A Niched Pareto Genetic Algorithm for Multiobjective Optimization

Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg

PREPRINT
(camera-ready)

As accepted for publication in the Proceedings of the
First IEEE Conference on Evolutionary Computation,
IEEE World Congress on Computational Intelligence, Volume 1, 1994
(ICEC '94)
Piscataway, NJ: IEEE Service Center, pp. 82-87

(camera-ready preprint) ICEC '94 ©1994 IEEE (pp. 82-87)

A Niched Pareto Genetic Algorithm for Multiobjective Optimization

Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg

***Abstract*— Many, if not most, optimization problems have multiple objectives. Historically, multiple objectives have been combined ad hoc to form a scalar objective function, usually through a linear combination (weighted sum) of the multiple attributes, or by turning objectives into constraints. The genetic algorithm (GA), however, is readily modified to deal with multiple objectives by incorporating the concept of Pareto domination in its selection operator, and applying a niching pressure to spread its population out along the Pareto optimal tradeoff surface. We introduce the Niched Pareto GA as an algorithm for finding the Pareto optimal set. We demonstrate its ability to find and maintain a diverse "Pareto optimal population" on two artificial problems and an open problem in hydrosystems.**

I. Introduction

Genetic algorithms (GAs) have been applied almost exclusively to single-attribute[1] problems. But a careful look at many real-world GA applications reveals that the objective functions are really multiattribute. Typically, the GA user finds some ad-hoc function of the multiple attributes to yield a scalar fitness function. Often-seen tools for combining multiple attributes are constraints, with associated thresholds and penalty functions, and weights for linear combinations of attribute values. But penalties and weights have proven to be problematic. The final GA solution is usually very sensitive to small changes in the penalty function coefficients and weighting factors [9].

A few studies have tried a different approach to multicriteria optimization with GAs: using the GA to find all possible tradeoffs among the multiple, conflicting objectives. Such solutions are *non-dominated*, in that there are no other solutions superior in all attributes. In attribute space, the set of non-dominated solutions lie on a surface known as the *Pareto optimal frontier*[2]. The goal of a *Pareto GA* is to find a representative sampling of solutions all along the Pareto front.

II. Previous Work

We assume the reader is familiar with the simple GA [3]. Here we review previous approaches to multiobjective optimization with GAs.

In his 1984 dissertation [10], and later in [11], Schaffer proposed his Vector Evaluated GA (VEGA) for finding multiple solutions to multiobjective (vector valued) problems. He created VEGA to find and maintain multiple classification rules in a set covering problem. VEGA tried to achieve this goal by selecting a fraction of the next generation using one of each of the attributes (e.g., cost, reliability). Although Schaffer reported some success, VEGA seems capable of finding only extreme points on the Pareto front, where one attribute is maximal, since it never selects according to *tradeoffs* among attributes.

In his review of GA history, including Schaffer's VEGA, Goldberg [3] suggested the use of non-domination ranking and selection to move a population toward the Pareto front in a multiobjective problem. He also suggested using some kind of niching to keep the GA from converging to a single point on the front. A niching mechanism, such as sharing [5], would allow the GA to maintain individuals all along the non-dominated frontier.

Fonseca and Fleming [2], and, independently, Horn and Nafpliotis [7], implemented Goldberg's two suggestions, and successfully applied the resulting algorithms to difficult, open problems. Fonseca and

The authors are with the Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 117 Transportation Building, 104 South Mathews Ave., Urbana, IL 61801. Internet: jeffhorn@uiuc.edu, nick-n@uiuc.edu, goldberg@vmd.cso.uiuc.edu. Phone: 217/333-2346, Fax: 217/244-5705. The first author acknowledges support from NASA under contract number NGT-50873, while the remaining authors acknowledge support provided by the U.S. Army under Contract DASG60-90-C-0153.

[1] We use the terms "attribute", "objective", and "criteria" interchangeably to describe a scalar value to be maximized or minimized. "Decision variable" refers to the parameters of the problem encoded in the genome of the genetic algorithm.

