \( ge(4)_{n,x} \), etc. In general, we have the definition given by expression (2), where we omitted the generation number

\[
ge_n = (ge(minStr)_n, ge(maxStr)_n, ge(gender)_n, ge(actAge)_n, ge(maxAge)_n, ge(phLevel)_n, \ldots, ge(m)_n) \quad (2)
\]
The third part in the individual representation is devoted to the individual’s objective values. Objective values are codified in vectorial form, the size of the vector is determined by the number of objectives that the problem requires. For single objective (SO) problems, we will use only one \( ov \) value. For multiple objectives (MO) problems, we will use \( ov(1), \ldots, ov(M) \).

Fig. 3 shows an individual \( p_n = (gr_n, ge_n, ov_n) \) where \( gr_n = (gr(1)_n, \ldots, gr(2)_n) \) is a vector (a row) in the \( GR_{NxQ} \) matrix. The genetic effects \( ge_n \) are rows in the matrix \( GE_{NxR} \). Because HEM allows the optimization of SO and MO problems, we can have one or several objective values codified as vectors \( ov_n \), and they are rows in the matrix \( OV_{NxS} \). In this context, a population \( P \) at generation \( x \) is defined as \( P_x = (GR_x + GE_x + OV_x) \).

Using the attributes \( ge(minStr) \) and \( ge(maxStr) \) is possible to control the individuals’ structure size \( gr \) in problems where it is necessary to optimize the number of decision variable (i.e. the structure size) in addition of the objective values. For the attribute \( ge(gender) \) we have the valid set of values \{M, F, O\} for males, females and not used, respectively. We can set the maximum life expectancy using the attribute \( ge(maxAge) \). We use the attribute \( ge(phLevel) \) for keeping track of individuals that produce good successors.

The SOO method is divided in two parts, the first one corresponds to the initialization and setting up of the optimization problem, and the second one corresponds to the intelligent evolutionary process.

For the first part we have,

1. The human(s) will use an already established knowledge base of AIIS. At any time, the human(s) can interact with the evolutionary process for teaching, helping the evolution, and/or taking decisions. The main idea is that the human(s) can be able to teach the human decision making process to AIIS. The purpose is that eventually AIIS will be able to substitute the human(s).
2. Setting up of initial parameters (default parameters), for example, initial population size, upper and lower limit for population size, parameters for the recombination and mutation operators, etc.
3. Setting up the optimization problem, here we will establish the ranges for the decision variables, the quantity of objective values, the optimization criteria, and the cost functions.

<table>
<thead>
<tr>
<th>( gr )</th>
<th>( ge )</th>
<th>( ov )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genetic representation</td>
<td>Genetic effects</td>
<td>Objective values</td>
</tr>
<tr>
<td>Binary</td>
<td>Effect #1, \ldots</td>
<td>Objective value #1, \ldots</td>
</tr>
<tr>
<td>Floating point</td>
<td>Effect #n</td>
<td>Objective value #n</td>
</tr>
</tbody>
</table>

Fig. 3. Representation of one individual in \( HEM \). Each individual has a genetic representation \( (gr) \); a set of control parameters called genetic effects \( (ge) \); the evaluation of the individual can performed using one ore several objective functions, this is depending of problem’s nature, this evaluation is expressed as a set of fitness values \( (fv) \).
For the second part, we have:

1. For \( m = 1 \) to \( M \) where \( m \) is the objective function number, perform the next procedure.
2. Create an initial population \( P_0 \) of size \( N \). Here, we will create \( GR_0 \) of population \( P_0 \). In this part the genetic attributes \( GE_0 \) will be defined. Make \( GR_x = GR_0 \) and \( GE_x = GE_0 \).
3. Evaluate \( GR_x \). At this stage, we are going to evaluate \( ov(m) \), the corresponding objective value will be assigned to each individual. Here, we will obtain \( OV_x \).
4. Calculate the fitness value \( fv(m) \) of each individual in \( P_x \) and sort the whole population in ascending order. In this step, we will obtain the fitness vector \( FV_x \).
5. At each generation, the best individual \( P_{Best} \) is saved in a Tabu List (TL) as the element \( t_1 \). In this list, we can find all the solutions previously found. Repetition of individual's values is not allowed. The list is defined as \( TL = \{ t_i \}_{i=1}^{L} \), each individual \( t_i \) has a radius \( \rho \) to represent a subspace in the landscape, this radius is defined according the application precision requirements.
6. Apply to the whole population \( P_x \) the genetic operator \#1 (GO\#1). All the genetic operators work over the genetic effects \( GE_x \) of HEM. In this case, \( GO\#1 \) will add “1” to the genetic effect “age” of each individual of the whole population.
7. Apply predation operator \#1 (Pred\#1) to the population \( P_x \). This operator verifies the actual age of each individual in \( P_x \), and it will eliminate all the individuals that are over this age. Moreover, this operator will eliminate the amount of individuals suggested by AIIS via the variables “Create” and “Delete”. To achieve this elimination process, the individuals with less fitness in the actual population will be selected.
8. The task of the genetic operator \#2 (GE\#2) is to mutate some of the genetic effects of the individuals. Functionally, the most evident is to mutate the genetic effect “gender”.
9. Predator \#2 (Pred\#2) will analyze the actual population to verify the gender balance; it is desirable to know if there is a numeric balance or a valid proportion of male and female individuals. If the population is balanced the algorithm will not perform any predation. If the population is unbalanced the operator will proceed to balance it. For achieving the balance, the algorithm will select randomly as many individuals as necessary of the biggest population and it will change their gender. We preferred to make this change instead of really eliminating the individuals because we could eliminate very good individuals slowing down the evolution.
10. Select individuals according to a specific rule. We are using the pyramidal selection rule (see Fig. 4). This rule will tell us that the new population \( P \) at generation \( x + 1 \) should be formed using a combination of individuals from several sources. The pyramidal form is related with the quantity of individuals to be selected.

![Pyramidal selection rule](image)

**Fig. 4.** Pyramidal selection rule. Individuals to generate the new population are selected using this criterion. The base of the pyramid represents the biggest quantity of individuals to be selected.
selected; for example, the base is the biggest part of the pyramid and it is representing the biggest quantity of determined class of individuals to be selected, in this case a percentage of the best individuals of the actual population $p_1(x)$. In the next level, at the middle, we have a percentage of individuals with high pheromone level $p_2(x)$. At the top, we have individuals selected from other optimization methods $p_3(x)$. In this way, we are allowing a natural hybrid functionality.

11. Utilize the genetic operator #3 (GO#3) to increase the pheromone level of the best individuals (base of pyramid) that were selected in step 10.

