A Non-Generational Genetic Algorithm for Multiobjective Optimization

Manuel Valenzuela-Rendón and Eduardo Uresti-Charre
Centro de Inteligencia Artificial
ITESM, Monterrey
Sucursal de Correos "J," C.P. 64849 Monterrey, N.L., Mexico
mvalenzu@campus.mty.itesm.mx, euresti@campus.mty.itesm.mx

Abstract

This paper describes a non-generational genetic algorithm for multiobjective optimization. The fitness of each individual in the population is calculated incrementally based on the degree in which it is dominated in the Pareto sense, or close to other individuals. The closeness of individuals is measured using a sharing function. The performance of the algorithm presented is compared to previous efforts on three multiobjective optimization problems of growing difficulty. The behavior of each algorithm is analyzed with regard to the visited search space, the quality of the final population attained, and the percentage of non-dominated individuals in the population through time. According to all these performance measures, the algorithm presented clearly outperforms previous efforts based on generational genetic algorithms.

1 INTRODUCTION

It can be said that true optimization must be multiobjective; *real problems* usually involve more than one objective function to be optimized; usually, these functions cannot be combined into a single one. Consider, for example, the design of an electronic device. One could wish to maximize the performance of the device given a set of desired electronic characteristics, to minimize the mean time between failures of the device, and to minimize the cost of manufacturing the device. These objectives cannot be combined into a single function because they are *incommensurables,* i.e., they measure different properties that cannot be related to each other directly. One particular design might give very high performance at a high cost; another design might achieve low cost by reducing the performance of the device and increasing the time between failures. Without any information regarding the preferences of the target user of the device, one cannot say that one design is better than another.

Genetic algorithms (GAs) (Goldberg, 1989; Holland, 1975) evolve populations of candidates solutions; they are not methods based on moving a single point in the search space. Because of this, the product of a GA search is not a single individual, but rather the information contained in the whole population. Nevertheless, in many applications GAs are used to find a single solution. One could say that in most applications GAs are used in a way that does not exploit their natural abilities. The solution to a multiobjective optimization problem is a set of many, possible infinite, points. We believe, based on this fact, that multiobjective optimization is an appropriate ground for a GA approach to display superior performance.[1] Louis and Rawlings (1993) recognized this point before; they suggested that multiobjective optimization problems "may provide fertile ground for finding search spaces in which GAs do better than other blind search algorithms."

In this paper, we present an approach to multiobjective optimization that employs a non-generational GA similar to the one usually employed in learning classifier systems (LCSs). We believe that the multiobjective optimization problem has characteristics similar

[1] It is possible to view a set of points that constitute a solution to a multiobjective optimization problem as a single point in a very complex space. For example, a *genetic programming* approach (Koza, 1992) could be implemented to find an analytical expression for the curve that touches all the points in the solution set. Nevertheless, this approach to multiobjective optimization seems hard to implement given the difficulty in comparing and assigning an evaluation to tentative solutions.

to those faced by the GA in an LCS, and thus, a non-generational GA approach should perform significantly better than a generational GA.

2 DEFINITIONS

In a multiobjective optimization problem, the objective is to optimize a set of functions that depend on a set of independent variables. Without loss of generality, we will limit ourselves to pure minimization problems, i.e., problems in which all the objective functions are to be minimized.

Definition 1 *A point* $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k})$ *is said to* dominate *another point* $\mathbf{y} = (\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_k})$ *if* $\forall i \; x_i \leq y_i$ *and* $\exists \; i_0 \mid x_{i_0} < y_{i_0}$.

Definition 2 *A point* $\mathbf{x}$ *is a member of the* Pareto front *if there is no other point* $\mathbf{y}$ *such that* $\mathbf{y}$ *dominates* $\mathbf{x}$.