[2] We assume familiarity with the concept of Pareto optimality, but note here that the Pareto front often goes by the names Pareto optimal set, non-dominated frontier, *efficient* points, and *admissible* points.

Fleming found many good tradeoffs in a four attribute gas turbine design problem. Horn and Nafpliotis concentrated on a series of two attribute problems, which we describe later in this paper.

III. The Niched Pareto GA

The specifics of the *Niched Pareto GA* are localized to implementation of selection for the genetic algorithm. One of the most widely implemented selection techniques for GAs is tournament selection. In tournament selection a set of individuals is randomly chosen from the current population and the best of this subset is placed in the next population. By adjusting the size of the tournament we can exert some control over the amount of selection pressure and hence convergence speed. Thus the smallest tournament size of two (binary tournament) exhibits slower convergence than any larger tournament size.

Tournament selection assumes that we want a single answer to the problem. After a certain number of generations the population will converge to a uniform one. To avoid convergence and maintain multiple Pareto optimal solutions, we have altered tournament selection in two ways. First we added *Pareto domination tournaments*. Second, when we have a non-dominant tournament (i.e., a tie), sharing is implemented to determine the winner.

A. Pareto domination tournaments

The binary relation of domination leads naturally to a binary tournament in which two randomly selected individuals are compared. If one dominates the other, it wins. Initially, we used such a small *local domination criterion*, but we soon found that it produced insufficient domination pressure. There were too many dominated individuals in later generations. It seemed that a sample size of two was too small to estimate an individual's true "domination ranking"[3].

Because we wanted more domination pressure, and more control of that pressure, we implemented a sampling scheme as follows. Two candidates for selection are picked at random from the population. A comparison set of individuals is also picked randomly from the population. Each of the candidates are then compared against each individual in the comparison set. If one candidate is dominated by the comparison set, and the other is not, the latter is selected for reproduction. If neither or both are dominated by the comparison set, then we must use sharing to choose a winner, as we explain later. The sample size t_{dom} (size of comparison set) gives us control over selection pressure, or what we call *domination pressure.* The performance of the Niched Pareto GA is somewhat sensitive to the amount of domination versus sharing pressure applied [7].

A problem will arise if both candidates are on the current non-dominated front since neither will be dominated. Even off the front, a small t_{dom} could mean that neither appears dominated. And of course both could be dominated. How is a winner then chosen in such a "tie"? If we choose the winner at random, genetic drift will cause the population to converge to a single region of the Pareto front. To prevent this we implement a form of sharing when there is no preference between two individuals.

B. Sharing on the non-dominated frontier

Fitness sharing was introduced by Goldberg and Richardson [5], analyzed in detail by Deb [1], and applied successfully to a number of difficult and real world problems. The goal of fitness sharing is to distribute the population over a number of different peaks in the search space, with each peak receiving a fraction of the population in proportion to the height of that peak[4].

To achieve this distribution, sharing calls for the degradation of an individual's *objective fitness* f_i by a *niche count* m_i calculated for that individual. This degradation is obtained by simply dividing the objective fitness by the niche count to find the *shared fitness*: f_i/m_i. The niche count m_i is an estimate of how crowded is the neighborhood (niche) of individual i. It is calculated over all individuals in the current population: $m_i = \sum_{j \in Pop} Sh[d[i,j]]$, where $d[i,j]$ is the distance between individuals i and j and $Sh[d]$ is the *sharing function.* $Sh[d]$ is a decreasing function of $d[i,j]$, such that $Sh[0] = 1$ and $Sh[d \geq \sigma_{share}] = 0$. Typically, the *triangular sharing function* is used, where $Sh[d] = 1 - d/\sigma_{share}$ for $d \leq \sigma_{share}$ and $Sh[d] = 0$ for $d > \sigma_{share}$. Here σ_{share} is the *niche radius*, fixed by the user at some estimate of the *minimal separation* desired or expected between the goal solutions. Individuals within σ_{share} distance of each other degrade each other's fitness, since they are in the same niche. Thus convergence occurs within a niche, but convergence of the full population is avoided. As one niche "fills up", its niche count increases to the point that its shared fitness is lower than that of other niches.

