Multiobjective Optimization Using
The Niched Pareto Genetic Algorithm

Jeffrey Horn and Nicholas Nafpliotis
Department of Computer Science
Department of General Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801-2996

IlliGAL Report No. 93005
July 1993

(Published, in part, 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, 82–87)

Illinois Genetic Algorithms Laboratory (IlliGAL)
Department of General Engineering
University of Illinois at Urbana-Champaign
117 Transportation Building
104 South Mathews Avenue
Urbana, IL 61801-2996

Multiobjective Optimization Using the Niched Pareto Genetic Algorithm

Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg
Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 South Mathews Avenue
Urbana, IL 61801-2996
Internet: jeffhorn@uiuc.edu, nick-n@uiuc.edu
Phone: 217/333-2346, Fax: 217/244-5705

Abstract

Many, if not most, optimization problems have multiple objectives. Historically, multiple objectives (i.e., attributes or criteria) 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 most recent development in the field of decision analysis has yielded a rigorous technique for combining attributes multiplicatively (thereby incorporating nonlinearity), and for handling uncertainty in the attribute values. But *MultiAttribute Utility Analysis* (MAUA) provides only a mapping from a vector-valued objective function to a scalar-valued function, and does not address the difficulty of searching large problem spaces. Genetic algorithms (GAs), on the other hand, are well suited to searching intractably large, poorly understood problem spaces, but have mostly been used to optimize a single objective. The direct combination of MAUA and GAs is a logical next step for multiobjective GA optimization. However, there is an alternative approach. It turns out that the GA 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. In this report, we discuss the general issues involved in searching large problem spaces while trying to optimize several objectives simultaneously. We explore various combinations of decision analysis techniques, specifically MAUA, and GAs. Finally, we introduce the Niched Pareto GA as an algorithm for finding the Pareto optimal set. We compare and contrast the Niched Pareto GA with MAUA. And we demonstrate the ability of the Niched Pareto GA to find and maintain a diverse "Pareto optimal population" on two artificial problems, and an open problem in hydrosystems.

1 Introduction

Genetic algorithms (GAs) have been applied almost exclusively to single-attribute[1] problems. But a careful look at many, if not most, of the 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.

One approach to improve the GA's handling of multiple attributes is to use a more systematic and robust method for combining the multiple attributes into a single scalar fitness function. At the very

[1]Throughout this report we use the terms "attribute", "objective", and "criteria" interchangeably. They all 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.

least, such a method should account for nonlinearities in attribute interactions. The technique known as MultiAttribute Utility Analysis (MAUA) is a recent development in the field of decision analysis. MAUA prescribes a multiplicative combination of individual *utility* values, which are functions of attribute values and the uncertainties in those values. Thus MAUA addresses multiple criteria and uncertainty, but does not address the issue of search. GAs and MAUA are therefore complementary techniques for optimization and design. MAUA has no specific method for handling intractable search spaces while traditional GAs assume a single attribute. After conducting a limited search of the literature on genetic algorithms, we concluded that GAs and MAUA have never before been combined in any manner to solve a particular problem.

The MAUA approach has some limitations, however. First, it only incorporates pairwise, multiplicative nonlinearities and not higher-order nonlinear dependencies. Second, to estimate the coefficients of the singular and pairwise terms, MAUA requires a lengthy interview with a decision maker in which the decision analyst poses a series of lottery questions. This process can be time consuming and, if not carefully performed, can introduce errors due to bias on the part of the decision analyst, etc. Finally, the estimated scalar utility function $U[\vec{x}]$ pertains only to a single decision maker (or a single group of decision makers, if they can all agree on their answers to the lottery questions!). Thus the optimal solution found by a GA for a particular $U[\vec{x}]$ is of little use to a different decision maker. All of these problems are avoided in the Pareto approach described next.

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. These solutions (tradeoffs) 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 and maintain a representative sampling of solutions on the Pareto front. Previous studies, such as Schaffer's 1984 dissertation (Schaffer, 1984), have succeeded in finding some points on the front, in particular near the extreme ends, where one attribute is optimized. But to find a *representative* sampling, with the final population spread all along the front, niching is required (Goldberg, 1989).