Definition 3 *Given a set of* m *real functions defined over a subset* D *of* $\Re^k$, $f_j : D \to \Re$, *where* $j = 1, 2, \ldots, m$, *the* multiobjective optimization problem *consists in finding a subset* P_f *of* $D \in \Re^k$ *such that the image of* P_f *is the Pareto front.*

The multiobjective optimization problem has characteristics different from a single optimization problem where only one solution is sought. These characteristics require a different approach if a GA is applied to find its solution: the solution is a set of points, the performance of an individuals depends on other individuals, individuals cannot be evaluated absolutely, and the solution set is potentially infinite. The last characteristic supports the idea of using a GA with some mechanism for niche formation.

3 FITNESS SHARING

Fitness sharing (Goldberg & Richardson, 1987; Deb & Goldberg, 1989) is the most successful mechanism for niche formation. Through the use of fitness sharing, a GA can find all the maxima of a multimodal optimization problem in a single run. Fitness sharing sees each maximum as an ecological niche. Individuals that belong to the niche must share resources. Each individual calculates a *niche count*, or m_i, which is a weighted sum of the number of individuals belonging to its niche, i.e., that lie close to it according to a given definition of distance: $m_i = \sum_{j \in \text{pop}} \text{sh}(d_{ij})$, where d_{ij} is the distance between the individual i and the individual j. The function $\text{sh}(d)$ defines a neighborhood and the degree of sharing for each individual:

$$\text{sh}(d) = \begin{cases} 1 - \left(\dfrac{d}{\sigma_s}\right)^{\alpha}, & \text{if } d < \sigma_s; \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The fitness of an individual is calculated by taking the the evaluation of the objective functionand dividing it by the niche count: $f_i' = f_i / m_i$. A GA with fitness sharing will produce stable subpopulations on the maxima of a multimodal optimization problem.

4 PREVIOUS EFFORTS

Most multiobjective optimization problems are usually transformed into single objective problems. Other efforts using GAs have been made to attack the complete problem in order to find a Pareto front (Schaffer, 1985; Fonseca & Fleming, 1993; Horn, Nafpliotis, & Goldberg, 1994; Srinivas & Deb, 1995).

Horn, Nafpliotis, & Goldberg developed the *niched Pareto GA* (nPGA). The niched Pareto GA (nPGA) is a generational GA with no generational overlap[2] and *binary tournament selection.* In binary tournament selection (Goldberg & Deb, 1991), two individuals are chosen from the population and compared. The best individual according to the definition of fitness is selected. This procedure is repeated until the mating pool is filled. In the nPGA, each time the tournament selection requires to compare two individuals, the best one is determined according to the following factors: domination of a by b or domination of b by a, domination of a or b by at least one individual in a subset of the population, and proximity of other individuals in the population according to a sharing function. These factors are considered in a hierarchical fashion. Figure 1 gives the pseudo-code that implements the algorithm proposed in (Horn & Nafpliotis, 1993; Horn et al., 1994). This algorithm incorporates the idea of *continuously updated sharing* (Oei, Goldberg, & Chang, 1991); according to it, the niche count, m, of an individual is calculated comparing it to a sample of the partially filled next generation. This idea was proposed to avoid the chaotic behavior that is produced by performing fitness sharing and tournament selection in conjunction.

Horn et al., set the nPGA to solve several problems, including two problems presented in (Schaffer, 1985). The results they obtained indicate that the nPGA finds a more evenly distributed population over the Pareto front.

[2] *Generational overlap* is the fraction of the population that is not replaced from one generation to another in a generational GA.

```
Select(a, b)
(* This function receives two individuals a and b
and selects one. *)

   (* Domination of a by b or vice-versa *)
   if a dominates b then return a
   if b dominates a then return b
   (* Domination by a random set of individual *)
   N ← n1 individuals randomly selected
   if no individual in N dominates a and
   at least one dominates b
      then return a
   if no individual in N dominates b and
   at least one dominates a
      then return b
   (* Continuously updated sharing *)
   m_a ← 0
   m_b ← 0
   N ← n2 individuals randomly selected from the
      partially filled next generation
   for-each j ∈ N
      m_a ← m_a + sh(d_aj)
      m_b ← m_b + sh(d_bj)
   end-for-each j
   if m_a < m_b
      then return a
      else return b
```

Figure 1: Selection algorithm in the *nPGA* (Horn & Nafpliotis, 1993). Notice that n_1 is what Horn et al. call t_{dom}.

