Journal of Global Optimization **19:** 265–289, 2001.

Reliability and Performance of UEGO, a Clustering-based Global Optimizer ⋆

PILAR M. ORTIGOSA[1], I. GARCíA[1] and MÁRK JELASITY[2]
[1]*Computer Architecture & Electronics Department, University of Almería, Cta. Sacramento SN, 04120 Almería, Spain (e-mail: ortigosa@ual.es)*
[2]*Research Group on Artificial Intelligence MTA-JATE, Szeged, Hungary*

Abstract. UEGO is a general clustering technique capable of accelerating and/or parallelizing existing search methods. UEGO is an abstraction of GAS, a genetic algorithm (GA) with subpopulation support, so the niching (i.e. clustering) technique of GAS can be applied along with any kind of optimizers, not only genetic algorithm. The aim of this paper is to analyze the behavior of the algorithm as a function of different parameter settings and types of functions and to examine its reliability with the help of Csendes' method. Comparisons to other methods are also presented.

Key words: Global optimization, Stochastic optimization, Evolutionary algorithms

1. Introduction

UEGO stands for *Universal Evolutionary Global Optimizer*. Though this method is not 'evolutionary' in the usual sense, we have kept the name for historical reasons. The predecessor of UEGO was GAS, a steady-state genetic algorithm with subpopulation support. GAS offers a solution to the so-called niche radius problem which is a common problem of many simple niching techniques such as *fitness sharing* (Deb, 1989; Deb and Goldberg, 1989), *simple iteration* or the *sequential niching* Beasley et al. (1993). This problem is related to functions with multiple locals that are unevenly spread throughout the search space. The solution of GAS involves a 'cooling' technique, which enables the search to focus on the promising regions of the space, starting off with a relatively large radius that decreases as the search proceeds. In multimodal optimization problems where the objective function has multiple local optima it may be useful to ensure that the optimizer does not waste its time exploring the same region several times and is able to visit new promising regions where an optimum may exist. This goal can be achieved by applying a non-overlapping set of clusters which define sub-domains for the applied optimizer. Based on the results of the optimizer, the search process can be directed towards smaller regions by creating a new set of non-overlapping clusters that consists of smaller sub-domains. This process is a kind of cooling method similar to simulated

⋆ This work was supported by the Ministry of Education of Spain (CICYT TIC99-0361)

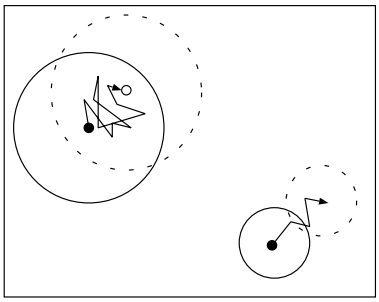

Figure 1. Concept of species.

annealing. A particular cluster is not a fixed part of the search domain; it can move through the space as the search proceeds. The non-overlapping property of the set of clusters is maintained however. For more details on GAS the reader should consult Jelasity and Dombi (1998).

In Jelasity (1998) an introduction to the history, motivation behind developing UEGO and its evaluation for a combinatorial problem is given. The common part of UEGO with GAS is the *cluster-management* (or species creation) mechanism and the *cooling* method. However, the *species creation* and *cooling* mechanism has been logically separated from the actual optimization algorithm, so it is possible to implement any kind of optimizers that work 'inside a species'. This allows the adaptation of the method to a large number of possible search domains using existing domain specific optimizers while enjoying the advantages of the old GAS-style subpopulation approach. In this paper, an algorithm called SASS, proposed by Solis and Wets Solis and Wets (1981), has been used as the optimizer algorithm.

The paper is organized as follows: Section 2 contains a short description of UEGO, the optimization algorithm; Section 3 presents the methodology that has been used to analyze the performance of UEGO with respect to its user-given parameters, and hence, a robust parameter setting can be presented. Section 4 is devoted to testing the reliability of UEGO using a robust parameter setting. Section 5 shows comparisons to other methods. Finally, in Section 6, the performance of UEGO is tested using a wide set of known test functions.

2. Description of UEGO

In this section the basic concepts, the algorithm, and the setting of the parameters are outlined. In UEGO, a domain specific optimizer (i.e. SASS) has to be implemented. Wherever we refer to 'the optimizer' in the paper we mean this optimizer.

2.1. BASIC CONCEPTS

A key notion in UEGO is that of a **species**. A species would be equivalent to an `individual` in a usual evolutionary algorithm. A species can be thought of as a

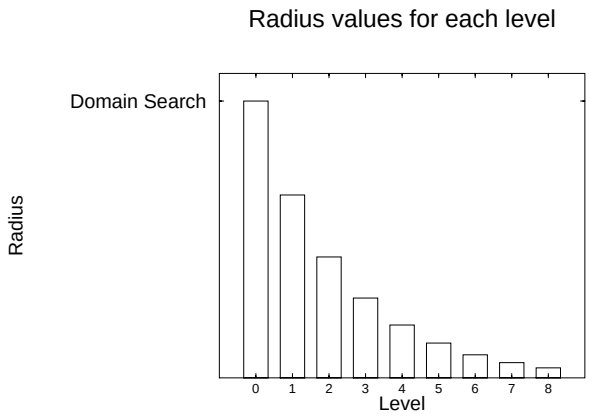

Figure 2. Radius values for the levels based on an exponentially decreasing function.

window (sphere) on the whole search space (see Figure 1). This window is defined by its *center* and a *radius*. The center is a solution, and the radius is a positive number. Of course, this definition assumes a *distance* defined over the search space. The role of this window is to 'localize' the optimizer that is always called by a species and can 'see' only its window, so every new sample is taken from there. This means that any single step made by the optimizer in a given species is no larger than the radius of the given species. If the value of a new solution is better than that of the old center, the new solution becomes the center and the window is moved while it keeps the same radius value.

The radius of a species is not arbitrary; it is taken from a list of decreasing radii, the **radius list** that follows a `cooling schedule` (see Figure 2), in such a way that given the smallest radius and the largest one (r_l and r_1) the remaining radii are expressed by the exponential function

$$r_i = r_1 \left(\frac{r_l}{r_1} \right)^{\frac{i-1}{l-1}} \quad (i = 2, \ldots, l). \tag{1}$$

The first element of this list is always the diameter of the search space. If the radius of a species is the ith element of the list, then we say that the *level* of the species is i.

The parameter **levels** indicates the maximal number of levels in the algorithm, i.e. the number of different 'cooling' stages. Every level i (i.e. for levels from [1,**levels**]) has a radius value ($\mathbf{r_i}$) and two maxima on the number of function evaluations (f.e.) namely $\mathbf{new_i}$ (maximum f.e. allowed when creating new species) and $\mathbf{n_i}$ (maximum f.e. allowed when optimizing individual species).

During the optimization process, a list of species is kept by UEGO. This concept, **species-list**, would be equivalent to the term `population` in an evolutionary algorithm. UEGO is in fact a method for managing this **species-list** (i.e. creating,

deleting and optimizing species). The maximal length of the species list is given by **max_spec_num** (`maximum population size`).