12. Create a new population of size $actPs$ (actual population size) using the subpopulation formed by the selected individuals at step 10.
   a. Parent selection is made according to what was programmed in the genetic effect “gender”. If we selected to use a population of male and female individuals, then we have to select randomly one individual of each classification to perform the recombination. On the other hand, if we are not considering the gender, then we have to select two individuals from the same population. We used the Extended intermediate recombination operator given by Eq. (3) [22,24]. One strategy that gives good results is to achieve the parent’s selection with a probability of 80% of being of different sex (class), the rest can be of the same sex, and even we can select the same individual twice. Randomly, create and assign the genetic effects to the new individual.
   b. Apply the mutation operator to the genetic representation (gr) of the new individual. We are using the mutation operator defined by Eq. (4).
   c. Apply Predator #3 (Pred#3) to the recently created individual. This operator is dedicated to eliminate all the individuals that are out of the valid search space. This operator can be programmed in two ways: One is to eliminate individuals, which parametric values $gr$ went out the search space; this is a drastic solution because we can eliminate individuals that are very near to the optimum. A better solution is to adjust (correct) the decision variables that are out of range in such a way that the corresponding values fall in the nearest boundary.
   d. Calculate the objective values of the new individual.

13. Considering the best individual of the last generations (we used the last 10 generations), calculate the variance value of these individuals using their fitness value. We are using the variable “Variance” to store this value.

14. We are using the variable “Cycling” to indicate the number of generations that the evolution has not had any significant change. We are measuring this change using the variable “Variance”, if the obtained variance value is less than 0.05 in the last generations (10 generations), then increase the value of the variable “Cycling”, otherwise reset its value.

15. Using AIIS calculate the amount of individuals that we have to eliminate, as well as the amount of individuals that we have to create.

16. If AIIS determined that it is necessary to create more individuals, then we have two ways to create them. AIIS will determine the best option.
   a. Randomly, create a quantity of new individuals, this quantity is calculated by AIIS. This option is very fast and the best option at the first stages because in average is very likely that the new generated individuals do not repeat the same search subspace, i.e. they do not fall in a previously visited point.
   b. Create an individual out of the tabu regions, i.e., an individual that has not been included in the Tabu List $TL$ and/or in the Visited Region List ($VRL$). Note that always at the first time $VRL$ is empty, and it is filled with individual that has been selected as valid individual to complete the population size inferred by AIIS, this is to avoid repeating the selection of the same regions; i.e., individuals that previously were created and selected. The $VRL$ uses the same value of $\rho$ that was used in step 5 for $TL$. The $VRL$ is a complementary list of $TL$, and the use of both is very important since they allow the algorithm to perform an exhaustive search in the whole landscape.

17. Repeat steps 4–17, until we fulfill the termination criterion.

18. Save the decision variables for the current objective function, as well as the corresponding $ov$.

19. End for. (Of step 1).
20. Continue with the next objective, i.e. \( m = m + 1 \); otherwise proceed with step 21.
21. End.

We are showing in Fig. 5 an abstraction of the general idea for controlling the population size and the progress of evolution by eliminating and creating new individuals in an intelligent way. In essence, this method uses two variables as inputs to an inference system, which in turn, it will decide the quantity of individuals to create and the quantity of individuals to eliminate. Although, the inference system can be implemented using traditional fuzzy logic, we used Mediative Fuzzy Logic (MFL) because can handle doubtful and contradiction in the experts' knowledge. In Fig. 5, from line 1 to 5, we can see that the method needs an initial population size \((N)\). The upper bound \((ub)\) and the lower bound \((lb)\) are used to limit the population size. The maximum number of generations \((maxGen)\) that the algorithm can run and the "actual population size" are given in lines 4 and 5. Lines 6–31 are repeated "maxGen" times. In line 7, we have a box indicating that can contain an evolutionary algorithm, and of course, we are using the method described above. After 12 generations (this is an heuristic value), the algorithm calculates for each generation the variance value of the last 10 best individuals obtained from the last ten generations (also, this is an heuristic value), and the result is assigned to the variable "Variance" (lines 8–10). This method is using the variable “Cycling” for keeping track the number of times that the algorithm is still obtaining fitness values without a significant change, for calculating this value the algorithm uses the “Variance” value. It verifies whether this value is less than 0.05 (heuristic value); if so,
then the variable "Cycling" is incremented (line 12), otherwise the variable is reset (line 14). The MFL inference system will calculate the quantity of individuals that should be created and deleted, the result is assigned to the variable "y_Axis" (line 16); using this value, in line 17 the algorithm obtains an integer value of individuals to create, the result is assigned to the variable "Create". Similarly, in line 18 the quantity of individuals that have to be deleted are calculated, and the resulting value is assigned to the variable "Delete". Lines 19-25 are dedicated to delete the quantity of individuals contained in the variable Delete. This elimination is made provided the actual population size still being greater than the lower bound value (lb). Using line 21 the worst individual in the population is eliminated, line 22 updates de quantity of individuals to be eliminated, and in line 23 the actual population size (variable "actPs") is updated. Lines 26-32 are used in the process of creating new individuals, the quantity is indicated in the variable "Create". The algorithm will create new individuals as long as the actual population size does not exceed the upper bound (ub). A new individual can be created at random, using a tabu like method, and/or using a combination of both methods. After that, the variables "Create" and "actPS" are updated. Note that new population is still being created in line 7, in other words, the evolutionary process creates the new population using the common procedures for selection, recombination and mutation. The difference stems in that this method exploits determined region with the less individuals as can be possible, for speeding up the process; of course, diversity is lost but we do not take care for it, because HEM is able to advance the evolution when the algorithm has exploited exhaustively the region, i.e., we cannot get better fitness values.

2.2. Genetic operators to modify the genetic representation

2.2.1. Recombination operator

We are using the Extended Intermediate Recombination (EIR) operator. To use this operator, we are considering that we selected two parents \(X = (x_1, \ldots, x_n)\) and \(Y = (y_1, \ldots, y_n)\) to obtain the successor \(Z = (z_1, \ldots, z_n)\). Classically, for this operator the parents are selected such that \(X \neq Y\). Because, the process of testing this last condition takes time, we obtained an 80% of individuals using the original approach, we separated the individuals in two classes, male and females, and we selected one individual of each class to achieve the recombination. For the remaining 20%, we allowed to mating individuals of the same class. The EIR operator is given by Eq. (3)

\[
z_i = x_i + \alpha_i(y_i - x_i), \quad i = 1, \ldots, n
\]  

(3)

where \(\alpha_i \in [-d, 1 + d]\) is chosen with uniform probability for each \(i\), and \(d \geq 0\), a good choice is \(d = 0.25\), which is the one that was used.