5 NON-GENERATIONAL GAs IN OPTIMIZATION

Traditionally (Goldberg, 1989; Holland, 1975), GAs applied to optimization problems have been constructed with *generational selection* schemes. In a generational GA the total population, or a very large portion of it, is replaced each generation. In De Jong's doctoral dissertation, it was determined that having a small *generational gap*[3] did not contribute to improve the performance of a GA (cf. Goldberg, 1989). These results seem to indicate that there is no advantage in allowing a generational overlap, thus supporting the idea that generational selection schemes perform better in optimization problems.

In (Whitley, 1989) a new trend was started by which *non-generational selection*[4] was implemented in GAs. In a non-generational GA there is no concept of generation. Each cycle, only a small portion (usually a single individual) is replaced by new individuals. Theoretical analysis of selection schemes (Goldberg & Deb, 1991) seems to indicate that there is no intrinsic advantage in non-generational selection, that the main difference with generational selection is an increase in selective pressure, and that this increase in selective pressure can be easily accomplished in traditional generational selection schemes.

[3]A small generational gap implies a large generational overlap as defined above.

[4]The term *steady-state genetic algorithm* has been used to refer to such GAs. We purposely avoid it because we believe the term to be confusing given that there is no difference in the transient/steady-state behavior between generational and non-generational GAs; in fact, the interesting properties of the non-generational GAs for optimization should appear, if so, in the transient rather than in the steady-state.

6 NON-GENERATIONAL GAs IN LCSs

A learning classifier system (LCS) is a learning model capable of reinforcement learning that applies a GA to the discovery of a set of coadapted rules[5] (Holland et al., 1986; Goldberg, 1989). These rules or *classifiers* are seen by the GA as single individuals and are evaluated according to their contribution to the system's goal of achieving payoff. The GA must discover a set of classifiers that performs correctly as a group, i.e., a *co-adapted* rule set in which classifiers cooperate with each other. This contrasts with the task that a GA performs when faced with a strict optimization problem.

In LCSs, the GA usually implemented has a non-generational selection scheme, and it is very unlikely a generational selection scheme could conduct to good results. Classifiers, seen as single individuals by the GA, depend on each other; removing or adding, or even changing the strength of a single classifier, can potentially affect all the others in the classifier list. The strength of firing classifiers is adjusted every time cycle; in this way, the strength of a classifier may never reach a fixed steady state (Valenzuela-Rendón, 1989) because the individual fires to different situations, because of changes in the environment, or because of changes in the population. With a given frequency, the system is stopped and the GA is run. The GA deletes a few week classifiers and replaces them with new classifiers created through selection, crossover, and mutation. This approach allows for good individuals to remain in the population, so they do not have to be rediscovered by the GA, unless fitter individuals are found. The reason that this kind of non-generational GA is used is that the LCS is seeking for a set of individuals, not a single individual, and that one cannot expect this set to appear suddenly, but rather, it must be found incrementally.

We believe that the characteristics that require a non-generational GA in LCSs are present in the multiobjec-

[5]We will consider only Michigan type LCSs.

tive optimization problem. A reasonable GA approach to multiobjective optimization should find individuals that are closer and closer to the Pareto front as the evaluations accumulate. A generational GA for multiobjective optimization would probably wander in a rather random fashion over the search space because it would constantly be deleting good individuals. It is a common practice in optimization applications, where a single solution is sought, to keep a copy of the best individual found so far separate from the population; in this way, the result of the complete run is the best individual found. This type of approach does not seem useful in multiobjective optimization where the solution is formed by a set of individuals. In contrast to a generational GA, a non-generational GA could incrementally zoom into the Pareto front deleting individuals only when they cease to be good because of changes in the whole population.

