A Two-Population Evolutionary Algorithm for Constrained Optimization Problems

P. A. Simionescu, G. V. Dozier and R.L. Wainwright

***Abstract*—A new approach to solving constrained nonlinear programming problems using evolutionary computations is discussed. According to the method two populations are evolved, one population (females) is evolved inside the feasible domain of the design space and a second population (males) is evolved outside this feasible domain. Both populations can be independently subject to crossover and mutation operations and the design space explored. Female-male crossover however ensures the desirable increase in the search pressure upon the boundaries of the feasible space - it is known that in many optimization problems the global optimum is bounded. The experiments performed on three test objective functions of two variables show some promise of the proposed approach in that it can cope with both linear and nonlinear constraints and with nonconvex feasible domains.**

I. Introduction

THE general nonlinear programming (NLP) problem requires finding the optimum point (minimum or maximum) of a function of *n* real variables:

$$F(x_1,...x_i,...x_n) \quad (1 \le i \le n) \tag{1}$$

subject to the following side constraints:

$$x_{i\min} \le x_i \le x_{i\max} \quad (1 \le i \le n) \tag{2}$$

and inequality and equality constraints:

$$g_j(x_1,...x_n) \le 0 \quad (1 \le j \le m) \tag{3}$$

$$h_j(x_1,...x_n) = 0 \quad (1 \le j \le p). \tag{4}$$

Since equality constraints can be embedded in the objective function using penalty methods, or can be eliminated together with some of the variables [1],[2], only NLP problems subject to inequality constraints will be discussed.

In recent years many researcher have reported on solving the general NLP problem using evolutionary algorithms [3]-[7]. One of the main differences between various approaches lay in constraint-handling the techniques employed as follows: **1)** various implementations of the penalty function 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) **5)** hybrid algorithms etc.

According to [6],[7], one of the main reasons for difficulties in locating the global solution is the inability of evolutionary systems to search precisely the boundary area between the feasible and infeasible region of the search space. It is known that in many constrained optimization problems, some of the constraints are active at the global optimum point, making it more difficult to locate.

The present paper proposes a two-population evolutionary computation approach that aims at increasing the search pressure upon the boundaries of the feasible space. This is done by evolving one population of feasible individuals, and a second population of infeasible individuals. The individuals evolving inside (including the boundaries of the feasible space) are called *females*, while the individuals evolved outside the feasible space are called *males*. The female and male populations can be subject to asexual crossover and independent mutation (but not necessarily). The boundary-search effect however is ensured by female-male crossover, which can be performed between two or more feasible and infeasible individuals, and with, or without, favoring the better fit females or the better ranked males. Male ranking can be done based on the number of constraints they violate, on their mating successes or on their capacity of generating (superior) offspring.

Segregated, speciated and multi-population evolutionary models have been proposed by a number of researchers in the past [8]-[16]. It is believed however that this is the first time when two populations, one of feasible and one of infeasible individuals, are distinctively evolved, and that these two populations interact systematically (rather than occasionally) for the purpose of exploring the boundaries of the feasible space. Since the criterion based on which individuals are assigned to the two populations does not change during evolution, the behavior of the two populations is easier to understand. Visualizations of the two populations (as accumulated graphs and animated frames) as they explore the landscape of several test objective functions of two variables, are also provided in the paper. Such visualizations can further enhance the understanding of how various implementations of the algorithm work, helping with parameter tuning (which is known to be an optimization problem in itself).

II. General Description of the Algorithm

Implementations of the proposed approach will be further named *female-male evolutionary algorithms*, or F-M(μ_F,μ_M)

P. A. Simionescu is with the Department of Mechanical Engineering, The University of Tulsa, 600 S. College Ave., Tulsa, OK 74104 USA (corresponding author: e-mail: psimionescu@utulsa.edu).

G. V. Dozier is with the Department of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849 USA, (e-mail: gvdozier@eng.auburn.edu).

R.L. Wainwright is with the Department of Mathematical and Computer Sciences, The University of Tulsa, 600 S. College Ave., Tulsa, OK 74104 USA (e-mail: rogerw@utulsa.edu).

in short, where μ_F is the size of the female population and μ_M is the size of the male population.

