Constrained Optimization Problem Solving Using Estimation of Distribution Algorithms

P.A. Simionescu
Dept. of Mechanical Engineering
202 Ross Hall
Auburn University
Auburn, AL 36849
Email: simiope@auburn.edu

D. G. Beale
Dept. of Mechanical Engineering
202 Ross Hall
Auburn University
Auburn, AL 36849
Email: bealedg@auburn.edu

G. V. Dozier
Dept. of Computer Science
109 Dunstan Hall
Auburn University
Auburn, AL 36849
Email: doziegv@auburn.edu

***Abstract*-Two variants of Estimation of Distribution Algorithm (EDA) are tested against solving several continuous optimization problems with constraints. Numerical experiments are conducted and comparison is made between constraint handling using several types of penalty and repair operators in case of both elitist and non-elitist implementations of the EDA's. Graphical display and animations of representative runs of the best and worst performers proved useful in enhancing the understanding of how such algorithms work.**

I. INTRODUCTION

Estimation of Distribution Algorithms (EDA) are relatively new comers to the field of Evolutionary Computation [1, 2]. Their appealing features over other evolutionary algorithms are a simple structure and an intuitive dynamics of the population which facilitate choosing the values of the control parameters. In standard EDA there are no crossover and mutation operations, the new population being generated by sampling the probability distribution of a number of superior individuals selected from the current population. As highlighted in [3], the known EDA implementations differ by the probability distributions employed and by the survival selection schemes.

Several authors have reported on solving combinatorial, discrete and continuous optimization problems using EDA [2, 4, 5, 6]. Work remains to be done however on testing the capabilities of EDA in solving constrained optimization problems. In this respect the present paper investigates the use of *Univariate Marginal Distribution Algorithm* (UMDA) and a *Population Based Incremental Learning Algorithm* (PBIL) on three continuous objective functions with constraints. Comparison is made between constraint handling using penalty and repair techniques through numerical experimentation.

II. ALGORITHMS TESTED

Two estimation of distribution algorithms have been implemented and tested in both elitist and non-elitist variants.

A. The UMDA Algorithm

The first algorithm considered, a *Univariate Marginal Distribution Algorithm* [3, 5], was coded in the following structure:

Step 1: Generate M uniform random points within the imposed boundaries of the design variables $[x_{i\,\min} \ldots x_{i\,\max}]$ (i=1...n) or until at least one feasible individual has been generated. The population size, M, is a constant specified by the user.

repeat Step 2 and Step 3 until a certain *stopping criteria* is met;

Step 2: Select the best N individuals in the population and evaluate the average and standard deviation vectors:

$$\{\mu_i\} = \left\{ \frac{1}{N} \sum_{k=1}^{N} (x_i)_k \right\} \quad (i = 1...n) \tag{1}$$

$$\{\sigma_i\} = \left\{ \sqrt{\frac{1}{N-1} \sum_{k=1}^{N} \left[(x_i)_k - \mu_i \right]^2} \right\} \quad (i = 1...n) \tag{2}$$

In the above formulae N is a specified integer restricted to $1 < N < M$.

Step 3: Replace the whole current population by generating M normally distributed random points $\{x_i\}$, (i=1...n) with the averages and standard deviations given by equations (1) and (2) respectively. In order to ensure that the newly generated individuals satisfy the imposed side constraints, the following corrections were performed:

$$\begin{aligned} &\text{IF } x_i < x_{i\min} \text{ THEN } x_i = x_{i\min} \\ &\text{IF } x_i > x_{i\max} \text{ THEN } x_i = x_{i\max} \end{aligned} \tag{3}$$

Additionally, a record of the best-fit individual generated so far is kept to be provided as solution of the search.

The *stopping criteria* can be either attaining an imposed maximum number of generations G_{max} or exceeding a prescribed maximum number of function evaluations NF.