7 NON-GENERATIONAL GA FOR MULTIOBJECTIVE OPTIMIZATION

We propose a GA for multiobjective optimization with the following characteristics: a non-generational selection scheme, the fitness of an individual is calculated incrementally, the goal of the algorithm is to find a set of coadapted individuals, and the fitness of an individual is potentially dependent on all the other individuals in the population. The operation of our proposed GA is very similar to the GA in an LCS. The algorithm has an evaluation part in which the fitness of the individuals in the population is adjusted, and a GA part in which new individuals are created by means of selection, crossover, and mutation.

The evaluation of an individual is based on two figures of merit: δ, a running weighted average of the number of individuals that have dominated this individuals so far; and ω, a running weighted average of the number of individuals that lie close according to a sharing function. We will call δ the *domination count* and ω the *moving niche count.* The evaluation for reproductive purposes of an individual i is given as:

$$f_i = c_\delta \delta_i + c_\omega \omega_i. \tag{2}$$

The evaluation of individuals in the population is adjusted by pair-wise comparisons. Every time an individual is found to be dominated in one of these comparisons its domination count is adjusted according to the following equation:

$$\delta(t+1) = \delta(t) - k_\delta \delta(t) + \mathcal{D}(t), \tag{3}$$

where $\mathcal{D}(t)$ is 1 if the individual is dominated in comparison t, and 0 if it is not. The moving niche count of an individual is adjusted according to a sharing function of the distance to the individual it is being compared with in the following way:

$$\omega(t+1) = \omega(t) - k_\omega \omega(t) + \mathrm{sh}(d), \tag{4}$$

where d is the distance between the two individuals being compared.[6]

The objective of the GA is to find individuals with low domination and moving niche counts. The constants c_δ and c_ω express the compromise between the two objectives set to the GA. It is interesting to note that our approach transforms a multiobjective optimization problem into a two objective problem, minimization of domination count and moving niche count, and that this problem is then attacked by transforming it into a single objective problem by taking a linear combination of these two objectives. This is departure from the hierarchical fashion in which these objectives are treated in the nPGA, gives our approach the possibility to explicitly control the compromise between finding points that lie close to the Pareto front and finding points that are approximately evenly spread out.

The pseudo-code presented in Figure 7 implements our non-generational GA for multiobjective optimization. During the *initialization* each individual in the population is compared to n_1 randomly selected individuals. After these comparisons, the domination count is set to the number of individuals that dominated each individual. The moving niche count of individual i is set to $\omega_i = \sum_{p=1}^{n_1} \mathrm{sh}(d_{ip})$, where d_{ip} is the distance to individual p.

The *main cycle* performs two functions. First, n_2 comparisons are performed; in these, the values of δ, ω, and f of the individuals participating are adjusted and set according to Equations 2, 3, and 4. Then, one step of a GA is performed. In the GA step, two individuals are chosen proportionally to $(f_{\max} - f)$, where $f_{\max}$ is the maximum value of f in the population; crossover and mutation are applied to these individuals to produce a new individual which substitutes the worst individual. For an initial evaluation of the newly created individual, n_1 comparisons are performed. After these comparisons its domination count is set to the number of individuals that dominated it, and its moving niched count is set to $\omega_i = \sum_{p=1}^{n_1} \mathrm{sh}(d_{ip})$.

[6] Notice that Equations 3 and 4 are very similar to the equation that governs the strength S of a classifier in isolation that receives a reward $R(t)$ and that bids a fraction k of its strength (Valenzuela-Rendón, 1989): $S(t+1) = S(t) - kS(t) + R(t)$.