2.2. THE ALGORITHM

The UEGO algorithm for maximizing a multimodal function has the following structure:

Begin UEGO
  init_species_list
  optimize_species(n_1)
  for i = 2 to **levels**
    Determine r_i, new_i, n_i
    create_species(new_i/length($species\_list$))
    fuse_species(r_i)
    shorten_species_list(max_spec_num)
    optimize_species(n_i/max_spec_num)
    fuse_species(r_i)
  end for
End UEGO

Init_species_list:
A new species list consisting of one species with a random center at level 1 is created.

Create_species (evals): For every species in the list, random trial points in the 'window' of the species are created, and for every pair of trial points the objective function is evaluated at the middle of the *section* connecting the pair (see Figure 3). If the objective function value of the middle is worse than the values of the pair, then the members of the pair are inserted in the species list. Every newly inserted species is assigned the actual level value (**i**). As a result of this procedure the species list will eventually contain several species with different levels (hence different radii). The motivation behind this method is to create species that are on different 'hills' so ensuring that there is a valley between the new species. For unimodal problems members are never added to the species list so a single species is maintained. The parameter of this procedure (evals) is an upper bound of the number of function evaluations. Note that this algorithm needs a definition of section in the search space. In terms of genetic algorithms, it could be thought that, in this procedure, a single parent (species) is used to generate offspring (new species), and all parents are involved in the procedure of generating offspring.

Fuse_species (radius): If the centers of any pair of species from the species list are closer to each other than the given radius, the two species are fused (see Figure 4). The center of the new species will be the one with the better function value while the level will be the minimum of the levels of the original species (so the radius will be the larger one).

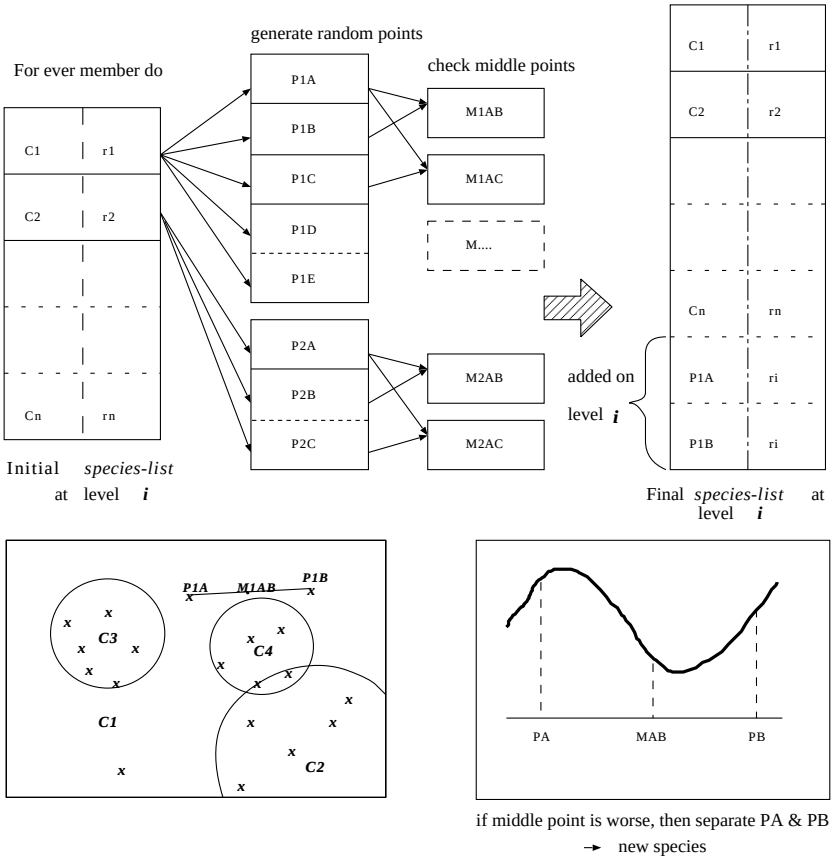

Figure 3. Creation procedure.

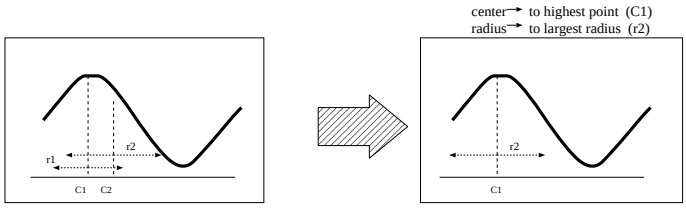

Figure 4. Fusion procedure.

Shorten_species_list(max_spec_num): It deletes species to reduce the list length to the given value. Higher level species are deleted first, therefore species with bigger radii are always kept. For this reason one species at level 1 whose radius is equal to the diameter of the search domain always exists, making it possible to escape from local optima. In the implementation used in the paper no special method is applied for selecting the species to delete, except the above constraint.

Optimize_species(budget_per_species): Execute the optimizer (in this paper: SASS) for every species with a given number of evaluations (budget_per_species)

(see Figure 1). At level i the budget_per_species is n_i/max_spec_num, so this budget depends on the max_spec_num (maximum species number or maximum population size).

It is clear that if for some level i the species list is shorter than the allowed maximal length, **max_spec_num**, the overall number of function evaluations will be smaller than n_i. Therefore the number of function evaluations tends to decrease as **max_spec_num** increases whenever the number of found species is smaller than this maximal length. Genetic algorithms (even for GAS) typically run until a maximum number of function evaluations has been computed whereas UEGO may terminate before this limit has been reached simply because it has executed all of its levels.

2.3. PARAMETERS OF UEGO

In UEGO the most important parameters are those defined at each level: the radii (r_i) and the function evaluations numbers for species creation (new_i) and optimization (n_i). These parameters are computed from some user-given parameters that are easier to understand:

evals (N): The maximal number of function evaluations the user allows for the whole optimization process. It could be called as *Whole Budget*. Note that the actual number of function evaluations may be less than this value.

levels (l): The maximum number of levels, i.e. the number of cooling stages.

max_spec_num (M): The maximum length of the species list or the maximum allowed population size.

min_r (r_l): The radius that is associated with the maximum level, i.e. **levels**.

Discussing the algorithm for computing the parameters from these user given values in detail are out of the scope of this paper. The reader should consult Jelasity (1998) to understand the justification of the following equations. The basic idea is that species move in the space at a given *speed* ($v(r)$) (distance per function evaluations) which depends on the applied search algorithm and the level of the given species. The *speed* in a level i can be computed as:

$$v(r_i) = \frac{\binom{n}{\frac{n-1}{2}}}{2^{n+1}} \cdot r_i \quad (i = 2, \ldots, l) \tag{2}$$

Using this notion, we make sure that even when the length of the species list is maximal, the species at different levels can explore the same volume of the search space.