B. The PBIL Algorithm

The second estimation of distribution algorithm tested was a variant of the *Population Based Incremental Learning Algorithm* [6]. The algorithm employs the same Steps 1 and 2 and stopping criteria listed above, but uses a different population-generation scheme on Step 3:

Step 3: Generate M new points $\{x_i\}$, (i=1...n, r=1...M) to replace the current population, using the standard deviations (2) and the following vector of corrected average values:

$$\{\mu_i^*\} = \{(1-\alpha) \cdot \mu_i + \alpha \cdot (x_i)_{best}\} \tag{4}$$

0-7803-8515-2/04/$20.00 ©2004 IEEE

where μ_i are given by the same formula (1) and α is a variable parameter:

$$\alpha = w \cdot \left(G_c / G_{max}\right)^n \quad (5)$$

with G_c the number of the current generation and w a chosen constant between 0 and 1. It is to be noticed that for w=0 the algorithm becomes a UMDA algorithm. In order to ensure that the imposed side constraints are satisfied, same tests (3) are applied to the newly generated points. Similarly to UMDA, the best fit individual encountered so far is recorded to be provided as solution of the search.

In case of elitist implementations of the above two algorithms, further referred to as E-UMDA and E-PBIL, Steps 3 must be modified so that only M-1 new individuals are generated and the best fit individual in the population is not destroyed; evidently, there will no longer be necessary to keep a record of the best fit individual generated so far.

III. CONSTRAINT HANDLING TECHNIQUES

There are numerous constraint handling techniques used in evolutionary computation as follows [7, 8, 9]: 1) various implementations of the penalty method, 2) specialized representations and operators, 3) repair algorithms, 4) separation of objectives and constraints (behavioral memory, superiority of feasible points, multi-objective optimization techniques) 4) hybrid algorithms etc.

Of the known constraints handling techniques, penalty and repair methods will be numerically tested in association with UMDA and PBIL algorithms described in the previous paragraph.

A. Penalty methods

Three penalty methods have been experimented with in this paper; all of them operate by providing some fitness value to the infeasible individuals in the population that will further help with their ranking. Two of the considered methods are step-type penalties while a third method employs the Euclidean distance from the considered infeasible point to the closest feasible point as a measure of its infeasibility.

1) The first penalty method tested, of the step type, will be further referred to as *1K-Penalty* and has the form:

$$\text{fitness}(x_1 \ldots x_n) = \begin{cases} F(x_1 \ldots x_n) & \text{if feasible} \\ K & \text{if infeasible} \end{cases} \quad (6)$$

where K a constant about one order of magnitude greater than the expected global maxima of the constrained function. Such a penalty is very easy to implement but has the main drawback that the search is difficult to initiate in case of highly constrained problems with landscape resembling flat plateaus with scattered crevasses (or only one such crevasse).

2) A slightly more elaborate penalty method tested resembling the K-method in [10], further called *vK-Penalty* was:

$$\text{fitness}(x_1 \ldots x_n) = \begin{cases} F(x_1 \ldots x_n) & \text{if feasible} \\ v \cdot K & \text{if infeasible} \end{cases} \quad (7)$$

with v is the number of constraints violated at point $(x_1 \ldots x_n)$. In this form some rough information about the degree of constraint violation at a certain point can be acquired, which can help directing the search toward the feasible domain. However, as will be seen in case of the first test problem below, the method is less effective when the global optima is bounded by more than one active constraint.

3) A third penalty method tested named *DK-Penalty*:

$$\text{fitness}(x_1 \ldots x_n) = \begin{cases} F(x_1 \ldots x_n) & \text{if feasible} \\ D^2 \cdot K & \text{if infeasible} \end{cases} \quad (8)$$

employs the distance D between the considered infeasible point and the closest to it feasible point in the population [11]. This will require evaluating the Euclidean distance (or of some other norm) between the current point and all feasible points in the population, slowing down the algorithm.

B. Infeasible-individual repair

These constraint-handling techniques require that at least one feasible individual exists in the current population. It involves a line searching (or some other crossover operation) between the current infeasible point and a selected feasible individual in the population. In the present paper the following repair methods have been experimented with:

Repair 1 - repair by line search: Assign to the infeasible individual to be repaired the closest feasible individual in the population. If there are no feasible individuals in the current population, the repair operation must be suspended and the infeasible points treated in a simple *1K-Penalty* manner (this is the form in which the method was implemented in the numerical experiments performed). Alternatively, in case of non-elitist algorithms, the best point encountered so far can be used as a second point for the line search operation. After the infeasible-feasible pairs have been made, a random search is performed along the line connecting the two points until a second feasible point is generated to be introduced in the population in replace to the considered infeasible individual [12].

Repair 2 - repair by crossover: Instead of doing a line search, which requires a number of objective function evaluations, one single crossover operation can be performed (for example a midpoint crossover) between the current infeasible and its closest feasible individuals. Since the offspring that will replace the infeasible parent may in turn be infeasible, the method is more of an *incomplete repair*.