A general structure of the algorithm is given below, with the observation that the square-bracketed steps, or portions of a step, can be omitted in some implementations:

Step 1) Generate the initial female population of μ_F individuals and the male population of μ_M individuals;
repeat
Step 2: [Rank females];
Step 3: [Rank males];
Step 4: [Mutate females];
Step 5: [Mutate males];
Step 6: Make female-male pairs and generate offspring;
Step 7: Select survivors;
until a certain **Stopping Criteria** *is met.*

The following is a more detailed discussion of the steps introduced above:

Step 1: Uniform random points are generated within the extended intervals:

$$\left[x_{i\min} - b_L \cdot \Delta x_i, x_{i\max} + b_R \cdot \Delta x_i\right] \quad (1 \le i \le n) \qquad (5)$$

where

$$\Delta x_i = \left(x_{i\max} - x_{i\min}\right). \qquad (6)$$

Coefficients b_L and b_R (usually between 0 and 1) control the amount in which the infeasible space is expanded so that a male population can be generated and evolved, even when only side constraints are imposed to the optimization problem. If additional constraints are present, the infeasible region may be sufficiently large and the respective coefficients can be set to smaller values, including zero. If desired, different b_L and b_R coefficients can be assigned for different variables x_i, as well as extending only the lower boundaries or only the upper boundaries from their initial values $x_{i\min}$ and $x_{i\max}$. Evidently, when the objective function is evaluated, the side constraints are verified as they were posed in the original problem.

If a randomly generated individual belongs to the feasible region of the design space (i.e. no constraints are violated), then it will be assigned to the female population, else will be assigned to the male population. The process continues until μ_F females and μ_M males are generated.

As the program iterates toward generating the initial female and male populations, further useful information about the problem at hand can be acquired: For example, the ratio of the size of the feasible-space over the total size of the search space can be estimated. This ratio, noted ρ, was determined experimentally in [5], prior to solving the optimization problem, by generating a large number of uniform random points inside the search space and counting the number of occurrences of feasible points from the total number of points generated. Knowledge about this ratio can be used in female-male crossover schemes, or in dynamically adjusting the upper and lower limits of the search space, so that the probabilities of randomly generating female and male individuals are balanced.

On the other hand, if ρ remains zero after a given large number of function-calls, it is a sign that the optimization problem is highly-constrained and the user can be prompted to reformulate it, or to provide at least one initial feasible point to the algorithm. In more elaborate implementations, similar to [17],[18], the infeasible individuals can be ranked based on the number of constraints they violate and evolved using asexual crossover and/or mutation operations, until an initial pool of feasible individual are generated, so that the algorithm can continue with the iteration of **Steps 2-7**.

Step 2 (optional step): Rank females based on their fitness, using one of the known deterministic or stochastic procedures (i.e. complete sorting, partial sorting, roulette wheel, tournament selections etc.), or the step can be limited to identifying the best-fit female only which will be further called α-female.

Step 3 (optional step): Rank males based on the number of constraints they violate, on their capacity to generate [improved] offspring, on their mating successes (number or "quality" of mates), or simply assign to them the rank of the [best fit] female they mated with.

Step 4 (optional step): Mutate females by replacing [the worst ranked] p_F % females with randomly generated new females. If female ranking was performed, the newly created females can take the rank of the females they replace (rank inheriting), or **Step 2** can be applied one more time until every female has a rank of her own. If no female ranking is performed, then females are mutated at random with, or without preserving the α-female.

Step 5 (optional step): Mutate males (the same as it was done with the female population), with a mutation rate of p_M %, with or without preserving the α-male (the male that mated during the previous step with the α-female). If no male ranking was performed, simply mutate at random p_M % of the male population.

Step 6 : Form female-male pairs by assigning [at least] one male to each female based on their closeness in the *n*-dimensional Euclidean space. If females ranking was applied, then females can choose their mates in a rank-decreasing order. In some implementations further called *monogamous-male algorithms*, once a male has been assigned to a female, the respective male will not be available for further mating during the same generation. In other implementations called *polygamous-males algorithms*, males are permitted to recombine with more than one female. It is obvious that for *monogamous-male algorithms* with $\mu_F > \mu_M$, there will be some females that will not participate in the crossover operation, while for *polygamous-male algorithms* this can be avoided. *Polygamous-male algorithms* can be *unrestricted* i.e. a male can crossover with any number of females, or *restricted* when the number of crossovers a male can perform in one generation is limited to a fraction of the total female population.

After female-male pairs are formed, offspring are generated using one of the known crossover schemes [16]. The offspring can result inside the feasible space (viz. they are females) or outside the feasible space (viz. they are

males). Alternatively, a [random] search can be performed along the direction connecting the female with her pair male, until one female and/or one male offspring are generated to replace either one or both parents.

It is to be expected that in *unrestricted polygamous-male algorithms*, where females mate with their closest males, the offspring will rapidly migrate towards the feasible-unfeasible boundary region (particularly if midpoint crossover or other convex combination scheme is employed), which can lead to a stagnation of the two populations. In order to reduce this phenomenon and ensure an exploration of the design space parallel to the feasible-unfeasible boundary, multi-parent recombination [19], mutation operations and/or asexual crossover could be applied. In some instances however this can be a favorable characteristic which can be used in conducting [final] localized searches, (by adding that "killer instinct" most evolutionary algorithms lack), or in conducting post-optimal constraint-activity analyses.