In the creation mechanism it must be ensured that even if the length of species list is maximal, there is a chance of creating at least two more species for each old species. It also makes a strong simplification that all the evaluations should be set

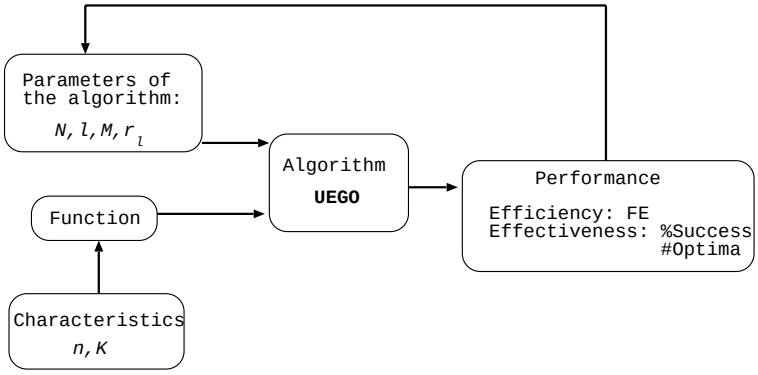

Figure 5. Experimental methodology.

to the same constant value. Equation 3 shows how new_i is computed.

$$new_i = 3M \quad (i = 2, \ldots, l) \tag{3}$$

The number of function evaluations in the optimization process n_i at every level i can be expressed by Equation 4, where ν is a threshold that directly controls the distance a species is allowed to cover.

$$n_i = \frac{r_1 \nu M}{v(r_i)} \quad (i = 2, \ldots, l) \tag{4}$$

Let us define $new_1 = 0$ for the sake of simplicity since new_1 is never used by UEGO. The decomposition of N results in the trivial equation

$$\sum_{i=1}^{l} new_i + \sum_{i=1}^{l} n_i = (l-1)3M + \sum_{i=2}^{l} \frac{r_1 \nu M}{v(r_i)} = N. \tag{5}$$

From (4) and (5) parameters ν and n_i can be computed.

3. Testing Experiment Settings

In this section, experimental results on real functions will be presented. For real functions the optimizer used by UEGO was the derivative-free and stochastic hill climber suggested in Solis and Wets (1981) (SASS), where the parameter ρ_{ub}, that controls the maximum step size was set to the value of the radius of the species from which the optimizer is called; and the accuracy of the search was set to $\min(\rho_{ub}/10^3, 10^{-5})$. No fine-tuning of the parameters of the optimizer was done.

Due to the stochastic nature of UEGO, all the numerical results given in this work are average values of hundred executions.

Our experimental methodology (see Figure 5) can be split into two stages: the first stage of training is intended to determine the values of the free parameters of

Table 1. Type and number of maxima of the four test functions

	Y1	Y2	Y3	Y4
Type	$[0,1]^2 \to R$	$[0,1]^2 \to R$	$[0,1]^{30} \to R$	$[0,1]^{30} \to R$
# Optima	5	125	5	125

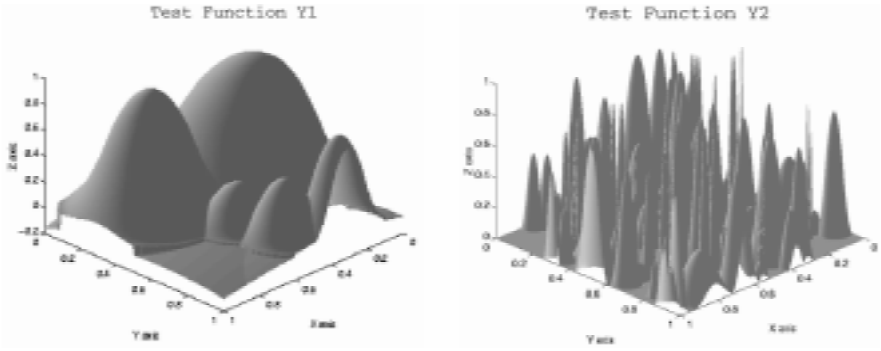

Figure 6. The plot of the test functions Y1 and Y2.

UEGO which produce good solutions (Section 3); the second stage of testing has been designed for comparing UEGO to other methods (Section 5) and for evaluating UEGO with a set of known test functions (Section 6).

3.1. CHARACTERISTICS OF TRAINING TEST FUNCTIONS

The first stage of experiments has been carried out on a set of four different test functions (Y1, Y2, Y3, Y4). The main characteristics of these functions (dimension and number of maxima) are described in Table 1.

We decided not to use well-known benchmark functions in this stage of experiments. The main reason is that we agree with the ideas discussed in Hooker (1995), namely that for doing scientific tests it is more convenient to use functions that differ only in controllable features. This will allow us to analyze the effect of only one separated feature of the test problem: the number of local optima K and the dimension n of the function.

The construction of these functions starts with a user-given list of local optimum sites (o) and the corresponding function values (f_o), which must be positive values.

In the first step, we define bell shapes for every site to create the local optima. The height of a bell is given by the function value f_o of its site o, and its radius r is the distance from o to the closest site. The height of the bell at a distance x from

Table 2. Values of the UEGO parameters

Levels (l)	max_spec_num (M)	min_r (r_l)
2, 5, 10	5, 20, 50, 200	0.8, 0.5, 0.3, 0.1, 0.03

o is $f_o g(x)$ where

$$g(x) = \begin{cases} 1 - \dfrac{2x^2}{r^2} & \text{if } x < \dfrac{r}{2} \\ \dfrac{2(x-r)^2}{r^2} & \text{if } \dfrac{r}{2} \leqslant x < r \\ 0 & \text{otherwise} \end{cases}$$

The value of the objective function at any location is the sum of these bells. In the case of our test functions, the coordinates of the maximum sites and function values were randomly taken from [0, 1] using a uniform distribution. We made the random choice of the maximum sites taking in account that the distance among them should be greater than 0.04. Examples of such functions (Y1,Y2) have been drawn in Figure 6.

3.2. THE EXPERIMENTS

For the purpose of analyzing the effects of the parameters **levels**, **max_spec_num** and **min_r**, a set of experiments using the four training test functions were made. The values of these parameters are shown in Table 2. Experiments were performed for all combinations of these parameter settings. Some results are shown in Tables 3, . . . , 6 that are average values over 100 runs.

Results in Tables 3 and 4 go with the experiments with the fixed value of **evals** (N) equal to 100, 000. In this table, first column shows the values of the parameter **max_spec_num** (M), and second column shows the values of minimum radius (r_l) for each value of M. First row shows the values of the parameter **levels** (l). For each combination of M, r_l and l, the values of two performance measures are indicated: the average number of function evaluations UEGO uses when $N = 100,000$ (FE), and the average number of maxima (species) UEGO detects (S). The number of maxima is computed as the number of species existing at the end of UEGO. If 100% of success in finding the global solution is not reached using a certain combination of parameters, then the corresponding FE value is an $*$. A success happens when $\hat{f} \geqslant f^* - \epsilon$, where $\hat{f}$ is the value of the objective function for the found maximum, f^* is the maximum value of the objective function and $\epsilon = 10^{-6}$.