Repair 3 - repair by cloning: Replace the infeasible individual with an identical copy of the feasible individual that is closes to it. When only one or two feasible individuals are available in the population, in order to preserve diversity (particularly for elitist algorithms), it might become necessary to repair only part of the infeasible individuals (a *partial repair*) to avoid standard deviation becoming too small, or to impose a lower limit upon the components of the standard deviation vector.

Combined repairs: Combination of the above approaches can also be employed, like for example repairing half of the infeasible individuals using cloning and the other half using some crossover operation.

Even if they don't always eliminate the infeasible individuals, the above listed repair methods contribute to a favorable confinement of the population toward the feasible domain(s) of the search space. Methods 2 and 3 have the appealing feature that require less or no additional function evaluations. They are also suitable in case of discrete or integer optimization problems, when the feasible space is very fragmented or is reduced to only scattered points.

IV. TEST PROBLEMS

Several numerical experiments have been performed on solving three constrained objective functions. Since graphical representation and animation of the successive populations can provide a valuable insight into how algorithms work, preference has been given to the following test functions of two variables:

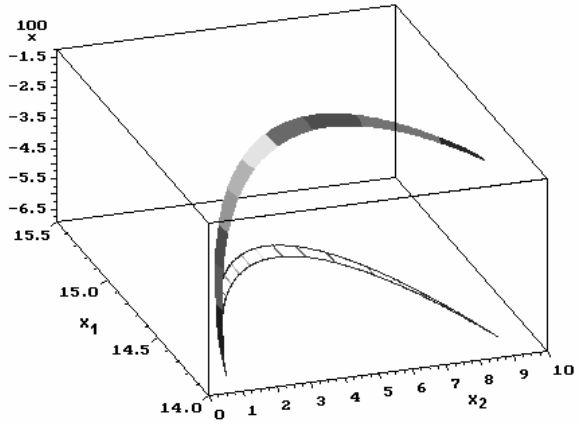

Figure 1. Graph of the Sickle function

Test Problem1 - Sickle function

This is a slightly modified version of problem G6 in reference [13] which requires minimizing the function:

$$F(x_1, x_2) = (x_1 - 20)^3 + (x_2 - 10)^3 \quad (9)$$

subjected to:

$$g_1 = (x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geq 0$$
$$g_2 = -(x_1 - 5)^2 - (x_1 - 6)^2 + 82.81 \geq 0 \quad (10)$$

and the side constraints:

$$0 \leq x_1 \leq 10 \quad \text{and} \quad 14 \leq x_2 \leq 15.5 \quad (11)$$

In its original form [13], the side constrains were over 10 times wider, making the ratio between the feasible and the infeasible spaces very small and therefore a starting feasible point hard to find. The global minimum point is located at x_1=14.095 and x_2=0.84296 for which the function value is -6961.8139 and both constraints are active. The maximum point, also double bounded, is located at x_1=14.095 and x_2=9.15704 and equals -1206.13556. As shown by the plot in Figure 1, the feasible domain of this function is not convex.

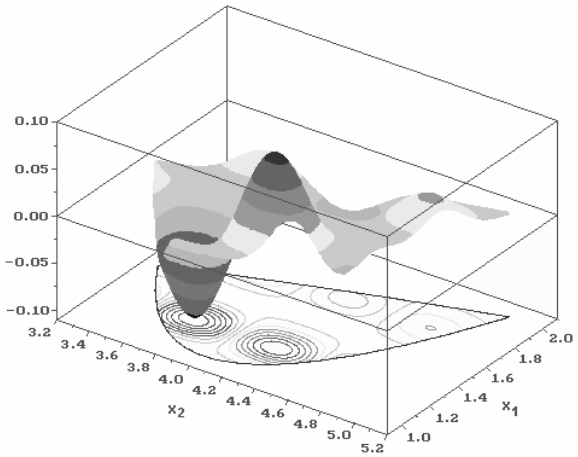

Figure 2. Graph of Koziel and Michalewicz's G6 function

Test Problem2 - Koziel and Michalewicz G6 function

This second problem [13] requires finding the maximum point of:

$$F(x_1, x_2) = \frac{\sin(2\pi x_1) \cdot \sin^3(2\pi x_2)}{(x_1 + x_2) \cdot x_2^3} \quad (12)$$

subjected to:

$$g_1 = x_1 - x_2^2 - 1 \geq 0$$
$$g_2 = -(x_1 - 4)^2 + x_2 - 1 \geq 0 \quad (13)$$

and the side constraints (modified as compared to the original form in [13] for the same reason as before):

$$3.2 \leq x_1 \leq 5.2 \quad \text{and} \quad 0.9 \leq x_2 \leq 2.1 \quad (14)$$