2.2.2. Mutation operator

The goal of the mutation operator is to modify one or more parameters of \(z_i\), the modified objects (i.e., the offsprings) that appear in the landscape within a certain distance of the unmodified objects (i.e. the parents). In this way, a coefficient \(z'\) of the offspring \(Z\), is given by

\[
z' = z_i \pm \text{range}_i \cdot \delta
\]  

(4)

where \(\text{range}_i\) defines the mutation range, and it is calculated as

\[
\text{range}_i = \lambda \cdot \text{searchinterval}_i
\]  

(5)

In the Discrete Mutation operator (DM), \(\lambda\) is normally set to 0.1 or 0.2 and it is very efficient with some functions, but also we can set it to 1. This is the case of the Broad Mutation Operator (BM) proposed in [29] to solve problems where the distance between the optimum and the actual position is larger than the DM operator could reach. We will have in one step, that the mutation operator can reach points from \(z_i\) only within the distance given by \(\pm \text{range}_i \cdot \delta\). The sign + or – is chosen with probability of 0.5. The variable \(\delta\) is computed by

\[
\delta = \sum_{i=0}^{k-1} \varphi_i 2^{-i}, \quad \varphi_i \in \{0, 1\}
\]  

(6)
Before mutation is applied, we set each $\varphi_i$ equal to 0, and then each $\varphi_i$ is mutated to 1 with probability $p_\delta = \frac{1}{\delta}$, and only $\varphi_i = 1$ contributes to the sum. On the average there will be just one $\varphi_i$ with value 1, say $\varphi_j$; this is done for each coefficient, in this way several coefficients of the same vector might be mutated. Then $\delta$ is given by [7,23]

$$\delta = 2^{-1}$$  

In formula (6), $K$ is a parameter originally related to the machine precision, i.e. the number of bits used to represent a real variable in the machine that we are working with, traditionally $K$ uses values of 8 and 16. In practice, however, the value of $K$ is related to the expected number of mutation steps, in other words, the higher the value of $K$ is, the more fine-grained is the resultant mutation operator [2].

### 2.3. Mediative fuzzy logic

MFL is a novel method for performing inference in presence of doubtful and contradictory information [20,21]. MFL was proposed as an extension of traditional fuzzy logic and includes Intuitionistic fuzzy Logic (IFL) in the Atanassov sense [1,4,15,33].

A set $C$ in MFL is given by

$$C = \{ (x, \mu_C(x), \nu_C(x)) | x \in X \}$$  

where $\mu_C: X \rightarrow [0,1]$ and $\nu_C: X \rightarrow [0,1]$ are the agreement and non-agreement membership functions.

In the same way as in intuitionistic fuzzy logic in the Atanassov sense, for a fuzzy set $C$ we have a “hesitation margin” $\pi_C(x)$ defined by equation

$$\pi_C(x) = 1 - \mu_C(x) - \nu_C(x)$$  

where

$$0 \leq \mu_C(x) + \nu_C(x) \leq 1$$  

and $\mu_C(x), \nu_C(x) \in [0,1]$ denotes a degree of membership and a degree of non-membership of $x \in C$, respectively [3].

A contradiction fuzzy set $C$ in $X$ is given by

$$\zeta_C(x) = \min(\mu_C(x), \nu_C(x))$$  

we can say that for each $x \in X$ we have a “contradiction index” value $\zeta_C$ that indicates the degree of contradiction in the knowledge for the $x$ value. For calculating the inference at the system’s output, we are using

$$MFS = \left( 1 - \pi - \frac{\zeta}{2} \right) FS_{\mu} + \left( \pi + \frac{\zeta}{2} \right) FS_{\nu}$$  

where $FS_{\mu}$ is the inference at the output using only the agreement membership functions, and $FS_{\nu}$ is the output using only the non-agreement membership functions.

### 2.4. The artificial intelligent intuitive system

The control of population size is achieved via the AIIS and it is feedback method based in MFL. We preferred to use MFL in the inference system because can handle the knowledge in a more robust way since MFL can make inferences taking into account doubtful and contradictory knowledge. Moreover, if there is no contradiction or doubtful, the MFL system auto-reduces itself to behave as a traditional or intuitionistic fuzzy logic system.

Basically, at present time, the MFL method in HEM uses two variables. The variable “Variance” to control the variance value, and the variable “Cycling” to control the number of generations that the population remains in the same landscape region and/or without any significative change in the variance measurement. For the first one, we use the fitness value of the best individual of previous generations (we used 10 generations), to calculate their variance. For measuring the degree of cycling, at each generation we calculate the variable “Variance” and we make an increment in the variable “Cycling” each time the variable “Variance” is below a predetermined threshold value, we fixed this value to 0.05; otherwise reset the variable “Cycling”, i.e. “Cycling = 0”. The threshold value was obtained experimentally, testing several complex test functions.
The intelligent system will calculate the quantity of individuals that should be eliminated from the actual population, as well as the individuals that should be created in the actual population. We have two choices for creating individuals, the first one is to create individuals randomly, the second way is to use a Tabu search method for avoiding the tabu regions and create the new individual out these regions. Always, the operation of deletion is performed first, then we proceed to create the new individuals.

We used two Sugeno inference systems of order 0, one for the agreement side, and the other one for the non-agreement side. In Table 1, we are summarizing the linguistic variables that we used in the fuzzy system. Each inference system has two output variables named “Delete” and “Create”. The values of these two variables are related to the amount of individuals that should be eliminated or created. Each output variable has three constant terms, which are: “little”, “regular”, and “many”. We assigned the values of “0”, “0.5”, and “1”, respectively. We used the same name and values at the output of each inference system (agreement and non-agreement).

At the agreement side we have the following 16 rules:

1. If (Cycling is SmallC) and (Variance is SmallV) then (Create is little)(Delete is little)
2. If (Cycling is SmallC) and (Variance is RegularV) then (Create is regular)(Delete is little)
3. If (Cycling is SmallC) and (Variance is LargeV) then (Create is regular)(Delete is regular)
4. If (Cycling is RegularC) and (Variance is SmallV) then (Create is many)(Delete is regular)
5. If (Cycling is RegularC) and (Variance is RegularV) then (Create is regular)(Delete is regular)
6. If (Cycling is RegularC) and (Variance is LargeV) then (Create is regular)(Delete is regular)
7. If (Cycling is LargeC) and (Variance is SmallV) then (Create is many)(Delete is many)
8. If (Cycling is LargeC) and (Variance is RegularV) then (Create is many)(Delete is regular)
9. If (Cycling is LargeC) and (Variance is LargeV) then (Create is little)(Delete is little)
10. If (Cycling is SmallC) and (Variance is XLargeV) then (Create is little)(Delete is regular)
11. If (Cycling is RegularC) and (Variance is XLargeV) then (Create is regular)(Delete is little)
12. If (Cycling is LargeC) and (Variance is XLargeV) then (Create is little)(Delete is little)
13. If (Cycling is XLargeC) and (Variance is SmallV) then (Create is many)(Delete is many)
14. If (Cycling is XLargeC) and (Variance is RegularV) then (Create is many)(Delete is many)
15. If (Cycling is XLargeC) and (Variance is XLargeV) then (Create is many)(Delete is many)
16. If (Cycling is XLargeC) and (Variance is LargeV) then (Create is many)(Delete is many)