Fitness sharing was originally combined with fitness proportionate (a.k.a., roulette wheel) selection. When sharing is combined with the more popular

[3] Note that any partial order determines a unique ranking, in which *maximal* individuals are ranked first, then removed. The remaining individuals are reordered, and the maximal individuals of this set are ranked second, and removed, etc. This is the domination ranking scheme suggested by Goldberg [3].

[4] The authors sometimes refer to this form of niching as fitness *proportionate* sharing.

tournament selection, however, the niched GA exhibits chaotic behaviour [8]. The wild fluctuations in niche subpopulations induced by the "naive" combination of sharing and tournament selection can be avoided. Oei, Goldberg, and Chang [8] suggest the use of tournament selection with *continuously updated sharing*, in which niche counts are calculated not by using the current population, but rather the partly filled next generation population. This method was used successfully by Goldberg, Deb, and Horn [4] on a "niching-difficult" problem. Also in [4], it was found empirically that sampling the population was sufficient to estimate the niche count and so avoid the $O(N^2)$ comparisons needed to calculate *exactly* the m_i. We incorporate both techniques (continously updated sharing and niche count sampling) in the Niched Pareto GA.

In any application of sharing, we can implement genotypic sharing, since we always have a genotype (the encoding). But Deb's work [1] indicated that in general, phenotypic sharing is superior to genotypic sharing. Intuitively, we want to perform sharing in a space we "care more about", that is, some phenotypic space. Since we are interested in maintaining diversity along the phenotypic Pareto optimal front, which exists only in *attribute space*, it makes sense to perform our sharing in attribute space[5].

When the candidates are either both dominated or both non-dominated, it is likely that they are in the same equivalance class (in the partial order induced by the domination relation). Because we are interested in maintaining diversity along the front, and most of the individuals in these equivalence classes can be labeled "equally" fit, we do not implement any form of fitness degradation according to the niche count. Instead, the "best fit" candidate is determined to be that candidate which has the least number of individuals in its niche and thus the smallest niche count. We call this type of sharing *equivalence class sharing*[6].

Figure 1 illustrates how this form of sharing should work between two non-dominated individuals. Here we are maximizing along the x-axis and minimizing on the y-axis. In this case the two candidates for selection are not dominated by the comparison set. Thus the two candidates are in the Pareto optimal subset (the dashed region) of the *union* of the comparison set and the candidates. From a Pareto point of view, neither candidate is preferred. But if we want to maintain useful diversity (i.e., a representative sampling of the Pareto frontier), it is apparent that it would be best to choose the candidate that has the smaller niche count. In this case, that is candidate 2.

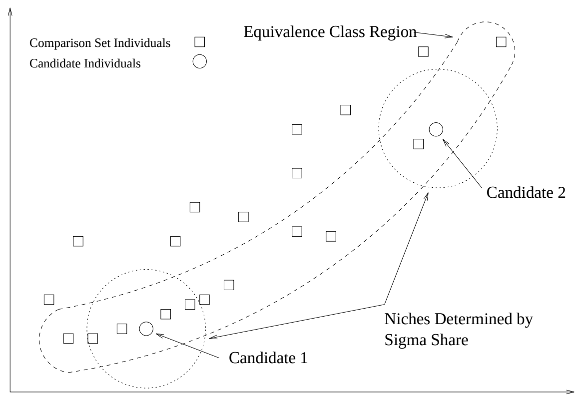

Figure 1: Equivalence class sharing.