Conversely, *monogamous-male algorithms* have the inherent property that the search is performed both towards and parallel to the feasible-infeasible boundary, which ensures a better global searching capability.

Step 7: For convenience, the selection step should be performed as offspring are generated rather than a distinct step, following one of the rules: If the child results outside the feasible space, then he may or may not replaces his father. If the child is a female, she replaces her mother either unconditionally or only if the child has a better fitness than that of the mother. In *polygamous-male* type implementations of the algorithm, both parents can be replaced, or only the mother (either unconditionally or only if there is a fitness improvement between mother and child).

Stopping Criteria: Steps 2 through 7 are repeated until an imposed condition is satisfied; this can be exceeding a maximum number of function calls, attaining a given threshold fitness, or recording the same α-female over a given number of generations.

III. Test Functions

The effectiveness of one *monogamous male* and one *polygamous male* implementations of the F-M(μ_F,μ_M) algorithm was tested against solving the following three problems:

The first two test problems require minimizing the classical Rosenbrock's function:

$$F(x_1, x_2) = 100 \cdot (x_2 - x_1^2)^2 + (x_1 - 1)^2 \tag{7}$$

subject to the following constraints:

$$-2 \le x_1 \le 2 \quad \text{and} \quad -1 \le x_2 \le 3. \tag{8}$$

$$g_1(x_1, x_2) \le 0 \ \text{ AND } \ g_2(x_1, x_2) \le 0 \ \text{ for problem 1} \tag{9}$$

$$g_1(x_1, x_2) \ge 0 \ \text{ OR } \ g_2(x_1, x_2) \ge 0 \ \text{ for problem 2} \tag{10}$$

with g_1 and g_2 cubic and linear functions as follows:

$$\begin{aligned} g_1(x_1, x_2) &= (x_1 - 1)^3 - x_2 + 1 \\ g_2(x_1, x_2) &= x_1 + x_2 - 2 \end{aligned} \tag{11}$$

Both problem 1 and 2 have their global minimum equal to 0 located at (1,1) where g_1 and g_2 are active. The feasible space of problem 2 is convex, while that of problem 1 is not (see also Appendix). In case of problem 1 there is also a local minima equal to 1 at (0,0) where only g_1 is active.

The third test problem from [17] requires minimizing:

$$F(x_1 ... x_n) = \frac{\left| \sum_{i=1}^{n} \cos^4(x_i) - 2 \prod_{i=1}^{n} \cos^2(x_i) \right|}{\sqrt{\sum_{i=1}^{n} i \cdot x_i^2}} \tag{12}$$

subject to:

$$0 \le x_i \le 10 \quad \text{for} \quad 1 \le i \le n \tag{13}$$

and

$$\begin{aligned} g_1 &= \sum_{i=1}^{n} x_i - 7.5n \le 0 \\ g_2 &= 0.75 - \prod_{i=1}^{n} x_i \le 0 \end{aligned} \tag{14}$$

This is a highly multimodal function the optimum of which is constrained by g_2 only. For n=2 the known optimum -0.36497974587 occurs for x_1=1.60086042842878 and x_2=0.46849805684566.

IV. Numerical Results

(1) The *female monogamous-male algorithm*, or F-*m*M(μ_F,μ_M) in short, was implemented and tested in the following form:

Step 1: Generate the initial μ_F female and μ_M male populations by uniform random generating points within the extended intervals.

Repeat **Steps 2-7** *below, until* **NF** *function calls has been exceeded* (**NF**=4000).

Step 2: Do a complete sorting of the female population based on fitness.

Step 3: If a new α-female emerged, then skip **Step 4** for *ns* successive generations (*ns* was set equal to the number of variables of the objective function). This will allow the search to further advance toward a possible bounded optimum, undisturbed by mutation.

Step 4: Generate uniform random points (feasible or infeasible) within the extended intervals. Replace with these newly generated random points some of the female individuals, starting with the worst fit female, and at random some of the male individuals until p_F% new females are replaced OR p_M% new males are replaced. (The OR operator ensures that no unnecessary function calls are made, although some coupling between the interval expansion coefficients b_L and b_R and the mutation rates p_F and p_M is introduced. The *polygamous-male* implementation of the algorithm, that will be described next, employs a slightly different mutation step, which avoids such a coupling and which can be used in this present algorithm as well).