Table 3 shows some results for Y1 test function, which is two-dimensional ($n = 2$) and it only has 5 optima ($K = 5$). In this table it can be seen that for a fixed value of M and l, the number of detected species (S) increases when the radius (r_l) decreases. This growth in the number of species lies in the fact that the

Table 3. Results for Y1 ($n = 2,\ K = 5$)

		$l = 2$		$l = 5$		$l = 10$	
M	r_l	FE	S	FE	S	FE	S
	0.80	6,301	1.0	12,154	1.0	14,054	1.0
	0.50	6,497	2.9	11,969	2.7	14,473	2.5
5	0.30	6,669	2.9	13,412	3.0	16,055	3.0
	0.10	6,732	2.7	16,184	3.0	18,950	3.1
	0.03	6,322	2.6	17,594	3.2	20,010	3.4
	0.80	2,131	1.0	3,907	1.0	4,836	1.0
	0.50	2,501	2.6	4,310	2.5	5,516	2.3
20	0.30	2,723	3.0	5,094	3.0	6,448	2.9
	0.10	5,186	4.2	9,115	4.2	12,447	4.3
	0.03	5,155	4.6	11,516	4.9	16,453	5.0
	0.80	1,191	1.0	2,256	1.0	3,237	1.0
	0.50	1,510	2.6	2,707	2.5	3,915	2.4
50	0.30	1,872	3.0	3,258	3.0	4,725	3.0
	0.10	5,845	4.5	6,906	4.3	9,498	4.1
	0.03	9,054	5.0	10,191	4.9	12,282	5.0
	0.80	827	1.0	2,477	1.0	4,979	1.0
	0.50	1,128	2.5	2,885	2.6	5,178	2.6
200	0.30	1,414	3.0	3,189	2.9	5,504	3.0
	0.10	5,177	4.6	5,882	4.3	8,330	4.6
	0.03	8,636	4.9	7,743	5.0	10,705	5.0

number of fused species decreases when the radii are smaller. Since each species had been assigned a certain maximum number of function evaluations (budget per species) in the optimization processes, the more species there are, the more function evaluations are consumed. Consequently, the number of function evaluations FE also increases when the radius decreases.

For Table 3 under consideration and for fixed values of l and r_l, a tendency to find more species (S) when M increases can be appreciated. This tendency is not too clear due to the fact that Y1 test function only has 5 optima. This increment in the number of found species can be explained by the fact that a larger population size is allowed and hence, it results easier to explore the search space. However, for those fixed parameters, the number of function evaluations (FE) decreases when M increases owing to the decrease in the budget per species (n_i/M) in the optimization processes and to the fact that the number of species remains small for this test function. It is clear that this effect is not a linear function of M, and it can be seen clearer between $M = 5$ and $M = 20$. Increasing M further does not show

Table 4. Results for Y2 ($n = 2,\ K = 125$)

		$l = 2$		$l = 5$		$l = 10$	
M	r_l	FE	S	FE	S	FE	S
	0.80	12,820	1.0	44,457	1.0	57,730	1.1
	0.50	13,231	2.6	38,917	2.6	50,499	2.6
5	0.30	12,677	3.7	36,745	3.8	49,529	3.9
	0.10	*	3.8	37,517	3.9	51,590	4.0
	0.03	*	3.9	49,064	4.0	74,609	4.0
	0.80	*	1.0	15,147	1.0	19,628	1.1
	0.50	*	2.4	14,074	2.4	18,499	2.6
20	0.30	*	6.3	14,754	6.2	20,639	6.2
	0.10	*	14	21,224	14	31,548	15
	0.03	*	16	29,911	16	42,630	18
	0.80	*	1.0	*	1.0	10,384	1.1
	0.50	*	1.0	*	2.3	10,308	2.5
50	0.30	*	1.0	*	6.8	12,315	6.1
	0.10	10,138	23	18,424	23	26,458	24
	0.03	10,399	34	27,197	35	40,977	38
	0.80	*	1.0	*	1.2	6,632	1.2
	0.50	*	2.1	*	2.1	6,855	2.1
200	0.30	*	5.7	*	5.9	8,692	5.9
	0.10	7,873	21	13,629	24	20,628	23
	0.03	17,909	55	39,510	62	49,338	64

the effect on S since $M = 20$ is already sufficient due to the small number of local optima.

Results in Table 4 for Y2 test function, which is two dimensional ($n = 2$) and it has 125 optima ($K = 125$), show that the tendencies in the performance of UEGO with respect to the parameters M, r_l and l are similar to the tendencies analyzed for Y1 test function.

Accordingly with the tendencies shown for Y1 in Table 3, the number of detected species (S) increases when M and l increase and r_l decreases. In the same way, the number of function evaluations (FE) increases when l increases and M and r_l decrease.

Y2 is a more difficult test function than Y1, in such a way that for some combinations of the parameters, UEGO is not able to find the global solution with 100% of success. The symbol $*$ in Table 4 indicates that for the corresponding combination of parameters UEGO does not reach the global optimum with 100% of success. The results in Table 4 show that UEGO did not get trapped in local optima only when the number of levels is high ($l = 10$), following that the cooling process allows the

Table 5. Results for Y3 ($n = 30,\ K = 5$)

		$l = 2$		$l = 5$		$l = 10$	
M	r_l	FE	S	FE	S	FE	S
	0.80	397,208	2.5	406,303	3.2	407,825	3.6
	0.50	397,154	2.6	406,928	3.3	407,631	3.6
5	0.30	396,184	2.7	407,118	3.2	408,608	3.6
	0.10	396,074	2.7	407,542	3.4	409,258	3.7
	0.03	396,070	2.6	407,375	3.3	409,550	3.6
	0.80	124,145	4.2	122,147	4.0	119,658	3.9
	0.50	124,050	4.2	123,929	4.0	123,904	4.1
20	0.30	124,292	4.2	124,583	4.0	124,684	4.1
	0.10	124,493	4.2	125,214	4.2	125,598	4.2
	0.03	124,158	4.2	124,221	4.1	126,039	4.1
	0.80	83,600	4.4	51,074	3.5	52,176	3.6
	0.50	80,965	4.3	53,966	3.8	54,322	3.5
50	0.30	85,077	4.4	54,080	3.7	54,584	2.7
	0.10	84,694	4.4	54,166	3.8	53,902	3.7
	0.03	86,666	4.7	55,363	3.8	53,347	3.6
	0.80	25,074	3.7	18,441	2.8	21,130	2.9
	0.50	26,237	3.7	18,437	2.6	20,873	2.9
200	0.30	25,640	3.7	18,877	2.8	$*$	2.9
	0.10	27,273	3.7	$*$	2.9	$*$	2.9
	0.03	24,505	3.8	$*$	3.3	$*$	3.8

algorithm to escape from local optima. When the number of levels is small ($l = 2$), the algorithm can find the global optimum with 100% of success either when M is small and r_l is big or when M is big and r_l is small (the large amount of allowed species cover most of the search space, detecting the global attraction area).

Function Y3 has only 5 optima ($K = 5$), but it is defined in a 30-dimensional domain space ($n = 30$), a characteristic that increases the difficulty of locating the maxima, in such a way that UEGO could not find the global solution with 100% of success for any combination of parameters. In several cases, the algorithm could find the global maximum attraction region, but it was not able to reach the maximum position with enough precision. For these reasons we tried to run the algorithm with a larger value of the maximum number of function evaluations, i.e. $N = 1,000,000$. Table 5 shows some results for these new experiments. Nevertheless, it can be seen that 100% of success in finding the global maximum was not reached for large values of M ($M = 200$), where the number of function evaluations is quite small. This fact happens because when M increases, the budget

Table 6. Results for Y4 (n=30, K=125). Parameters: M=50, N=20,000,000

	$l=2$			$l=5$			$l=10$		
r_l	FE	S	%	FE	S	%	FE	S	%
0.80	420,625	33.3	94	474,177	34.9	94	468,973	35.1	88
0.50	422,896	34.4	94	474,322	35.3	94	479,947	36.7	87
0.30	420,084	34.5	89	477,121	36.7	88	483,702	39.1	94
0.10	423,259	35.4	80	472,516	39.6	96	486,743	40.1	98
0.03	420,875	36.7	78	473,898	40.8	100	495,103	42.2	100

per species in the optimization process (n_i/M) decreases; and this small number of points the optimizer is allowed to evaluate is not enough in a 30-dimensional space. These difficulties of UEGO in finding the global maximum with enough precision would be mainly due to the fact that we are using a rather inefficient local search procedure that does not converge very well in high dimensions. Therefore better results could be achieved if a faster local search optimizer were used.