At the non-agreement side we have the following four rules:

1. If (Cycling is nLargeC) and (Variance is nLargeV) then (Create is little)(Delete is little)
2. If (Cycling is nLargeC) and (Variance is nSmallV) then (Create is little)(Delete is little)
3. If (Cycling is nSmallC) and (Variance is nLargeV) then (Create is many)(Delete is many)
4. If (Cycling is nSmallC) and (Variance is nSmallV) then (Create is regular)(Delete is regular)

Table 1
Name, type, and parameters of the linguistic variables for each Fuzzy Inference System (FIS)

<table>
<thead>
<tr>
<th>FIS name</th>
<th>Variable name</th>
<th>Term name</th>
<th>Type</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIISa (FS0)</td>
<td>Cycling</td>
<td>SmallC</td>
<td>Trapezoidal</td>
<td>[-1, -1, 1.5, 30]</td>
</tr>
<tr>
<td>AIISa (FS0)</td>
<td>Cycling</td>
<td>RegularC</td>
<td>Triangular</td>
<td>[10, 30, 50]</td>
</tr>
<tr>
<td>AIISa (FS0)</td>
<td>Cycling</td>
<td>LargeC</td>
<td>Triangular</td>
<td>[30, 50, 70]</td>
</tr>
<tr>
<td>AIISa (FS0)</td>
<td>Cycling</td>
<td>XLargeC</td>
<td>Trapezoidal</td>
<td>[50, 70, 5e+15, 5e+15]</td>
</tr>
<tr>
<td>AIISa (FS0)</td>
<td>Variance</td>
<td>SmallV</td>
<td>Trapezoidal</td>
<td>[-1, -1, 0.04, 0.2]</td>
</tr>
<tr>
<td>AIISa (FS0)</td>
<td>Variance</td>
<td>RegularV</td>
<td>Triangular</td>
<td>[0, 0.2, 0.4]</td>
</tr>
<tr>
<td>AIISa (FS0)</td>
<td>Variance</td>
<td>LargeV</td>
<td>Triangular</td>
<td>[0.2, 0.4, 0.6]</td>
</tr>
<tr>
<td>AIISa (FS0)</td>
<td>Variance</td>
<td>XLargeV</td>
<td>Trapezoidal</td>
<td>[0.4, 0.6, 5e+15, 5e+15]</td>
</tr>
<tr>
<td>AIISa (FS0)</td>
<td>Cycling</td>
<td>nLargeC</td>
<td>Trapezoidal</td>
<td>[-1, -1, 10, 60]</td>
</tr>
<tr>
<td>AIISna (FS)</td>
<td>Cycling</td>
<td>nSmallC</td>
<td>Trapezoidal</td>
<td>[10, 60, 5e+15, 5e+15]</td>
</tr>
<tr>
<td>AIISna (FS)</td>
<td>Variance</td>
<td>nLargeV</td>
<td>Trapezoidal</td>
<td>[-1, -1, 0.05, 1]</td>
</tr>
<tr>
<td>AIISna (FS)</td>
<td>Variance</td>
<td>nSmallV</td>
<td>Trapezoidal</td>
<td>[-1, -1, 0.09, 1.5e+15]</td>
</tr>
</tbody>
</table>

In the agreement side, we are identifying the agreement side using (AIISa), and the non-agreement side with AIISna.
The agreement and non-agreement membership function were established by two different experts. At the agreement side, for example, considering rule 1, we have that for small values of the variables "Cycling" and "Variance" it is necessary to "Delete" a "little" quantity of individuals and then "Create" a "little" quantity of individuals. In other words, if AIIS has begin to detect that the algorithm in the last generations has not had a significative increment in the fitness value of the best individuals then is necessary to "Delete" a small quantity of individuals (the less fit individuals), then "Create" a small quantity of individuals to increase the explorative skills of the algorithm. If "Variance" remains being "Small" the value of the variable "Cycling" is increased (rule 4 at the agreement side) and the algorithm should "Delete" more individuals (a regular quantity) and "Create" many individuals, and so on.

2.5. Generating the new population

HEM has three ways to generate new individuals, they are:

1. The basic method is to use a subpopulation of individuals that were selected with the pyramidal rule (step 10 of HEM), then the new individuals are obtained applying the recombination and mutation operators (steps 12a and 12b of HEM), this part is very similar to others EAs; the difference is the use of the operator Pred#3 that basically is repairing individuals with decision variables out of a valid range (step 12c of HEM).

2. AIIS can determine whether is necessary to create new individuals using random search (step 16a). The quantity of new individuals is variable and depends on the degree of cycling that the algorithm is having. Moreover, AIIS determines the quantity of individuals in the actual population that we have to eliminate. AIIS will select the less fit individuals in the population to eliminate them. The new population will be formed by a combination of individuals generated using steps 1 and 2 of this Section 2.5; also, a possibility is that new populations will be formed only by individuals randomly generated.

3. The third way is more complex and time consuming but it is very efficient when we could not advance in evolution using steps 1 and 2 of this section. Similarly than in step 2, AIIS will determine the amount of individuals to create and eliminate, but the new individuals will be created using tabu regions for avoiding create a successor in this region. HEM basically is using a Tabu List (TL) where the best solutions at each generation is saved, and a Visited Region List where HEM is saving the already visited regions in the process of creating new successors [14]. The TL is defined as \( TL = \{t_i\}_{i=1}^L \), in this list we are saving the whole individual \( p_m \). VRL is defined by \( VRL = \{(\zeta_i, \rho_i, \varphi_i)\}_{i=1}^V \) where \( V \) is the number of all visited regions; \( \zeta_i \) is the center of a visited region which is a sphere with radios \( \rho_i \); and the frequency of visiting this region is represented by \( \varphi_i \). Step 2 is faster than step 3, and it is an option for going out traps.

3. Benchmark functions description

As it was mentioned in the introduction, several classical benchmark functions as well as a novel composite test functions suite specially designed for testing optimization algorithms were used. Next, we are giving a brief description of them.

3.1. Classical test functions

The Sphere function (F1) is quadratic, continuous, convex, and unimodal. For \( D \) dimensions the minimum is 0 at the origin; i.e. \( x^* = [0_1, 0_2, \ldots, 0_D] \). This function provides an easily analyzable first test for the optimization algorithm.

The Rosenbrock function (F2) is quadratic, continuous, non-convex, unimodal. For \( D \) dimensions, this function has its global minima at \( x^* = [1_1, 1_2, \ldots, 1_D] \). This function is considered as a nightmare for most of the optimization algorithms because it has deep parabolic valley along the curve \( x_{i+1} = x_i^2 \). Algorithms that are not able to discover good directions underperform in this problem.