This multimodal function has its global maximum at x_1=1.24539 and x_2=4.2425 and equals 0.09582504. The global minimum is located at x_1=1.24492 and x_2=3.74154 where the function value is -0.10363448. Both the global minimum and the global maximum points are unbounded i.e. they are located inside the feasible domain (Figure 2).

Test Problem3 - Keane's function

The third test problem, due to Keane, also listed as problem G2 in [6], requires minimizing the function:

$$F(x_1 \ldots x_n) = \frac{\left| \sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i) \right|}{\sqrt{\sum_{i=1}^{n} i \cdot x_i^2}} \quad (15)$$

subjected to:

$$g_1 = \sum_{i=1}^{n} x_i - 7.5n \le 0$$
$$g_2 = 0.75 - \prod_{i=1}^{n} x_i \le 0 \quad (16)$$

and to the side constrains:

$$0 \le x_i \le 10 \quad \text{for} \quad 1 \le i \le n \quad (17)$$

This is a highly multimodal function that has its global minimum constrained by g_2. For n=2 its optimum equals -0.36497974 and occurs for x_1=1.60086 and x_2=0.468498. According to [6], for n=20 the minimum value found so far equals -0.8036.

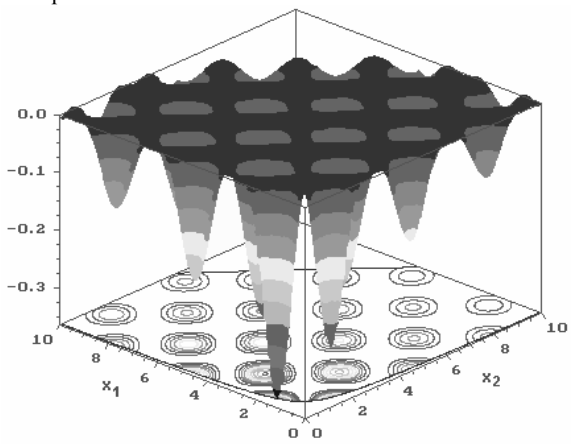

Figure 3. Graph of Keane's function with n=2

V. NUMERICAL RESULTS

A set of numerical experiments have been conducted to test the capabilities of the Estimation of Distribution Algorithms and constraint handling techniques presented and the results are summarized in Tables 1-3.

No attempt has been made during these experiments to fine tune the N, M or w parameters so that performances are maximized (in all cases N=50, M=25 and w=1 while the stopping criteria was limiting the maximum number of function evaluations to NF=5000). The main purpose of these tests was to identify some promising combinations of Estimation of Distribution Algorithms and constraint handling techniques, their potential for improvement and reasons why they performed or did not perform well.

Problem 1 has a non-convex feasible space with only one minimum and one maximum (both double constrained). It is therefore not surprising that the elitist E-UMDA (and E-PBIL algorithm) with line-search repair performed well. This is because only feasible individuals were sampled during the search and the monotonicity of the function favored a constant downhill migration of the population.

TABLE I. RESULTS OBTAINED FOR 500 RUNS OF PROBLEM 1 FOR M=50 AND N=25 (KNOWN GLOBAL OPTIMUM: -6961.81)