It is interesting to point out that, as can be seen in Table 5, the average number of detected species (S) and the average number of function evaluations (FE) hardly change with the value of the parameter r_l. This result can be put down to the relatively far distance among the locations of the maxima in a 30-dimensional space, allowing the detection of the same number of species using different radii values. In other words, shrinking the radius size the way done in the experiments did not result in decreasing the number of detected species. To achieve this effect the radius should have been decreased much steeper. However for a clearer comparison we used the same values in every experiment.

Another similar effect with respect to the number of consumed function evaluations can be appreciated for different **levels** values. Taking into account that for Y3 it is quite hard to detect the positions of the maxima, the optimizer consumes the whole budget per species it has been assigned (n_i/M).

Test function Y4 (30-dimensional with 125 optima) has both a large number of maxima and high dimensionality. Consequently 100% of success in finding the global optimum was not reached for $N \leqslant 1,000,000$. Therefore, experiments for Y4 test function were run for the parameters M and N fixed to $M = 50$ and $N = 20,000,000$. From these experiments the effects of the number of levels l and minimum radius r_l were analyzed. Table 6 illustrates the average results from those experiments, where FE indicates the average number of consumed function evaluations, S the average number of found optima (species), and % the percentage of success in reaching the global optimum.

Results in Table 6 show that the number of optima the algorithm is able to find increases when the minimum radius decreases and the number of levels increases. It can be seen that the number of function evaluations also increases with the number of found species; though this increment is not too high as happened for Y3 test

function. It is interesting to remark that the largest percentage of success is achieved for higher number of levels and smaller radii, as happened for Y2 test function. In this way, it can be seen that for $l = 2$, the percentage of success decreases when r_l decreases due to the fact that with $l = 2$ and small radii the algorithm cannot cover the whole search space and hence, it can get trapped in a local optimum. However, when the number of levels is high, due to the cooling mechanism, the algorithm is not trapped in a local optimum, and small final radii allow to obtain more accurate solutions.

As conclusion of the above experiments it could be said that a robust parameter setting consists of a large enough number of levels (l), a small minimum radius (r_l), a sufficient maximum number of species (M) and a large value of N in order to get a minimum budget per species which is sufficient in the optimization process. An example of robust parameter setting could be: $r_l = 0.03$, $l = 10$, $M = 50, 100$ and $N = 1,000,000$. We have to point out that the optimal values depend on the problem domain at hand. Based on the above results, after preliminary experiments, it is possible to fine-tune the parameters. For instance if the number of species found is small then we can use a smaller M, and a smaller N.

4. Reliability Measurement

In Csendes (1988) a new global optimization test problem is suggested. This problem can be used for measuring the reliability of a global optimization algorithm and testing the degree of difficulty of global optimization problems that can be solved with it.

The suggested n-dimensional test function to be minimized can be described as:

$$F(x) = \sum_{i=1}^{n} f_i(x_i) \tag{6}$$

where for every $i = 1, 2, \ldots, n$:

$$f_i(x_i) = x_i^6(\sin(1/x_i) + 2)$$

if $x_i \neq 0$, and

$$f_i(0) = 0$$

The function $F(x)$ has a countable infinity of local minima and maxima and all these extrema are in the hypercube:

$$-1 \leqslant x_i \leqslant 1 \quad i = 1, 2, \ldots, n \tag{7}$$

In Csendes (1988) it was proved that the region of attraction of the global minimum is of zero measure. The most important property of $F(x)$ is that the smaller the local minimum, the smaller the measure of the region of attraction

Table 7. Results for a set of test functions. UEGO parameters: N=1,000,000, M=20, l=10 y r_l=0.03

n	$f(x^*)$	FE	S
1	$-0.0000000000e+00$	45, 847	1
2	$-0.0000000000e+00$	46, 282	1
3	$-5.8190367120e-54$	53, 901	1
4	$-7.4351932756e-46$	62, 275	6.3
5	$-1.6196189608e-30$	60, 175	1.5
6	$-6.8963964499e-28$	60, 799	2.3
7	$-1.9740668478e-27$	57, 333	2.9
8	$-1.9975941753e-25$	58, 032	3.6
9	$-1.5502053326e-23$	54, 117	4.4
10	$-1.5597071749e-19$	56, 585	7.6

relating to this local minimum. This feature can be used to assess the degree of difficulty of global optimization problems that can be solved by the given method. The local minimizers of the one-dimensional version of the test function can be ordered according to the magnitude of the function value. The serial number N_x of the local minimizer x can be calculated using the equation

$$N_x = 2\lfloor |1/x|/2\pi \rfloor - 1 + (sgn(x) - 1)/2 \tag{8}$$

where $\lfloor . \rfloor$ denotes the largest integer not greater than the argument, and sgn stands for the signum function. In the one-dimensional case the size of the region of attraction A_x of local minimizer x can be well estimated by:

$$A_x \approx \frac{2}{\frac{1}{x^2} - \pi^2} \tag{9}$$

where A_x is approximately equal to the distance between the two minima that are adjacent to x.

4.1. RESULTS FOR THIS TEST FUNCTION DEFINED IN SEVERAL DIMENSIONS

In order to measure the reliability of UEGO algorithm, the opposite of the above test function ($-F(x)$; $x \in [-1, 1]^n$), see Equation (6), in several dimensions n, has been maximized. The values of the parameters were: $N = 1,000,000$, $M = 20$, $l = 10$ y $r_l = 0.03$. All experiments have been executed 100 times, and results in Table 7 are average values (double precision) of the found maximum ($f(x^*)$), the average number of function evaluations (FE), and the average number of detected species (S). In the left hand column dimension of the test problems has been represented.

The results in Table 7 show that for $n = 1$ and $n = 2$ UEGO reaches the real global maximum equal to 0.0. For the remaining test functions, the differences

Table 8. Comparison to Csendes algorithm

n		Csendes	UEGO
1	$f(x^*)$	$-0.319144e-23$	$-0.000000e+00$
	FE	22, 137	45, 847
	s.t.u	33.5	61.04
	s.t.u./eval	$1.51e-3$	$1.33e-3$
4	$f(x^*)$	$-0.272099e-8$	$-7.435193e-46$
	FE	22, 137	62, 275
	s.t.u	46.1	110.09
	s.t.u./eval	$2.09e-3$	$1.77e-3$

between the real maxima and the optima reached by UEGO increase with the dimension of the problem, and hence with the complexity of the test problem. However, the found optima are still quite small. In Csendes (1988), after a reliability test using these test functions, Csendes concludes that his algorithm can be tuned to solve most practical problems with satisfying reliability.

Table 8 shows results for Csendes' and UEGO algorithms for the test functions with $n = 1$ and $n = 4$. The magnitude 's.u.t' stands for standard time unit which indicates the consumed time in 1000 evaluations of the Shekel 5 function at $x^T = (4.0, 4.0, 4.0, 4.0)^T$. The magnitude s.t.u./eval (s.t.u/number_of_function_evaluations) is a measurement of the speed of the algorithm. Results show that UEGO reaches the solution with more accuracy though it needs more function evaluations and therefore more time. However it can be seen that UEGO is faster than Csendes' algorithm in the sense that the values of 's.t.u./eval' are smaller for UEGO for both test functions.