The Step function (F3) is discontinuous, non-convex, and unimodal. This function is used as a representation of problems with flat surfaces that are considered as obstacles for optimization algorithms, because they
do not give any information about which is the feasible direction. The main idea of this function is to make the search more difficult by introducing small plateaus to the topology. Using the range \([-5.12, 5.12]\), the minimum of this function for two dimensions is \(-10\); similarly, for four dimensions the minimum is \(-20\).

The Quartic function (F4) is quadratic, continuous, convex, unimodal padded with Gaussian noise. The fact of introducing noise to the function causes that the algorithm never gets the same value on the same point. Algorithms that do not work well optimizing this function will work poorly on surfaces with noisy data.

The Foxholes function (F5) is continuous, non-convex, non-quadratic, two-dimensional with 25 local minima located approximately at the point \(\{(a_{1j}, a_{2j})\}_{j=1}^{25}\), and the value at the points \((a_{1j}, a_{2j})\) is approximately \(c_j\). De Jong in his work defined the next values for \(a_{ij}\),

\[
[a_{ij}] = \begin{bmatrix}
-32 & -16 & 0 & 16 & 32 & -32 & -16 & \cdots & 0 & 16 & 32 \\
-32 & -32 & -32 & -32 & -16 & -16 & \cdots & 32 & 32 & 32 \\
\end{bmatrix}
\]

where \(c_j = j\) and \(K = 500\).

### Table 2

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
<th>Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere function F1</td>
<td>(F_1(X) = \sum_{i=1}^{N} x_i^2)</td>
<td>(-10 &lt; x_i &lt; 10)</td>
</tr>
<tr>
<td>Rosenbrock function F2</td>
<td>(F_2(X) = \sum_{i=1}^{N-1} (100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2))</td>
<td>(-2.048 &lt; x_i &lt; 2.048)</td>
</tr>
<tr>
<td>Step function F3</td>
<td>(F_3(X) = \sum_{i=1}^{N}</td>
<td>x_i</td>
</tr>
<tr>
<td>Quartic function F4</td>
<td>(F_4(X) = \sum_{i=1}^{N-1} x_i + \text{GAUSS}(0, 1))</td>
<td>(-1.28 &lt; x_i &lt; 1.28)</td>
</tr>
<tr>
<td>Foxholes F5</td>
<td>(\frac{1}{F_5(X)} = \frac{1}{K} + \sum_{j=1}^{25} \frac{1}{f_j(X)})</td>
<td>(-65.536 &lt; x_i &lt; 65.536)</td>
</tr>
<tr>
<td></td>
<td>where (f_j = c_j + \sum_{i=1}^{2} (x_i - a_{ij})^6)</td>
<td></td>
</tr>
<tr>
<td>Rastrigin F6</td>
<td>(F_6(X) = A \cdot n + \sum_{i=1}^{N} x_i^2 - A \cdot \cos(w \cdot x_i))</td>
<td>(-10 &lt; x_i &lt; 10)</td>
</tr>
<tr>
<td>Weierstrass's function F7</td>
<td>(F_7(X) = 20 \sum_{i=1}^{D} \left(\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k(x_i + 0.5))]\right) - D \sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k \cdot 0.5)])</td>
<td>(a = 0.5, b = 3, k_{max} = 20, x \in [-0.5, 0.5]^D)</td>
</tr>
<tr>
<td>Griewank's function</td>
<td>(f(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos \left(\frac{x_i}{\sqrt{2}}\right) + 1)</td>
<td>(x \in [-100, 100]^D)</td>
</tr>
<tr>
<td>Ackley's function</td>
<td>(f(x) = -20 \exp \left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2}\right) - \exp \left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + e)</td>
<td>(x \in [-32, 32]^D)</td>
</tr>
</tbody>
</table>

For all the functions \(N\) is the number of dimensions, in F3, \([x_i]\) represents the greatest integer less than or equal to \(x_i\); in F6, \(A\) is the amplitude and \(w\) the frequency.

We tested functions F1 and F6 using a range of \([-10, 10]\).
Rastrigin's function (F6) is continuous, non-convex, and multimodal. It is a fairly difficult problem for evolutionary algorithms due to the large search space and large number of local minima. The surface of the function is determined by the variables $A$ and $w$, which control the amplitude and frequency of the modulation, respectively. The global minima is located at $x^* = [0,0,\ldots,0]$.

3.2. Composite test functions

The basic set of composite test functions are composed by 10 basic functions ($f_{1-10} = f_1 \cdots f_{10}$) for 10-D (Dimensions), the search range for all the functions is $[-5,5]^D$. The basic set is given by the Sphere, Rastrigin, Weierstrass, Griewank, and Acley's functions, see Table 2 for the specific formulas. In [16] a method is given for constructing these Composite Functions.

![Fig. 6. 2-D Composite Function 1 (CF1).](image)

![Fig. 7. 2-D Composite Function 2 (CF2).](image)
In general we have that the multimodal Composite Function CF1 (Fig. 6) was constructed using 10 unimodal sphere functions \( f_{1-10} \), this resulting function has one global optimum and nine local optima.

CF2 (Fig. 7), and CF3 (Fig. 8) were constructed using 10 Griewank’s and Rastrigin’s functions \( f_{1-10} \), respectively. Because they have more complex multimodal functions, localizing their local optima becomes more complex. For this function the global optimum is very difficult to reach even when the global optimum area has been found.

CF4, CF5, and CF6 are all called hybrid functions because they were constructed with different basic functions.

To construct CF4 (Fig. 9) we have: \( f_{1-2}(x) \): Acley’s function, \( f_{3-4}(x) \): Rastrigin’s function, \( f_{5-6}(x) \): Weierstrass function, \( f_{7-8}(x) \): Griewank’s function, \( f_{9-10}(x) \): Sphere function.
CF5 (Fig. 10) is composed by the functions: \( f_{1-2}(x) \): Rastrigin’s function, \( f_{3-4}(x) \): Weierstrass function, \( f_{5-6}(x) \): Griewank’s function, \( f_{7-8}(x) \): Acley’s function, \( f_{9-10}(x) \): Sphere function.

CF6 (Fig. 11) is composed by the functions: \( f_{1-2}(x) \): Rastrigin’s function, \( f_{3-4}(x) \): Weierstrass function, \( f_{5-6}(x) \): Griewank’s function, \( f_{7-8}(x) \): Acley’s function, \( f_{9-10}(x) \): Sphere function.

The construction of composite functions requires a methodology, like the one given in [16].

4. Experimental results

Several experiments were performed, and they were classified in two groups: experiments with classical test functions, and experiments with composite test functions.
For all the tests the HEM model was used, as was described in Section 2. The HEM model was not allowed to work in hybrid mode because the idea is to test the functionality of the method without the intervention of any other algorithm. New individuals were created using the procedures explained in Section 2.5 (sections 1 and 2). For achieving the recombination process, we selected a percentage of the best individuals (20%), and a percentage of individuals with highest pheromone level (2%) (value of the genetic effect $ge(phLevel)$ of each individual). Others specific HEM’s parameters are given in the corresponding tests in Tables 4–6.