IV. Application to Three Problems

A. Problem 1: A simple test function

We begin testing our new algorithm by constructing a simple, artificial problem with an easily calculated Pareto optimal front. We call problem 1 "unitation versus pairs" for the two attributes of unitation and complementary adjacent pairs. Unitation $Unit[s]$ is simply the number of ones in the fixed length bit string s, thus $Unit[01110010] = 4$. Pairs $Prs[s]$ is the number of pairs of adjacent complementary bits, either 01 or 10. Thus $Prs[01110010] = 4$. The problem is to maximize both attributes. Among strings of a particular unitation, those strings with the greatest "mixing" of ones and zeroes dominate those who "clump" their ones (e.g., 01010 dominates 01100, though both have the same unitation).

In Figure 2, we plot the feasible region of the two-dimensional attribute space $Unit$ versus Prs for a 12-bit problem. P indicates a point on the Pareto front, while "-" indicates a feasible point that is dominated by some member(s) of the Pareto set.

In Figure 3, we plot the initial population (generation 0) of 100 randomly generated 12-bit individuals[7]. The numbers plotted in attribute space are the numbers of individuals with attribute values corresponding to the coordinates in attribute space.

Figure 4 shows the population after 100 generations. We can see that the GA has succeeded in finding all but one member of the Pareto set, and

[5] To impose a meaningful metric on attribute space, the attributes should be scaled to the same numerical range, (e.g., 0 to 1). This is possible if we have some idea of the extreme values theoretically attainable by each attribute.

[6] This technique might also be called *flat fitness sharing*, since the same effect is produced in normal (single-attribute) fitness sharing when two individuals have the exact same fitness.

[7] Tournament size $t_{dom} = 10$, niche size $\sigma_{share} = 2.0$, single-point crossover probability $p_c = 0.9$, and mutation rate $p_m = 0.01$.

```
Unitation
12 —   P
11 —       -   P
10 —       -   -   -   P
09 —       -   -   -   -   -   P
08 —       -   -   -   -   -   -   -   P
07 —       -   -   -   -   -   -   -   -   -   P
06 —       -   -   -   -   -   -   -   -   -   -   P
05 —       -   -   -   -   -   -   -   -   -   -
04 —       -   -   -   -   -   -   -   -
03 —       -   -   -   -   -   -
02 —       -   -   -   -
01 —       -   -
00 —   -
-----------------------------------------------------------
       0   1   2   3   4   5   6   7   8   9  10  11  12      Pairs
```

Figure 2: Problem 1's discrete, two dimensional attribute space, with feasible (-) and Pareto (P) points indicated.

```
Unitation                              Generation 0
12 —   0
11 —       0   2
10 —       0   0   1   4
09 —       0   0   1   2   1   2
08 —       0   0   0   4   8   1   2   1
07 —       0   0   1   2   5   3   6   1   0   0
06 —       0   0   2   2   6   3   5   6   0   0   0
05 —       0   0   6   2   2   2   1   6   0   0
04 —       0   2   0   1   1   3   0   1
03 —       0   0   0   0   0   1
02 —       0   1   0   0
01 —       0   0
00 —   0
-----------------------------------------------------------
       0   1   2   3   4   5   6   7   8   9  10  11  12      Pairs
```

Figure 3: Distribution of the randomly generated initial population.

appears to be maintaining substantial subpopulations at each such point. Moreover, there are few (none here) dominated individuals in the current population. Although not shown, we have plotted population distributions over many generations, and noticed that the GA does indeed maintain roughly equal size subpopulations at each Pareto point over many generations. Dominated solutions regularly appear, due to crossover and mutation, but are not maintained.

We have observed similar behaviour over many runs of the GA on different initial population distributions (all random, but using different random seeds). We have also successfully tried larger problems ($l > 12$), with correspondingly larger population sizes N, such as 400 individuals on a 28 bit problem [7].