In the remainder of this report, we compare and contrast the GA and MAUA. In particular, we develop a version of the GA to find the Pareto optimal frontier in a multiobjective problem, and compare this approach to that of a simple combination of MAUA and GAs.

2 Background

We assume the reader understands the basic genetic algorithm and the techniques of decision analysis, in particular MAUA. We refer the less prepared reader to Goldberg (1989) for an introduction to GAs and to de Neufville (1990) for background on decision analysis and MAUA. Below we present first a brief summary of the two methods (MAUA and GAs). Then we compare the basic strengths and shortcomings of the two algorithms, and suggest how the abilities of one can help address the shortcomings of the other. We also review what little previous work we have found in the literature that attempts to extend the GA to multiobjective optimization.

2.1 MAUA

MAUA uses Keeney-Raiffa multiattribute utility $U[\vec{a}]$ as the scalar function to be optimized (Keeney & Raiffa, 1976). Here $\vec{a}$ is a vector of attributes, and is itself a function A of the decision variable settings[3]: $\vec{a} = A[\vec{x}]$. Rather than combining the attribute values directly, MAUA takes advantage of *utility theory*

[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 (Sawaragi, Nakayama, & Tanino, 1985), and *admissible* points (Guddat, Vasquez, Tammer, & Wendler, 1985). We endeavor to be consistent in using the word "Pareto" to describe the set, front, frontier, or surface.

[3] We will write $U[\vec{a}]$ when refering to U as a function of attributes, and $U[\vec{x}]$, instead of $U[\,A[\vec{x}]\,]$, when refering to U as the composite function $U \circ A$ over the decision space.

(de Neufville 1990). Separate utility functions $u_i[a_i]$ for each attribute a_i are determined for a particular decision maker in a systematic way. The $u_i[a_i]$ can be any strictly non-decreasing function of a_i, and can thus incorporate nonlinearities with respect to a single attribute (e.g., $u_i[a_i] = a_i^2$).

It is these single-attribute utility functions that are then combined, multiplicatively, to form a single overall utility function $U[\vec{a}]$. Specifically, de Neufville (1990) defines the expression for a utility function U of n attributes:

$$K\mathbf{U}[\vec{a}] + 1 = \prod_{i=1}^{n}(Kk_i u_i[a_i] + 1)$$

Every decision maker has a different relative weighting for each attribute and attribute pair (multiplication is pairwise). These attitudes towards the tradeoffs are accounted for in the coefficients[4] of the $u_i[a_i]$ and $u_i[a_i]u_j[a_j]$ terms of $U[\vec{a}]$. MAUA prescribes a systematic method for determining these coefficients.

It is worth noting here that utility theory addresses uncertainty in the attribute values, so that each $u_i[a_i]$ incorporates the decision maker's attitude toward uncertainty in the actual value of a_i. Thus MAUA handles uncertainty as well as some nonlinear interactions among the attributes, assuming that the decision analyst succeeds in closely approximating the decision maker's actual $U[\vec{a}]$.

But for this approximation to be meaningful, several assumptions must be made. These are embodied in the principle of monotonicity, and the axioms of preferential independence and utility independence. Monotonicity simply requires that a decision maker's utility functions be mononotically non-decreasing with improvement in the attribute value. Preferential independence implies that the order of preference between any *pair* of attribute values is independent of other attributes. Finally, utility independence is the requirement that the decison maker's attitude toward uncertainty, which helps determine $u_i[a_i]$, is independent of any other attribute a_j, $j \neq i$. If these three requirements are met, then the multiplicative $U[\vec{a}]$ accounts for *all* of the nonlinearities in the attribute interactions.

Although it is likely that these principles and axioms are satisfied by many, if not most, decision makers' preferences toward multiobjective problems, it is not possible to "prove" such satisfaction. Instead, MAUA calls for some testing to provide evidence of satisfaction. A Pareto approach can help relax these requirements.