All the tests were done in a Pentium 4 computer (DELL Optiplex GX620) with Windows XP platform. All the programs were tested in Matlab programming language version 7.1.

4.1. Group 1. Experiments with classical test functions

We performed experiments using the De Jong’s test function suite, as well as the Rastrigin function. In Table 2 we show the formula for each function (F1 to F6). We compared our results against the results obtained using a Genetic Algorithm (GA). We decided to use the Genetic Algorithm of Matlab’s Toolbox to perform the comparison because it provides an easy way to reproduce the results presented in this work. In Table 3 are the selected options for the Matlab’s GA.

We made comparisons of processing time and average precision for each test function and for each method. Each test was conducted in the next form:

a. For each test function (F1 to F6), we selected the most suitable parameters for the GA for achieving a good statistical result for 100 runs for the corresponding function.

b. We executed the GA 100 times and we saved the best fitness value (maximal fitness), the minimum fitness value, and the statistical values of mean, median and standard deviation achieved in the 100 runs.

c. We chose the parameters of HEM in such a way to obtain a notable difference in precision in favor of HEM. We used statistical t-Student test to ensure that statistical processes are different. In the same way than in step c, we saved the results of the 100 tests.

d. We compare the average time of both algorithms, and we obtained the statistical $t_0$ value [19].

Steps “a” to “d” were done for two and eight variables (2 and 8 dimensions). In Tables 4 and 5, we are showing a comparison between the results obtained using HEM and the GA of Matlab for the six abovementioned test functions F1 to F6. In Table 4 we used two decision variables, and eight variables for Table 5. In all of the cases, we executed each algorithm 100 times for obtaining statistical meaningful results. In these

Table 3
We used these options in the GA Toolbox of Matlab for testing the Test Suite of Table 2

<table>
<thead>
<tr>
<th>Options</th>
<th>InitialPopulation</th>
<th>InitialScores</th>
<th>InitialPenalty</th>
<th>PenaltyFactor</th>
<th>CreationFcn</th>
<th>FitnessScalingFcn</th>
<th>SelectionFcn</th>
<th>CrossoverFcn</th>
<th>HybridFcn</th>
<th>Display</th>
<th>PlotFcns</th>
<th>OutputFcns</th>
<th>Vectorized</th>
<th>TolFun</th>
<th>TolCon</th>
</tr>
</thead>
<tbody>
<tr>
<td>PopulationType</td>
<td>'doubleVector'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PopInitRange</td>
<td>[2 x 1 double]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PopulationSize</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EliteCount</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CrossoverFraction</td>
<td>0.8000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MigrationDirection</td>
<td>'forward'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MigrationInterval</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MigrationFraction</td>
<td>0.2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generations</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TimeLimit</td>
<td>Inf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FitnessLimit</td>
<td>-Inf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StallGenLimit</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StallTimeLimit</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TolFun</td>
<td>1.0000e-006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TolCon</td>
<td>1.0000e-006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 4
Two dimensions

<table>
<thead>
<tr>
<th>Generations</th>
<th>F1 HEM</th>
<th>F1 GA</th>
<th>F2 HEM</th>
<th>F2 GA</th>
<th>F3 HEM</th>
<th>F3 GA</th>
<th>F4 HEM</th>
<th>F4 GA</th>
<th>F5 HEM</th>
<th>F5 GA</th>
<th>F6 HEM</th>
<th>F6 GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>100</td>
<td>60</td>
<td>200</td>
<td>40</td>
<td>500</td>
<td>500</td>
<td>50</td>
<td>200</td>
<td>40</td>
<td>200</td>
<td>40</td>
<td>200</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Age</td>
<td>10</td>
<td>20</td>
<td>300</td>
<td>100</td>
<td>20</td>
<td>300</td>
<td>10</td>
<td>300</td>
<td>10</td>
<td>40</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>Ni</td>
<td>600</td>
<td>1000</td>
<td>50</td>
<td>600</td>
<td>100</td>
<td>600</td>
<td>100</td>
<td>600</td>
<td>100</td>
<td>600</td>
<td>100</td>
<td>600</td>
</tr>
<tr>
<td>Nub</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Maximum fitness</td>
<td>4.20e9</td>
<td>7.5e4</td>
<td>5.25e10</td>
<td>1.72e6</td>
<td>-10</td>
<td>-6</td>
<td>116.71</td>
<td>85.28</td>
<td>1.002</td>
<td>1.002</td>
<td>1.3e9</td>
<td>6.1e3</td>
</tr>
<tr>
<td>Minimum fitness</td>
<td>133.35</td>
<td>15.63</td>
<td>121.38</td>
<td>145.83</td>
<td>-10</td>
<td>8</td>
<td>1.01</td>
<td>1.00</td>
<td>1.002</td>
<td>1.002</td>
<td>1.229</td>
<td>5.4</td>
</tr>
<tr>
<td>Mean value</td>
<td>1.12e8</td>
<td>1.66e3</td>
<td>5.44e8</td>
<td>3.19e4</td>
<td>-10</td>
<td>0.65</td>
<td>7.58</td>
<td>5.87</td>
<td>1.002</td>
<td>1.002</td>
<td>1.897</td>
<td>580.17</td>
</tr>
<tr>
<td>Median</td>
<td>3.14e5</td>
<td>255.01</td>
<td>1.24e5</td>
<td>2.83e3</td>
<td>-10</td>
<td>-5</td>
<td>2.29</td>
<td>2.30</td>
<td>1.002</td>
<td>1.002</td>
<td>4.60e5</td>
<td>110.17</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>5.79e8</td>
<td>7.78e3</td>
<td>5.25e9</td>
<td>1.78e5</td>
<td>0</td>
<td>6.25</td>
<td>17.30</td>
<td>10.90</td>
<td>2.59e11</td>
<td>1.79e4</td>
<td>1.31e8</td>
<td>1.15e3</td>
</tr>
<tr>
<td>Average time (s)</td>
<td>0.131</td>
<td>0.145</td>
<td>0.92</td>
<td>0.98</td>
<td>0.26</td>
<td>0.52</td>
<td>0.135</td>
<td>6.647</td>
<td>0.866</td>
<td>1.211</td>
<td>0.206</td>
<td>0.871</td>
</tr>
<tr>
<td>Time difference (s)</td>
<td>0.0138</td>
<td>0.06</td>
<td>0.26</td>
<td>6.512</td>
<td>0.345</td>
<td>0.665</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statistic t-test (t0)</td>
<td>-0.9240</td>
<td>-0.9359</td>
<td>17.0348</td>
<td>-1.924</td>
<td>0</td>
<td>-1.4447</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We show results of the optimization of the De Jong’s test suit (FI–F5), and the Rastrigin function for two decision variables. We use as optimization methods, the HEM and the GA from the Matlab’s toolbox. In the first column we are showing the variables values that we use for both evolutionary algorithms; note that the GA does not require some of them because they are specific for HEM.