Non-generational GA for multiobjective optimization

```
(* Initialization *)
Generate random initial population
for-each i ∈ population
    δ_i ← 0
    ω_i ← 0
    N ← n_1 individuals randomly selected
    for-each j ∈ N
        if j dominates i then δ_i ← δ_i + 1
        if i dominates j then δ_j ← δ_j + 1
        ω_i ← ω_i + sh(d_ij)
        ω_j ← ω_j + sh(d_ij)
    end-for-each j
end-for-each i
repeat (*Main cycle *)
    for i ← 1 to n_2
        (*Adjustment of evaluations *)
        i ← individual randomly selected
        j ← individual randomly selected
        if j dominates i
            then D_i ← 1 else D_i ← 0
        if i dominates j
            then D_j ← 1 else D_j ← 0
        δ_i ← δ_i − k_δ δ_i + D_i
        δ_j ← δ_j − k_δ δ_j + D_j
        ω_i ← ω_i − k_ω ω_i + sh(d_ij)
        ω_j ← ω_j − k_ω ω_j + sh(d_ij)
        f_i ← c_δ δ_i + c_ω ω_i
        f_i ← c_δ δ_j + c_ω ω_j
    end-for i
    (* Selection, crossover, and mutation *)
    f_max ← max(f_i)
    Select two parents proportionally to (f_max − f_i)
    Create new individual by two point crossover
    and mutation
    Delete an individual i with f_i = f_max
    with i ← new individual
        N ← n_1 individuals randomly selected
        for-each j ∈ N
            if j dominates i then δ_i ← δ_i + 1
            if i dominates j then δ_j ← δ_j + 1
            ω_i ← ω_i + sh(d_ij)
            ω_j ← ω_j + sh(d_ij)
        end-for-each j
    end-with i
until termination criteria are met
```

Figure 2: A non-generational GA for multiobjective optimization

Table 2: Multiobjective optimization problems.

problem	objectives	ranges
1	$\min_x f_1(x) = x^2$ $\min_x f_2(x) = (x-2)^2$	$x \in [-6, 6]$
2	$\min_{x,y} f_1(x,y) = \dfrac{1}{x^2+y^2+1}$ $\min_{x,y} f_2(x,y) = x^2 + 3y^2 + 1$	$x \in [-3, 3]$ $y \in [-3, 3]$
3	$\min_{x,y} f_1(x,y) = x + y + 1$ $\min_{x,y} f_2(x,y) = x^2 + 2y - 1$	$x \in [-3, 3]$ $y \in [-3, 3]$

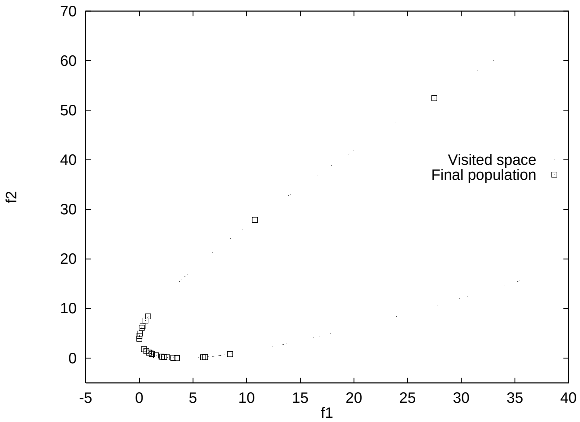
The value of k_δ was set to 0 so that the domination count of an individuals is monotonically non-decreasing; in this way, individuals that have been previously dominated, even if the dominating individual has disappeared from the population, are selected against. The value of k_ω was set to $1/n_1$; in this way, the expected value of the moving niche count after initialization is equal to the expected value of steady state of the moving niche count.

8 RESULTS

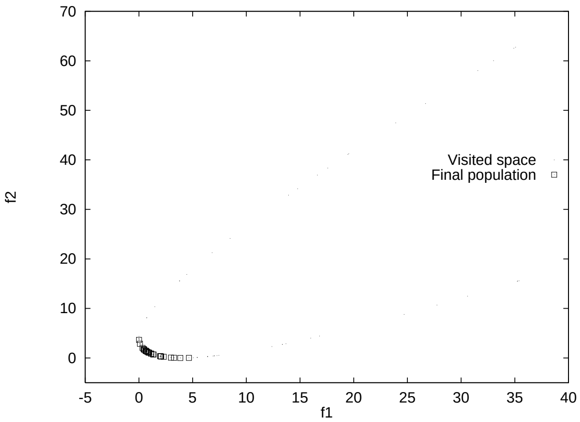
We tested the performance of our approach on three problems of increasing difficulty, and compared it to the nPGA. The objective functions that define each one of these problems are listed in Table 2. Table 1 summarizes the value of parameters for the nPGA and our non-generational GA in all experiments performed.