TABLE I
RESULTS OBTAINED WITH THE FEMALE-MONOGAMOUS-MALE ALGORITHM FOR 1000 TRIALS ON PROBLEMS 1, 2 AND 3

	Problem 1	Problem 2	Problem 3
b_L	0.1	0.1	0.2
b_R	0.1	0.1	0.0
μ_F	20	20	20
μ_M	15	15	15
p_F	15%	15%	25%
p_M	35%	35%	30%
NF	4000	4000	4000
Generations	196	196	187
F best	0.000000586	0.000000037	-0.364979727
g_1	-0.000013317	0.000069561	2.930685128
g_2	-0.000018270	-0.000094797	0.000000028
F worst	1.290982881	0.262304861	-0.306763216
g_1	-0.000007618	-0.565534200	13.077046033
g_2	-2.067034206	-0.817545972	0.000007806
F average	0.10283541	0.03382733	-0.36078237

Step 5: Make female-male pairs starting with the best fit female, such that the current female recombines with the closest available male. Once a male has been assigned to a female, he will not be allowed to mate again during the same generation. Evidently, if $\mu_F \neq \mu_M$ then either some of the worse fit females or some of the distant-from-the-boundary males will not participate in the crossover operation.

Step 6: Apply a mid-point crossover between the female and the male in every pair;

Step 7: If the offspring is a female, she replaces her mother unconditionally, less if the mother is the α-female, case in which the replacement takes place only if a new best-fit female emerged. If the offspring is a male, he always replaces his father.

The results obtained for 1000 runs over the above three test problems were summarized in Table I. In case of problem 1, for about 3% of the runs the algorithm converged to the (0,0) local optimum rather then the global optimum (Fig. 1). For problem 2, the 1000 solutions found are moderately dispersed along the curved valley which harbors the (1,1) optimum point (Fig. 2-above), suggesting that the algorithm has some capability of identifying unconstrained optima, not only bounded global optima. For problem 3, all 1000 solutions appear clustered along the boundary close to

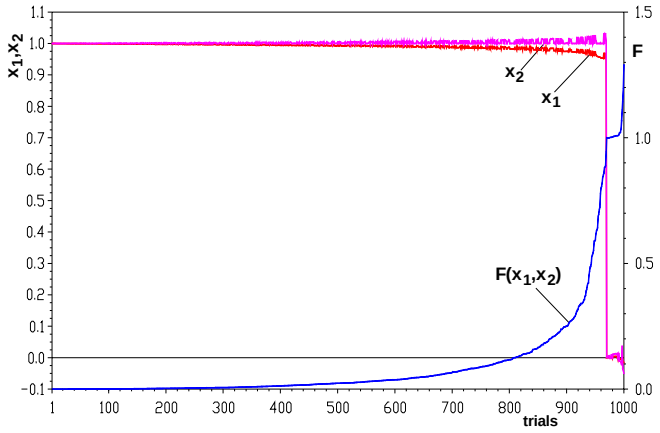

Fig. 1 Results obtained after1000 runs of the F-mM(μ_F,μ_M) algorithm for test problem 1 sorted by the best fitness found. The search converged to the global optimum point (1,1) for 96.9% of the trials.

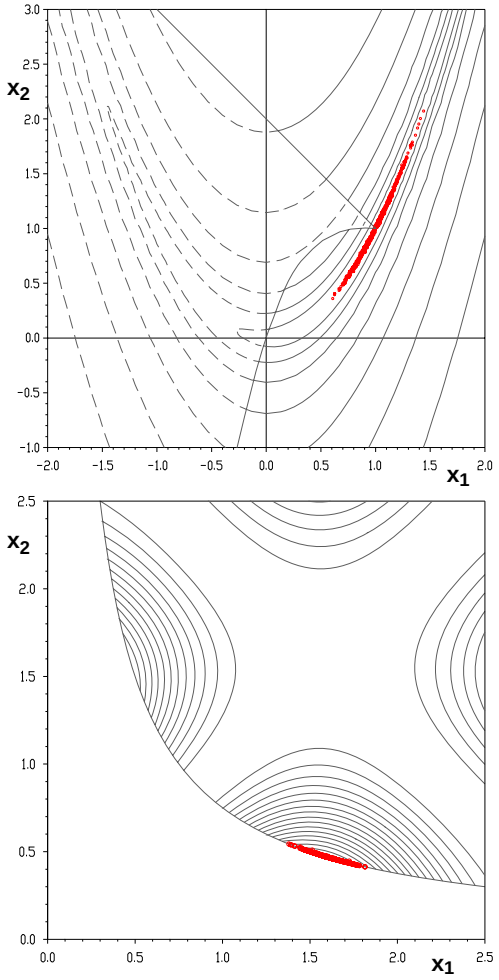

Fig. 2 Results obtained after 1000 runs of the F-mM(μ_F,μ_M) algorithm for test problems 2 (above), and test problem 3 (below).

where the global optimum is located (Fig. 2-below).