Finally, we note that this problem is GA-easy (in

```
Unitation                              Generation 100
12 —  26
11 —       0  11
10 —       0   0   0  14
09 —       0   0   0   0   0  12
08 —       0   0   0   0   0   0   0  16
07 —       0   0   0   0   0   0   0   0   0  21
06 —       0   0   0   0   0   0   0   0   0   0   0
05 —       0   0   0   0   0   0   0   0   0   0
04 —       0   0   0   0   0   0   0   0
03 —       0   0   0   0   0   0
02 —       0   0   0   0
01 —       0   0
00 —   0
-----------------------------------------------------------
       0   1   2   3   4   5   6   7   8   9  10  11  12      Pairs
```

Figure 4: Stable subpopulations on the Pareto front.

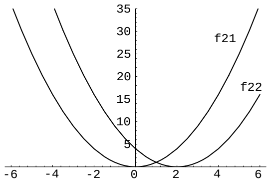

Figure 5: Schaffer's function F2, $P = \{x \mid 0 \leq x \leq 2\}$

that it is easy to find points on the front), but not necessarily easy for the Niched Pareto GA. Since there are actually many more solutions at middle points on the front, and only one or two at each end point of the front, it should be harder to maintain equal size subpopulations at the extreme points.

B. Problem 2: Schaffer's F2

Next we compare our algorithm to Schaffer's VEGA by running it on one of the test functions from Schaffer's dissertation [10]. This is the simple function F2, with a single decision variable, the real-valued x, and two attributes, $f21$ and$f22$ to be minimized:

$$f21(x) = x^2 \qquad f22(x) = (x-2)^2$$

The decision variable is mapped to a 14-bit string as a binary-coded integer. Thus $00000000000000 = x_{min} = -6.00$ and $11111111111111 = x_{max} = 6.00$. We plot $f21$ and $f22$ over this range of x in Figure 5. It is clear that the Pareto front is where the tradeoff exists between the two functions. That is, for $0 \leq x \leq 2.00$, one of the functions is decreasing to its best while the other is increasing away from its best.

Like Schaffer, we use a small population size $N = 30$. Our niche size $\sigma_{share} = 0.1$ and tournament size $t_{dom} = 4$. As Figure 6 illustrates, the Niched Pareto GA is able to maintain a fairly even spread of solutions along the Pareto front. There are a few dominated individuals in the population (to the right of $x = 2.00$), as in the VEGA run above, but most individuals are on the front. Although our population has several gaps in its distribution on the front, it appears more evenly distributed than generation 3 of the VEGA run[8]. Most importantly, the Niched Pareto GA exhibits stability in this population distribution for many more generations than were indicated for VEGA ($200 > 3$).

F2 is an easy problem for the GA: the initial population contains many individuals on the front already. However, this front is much denser than that of problem 1 above, challenging the Niched Pareto GA to maintain N subpopulations of size 1 along the front.

[8] Schaffer (1984) only gave results for generations 0, 1, and 3.

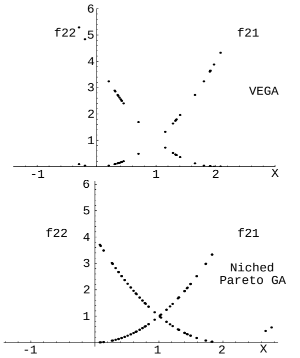

Figure 6: VEGA versus the Niched Pareto GA. Top: VEGA on F2's Pareto frontier, generation 3. Bottom: The Niched Pareto GA's distribution, generation 200.

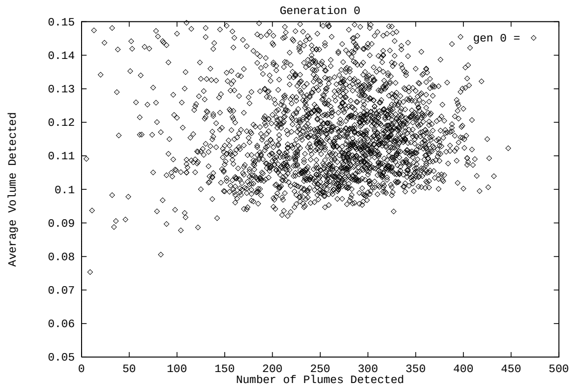

Figure 7: Initial population distribution, problem 3.