Problem 1 is attacked both in (Schaffer, 1985) and (Horn & Nafpliotis, 1993; Horn et al., 1994). This is a fairly simple problem where there is only one independent variable x. We have tried to match as closely as possible the values used in (Horn & Nafpliotis, 1993) from this problem. For our non-generational GA, we set up the number of cycles to match the number of evaluations in the nPGA. The parameters c_δ, c_ω, n_1, and n_2 where chosen arbitrarily. The parameters k_δ and k_ω where chosen as explained in Section 7. In Figure 3 we show the visited search space and the final population for both algorithms. Notice that the final population of the nPGA contains individuals that lie far away form the Pareto front. This contrasts with our non-generational GA in which all individuals lie close to the front.

Problem 2 is slightly more difficult than problem Problem 1; Problem 2 has two independent variables, x and y, and in the objective function plane, the set of valid points is an area. Figure 4 shows the visited space

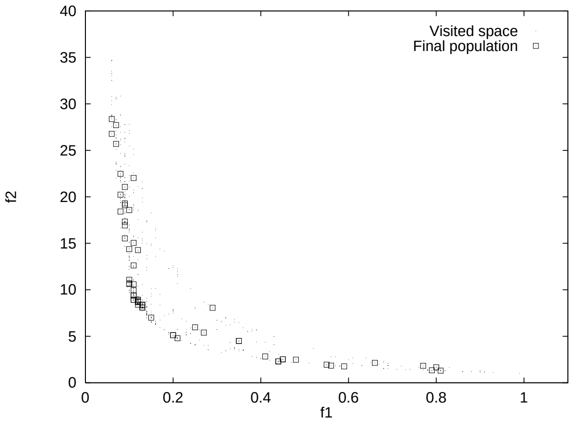


(a) Niched Pareto GA

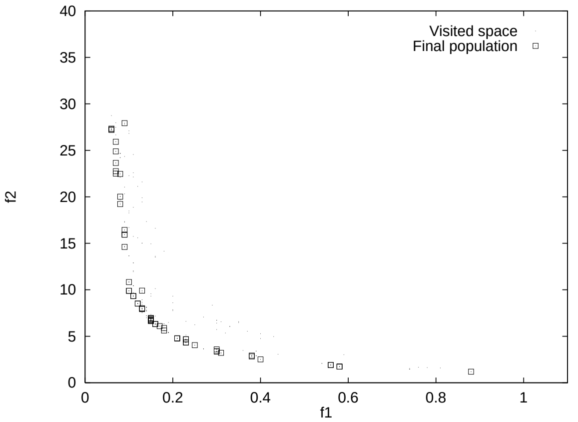


(b) Non-generational GA

Figure 3: Visited space and final population in the objective functions space for problem 1.