Thereby, it can be concluded that the algorithm shows a high degree of reliability even for quite complex test functions.

5. Comparison to Other Methods

In this set of experiments we wanted to compare UEGO to methods that have been developed to be used in similar environments. This is the reason why we did not include domain specific clustering methods in the test, only heuristics that have a similar general application area. The algorithms we chose are: a simple hillclimber (SHC), a multistart hillclimber (MHC), GAS and a GA with local search (GENESIS).

Another issue was to choose the problem domain for the comparison. According to a general feeling in the field of global optimization which is supported by theoretical results as well Wolpert and Macready (1997), every algorithm has its special area of application, and there is no algorithm that is better than some other algorithm on every task. Therefore a paper that discusses a new algorithm should try to characterize the situations in which the algorithm can be expected to perform

especially well. For this reason we chose a set of functions that can illustrate the adaptive search focusing capabilities of UEGO. The functions have a relatively large flat area with the interesting part located in a small cluster.

5.1. TEST FUNCTION SET

The set of test functions consists of the functions Y1-4 on extended domains, $[0, n]^d$ where d is 2 (for Y1, Y2) or 30 (for Y3, Y4), and n is such that the volume V of the domain is $V = 2^j$, where $j = 0, 1, 2, 3, 4, 5, 6, 7$. The function value outside $[0, 1]^d$ is the constant value -0.1 which is lower than the minimum of Y1-4. Therefore the new set of test function consists of the set of the 32 functions:

$$\mathrm{Y}i\mathrm{V}2^j, \quad i = 1, 2, 3, 4, \quad j = 0, 1, 2, 3, 4, 5, 6, 7$$

5.2. PARAMETER SETTING

In this section comparisons with a simple hill climber (SHC), a multistart hill climber (MHC), GAS (the ancestor of UEGO) and GENESIS (a GA with local search) are shown. All results are average results over 50 runs. The maximum number of function evaluations was set to $N = 100,000$ for Y1-2, and $N = 1,000,000$ for Y3-4, for all algorithms. The remaining parameters of the algorithms were set as follows.

UEGO algorithm was run using: $l = 10$, $M = 50$ and $r_l = 0.03$.

The hill climber (SHC) was the optimizer used by UEGO; it means that SHC is UEGO with $l = 1$.

In the multistart case, MHC, the number of restarts from a new random point is given by the value of M for UEGO, i.e. 50. Therefore it consists of 50 runs of UEGO($l = 1$) in such a way that N is an upper bound of the total number of function evaluations.

The parameters for GAS are very similar to those of UEGO, so the minimum radius was set to 0.03, the population size was set to 200 in such a way that the maximum number of species is $population\_size/4 = 50$ and the number of levels was set to 8, the maximum allowed by the algorithm.

GENESIS is a GA with local search (see Grefenstette (1990)). We have introduced local optimizer SASS in the algorithm in order to compare similar heuristics. The number of steps of the local optimizer in GENESIS algorithm was set to 20 according to the suggestion of Orvosh and Davis (1993). The parameters of GENESIS used in the experiments were: to use gray coding, 30 bit per dimension, mutation 0.01, and ranking elitist selection. The remaining parameters of the algorithm were set to the default values.

Table 9. Results of the comparison experiments for Y1

Function		UEGO	SHC	MHC	GAS	GENESIS
Y1V1	FE	12,282	188	9,447	172,893	100,114
	%Succ	100	30	100	100	100
	S	4.96	1.0	3.0	4.44	1.0
Y1V2	FE	9,129	186	9,327	159,199	1001,20
	%Succ	100	24	100	100	70
	S	4.88	1.0	3.0	3.68	1.0
Y1V4	FE	6,226	197	9,861	134,984	100,128
	%Succ	96	10	100	100	90
	S	4.26	1.0	2.8	2.62	1.0
Y1V8	FE	6,261	191	9,580	145,419	100,260
	%Succ	90	14	100	100	50
	S	4.06	1.0	2.0	1.96	1.0
Y1V16	FE	5,334	195	9,774	155,817	100,260
	%Succ	72	14	100	100	0
	S	3.46	1.0	2.0	2.24	1.0
Y1V32	FE	5,323	200	10,027	148,893	100,160
	%Succ	64	12	100	100	0
	S	2.74	1.0	2.0	1.84	1.0
Y1V64	FE	5,471	216	10,845	135,079	100,360
	%Succ	84	20	100	100	0
	S	3.52	1.0	2.0	1.72	1.0
Y1V128	FE	5,413	238	11,637	159,366	100,400
	%Succ	88	12	100	100	0
	S	3.44	1.0	2.0	1.84	1.0

5.3. RESULTS AND DISCUSSION

Results of the experiments for Y1-4 and their extensions are shown in Tables 9, 10, 11 and 12 respectively.

In these tables, first column shows the extension of the function, and second column shows the performance magnitudes that have been measured in the experiments. Columns third to seventh show the results for UEGO, SHC, MHC, GAS and GENESIS algorithms.

The SHC algorithm performs very poorly which is not surprising given the special structure of our domain.

Table 10. Results of the comparison experiments for Y2

Function		UEGO	SHC	MHC	GAS	GENESIS
Y2V1	FE	21,425	178	8,940	171,576	100,180
	%Succ	100	0	0	96	0
	S	38.2	1.0	16.67	31.02	1.0
Y2V2	FE	19,651	181	9,078	171,875	100,138
	%Succ	100	6	100	100	85
	S	36.42	1.0	17.13	26.2	1.0
Y2V4	FE	18,580	181	9,080	172,092	100,280
	%Succ	100	2	100	100	5
	S	36.06	1.0	18.6	24.48	1.0
Y2V8	FE	17,061	190	9,511	172,347	100,400
	%Succ	0.98453	8	100	100	0
	S	35.9	1.0	22.8	18.98	1.0
Y2V16	FE	16,601	194	9,718	172,509	100,360
	%Succ	94	4	100	98	0
	S	34.76	1.0	17.2	17.56	1.0
Y2V32	FE	15,632	183	9,159	172,787	100,380
	%Succ	100	8	100	98	0
	S	34.44	1.0	17.4	11.8	1.0
Y2V64	FE	14,906	195	9,789	172,870	100,020
	%Succ	96	8	100	96	0
	S	33.8	1.0	17.3	11.22	1.0
Y2V128	FE	14,414	181	10,287	173,037	100,200
	%Succ	90	6	80	84	0
	S	33.22	1.0	18.1	7.46	1.0

GENESIS also presents few success in finding the global solutions. On Y1 defined in large domains GENESIS does not reach the solution with enough precision using that upper bound in the number of function evaluations. On the remaining test functions GENESIS only reaches local solutions.

The performance of the other methods is more interesting. On the two dimensional functions the difference between the methods is in the number of function evaluations. GAS uses a lot more evaluations than the other two. An interesting effect is worth mentioning: UEGO is much more adaptive in terms of allocating function evaluations.