Algorithm	Constr. Handling	Best	Average	Worst
E-UMDA	Repair 1	-6945.91	-5607.47	-2997.05
E-PBIL	Repair 1	-6943.65	-5763.21	-3509.77
UMDA	Repair 1	-6939.16	-5553.78	-3372.03
E-UMDA	*v*K-Pen.	-6930.03	-5104.47	-2480.90
E-PBIL	*v*K-Pen.	-6912.24	-5159.63	-2249.93
PBIL	*1K*-Pen .	-6911.18	-5076.39	-2585.59
E-PBIL	*1K*-Pen .	-6903.78	-5244.17	-3117.43
E-PBIL	*DK*-Pen.	-6895.40	-5184.25	-2617.83
PBIL	Repair 1	-6892.11	-5661.05	-3373.31
E-UMDA	Repair 2	-6881.54	-4978.97	-1454.95
UMDA	Repair 3	-6874.920	-4135.440	-1321.91
E-PBIL	Repair 2	-6871.597	-5018.764	-1973.91
PBIL	Repair 2	-6870.774	-5050.324	-1636.22
UMDA	Repair 2	-6869.215	-5008.144	-1862.65
E-UMDA	*DK*-Pen.	-6860.034	-5068.846	-1542.06
UMDA	*DK*-Pen.	-6857.924	-5092.668	-2886.64
PBIL	*v*K-Pena.	-6847.324	-5120.544	-2624.04
E-UMDA	*1K*-Pena .	-6836.38	-5048.02	-2218.117
E-PBIL	Repair 3	-6821.19	-4113.49	-1286.33
E-UMDA	Repair 3	-6813.79	-4148.39	-1279.73
PBIL	Repair 3	-6763.21	-4187.59	-1396.38
UMDA	*v*K-Pen.	-6755.85	-5095.72	-1776.14
PBIL	*DK*-Pen.	-6677.15	-5132.26	-2123.77
UMDA	*1K*-Pen.	-6674.55	-5125.19	-2680.86

This is also illustrated by Figure 4-top where are plotted superimposed the individuals in all generations during one run of the E-UMDA + Repair 1 algorithm (less the intermediate points occurring during line searches).

The runs illustrated by the plots in Figures 4 (and also in Figures 5 and 6) were considered representative in that the best fitness found during the respective searches were very close to the average value recorded in Tables 1-3

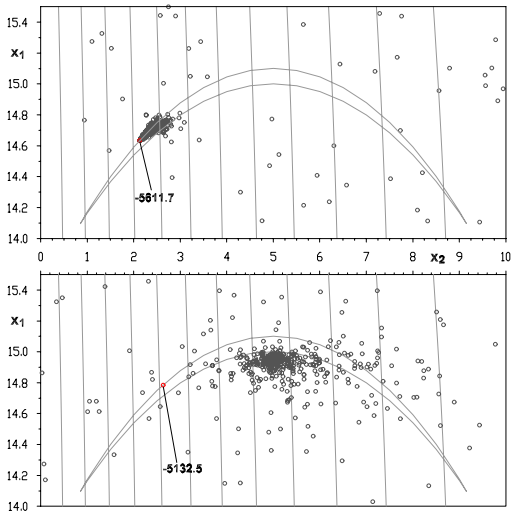

Figure 4. Superimposed plots of the points generated during one run of E-UMDA + Repair 1 (top) and UMDA + 1K-Penalty (bottom) algorithms on Test Problem 1.

(for Figure 4 these fitness values were -5612 vs. -5607 and -5133 vs. -5125).

The same frames plotted superimposed in Figure 4-up were animated and are available as *wmv* files at [14] (or upon request from the first author). From these animations it can be seen that the best fit individual emerged (most likely following a repair operation) during the second generation and was preserved unchanged all the way to the end of the run. As the search progressed, the rest of the population slowly moved toward this best fit individual.

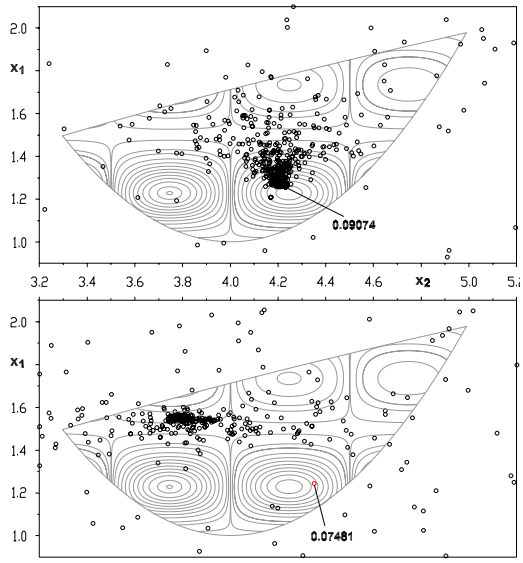

Figure 5. Superimposed plots of the points generated during one run of E-PBIL + *vK*-Penalty (top) and UMDA + Repair 3 (bottom) for Test Problem 2.