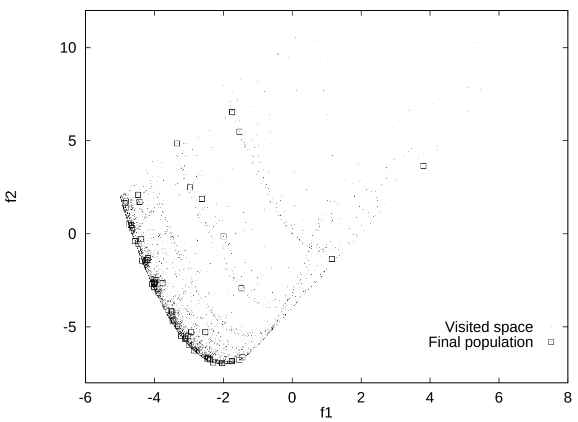


(a) Niched Pareto GA

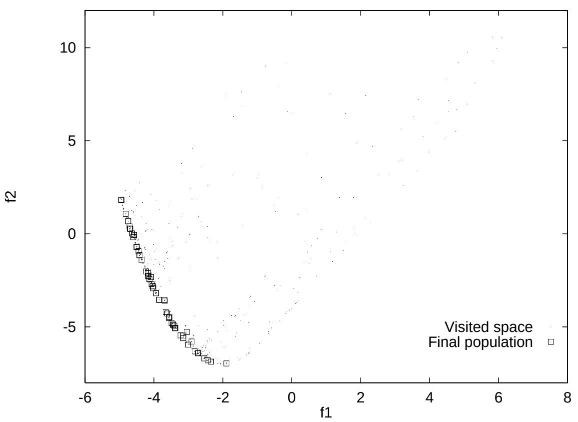


(b) Non-generational GA

Figure 4: Visited space and final population in the objective functions space for problem 2.



(a) Niched Pareto GA



(b) Non-generational GA

Figure 5: Visited space and final population in the objective functions space for problem 3.

Table 1: Parameters used in the experiments reported.

	Problem 1		Problem 2		Problem 3	
	nPGA	Non-generational GA	nPGA	Non-generational GA	nPGA	Non-generational GA
chromosome length	14	14	40	40	40	20
population size	30	30	60	60	60	60
mutation probability	0.033	0.033	0.033	0.033	0.033	0.033
crossover probability	0.85	0.85	0.70	0.70	0.85	0.85
generations/cycles	3	90	10	150	100	600
function evaluations	120	120	600	210	6000	660
σ_s	0.1	0.1	0.1	0.1	0.1	0.1
k_δ, k_ω		0, 0.9		0, 0.9		0, 0.9
c_δ, c_ω		10, 0.05		10, 0.05		10, 0.05
n_1, n_2	4, 10	10, 7	10, 10	10, 20	10, 10	10, 20

and the final population of the nPGA and our non-generational GA. Notice that the non-generational GA takes less evaluations to find a final population more evenly spread on the Pareto front, and that more individuals in the final population in the nPGA lie away from the Pareto front.

Finally, Problem 3 is even more difficult than Problems 1 and 2. Now, the valid region of the objective functions plane is larger, and a more complex relation exists between the independent variables, x and y, and functions f_1 and f_2. Figure 5 shows the visited space and the final population for the nPGA and our non-generational GA. There is a noticeable difference in performance between algorithms. Our non-generational GA visits almost one tenth of the space the nPGA does. Nevertheless, our algorithm produces a final population of individuals neatly spread out through the Pareto front. The nPGA seems to perform an almost exhaustive search and still leaves individuals in the final population far away from the Pareto front. For the three test problems, our non-generational GA explores a smaller portion of the search space and its final population contain less individuals that do not form part of the Pareto front. The nPGA appears to visit all the search space without a stronger concentration near the Pareto front, and its final population contains many dominated points.

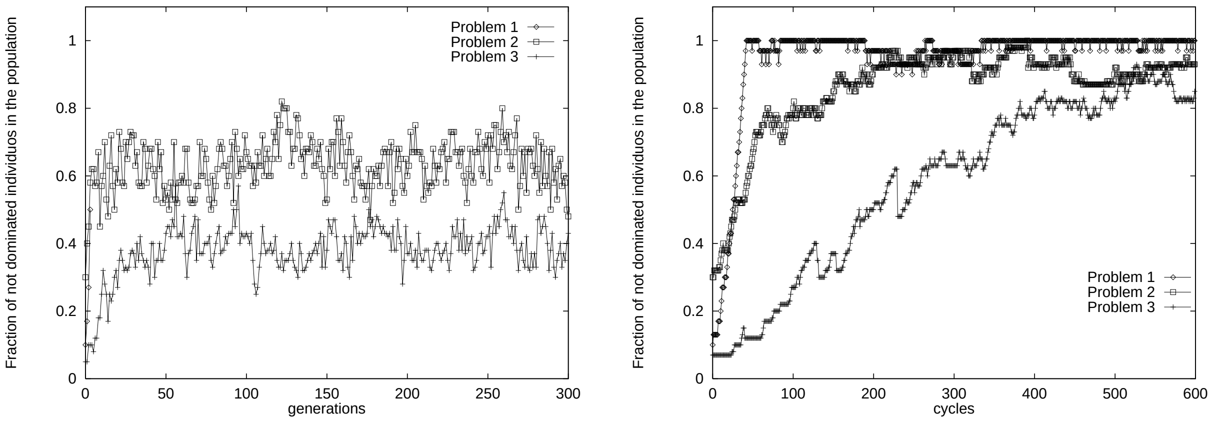
Figure 6 plots the fraction of non-dominated individuals in the population versus the the number of generations for the nPGA and of function evaluations for our practical non-generational GA. In both cases, the plots were obtained independently of the GAs being run to solve the multiobjective optimization problem. This means that the domination count that each individual carries in the non-generational GA was not used to create these graphs, but rather each individual was compared to all others in the population. The comparisons performed to produce the plots did not affect the individuals' domination or moving niche count. The fraction of non-dominated individuals in the population does not have a stable behavior for the nPGA. As the problem difficulty increases, the value of the plot is reduced, indicating that the algorithm delivers a solution of decreasing quality as the difficulty of the problem grows. In the non-generational GA the plot is an almost monotonically increasing curve. As the problem difficulty increases, the fraction of non-dominated individuals takes more evaluations to reach the value of 1.0, but once it reaches this values it stays very close to it. This behavior is easy to understand given that the non-generational GA deletes only the worst individual; in this manner, once the algorithm finds a population of not dominated individuals it will tend to delete all new individuals that are dominated.