From Figure A1 in Appendix, which shows superimposed on the function graph all male and female individuals from one sample run, it can be seen that the search was performed evenly over the boundary of the feasible space, with some occasional clustering. These graphs also indicate that by properly choosing the values of the boundary-expansion coefficients b_L and b_R (as it was done in case of problem 3), it is possible to voluntarily direct the search in certain areas of the design space.

(2) The *female-polygamous-male algorithm* or F-pM (μ_F,μ_M), was implemented in a *restricted* form, with the goal of enhancing its global searching capabilities, as follows:

Step 1: Generate the initial μ_F female and μ_M male populations by uniform random generating points within the extended intervals.

Repeat **Steps 2-6** *below, until* **NF** *function calls has been exceeded.*

Step 2: Do a complete sorting of the female population based on fitness.

Step 3: Make female-male pairs beginning with the best fit female, such that the current female recombines with the closest available male. Male are not allowed to mate more

TABLE II
RESULTS OBTAINED WITH THE FEMALE-POLYGAMOUS-MALE ALGORITHM FOR 1000 TRIALS ON PROBLEMS 1, 2 AND 3

	Problem 1	Problem 2	Problem 3
b_L	0.1	0.1	0.2
b_R	0.1	0.1	0.0
μ_F	20	20	8
μ_M	6	50	9
p_{FM}	25%	50%	25%
NF	4000	4000	4000
Generations	41	35	129
F best	0.000041251	0.000000882	-0.364950490
g_1	-0.000048463	0.000811285	12.928955299
g_2	-0.000248143	-0.001173672	0.000043306
F worst	1.215703072	0.516990985	-0.271542669
g_1	-0.016550336	-0.573121612	12.924986606
g_2	-1.940283631	-1.634762423	0.005108947
F average	0.27181687	0.07094794	-0.36197287

than M times during the same generation. It means that for $\mu_F > M \cdot \mu_M$, some of the worse fit females will not take part in crossover. In the numerical tests performed, M was set to a constant value equal to $\mu_F/3$. M can be assigned any value between 1 and μ_F; this will change the characteristics of algorithm from a *monogamous-male* all the way to an *unrestricted polygamous male* algorithm. Preliminary investigations show that there is some promise in doing this adjustment during the search, in an adapting fashion, controlled by the number of successive generations that preserve the same α-female.

Step 4: Apply a mid-point crossover between the female and male forming a pair.

Step 5: If the offspring is a female, she replaces her mother only if there is an improvement in fitness. The male population does not change during crossover; though the males that mated with the best and the second best fit females (called the α-male and the β-male) will be marked so that they can receive different treatment during the mutation step (see **Step 6**).

Step 6: If the α-female did not change during **Step 5**, then the two populations are subject to mutation as follows: Generate $p_{FM}(\mu_F+\mu_M)$ uniform random points (feasible or infeasible) within the extended intervals. Use these random points to replace the existing females starting with the worst fit female (female population ranking performed at **Step 2** is assumed unchanged), and replace the existing males at random, but only after the α-male and the β-male are replaced (the β-male is the male ranked second).

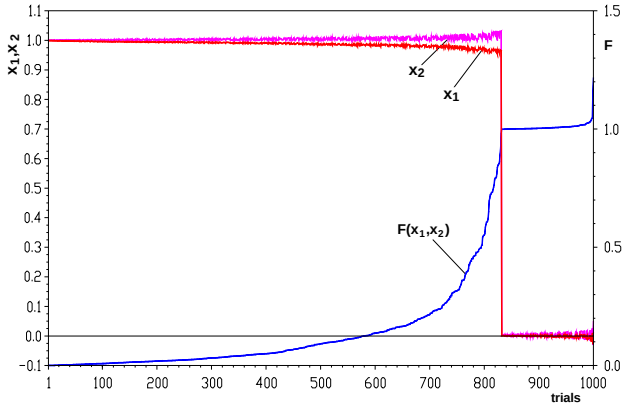

Fig. 3 Results obtained for 1000 trials of the F-pM(μ_F,μ_M) algorithm for test problem 1 sorted by the best fitness found. The search converged to the global optimum point (1,1) for 83.2% of the trials.

.