The set of functions based on Y1 have simple structure, here UEGO uses less evaluations than the MHC. On Y2 UEGO uses more due to the more difficult struc-

Table 11. Results of the comparison experiments for Y3

Function		UEGO	SHC	MHC	GAS	GENESIS
Y3V1	FE	30,339	1293	64,668	1,749,299	1,000,140
	%Succ	100	16	100	0	0
	S	3.5	1.0	38.2	1.0	1.0
Y3V2	FE	228,729	780,302	869,279	1,747,325	1,000,230
	%Succ	100	4	100	0	0
	S	7.16	1.0	11.24	43.48	1.0
Y3V4	FE	219,209	920,115	925,807	1,748,291	1,000,740
	%Succ	100	4	10	0	0
	S	10	1.0	11.20	47.14	1.0
Y3V8	FE	168,138	880,153	98,1471	1,748,962	1,000,640
	%Succ	100	6	0	0	0
	S	12.44	1.0	11.48	47.14	1.0
Y3V16	FE	121,885	980,024	1,000,050	1,748,729	1,000,720
	%Succ	100	2	0	0	0
	S	13.24	1.0	11.86	49.02	1.0
Y3V32	FE	110,809	980,030	1,000,050	1,749,003	1,000,400
	%Succ	100	0	0	0	0
	S	12.52	1.0	11.14	50.0	1.0
Y3V64	FE	105,950	980,030	1,000,050	1,749,287	1,000,760
	%Succ	94	0	0	0	0
	S	13.08	1.0	12.34	11.4	1.0
Y3V128	FE	109,765	980,027	1,000,050	1,746,830	1,000,108
	%Succ	94	0	0	0	0
	S	13.72	1.0	12.56	31.6	1.0

ture. In higher dimensions where the hillclimber in itself is much less effective the differences between the metaheuristics are more evident. The quality of the solutions of UEGO is slightly better than the solutions of GAS and much better than MHC, especially when the problems become harder (when the volume of the domain increases). UEGO finds the global optimum in significantly more runs than the other algorithms.

At the same time, due to the species creation mechanism and the evaluation allocation method to existing species (if there are few species, the total amount of evaluations will be small) the number of evaluations is significantly fewer in UEGO than in the other two algorithms.

Table 12. Results of the comparison experiments for Y4

Function		UEGO	SHC	MHC	GAS	GENESIS
Y4V1	FE	38,123	1263	63,155	1,747,766	1,000,780
	%Succ	20	0	0	6	0
	S	11.42	1	36.0	33.34	1.0
Y4V2	FE	30,433	580,599	534,043	1,747,503	1,000,460
	%Succ	28	0	1	4	0
	S	6.76	1.0	36.2	24.46	1.0
Y4V4	FE	25,881	780,294	776,851	1,747,855	1,000,600
	%Succ	14	0	0	4	0
	S	3.88	1.0	28.4	26.5	1.0
Y4V8	FE	22,925	900,139	88,8140	1,747,984	1,000,040
	%Succ	8	0	0	4	0
	S	2.0	1.0	11.6	21.56	1.0
Y4V16	FE	22,170	940,084	944,072	1,747,218	1,000,760
	%Succ	4	2	0	4	0
	S	1.44	1.0	11.64	24.5	1.0
Y4V32	FE	21,584	960,054	981,548	1,747,955	1,000,860
	%Succ	6	0	0	0	0
	S	1.04	1.0	11.82	20.6	1.0
Y4V64	FE	21,567	980,026	944,093	1,747,590	1,000,720
	%Succ	4	0	0	0	0
	S	1.08	1.0	12.2	25.5	1.0
Y4V128	FE	21,445	980,028	1,000,050	1,748,079	1,000,420
	%Succ	4	0	0	0	0
	S	1.0	1.0	12.4	12.5	1.0

6. Testing UEGO with a Set of Known Test Functions

Having the reliability of the algorithm tested, the next set of experiments was aimed to prove that a robust parameter setting of UEGO deduced from previous experiments can be used on other known test functions. To this end, a set of 48 test functions (see Appendix) was chosen; the parameter setting was: $N = 100,000$, $M = 20$, $l = 10$ y $r_l = 0.03$. The functions identified only by their names can be found in Törn and Žilinskas (1989), Dixon and Szego (1975) and Walster et al. (1985)

All experiments were run 100 times and results in Table 13 shows average values in the number of function evaluations (FE) and average values in the number

Table 13. Results for a set of test functions. UEGO parameters: $N = 100,000$, $M = 20$, $l = 10$ y $r_l = 0.03$

Function	FE	S	Function	FE	S
F1	9,168	18.8	F22	5,490	1.0
F2	13,028	8.4	F23	5,338	1.0
F3	5,631	1.0	F24	33,474	8.7
F4	12,733	3.0	F26	46,958	5.2
F5	49,841	2.0	F27	5,418	1.0
F6	12,037	8.0	F28	42,030	3.0
F7	11,444	17.1	F29	18,184	17.2
F8	5,475	1.0	F30	5,418	1.0
F9	13,642	4.0	F31	6,968	4.0
F10	6,295	2.0	F32	27,513	16.4
F11	5,475	1.0	F36	8,946	19.0
F12	22,608	2.0	F37	6,007	1.0
F13	25,510	3.0	F39	28,953	8.7
F14	11,137	4.8	F40_5	7,149	19.0
F15	9,757	2.7	F40_7	7,024	19.0
F16	10,018	18.4	F40_9	7,046	19.0
F17	23,315	18.0	F42_2	15,726	14.4
F18	39,571	14.8	F42_3	14,761	18.8
F19	5,428	1.0	F42_4	13,883	18.6
F20	20,590	4.0	F44	67,661	19.0

of detected species (S). Table 13 only gives results for the 40 test functions for which 100% of success in finding the global maximum was reached. It can be seen that the number of function evaluations depends on the complexity of the test functions. Recall that the parameter N is only an upper bound, and the actual number of function evaluations depends on the structure of the function. If it is simple, few evaluations will be used. Functions F40 and F42 have been built for several dimensions, i.e. function F40_5 is F40 function defined in a 5-dimensional space.

The rest of test functions did not reach the location of the global optimum with enough precision, so a more robust parameter setting was chosen, where the values of M and N were increased: $M = 100$, $N = 1,000,000$. For this stage of new experiments a 100% of success was reached for the whole set of test functions. Table 14 shows the number of optima of the test function (**K**), the average number of function evaluations (FE) and the average number of found species (S) for every test function. The average number of detected optima (S) is quite close to either the number of optima (K) or the maximum species number allowed (M). Additionally,

Table 14. Results for a set of hard test functions. UEGO parameters: N=1,000,000, M=100, l=10 and r_l=0.03

Function		Performance		Function		Performance	
Index	K	FE	S	Index	K	FE	S
F21	> 100	115,507	98.5	F35	10	141,662	9.7
F25	> 1000	143,622	95.4	F38	1	103,117	1.0
F33	5	126,441	4.9	F41	1	102,861	1.0
F34	7	131,722	6.9	F43	> 200	598,765	98.8

the average number of function evaluations (FE) depends not only on the number of optima but also in the kind of test function (i.e. the number of dimensions).

7. Concluding Remarks

In this paper, UEGO a general evolutionary algorithm has been investigated. Using a set of four training functions, the behavior of the algorithm has been analyzed and the set of robust parameter setting was found. The reliability of UEGO has been tested using the method proposed by Csendes, and the obtained results showed that UEGO is able to find accurate solutions for hard test problems. Comparisons to other methods have been carried out, and results show that UEGO presents a better global performance than the compared algorithms. The evaluation of UEGO over a set of forty-eight standard test functions has confirmed that it can be successfully used as a global optimization algorithm for finding the global optimum and many local optima for multimodal functions.