### Table 5
Eight dimensions

<table>
<thead>
<tr>
<th>Generations</th>
<th>F1 HEM</th>
<th>F1 GA</th>
<th>F2 HEM</th>
<th>F2 GA</th>
<th>F3 HEM</th>
<th>F3 GA</th>
<th>F4 HEM</th>
<th>F4 GA</th>
<th>F5 HEM</th>
<th>F5 GA</th>
<th>F6 HEM</th>
<th>F6 GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>250</td>
<td>5000</td>
<td>5200</td>
<td>100</td>
<td>5000</td>
<td>2000</td>
<td>1000</td>
<td>130</td>
<td>800</td>
<td>50</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>Age</td>
<td>8</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ni</td>
<td>50</td>
<td>50</td>
<td>200</td>
<td>800</td>
<td>20</td>
<td>1000</td>
<td>100</td>
<td>500</td>
<td>50</td>
<td>300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nub</td>
<td>600</td>
<td>250</td>
<td>200</td>
<td>200</td>
<td>100</td>
<td>800</td>
<td>800</td>
<td>600</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nlb</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum fitness</td>
<td>3.64e4</td>
<td>193.41</td>
<td>21.72</td>
<td>13.91</td>
<td>-40</td>
<td>-34</td>
<td>295.13</td>
<td>41.51</td>
<td>9.31e5</td>
<td>81.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum fitness</td>
<td>47.65</td>
<td>13.13</td>
<td>0.9230</td>
<td>0.9015</td>
<td>-36</td>
<td>-22.5</td>
<td>0.9994</td>
<td>0.9962</td>
<td>557.90</td>
<td>2.20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean value</td>
<td>3.78e3</td>
<td>45.92</td>
<td>13.62</td>
<td>0.9015</td>
<td>-39.41</td>
<td>-20.5</td>
<td>6.4299</td>
<td>4.6885</td>
<td>5.92e4</td>
<td>13.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td>1.24e3</td>
<td>34.84</td>
<td>13.57</td>
<td>0.5707</td>
<td>-40</td>
<td>-20.5</td>
<td>2.2034</td>
<td>2.1450</td>
<td>1.69e4</td>
<td>9.69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard deviation</td>
<td>6.80e3</td>
<td>33.20</td>
<td>3.39</td>
<td>1.54</td>
<td>1.0833</td>
<td>6.45</td>
<td>29.69</td>
<td>7.3404</td>
<td>1.26e5</td>
<td>12.40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average time (s)</td>
<td>0.4481</td>
<td>0.3970</td>
<td>263.74</td>
<td>286.85</td>
<td>1.1359</td>
<td>0.658</td>
<td>20.70</td>
<td>24.465</td>
<td>2.561</td>
<td>3.53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time difference (s)</td>
<td>0.1489</td>
<td>23.11</td>
<td>† (The GA found values very far the optimal value)</td>
<td></td>
<td>3.7650</td>
<td>0.9692</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statistic t-test (t0)</td>
<td>-5.4930</td>
<td>-34.15</td>
<td>60.34</td>
<td>-5.4930</td>
<td>-4.71</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Optimization of the De Jong’s test suite (F1–F4), and the Rastrigin function for eight decision variables. We use as optimization methods, the HEM and the GA from the Matlab’s toolbox. In this case, we do not optimize F5 since it is defined for two dimensions. Here, † means that the comparison does not apply because the GA could not find the optimal value, or the value founded is far from the optimal.

Tables, in the first column, the first five rows are parameters that we have to provide to the evolutionary algorithms, the next eight rows are statistical results. So, for the first column we have the next labels and their corresponding meaning: “Generations” refers to a fix number of generations that the algorithm run; “Age” is the maximal age allowed for an individual; “Ni” is the initial population size, “Nub” and “Nlb” are the upper and lower bounds for the population size, respectively; the “Maximum fitness”, “Mean”, “Median”, “Standard deviation”, show the statistical result that we obtained for 100 independent runs for each function. Next, we have the “Average time” of these 100 runs, and finally we show the $t$-Student test. Note that for the GA some cells are cancelled by a diagonal line, this is because the GA do not need to use these parameters, they are only useful for HEM.
In this group of tests for the recombination operator we used $d = 0.25$ to calculate $\alpha$ of Eq. (3); and for the mutation operator we used $\lambda = 0.1$ in Eq. (5) for all the cases, except for the Foxholes function where we used a value of 0.5 for this variable.

4.2. Group 2. Experiments with composite test functions

We considered as benchmark target values the results reported in [16] for the algorithms PSO, CPSO, CLPSO, CMA-ES, G3-PCX, and DE. In this work the best objective value reported for the composite test function is $5.7348 \times 10^{-8}$, so we used that value as the stopping criteria for HEM in this group of tests, a

<table>
<thead>
<tr>
<th>Table 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 dimensions Composite Functions optimization using HEM</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Generations</td>
</tr>
<tr>
<td>Age</td>
</tr>
<tr>
<td>Ni (initial population size)</td>
</tr>
<tr>
<td>Nub (population's upper bound)</td>
</tr>
<tr>
<td>Nlb (population’s lower bound)</td>
</tr>
<tr>
<td>Minimum objective value (best)</td>
</tr>
<tr>
<td>Maximum objective value</td>
</tr>
<tr>
<td>Mean value</td>
</tr>
<tr>
<td>Median</td>
</tr>
<tr>
<td>Standard deviation</td>
</tr>
<tr>
<td>Average time (s)</td>
</tr>
<tr>
<td>Success relation</td>
</tr>
<tr>
<td>Average of population size</td>
</tr>
</tbody>
</table>

We obtained statistics results running 20 times HEM for each composite function.

<table>
<thead>
<tr>
<th>Table 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparative results</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>HEM</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>PSO</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>CPSO</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>CLPSO</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>CMA-ES</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>G3-PCX</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>DE</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

HEM outperforms the results obtained with the others methods. The over all best optimization result obtained with determined algorithm (row) for the corresponding composition function (column) is highlighted (bold) and underlined. In addition, we highlighted the best result obtained with HEM for CF1 and CF6 because they are as good like the obtained with CLPSO for CF1, and DE for CF6. We underlined the best results obtained with any other algorithm different of HEM. Note: Optimization results for PSO, CPSO, CLPSO, CMA-ES, G3-PCX, and DE were taken from [16] as benchmarks.
second criteria was the number of generations. In Table 6, the first five rows are for the specific parameters values used in HEM to achieve each test. In the following six rows we collected information for 20 runs for each composite function CF1 to CF6 for 10 dimensions (Figs. 6–11), this is for being consistent with the result of the tests reported in [16]. We included one row labeled as “Success relation” for indicating a relation of successful test vs. number of runs. The last row is for annotating the average population size in 20 runs that each test required.