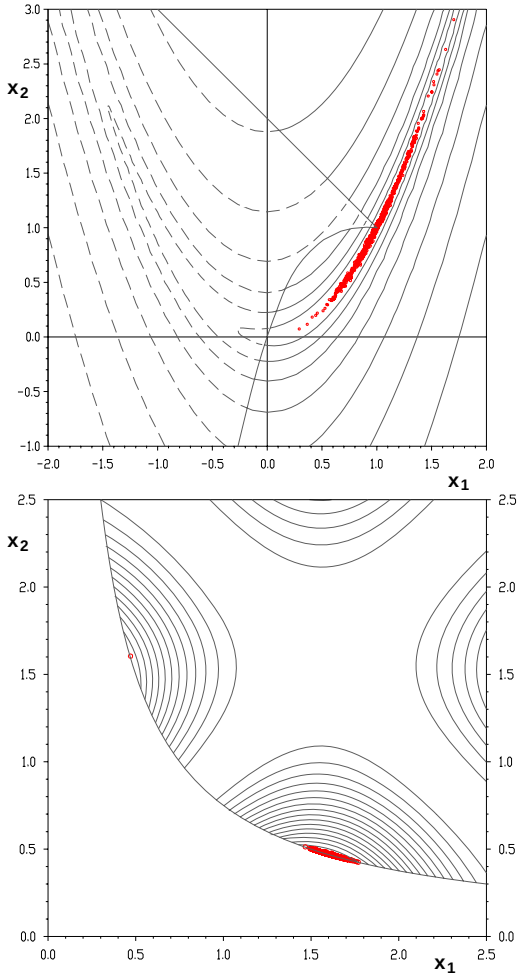

Fig. 4 Results obtained after 1000 runs of the F-pM(μ_F,μ_M) algorithm for test problems 2 (above) and test problem 3 (below).

The results obtained for 1000 runs in case of the same test problems are summarized in Table 2 and Figs. 3 and 4. For **NF**=4000 the success rates, variability of results and accuracy were lower than before for all three problems tested. This should not be automatically interpreted as a lack of merit of the proposed implementation, nor of the polygamous algorithms in general (which must performed additional function calls required by the directional searches). Actually the authors of the present paper experienced some promise with an *unrestricted* variant of the *female-polygamous-male algorithm*, showing that further investigation is needed before a final verdict can be reached.

Figure A2 in Appendix shows all male and female individuals for one complete run for the considered test problems (less the intermediate points generated during the directional searches). It can be seen that the clustering is now more prominent in the promising areas of the search

space for all three test functions, suggesting that this F-*p*M algorithm has indeed some favorable traits.

V. Conclusions And Further Studies

A new approach for solving constraint nonlinear programming problems using a co-evolutionary algorithm was described. The main goal was to induce an increased exploratory pressure upon the boundaries of the feasible domain, and it was ensured by the interaction between the feasible and infeasible individuals forming two distinct populations called females and males respectively.

Searching capabilities inside the feasible space and parallel to the feasible-infeasible boundary were induced by mutating the female and male populations and limiting the maximum number of mates a male can have during the same generation. BLX crossover [20] between females and males (rather then the midpoint crossover as currently experimented with), can also be employed in order to enhance the global searching capabilities of the algorithm.

One merit of the proposed approach is that the constraints are handled in a very simple form (when evaluated outside the infeasible region the objective function simply returns a constant large value). However, for highly constrained problems where a starting feasible point is not available, male ranking based on the number of constraints they violate can be applied, and asexual crossover performed within the infeasible population, until a small pool of feasible individuals emerge.

Two variants of the algorithm called *monogamous-male* and *restricted-polygamous-male algorithms* were tested against minimizing three constrained objective functions and some encouraging results obtained.

Since no penalty factor is involved in constraint handling, a clear distinction is made between the feasible and infeasible individuals. This trait, coupled with a small number of parameters to be adjusted at the beginning of the run make the proposed approach easy to comprehend by the inexperienced user, particularly if graphical representations and animations are studied for two dimension optimization problems as discussed in this paper.

Numerous other variants of *female-male evolutionary algorithms* remain to be implemented and tested, making them very appealing, similar to pair-figure-skating which is considered (at least by some viewers), more entertaining than individual elite skating.

Appendix

Fig. A1 shows superimposed plots of all the female (circles) and males (diamonds) generated for one run of the F-*m*M(μ_F,μ_M) algorithm in case of test problems 1 (top), problem 2 (middle) and problem 3 (bottom). Fig. A2 shows superimposed plot of all the female (circles) and males (diamonds) generate for one run of the F-*p*M(μ_F,μ_M) algorithm in case of test problems 1 (top), problem 2 (middle) and problem 3 (bottom). (The points generated during the directional searches were not represented).