The animation also reveals that imposing the repair search to be performed along the line pointing in the direction of the closest feasible individual the displacement of the population parallel to the boundary of the feasible space is significantly diminished. One remedy towards an increased exploration of the areas parallel to the boundaries of the feasible space (other than changing the line-repair strategy) can be to force (directly or indirectly) the components of the standard deviations vector to stay large during the first few generations.

As visible from Figure 4–bottom, the UMDA algorithm with *1K*-Penalty (that was ranked last) had difficulties in maintaining a pool of feasible individuals in the population and was therefore unable to direct the search toward promising areas of the design space. The *wmv* animation file in [14] generated using the same data as for Figure 4-bottom also shows that the actual solution (labeled -5132.5) was generated during the early generations, but no further exploration was performed in that same area.

Problem 2 Since the global maximum for this problem is not bounded, the constraint handling technique employed has very little effect upon the evolution of the population after several generations have passed, when more important become the hill climbing capabilities of the basic EDA employed. The elitist EDA's particularly the E-PBIL algorithm exhibits such traits and consequently performed better (although the favorable effect of a wider initial sampling of the landscape proved beneficial as the results in Table 2 show).

Least performers were the non-elitist EDA algorithms using *cloning repair* (and repair in general). Repair operations have the effect of reducing the variability of the initial populations by forcing its individuals inside the feasible space.

TABLE 2. RESULTS OBTAINED FOR 500 RUNS OF PROBLEM 2 FOR M=50 AND N=25 (KNOWN GLOBAL OPTIMUM: 0.095825)

Algorithm	Constr. Handling	Best	Average	Worst
E-PBIL	*vK*-Pen.	0.095825	0.090999	0.019106
E-PBIL	*DK*-Pen.	0.095825	0.090982	0.018469
E-PBIL	*1K*-Pen.	0.095825	0.090354	0.029143
E-PBIL	Repair 1	0.095825	0.089778	0.025175
E-PBIL	Repair 2	0.095825	0.089659	0.028708
E-UMDA	*vK*-Pen.	0.095825	0.089549	0.027295
E-UMDA	*1K*-Pen.	0.095825	0.089199	0.037292
E-UMDA	*DK*-Pen.	0.095825	0.088951	0.027408
E-UMDA	Repair 2	0.095825	0.088107	0.025067
PBIL	*DK*-Pen.	0.095825	0.086992	0.022776
UMDA	*vK*-Pen.	0.095825	0.086738	0.019890
PBIL	*vK*-Pen.	0.095825	0.086693	0.026722
UMDA	*DK*-Pen.	0.095825	0.086221	0.027787
UMDA	*1K*-Pen.	0.095825	0.086214	0.021905
PBIL	*1K*-Pen.	0.095825	0.086196	0.024969
PBIL	Repair 1	0.095825	0.086046	0.023640
PBIL	Repair 2	0.095825	0.085759	0.036746
UMDA	Repair 2	0.095825	0.085409	0.024786
E-PBIL	Repair 3	0.095825	0.082726	0.001645
E-UMDA	Repair 1	0.095825	0.082144	0.024559
UMDA	Repair 1	0.095825	0.079079	0.018287
E-UMDA	Repair 3	0.095825	0.078868	0.013925
PBIL	Repair 3	0.095825	0.075557	0.012989
UMDA	Repair 3	0.095825	0.074724	0.016606

Figures 5-top and the *wmv* file available on [14] show the ascending path followed by the successive populations in case of the E-PBIL+*vK*-Penalty algorithm leaded by the best-fit individual. However, because of the standard deviations becoming too small, this ascent ended prematurely, suggesting again that the standard deviation should be kept at larger values longer periods of time. On the other hand, as Figures 5-bottom shows, the run of the UMDA algorithm with *clone repair* was trapped in the neighborhood of a bounded local-optima without being able to advance towards it.