9 CONCLUSIONS

The interest in multiobjective optimization lies in that most *real problems* involve more than one optimization objective. The multiobjective optimization problem presents to a GA approach characteristics similar to those faced by the GA in an LCS. These common characteristics suggest a common GA approach. In the work reported in this paper, we have followed this idea. We have presented a non-generational GA to the multiobjective optimization problem. The GA presented incrementally calculates the fitness of individuals in the population based on all other individuals. The non-generational GA presented significantly outperforms previous efforts with respect to number of function evaluations and quality of final solution. Also, the dynamic behavior seen through the plots of the fraction of non-dominated individuals indicates that the non-generational GA has a more stable and reliable time response.

References

Deb, K., & Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function



(a) Niched Pareto GA (b) Non-generational GA

Figure 6: Fraction of not dominated individuals in the population.

optimization. *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 42–50.

Fonseca, C. M., & Fleming, P. J. (1993). Genetic algorithm for multiobjective optimization: Formulation, discussion and generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 416–423.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In G. J. E. Rawlins (Ed.), *Foundations of genetic algorithms* (pp. 69–93). San Mateo,CA: Morgan Kaufmann.

Goldberg, D. E., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pp. 41–49.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Cambridge, MA: MIT Press.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. (1986). *Induction: Processes of inference, learning, and discovery.* Cambridge, MA: MIT Press.

Horn, J., & Nafpliotis, N. (1993). *Multiobjective optimization using the niched pareto genetic algorithm* (IlliGAL Report No. 93005). Urbana-Champaign, IL: Department of General Engineering, University of Illinois at Urbana-Champaign.

Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994). A niched Pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, pp. 82–87, Piscataway, NJ. IEEE Service Center.

Koza, J. R. (1992). *Genetic programming: On the programming of computers by natural means of natural selection.* Cambridge, MA: MIT Press.

Louis, S. J., & Rawlings, G. J. E. (1993). Pareto optimality, GA-easiness and deception. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 118–123.

Oei, C. K., Goldberg, D. E., & Chang, S. (1991). *Tournament selection, niching, and the preservation of diversity* (IlliGAL Report No. 91011). Urbana-Champaign, IL: Department of General Engineering, University of Illinois at Urbana-Champaign.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. *Proceedings of an International Conference on Genetic Algorithms and their Applications*, pp. 93–100.

Srinivas, N., & Deb, K. (1995). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, pp. 221–248.

Valenzuela-Rendón, M. (1989). *Two analysis tools to describe the operation of classifier systems* (TCGA Report No. 89005). Tuscaloosa, AL: The University of Alabama. (Doctoral dissertation).

Whitley, D. (1989). The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 116–121.