In Table 7, we are comparing the results obtained with HEM for the composite functions against the results reported in [16]. To achieve this comparison HEM’s statistics from Table 6 was used.

In this group of tests for the recombination operator a value of \( d = 0.25 \) was used to calculate \( \alpha \) of Eq. (3); and for the mutation operator \( \lambda = 0.1 \) was used in Eq. (5) for all the cases.

5. Discussion

In this section, a discussion of the performed experiments is presented. The discussion was divided in two groups: “Group 1” for classical test function, and “Group 2” for the composite test functions.

5.1. Group 1. Classical test functions

We have that always HEM works better than the GA of the Matlab’s Toolbox. The best times and precision finding the minimizer for this group of functions were obtained using HEM.

In Tables 4 and 5, we can appreciate that the best fitness values and the lower processing times were achieved using HEM. We selected the parameters of HEM for obtaining a better statistical behavior than the one obtained using the GA, this with the aim of making comparisons of time, we noted that always the processing time and precision of HEM outperformed the results obtained using the GA of Matlab. We performed many tests using 8, 16 and 32 variables and always we obtained better results using REM. We noticed that the more difficult is the problem the better results were obtained using HEM. The Foxholes function (F5) is a difficult function that tests the explorative capacity of evolutionary algorithms, also in this test REM worked better, but we observed that in this case HEM always needed a mutation step size of 0.5, instead of 0.1 like in the other cases.

We conducted additional tests for 16, 32 and 64 decision variables for testing functions F1 to F6 of Table 2. For example, in Figs. 12 and 13, we show results of the optimization of a Sphere function with 64 decision

![Typical fitness plot for the Sphere function with 64 decision variables when it is optimized using HEM.](image)
variables. Fig. 12 is the fitness plot, and Fig. 13 shows the dynamics of population size for this problem. Note that in these figures how the population size increases when AIIS detects that the evolutionary process is losing fitness and it is getting stucked.

5.2. Group 2. Composite test functions

According to the results reported in [16], CLPSO was the only optimization method that was able to find the minimizer of CF1 but fails with the other functions, CF2 to CF6. The methods PSO, CPSO, CMA-ES, G3-PCX, and DE failed finding the minimizers of all the functions CF1 to CF6. All the tests were made for 10 dimensions. It is known that these methods have demonstrated their excellent performance on standard benchmark functions, the problem was that the composite function CF1 to CF6 are very complex.

On the other hand, with HEM we had the following results:

1. For CF1 we obtained a success relation of 20/20, i.e. a success ratio of 100%, which means that the algorithm in 20 runs always found the minimizer. The average population size for these tests was 49 individuals.
2. In CF2 we had a success relation of 19/20, then we had a success ratio of 95%. The average population size was of 475 individuals.
3. In CF3 we had a success relation of 5/20, then the success ratio is of 25%.
4. CF4 was the most difficult function to optimize using HEM, we had a success relation of 0/20, but a remark is that comparatively we got better results than the other methods.
5. For CF5 we had a success relation of 4/20, so the success ratio is 20%. The average population size was of 286 individuals.
6. For CF6 we had a success relation of 0/20. Only DE found a better mean value than HEM, the difference is minimal, and HEM has a smaller standard deviation. DE did not found the global minimizer.

Note that for CF1 we obtained an average population of 49 individuals although the initial population size was 600 individuals with an upper and lower bounds of 600 and 100 individuals respectively. This result is because we are not using an strict lower bound value, this can be better understood analyzing the pseudo-code of Fig. 5, between lines 19 and 20.

In Table 7, we are highlighting and underlying the best result among the methods. Note that for CF1 we also highlighted the value of HEM although CLPSO obtained a better result for this function. The reason is
that HEM also found the minimizer with a very good approximation to zero, in this sense HEM works as well as CLPO finding the minimizer for the composite function CF1. In CF6 we highlighted the statistic values of CF6 because we considered that they are as good as those obtained with DE since differences are minimal.

We executed the algorithms 300 generations for CF1 and CF3, and 500 generations for CF2, CF4, CF5, and CF6. HEM found solutions before the corresponding number of generations were achieved, for example for testing CF4, the best values were found around generation 50, so the time for finding this solution is more or less 200 seconds instead 1194 seconds. We have similar conditions in the other experiments. In Fig. 14 we are showing a typical fitness plot for the composite function CF2 for 10 decision variables, and in Fig. 15 we have its corresponding plot showing the dynamic behavior of population size for this test.

![Fitness behavior](image1)

**Fig. 14.** Typical fitness plot for the composite function CF2 with 10 decision variables when it is optimized using HEM.

![Dynamics of population size](image2)

**Fig. 15.** Typical dynamics of population size for the composite function CF2 with 10 decision variables when it is optimized using HEM.
6. Conclusions

The use of composite test functions was very important since these functions are hard to optimize, we had to reconsider and make a general review of each step of HEM as previously proposed in [20], for example, we were considering one operator for cloning the best individual in the population. This operator works great in our evolutionary model with classical test functions, but using the composite test function we realized that this operator was sticking the evolution.

HEM uses an intelligent strategy to “explore, exploit, …, exploit, explore, …” so we can say that HEM always searches for a promissory region for being exploited. AIIS will maintain or increase the population size; once a convenient region is found, generally HEM reduces its population size with the purpose of speeding up the algorithm, this is done taking care of no diminishing the performance of the evolution; and when AIIS determines that this region has been exploited enough, then AIIS decides to increase the population size by generating new individuals randomly and/or using a tabu regions for avoiding falling in the same subspace, this is for being more explorative. In general, HEM has several mechanisms that work synergetically for performing search in an intelligent way. Hence, HEM is an intelligent evolutionary algorithm that uses knowledge from experts, and it is able to handle doubt and contradiction among human experts. An interesting thing that we observed is that almost always the dynamics of the population size for the same type of problem is behaves very similar.

In general, according the comparisons that we made, we can conclude that HEM outperformed the evolutionary algorithms PSO, CPSO, CLPSO, CMA-ES, G3-PCX and DE for the composite test functions CF1, CF2, CF3, CF5 and CF6.

We have demonstrated that having an intelligent evolutionary algorithm with the ability of learning from a group of experts is an interesting and promising idea. Future work may focus on improving the AIIS and increasing its knowledge data base with the aim of controlling more parameters of this evolutionary model.

Acknowledgements

The authors thank Comisión de Operación y Fomento de Actividades Académicas del I.P.N., and Instituto Tecnológico de Tijuana for supporting our research activities.

References