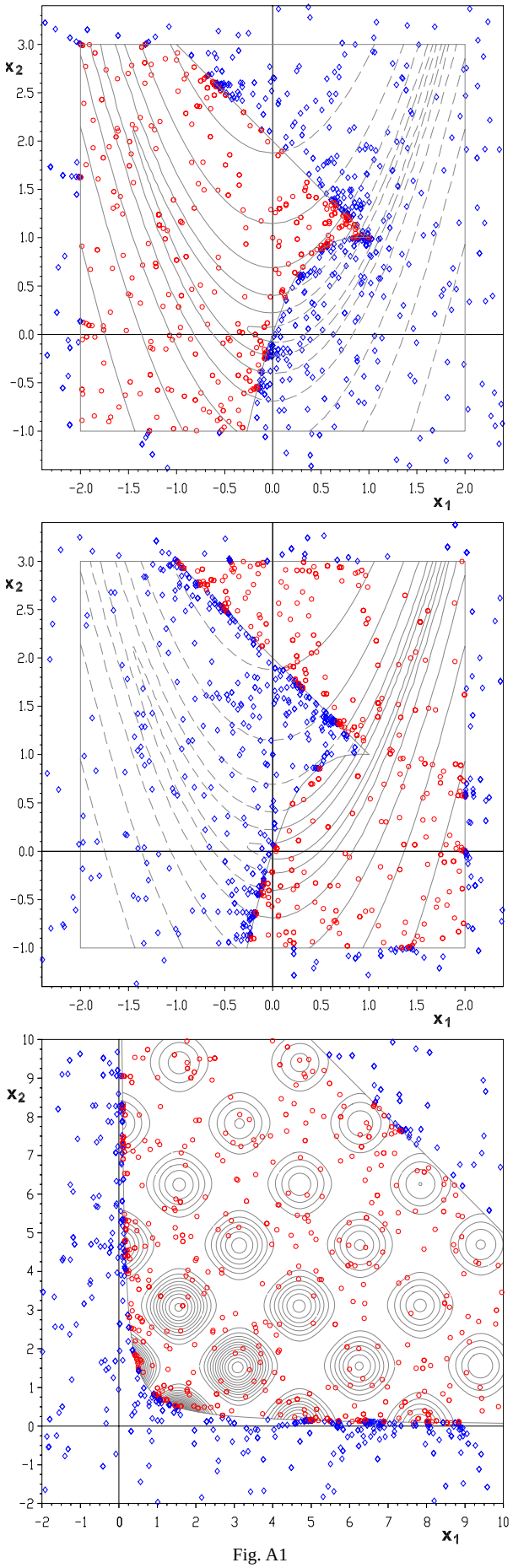

Fig. A1

Animation files of the successive populations used to generate Figs. A1 and A2 are available upon request.

REFERENCES

[1] P. Y. Papalambros and D. J. Wilde, *Principles of Optimal Design: Modeling and Computation*, 2nd edition, Cambridge University Press, 2000.

[2] T. Bäck, D. Fogel and Z. Michalewicz, *Handbook of Evolutionary Computation*, The Institute of Physics Publishing, 2000.

[3] C. A. Coello Coello, "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, No. 11-12, pp. 1245-1287, 2002.

[4] C. A. Coello Coello, "List of References on Constraint-Handling Techniques used with Evolutionary Algorithms," (last updated Jan. 2006): http://www.cs.cinvestav.mx/~constraint/

[5] Z. Michalewicz and M. Schoenauer, "Evolutionary Algorithms for Constrained Parameter Optimization Problems," Evolutionary Computation, Vol. 4, No. 1, pp. 1-32, 1996.

[6] Z. Michalewicz, D. Dasgupta, R. G. Le Riche and M. Schoenauer, "Evolutionary Algorithms for Constrained Engineering Problems," Computers and Industrial Engineering, Vol. 30, No. 4, pp. 851-870, 1996.

[7] M. Schoenauer and Z. Michalewicz, "Sphere Operators and Their Applicability for Constrained Parameter Optimization Problems," The 7th Annual Conf. on Evolutionary Programming, San Diego, CA, pp. 241-250, 1998.

[8] D. A. Sekharan and R.L. Wainwright, "Manipulating Subpopulations of Feasible and Infeasible Solutions in Genetic Algorithms," ACM/SIGAPP Symposium on Applied Computing (SAC'93), Indianapolis, Feb. 14-16, pp. 118-125, 1993.

[9] J. Paredis, "Co-evolutionary Constraint Satisfaction," The 3rd Conf. on Parallel Problems Solving from Nature, Springer, pp. 46-55, 1994.

[10] Z. Michalewicz and G. Nazhiyath, "Genocop III: A Co-evolutionary Algorithm for Numerical Optimization Problems with Nonlinear Constraints," The 2nd IEEE Intl Conf. on Evolutionary Computation, IEEE Press, pp. 647-651, 1995.

[11] R. Le Riche, C. Knopf-Lenoir and R. T. Haftka, "A Segregated Genetic Algorithm for Constrained Structural Optimization," The 6th Intl. Conf. on Genetic Algorithms, pp. 558-565, 1995.