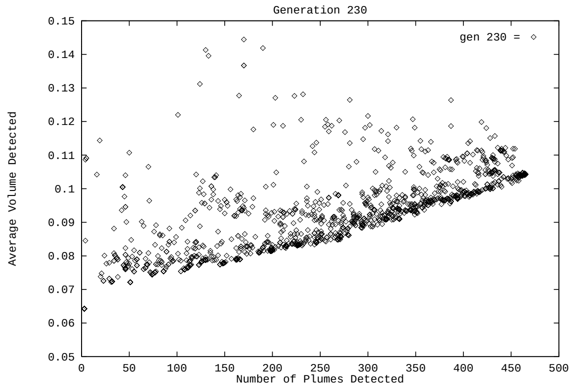

Figure 8: Final distribution, problem 3.

C. Problem 3: Open problem in hydrosystems

To challenge the Niched Pareto GA's ability to *search* for diverse tradeoffs, we chose a larger, real-world (i.e., unsolved) application for our third test problem: optimal well placement for groundwater contaminant monitoring[9]. The problem is to place a set of k out of a possible w wells in order to maximize the number of *detected* leak plumes from a landfill into the surrounding groundwater, and to minimize the volume of cleanup involved. These two objectives conflict. Simply optimizing for the minimum volume of cleanup will give us an answer with attributes $\{0, 0\}$, where we detect no plumes and therefore have no volume of contaminant to clean up. If we maximize the number of detected plumes our volume of cleanup increases dramatically.

It is important to note that this problem is intractable. The search space is of size $\binom{w}{k}$. In our specific example we have $w = 396$ and $k = 20$. The whole search space is then $\binom{396}{20}$ which is 2.269×10^{33}. This makes it impossible to know the actual Pareto optimal front from enumeration.

Monte-Carlo simulation was used to develop a set of possible leak plumes, the set of wells that detect each plume, and the volume leaked when each well detected the contaminant plume. Using these data, we constructed a vector-valued fitness function to return the number of plumes and average volume detected by any given set of wells.

In our first few runs, $N = 2000$, $\sigma_{share} = 40$, $t_{dom} = 40$, $p_c = 0.8$, and no mutation. In Figure 7 one can see that the random initial population is distributed throughout the search space. Figure 8 shows that after 230 generations the Niched Pareto GA has found an apparent front that is indeed improved over the initial population. It is promising to see that even after a large number of generations we are maintaining diversity over most of an apparent front. There is definite improvement as to the location of the front and the decrease in the number of dominated individuals in the population.

We do not know yet whether this is the actual Pareto optimal front or a sub-optimal front. But our first few runs indicate that the equivalence-class sharing and dominance tournaments are working together. We have shown that a tradeoff curve better than a random sampling can be developed by the Niched Pareto GA on an open problem.

[9] This open problem was developed by Wayland Eheart and his colleagues at the Civil Engineering Department at the University of Illinois at Urbana-Champaign. We are grateful to Dr. Eheart and his students S. Ranjithan, P. Stork, and S. Cieniawski for helping us implement it.

V. Discussion

These preliminary results on the application of the niched, Pareto technique are encouraging. Fonseca and Fleming [2] have also reported initial success with a similar algorithm. But we have found that the performance of the Niched Pareto GA is sensitive to the settings of several parameters. In particular, it is important to have a large enough population to search effectively and to sample the breadth of the Pareto front. Both [7] and [2] discuss the setting of σ_{share} and population size together to yield effective sampling.