Problem 3 Was a difficult one to solve due to the global optima being constrained and of the numerous local optima. Again the elitist E-PBIL algorithm performed

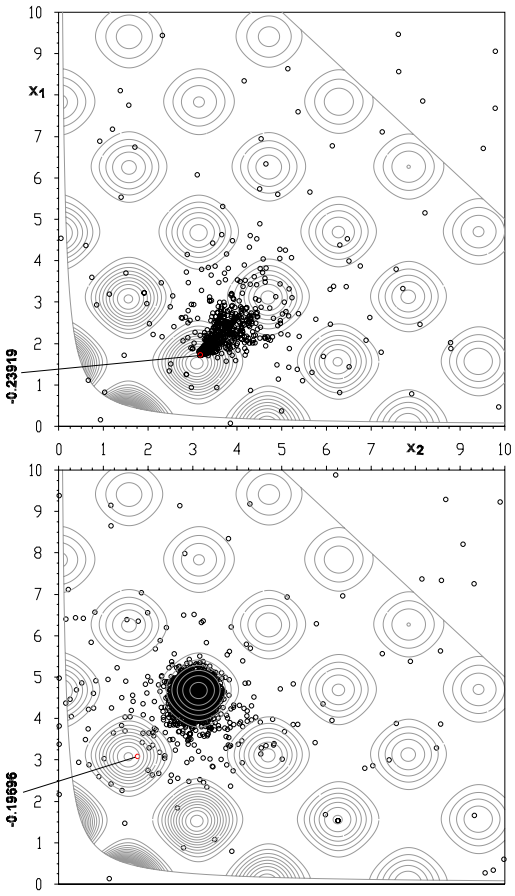

Figure 6. Superimposed plots of the points generated during one run of E-PBIL + Repair 1 (top) and E-UMDA + *DK*-Penalty (bottom) algorithms over test problem 3 with

better (see Table 3), this time in association with a line-search and crossover infeasible individual repair (remember that crossover repair is a variant of the line-search repair). It becomes evident that the favorable hill climbing characteristics of the E-PBIL algorithm were augmented by the boundary exploration capabilities provided by the repair method.

As happed before, and visible from Figure 6-up and the movie file available on [14], the search lost momentum due to a premature reduction to very small values of the components of the standard deviation vector.

For this third test problem, the lowest ranked was the elitist UMDA algorithm coupled with a *DK*-Penalty. The sample run shows an interesting behavior, in which the best fit individual is trapped around of a local optima and the rest of the population swarms around another slightly lower optima. After viewing the animated files generated with the same data as in Figure 6 below, it becomes obvious that the swarming can go forever because the standard deviation can neither go to zero nor increase enough so that the swarming population can join with the best fit individual trapped on the neighboring local optima.

TABLE 3. RESULTS OBTAINED FOR 500 RUNS OF PROBLEM 3 WITH *n*=2 FOR M=50 AND N=25 (KNOWN GLOBAL OPTIMUM: -0.3649797)

Algorithm	Constr. Handling	Best	Average	Worst
E-PBIL	Repair 1	-0.3649797	-0.239162	-0.133003
E-PBIL	Repair 2	-0.3649797	-0.235662	-0.109429
E-PBIL	*1K*-Pen.	-0.3649796	-0.224695	-0.109429
E-PBIL	Repair 3	-0.3649793	-0.223351	-0.107983
E-PBIL	*DK*-Pen.	-0.3649722	-0.226574	-0.109418
E-PBIL	*vK*-Pen.	-0.3649683	-0.227550	-0.133230
PBIL	Repair 1	-0.3649352	-0.211658	-0.109323
E-UMDA	Repair 1	-0.3639721	-0.206205	-0.101459
UMDA	Repair 2	-0.3619513	-0.205715	-0.110912
PBIL	*vK*-Pen.	-0.3606485	-0.199515	-0.101611
E-UMDA	*vK*-Pen.	-0.3600499	-0.200429	-0.109429
UMDA	Repair 3	-0.3544603	-0.192459	-0.099642
UMDA	Repair 1	-0.3529160	-0.203255	-0.100988
E-UMDA	Repair 2	-0.3528012	-0.207182	-0.117033
UMDA	*1K*-Pen.	-0.3526836	-0.200672	-0.121185
PBIL	Repair 3	-0.3525017	-0.196533	-0.099643
PBIL	*DK*-Pen.	-0.3505644	-0.20042	-0.108292
PBIL	Repair 2	-0.3493383	-0.208814	-0.104155
E-UMDA	Repair 3	-0.3474485	-0.196391	-0.108942
E-UMDA	*1K*-Pen.	-0.3461925	-0.200399	-0.109136
UMDA	*vK*-Pen.	-0.3434922	-0.195404	-0.114145
UMDA	*DK*-Pen.	-0.3397125	-0.199832	-0.103815
PBIL	*1K*-Pen.	-0.3231532	-0.197422	-0.105956
E-UMDA	*DK*-Pen.	-0.3087370	-0.197603	-0.107092

VI. CONCLUSIONS

Two Estimation of Distribution Algorithm viz the *Univariate Marginal Distribution Algorithm* and a variant of the *Population Based Incremental Learning Algorithm* were coded and tested in both elitist and non-elitist variants upon solving three continuous test objective functions with constraints.