[12] P. M. Todd and G. F. Miller, "Biodiversity through Sexual Selection," Proc. of the 5th International Workshop on the Synthesis and Simulation of Living Systems," Cambridge MA, MIT Press, pp. 289-299, 1997.

[13] P. B. Nair and A. J. Keane, "Coevolutionary Genetic Adaptation: A New Paradigm for Distributed Multidisciplinary Design Optimization", The 40th AIAA/ASME/ASCE/AHS/ASC Structures Structural Dynamics and Materials Conf., St. Louis, MO, AIAA Paper No. 1999-1428, 1999.

[14] C. A. Coello Coello, "Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems," Computers in Industry, Vol. 41, No. 2, pp. 113-127, 2000.

[15] Sana Ben Hamida and M. Schoenauer, "An Adaptive Algorithm for Constrained Optimization Problems," The 6th Conf. on Parallel Problem Solving from Nature, Springer, pp. 529-538, 2000.

[16] S. O. Kimbrough, L. Ming and D. H. Wood. "Exploring the Evolutionary Details of a Feasible-Infeasible Two-Population GA," The 8th Parallel Problem Solving From Nature, Birmingham UK, pp. 292-301, 2004.

[17] M. Schoenauer and S. Xanthakis, "Constrained GA Optimization," The 4th International Conference on Genetic Algorithms, Urbana Champaign, Morgan Kaufmann, 1993.

[18] R. Hinterding and Z. Michalewicz, "Your Brains and My Beauty: Parent Matching for Constrained Optimization," The 5th International Conf. on Evolutionary Computation, Anchorage, pp. 810-815, 1998.

[19] A.E. Eiben, "Multiparent Recombination in Evolutionary Computing," in A. Ghosh and S. Tsutsui, editors, *Advances in Evolutionary Computing, Natural Computing Series*, Springer, pp. 175-192, 2002.

[20] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," in Foundations of Genetic Algorithms-2. San Mateo, CA, Morgan Kaufman, pp. 187-202, 1993. BLX

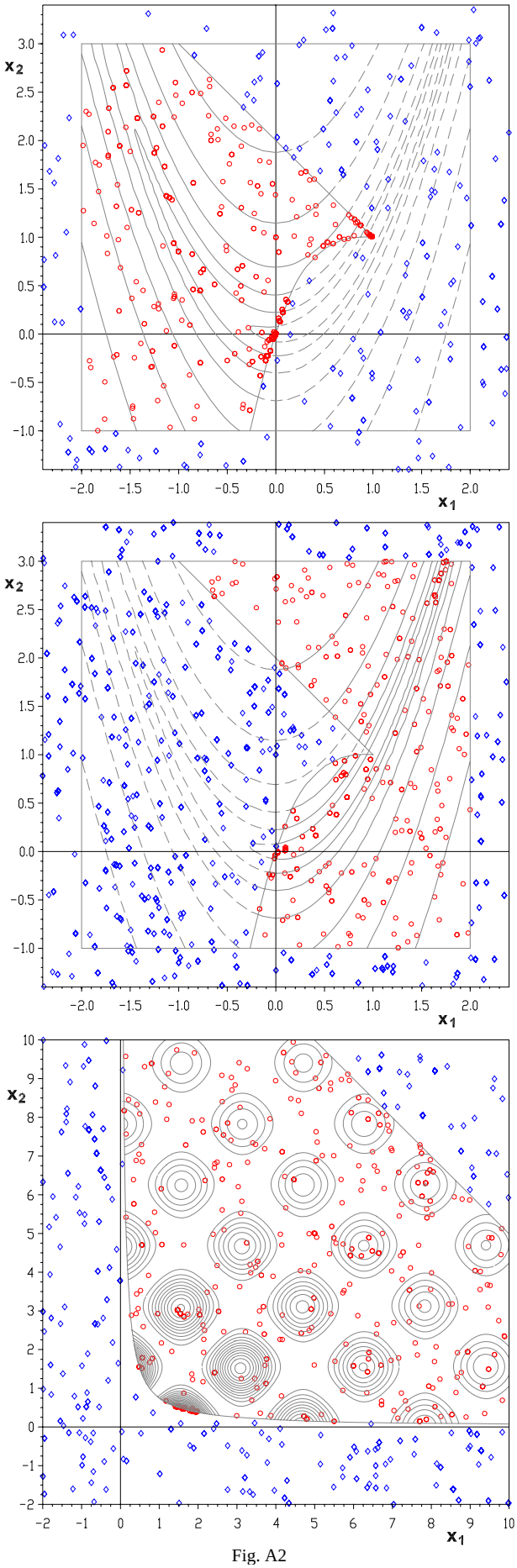

Fig. A2