Appendix

Table 15. Test Function Description. F: Index of test function. D: Search space. $f(x^*)$: Global maximum value. K: Number of optima

F	Function $f(x)$	D	$f(x^*)$	K
F1	$\sin(x)$	$[-5, 5]$	1.00	2
F2	$3 \mid x - \lfloor x \rfloor - 0.5 \mid + x \sin(x)$	$[-5, 5]$	2π	7
F3	$-24x^4 + 142x^3 - 303x^2 + 276x - 93$	$[-5, 5]$	-0.0064	1
F4	$-\sin x - \sin \frac{10x}{3} - \ln x + 0.84x$	$[2.7, 7.5]$	4.601308	3
F5	$-(x + \sin x) \cdot e^{-x^2}$	$[-10, 10]$	0.824239	1
F6	$\sum_{i=1}^{10} \frac{1}{(k_i \cdot (x - a_i))^2 + c_i}$	$[0, 10]$	14.59266	8
F7	$\sum_{i=1}^{5} i \cdot \sin((i + 1)x + i)$	$[-10, 10]$	12.03125	20
F8	$-(1 + g(x^2) - g(x^2 + 2xy + 2y))$	$[0, 10]$	-1.00	1

Table 15. Continued

F	Function $f(x)$	D	$f(x^*)$	K
F9	$x \cdot \sin(x)$	$[-10, 10]$	7.916727	2
F10	$x^2 \cdot (15 + x^2 \cdot (-27 - 250x^2)) - 1$	$[-10, 10]$	0.000944	4
F11	$-x^2$	$[0, 1]$	0.0	1
F12	$\begin{cases} (x_1-5)^2-(x_2-10)^2 & \text{if } x_1 \leqslant 10 \\ (x_1-15)^2-(x_2-10)^2 & \text{otherwise} \end{cases}$	$[0, 20]^2$	0.0	1
F13	$-\left(\frac{5}{\pi}x_1 - \frac{5.1}{4\pi^2}x_1^2 + x_2 - 6\right)^2 + -10 \cdot \left(1 - \frac{1}{8\pi}\right) \cdot \cos x_1 - 10$	$[-5, 10]^2$	9.602113	> 3
F14	– Six hump camel back	$[-2.5, 2.5]^2$	1.0316	6
F15	– Three hump camel back	$[-5, 5]^2$	0.0	3
F16	– Levy 3	$[-10, 10]^2$	176.542	> 100
F17	– Levy 13	$[-10, 10]^2$	0.0	> 10
F18	– Beale	$[-5, 5]^2$	0.0	> 3
F19	– Booth	$[-5, 5]^2$	0.0	1
F20	– Goldstein / Price	$[-2, 2]^2$	3.0	3
F21	– Griewank	$[-600, 600]^2$	0.0	> 10
F22	$x_1 + x_2$	$[-2, 2]^2$	4.0	1
F23	– Matyas	$[-10, 10]^2$	0.0	1
F24	– Ratz	$[-3, 3]^2$	0.106891	3
F25	– Levy 5	$[-10, 10]^2$	176.138	> 1000
F26	– Rosenbrock	$[-2, 8]^2$	0.0	1
F27	$-((1-x_1)^3 \cdot (x_1^2+1) + (x_1 - x_2^2)^2 + +(x_1-1)^2 + (x_1 - x_3^2)^2 + (x_3-1)^2)$	$[-10, 10]^2$	0.0	1
F28	– Hartman 3	$[0, 1]^3$	3.86278	4
F29	– Levy 8	$[-10, 10]^3$	0.0	> 1000
F30	– Schwefel1	$[-10, 10]^3$	0.0	> 10
F31	– Schwefel1	$[-1.89, 1.89]^3$	0.0	> 10
F32	Box 3D	$[-10, 30]^3$	0.0	14
F33	– Shekel 5	$[-10, 10]^4$	10.15320	5
F34	– Shekel 7	$[-10, 10]^4$	10.40294	7
F35	– Shekel 10	$[-10, 10]^4$	10.53641	10
F36	– Levy 9	$[-10, 10]^4$	0.0	> 100
F37	– Levy 15	$[-10, 10]^4$	0.0	1
F38	– Powell	$[-4, 5]^4$	0.0	1
F39	– Hartman 6	$[0, 1]^6$	3.322828	4
F40	– Ratz 6	$[0, 1]^n$	0.0	∞
F41	– Schwefel	$[-1.89, 1.89]^{30}$	0.0	1
F42	– Zabinsky_90	$[0.0, \pi]^n$	3.5	> 100
F43	$0.5 \cdot \sum_{i=1}^{n} (x_i - 2)^2$	$[-1, 1]^{20}$	90.0	> 200
F44	$0.5 \cdot i \cdot \sum_{i=1}^{n} (x_i - 2)^2$	$[-1, 1]^{20}$	< 4105	> 200

References

Beasley, D., Bull, D.R. and Martin, R.R. (1993), A Sequential Niche Technique for Multimodal Function Optimization. *Evolutionary Computation* 1(2): 101–125.

Csendes, T. (1988), Nonlinear parameter estimation by global optimization – efficiency and reliability. *Acta Cybernetica* 8: 361–370.

Deb, K. (1989), Genetic algorithms in multimodal function optimization. TCGA report no. 89002, The University of Alabama, Dept. of Engineering Mechanics.

Deb, K. and Goldberg, D.E. (1989), An investigation of niche and species formation in genetic function optimization. In: Schaffer, J.D. (ed.) *The Proceedings of the Third International Conference on Genetic Algorithms*.

Dixon, L. and Szego, G. (1975), *Towards Global Optimization*. North-Holland.

Grefenstette, J.J. (1990), *A User's Guide to GENESIS. Version 5.0*. John J. Grefenstette.

Hooker, J.N. (1995), Testing Heuristics: We Have It All Wrong. *Journal of Heuristics* 1(1): 33–42.

Jelasity, M. (1998), UEGO, an Abstract Niching Technique for Global Optimization. In: Eiben, A.E., Bäck, T., Schoenauer, M. and Schwefel, H.-P. (eds), *Parallel Problem Solving from Nature – PPSN V*, Vol. 1498 of *Lecture Notes in Computational Science*. pp. 378–387.

Jelasity, M. and Dombi, J. (1998), GAS, a Concept on Modeling Species in Genetic Algorithms. *Artificial Intelligence* 99(1): 1–19.

Orvosh, D. and Davis, L. (1993), Shall we repair? Genetic algorithms, combinatorial optimizations and feasibility constraints. In: Forrest, S. (ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms*.

Solis, F.J. and Wets, R.J.-B. (1981), Minimization by Random Search Techniques. *Mathematics of Operations Research* 6(1): 19–30.

Törn, A. and Žilinskas, A. (1989), *Global Optimization*, Vol. 350 of *Lecture Notes in Computational Science*. Springer-Verlag, Berlin.

Walster, G., Hansen, E. and Sengupta, S. (1985), Test results for global optimization algorithm. *SIAM Numerical Optimization 1984* pp. 972–287.

Wolpert, D.H. and Macready, W.G. (1997), No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation* 1(1): 67–82.