Since no attempt has been made to maximize the performances of any of the algorithms tested, no definitive conclusion can be drawn upon which implementation and constraint handling method is the most effective, the results described herein being rather preliminary. Further experiments will be required to confirm that elitist ED algorithms coupled with line-search repair techniques are more effective in case of multimodal problems or where a bounded optima is supposed to exist.

There were some suggestions that the function's local landscape and the way the current population is distributed in this landscape should not dictate alone the probability distribution used in generating the new individuals. When normal distributions are used, forcing the standard deviation values to remain relatively large during a longer period of the search is likely to improve performance by avoiding "sinking" the population prematurely into a local optimum area. Another phenomenon that can be avoided by controlling the standard deviation values is the localized swarming in

case of elitist algorithms applied to multimodal functions (as it was the case of test problem 3), when the best fit individual is trapped on one local optima while the rest of the population swarms around a neighboring lower optima.

Conversely, the same as the gradient value can be used as stopping criteria in deterministic optimization algorithms standard deviation values can be used as stopping criteria in EDA's. This was suggested by some of the numerical examples investigated, when the main part of the search was spent by generating (almost) identical individuals due to standard deviation vectors becoming almost zero.

REFERENCES

[1] Müßhlenbein H. and Paaß, G., "From Recombination of Genes to the Estimation of Distributions," in *Parallel Problem Solving from Nature IV*, Springer, pp. 178-187, 1996.

[2] Larrañaga, P. and Lozano, J. A., Eds., *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, 2002.

[3] Larrañaga, P., "A review of Estimation of Distribution Algorithms," in [2], pp. 57-100, 2002.

[4] Paul T. K. and Iba H., "Linear and Combinatorial Optimizations by Estimation of Distribution Algorithms," Proc. of the 9th MPS Symposium on Evolutionary Computation, IPSJ, Japan, 2002.

[5] Larrañaga, P., Etxeberria, R., Lozano, J.A. and Peña, J. "Optimization in Continuous Domains by Learning and Simulation of Gaussian Networks," Proc. of the 2000 Genetic and Evolutionary Computation Conference, GECCO 2000, Las Vegas, NE, pp. 201-204, 2000.

[6] Sebag, M. and Ducoulombier, A., "Extending population-based Incremental Learning to Continuous Search Spaces," in *Parallel Problem Solving from Nature V*, pp. 418-427, Springer, 1998.

[7] Michalewicz, Z., Dasgupta, D., Le Riche, R.G. and Schoenauer, M., "Evolutionary Algorithms for Constrained Engineering Problems," *Computers and Industrial Engineering*, vol. 30, pp. 851-870, 1996.

[8] Michalewicz, Z. and Schoenauer, M., "Evolutionary Algorithms for Constrained parameter Optimization Problems," *Evolutionary Computation*, vol. 4, pp. 1 32, 1996.

[9] Coello Coello, Carlos A., "Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, pp. 1245-1287, 2002.

[10] Kuri, A., "A Universal Eclectic Genetic Algorithm for Constrained Optimization," Proc. of the 6th European Congress on Intelligent Techniques & Soft Computing, EUFIT'98, pp. 518-522, 1998.

[11] Richardson, J.T., Palmer, M.R., Liepins, G. and Hilliard, M. "Some Guidelines for Genetic Algorithms with Penalty Functions," Proc. of the 3rd International Conference on Genetic Algorithms pp. 191-197, 1989.

[12] Michalewicz, Z. and Nazhiyath, G., "Genocop III: A Co-evolutionary Algorithm for Numerical Optimization Problems with Nonlinear Constraints," Proc. of the 2nd IEEE International Conference on Evolutionary Computation, Perth, Australia, 29 November– 1 December, vol. 2, pp.647-651, 1995.

[13] Koziel, S. and Michalewicz, Z., "Evolutionary Algorithms, Homomorphous Mappings and Constrained Parameter Optimization," Evolutionary Computation, vol. 7, pp. 19-44, 1999.

[14] http://www.auburn.edu/~simiope/CEC04