But the behaviour of the Niched Pareto GA seems to be most affected by the degree of selection pressure applied. Just as tournament size t_{size} is critical to selection pressure and premature convergence in a regular GA with tournament selection, so t_{dom} directly effects the convergence of the Niched Pareto GA. Horn and Nafpliotis [7] illustrate the effects of too little and too much dominance pressure. Here, we summarize their empirically-derived, order-of-magnitude guidelines:

- $t_{dom} \approx 1\%$ of N; results in too many dominated solutions (a very fuzzy front).
- $t_{dom} \approx 10\%$ of N; yields a tight and complete distribution.
- $t_{dom} \gg 20\%$ of N; causes the algorithm to prematurely converge to a small portion of the front. Alternative tradeoffs were never even found.

We have not yet addressed the critical issue of search, but we have some intuitions. Our intuition in the case of Pareto optimization is that the diversity along the currently non-dominated frontier actually helps the search for new and improved tradeoffs, thus extending the frontier. Individuals from very different parts of the front might be crossed to produce offspring that dominate a portion of the front lying between their parents. That is, information from very different types of tradeoffs could be combined to yield other kinds of good tradeoffs. Indeed, we see some evidence for this in problem 3. Because equivalence class sharing cannot be expected to maintain more than one copy of an individual (i.e., niche counts are approximately 1 for all niches at steady state), and because we used high crossover rates (typically 0.7-0.9), the maintenance of the front over hundreds of generations was largely due to the constant generation and regeneration of individuals on the front from the crossover of two different parents. Therefore, most crosses of parents on or near the front yielded offspring also on or near the front. This behaviour is evidence that Pareto *diversity* helps Pareto *search*.

Finally, we point out that the domination tournament does not rely strictly on a domination relation, but rather on an antisymmetric, transitive relation. Similarly, equivalence class sharing is useful not only on the Pareto optimal frontier, but in any equivalence class in a partial order. Thus the Niched Pareto GA can be used to search any partially ordered space, not just those induced by the Pareto approach to multiobjective problems.

References

[1] Deb, K. (1989). Genetic algorithms in multimodal function optimization. *MS thesis, TCGA Report No. 89002.* University of Alabama.

[2] Fonseca, C. M., & Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms.* Morgan-Kauffman, 416-423.

[3] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley.

[4] Goldberg, D. E., Deb, K., & Horn, J. (1992). Massive multimodality, deception, and genetic algorithms. *Parallel Problem Solving From Nature, 2*, North-Holland, 37-46.

[5] Goldberg, D. E., & Richardson, J. J. (1987). Genetic algorithms with sharing for multimodal function optimization. *Genetic Algorithms and Their Applications: Proceedings of the Second ICGA*, Lawrence Erlbaum Associates, Hillsdale, NJ, 41–49.

[6] Horn, J., (1993). Finite Markov chain analysis of genetic algorithms with niching. *Proceedings of the Fifth International Conference on Genetic Algorithms.* Morgan-Kauffman, 110-117.

[7] Horn, J., & Nafpliotis, N. (1993). Multiobjective optimization using the niched Pareto genetic algorithm. IlliGAL Report No. 93005. Illinois Genetic Algorithms Laboratory. University of Illinois at Urbana-Champaign.

[8] Oei, C. K., Goldberg, D. E., & Chang, S. J., (1991). Tournament selection, niching, and the preservation of diversity. IlliGAL Report No. 91011. Illinois Genetic Algorithms Laboratory. University of Illinois at Urbana-Champaign.

[9] Richardson, J. T., Palmer, M. R., Liepins, G., & Hilliard, M. (1989). Some guidelines for genetic algorithms with penalty functions. *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan-Kauffman, 191-197.

[10] Schaffer, J. D., (1984). Some experiments in machine learning using vector evaluated genetic algorithms, *Unpublished doctoral dissertation, Vanderbilt University.*

[11] Schaffer, J. D., (1985). Multiple objective optimization with vector evaulated genetic algorithms. In J. Grefenstette, ed., *Proceedings of an International Conference on Genetic Algorithms and their Applications*, 93–100.