2.2 GAs

Genetic algorithms search the space of possible settings of decision variables by processing a population of decision variable settings $\vec{x}$. Typically, the population is initialized with N random "strings" representing the $\vec{x}$. Each generation, each string is evaluated according to the scalar fitness function being optimized. The best strings of the current population are selected, stochastically, and recombined with each other to yield new, but similar, offspring to form the next generation. Selection drives the population toward better solutions, while recombination explores new areas of the search space (using knowledge already gained by ancestral generations to guide further search). Eventually, the population converges to a single solution which is likely the best found and hopefully the best that could ever be found. The stochastic operators of a GA allow the algorithm to search intractable spaces with some hope of finding optimal solutions within a computationally reasonable time frame.

2.3 MAUA and GAs: Complementary Techniques

There is no question that MAUA (and decision analysis in general) is a set of powerful techniques for optimization of vector-valued functions (mutliple attributes) under uncertainty. But in the decision analysis literature, the typical decision problem illustrating MAUA has only a few alternatives under consideration. This number is often at the minimum of two choices, and usually is no more than a few dozen. It is common for the number of attributes to exceed the number of design possibilities being considered. Even with only

[4] These coefficients are various combinations of the k_i and K.

two choices, the application of MAUA can be complex. This is evidence of its focus on multiobjective tradeoffs and uncertainty, and not on search.

It often seems the case, however, that the design possibilities considered are chosen as a representative sampling from a much larger set. The decision analyst frequently justifies limiting the number of design possibilities by assuming that certain ranges and combinations of decision variable settings would obviously result in infeasible or inferior solutions. But the highly nonlinear interaction of these variables on many real-world problems means that we generally can't justify such assumptions. If the decision analyst had at his or her disposal a powerful search technique such as a genetic algorithm, the analyst would be much more likely to search more of the space of feasible designs.

In the GA community, on the other hand, GA users are familiar with intractable search problems, but only for scalar valued functions. That is, GAs have been applied almost exclusively to single-attribute problems. But a careful look at many, if not most, of the real-world GA applications reveals that the objective functions were 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 thresholds with associated penalty functions, and weights for linear combinations of attribute values.

2.4 Previous Work

Although there has been much work using weighted sums and penalty functions to turn multiobjective problems into single-attribute problems for the GA, and much work using MAUA and other decision analysis techniques to search small, tractable spaces, there has been comparatively little effort expended on using GAs to search large, intractable *multiattribute* spaces. We have conducted a limited literature search, and have turned up two interesting approaches.

In his 1984 doctoral dissertation, Schaffer (1984) proposed his Vector Evaluated GA (VEGA) for finding multiple solutions to multiobjective (vector valued) problems. Schaffer was motivated by the difficulty a "scalar GA" had in the set covering problem. The GA could not maintain diverse rules that togther classified a set of patterns. He theorized that it was impossible to assess the quality of rules adequately and fairly with a single scalar value. Thus he created VEGA to find and maintain multiple solutions, each one addressing one of the multiple attributes. VEGA tried to achieve this goal by selecting a fraction of the next generation using one of each of the attributes. Thus, in a two attribute problem, say cost and reliability, VEGA would select half of the next generation's population using lowest cost as the selection criterion (scalar fitness function) and the other half using highest 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 (1989) 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 (Goldberg and Richardson, 1987), would allow the GA to maintain individuals all along the non-dominated frontier.

Ritzel (1992) took Goldberg's advice. He used non-dominated ranking and selection combined with deterministic crowding (Mahfoud, 1992) as the niching mechanism. His problem was an open one in groundwater remediation, with two attributes: cost and reliability. Though the actual Pareto front was unknown, Ritzel used the best tradeoff surface found by a domain-specific algorithm, MICCP, to judge the performance of the GA. Briefly, Ritzel found that selection according to Pareto non-domination was superior to both VEGA and non-domination with deterministic crowding, at least for *finding* points near or on the front found by MICCP. However, Ritzel did not investigate how well these algorithms *maintained* a distribution along their fronts. In addition, Ritzel did not try other niching mechanisms, such as fitness sharing.

3 Two Combinations

Below we present two alternative combinations of the two optimization techniques, in which one algorithm acts as a front end for the other. For MAUA as a front end to the GA, the two algorithms remain fairly separate, and their interface is straightforward. But for the GA as a front end to MAUA, there are many possibilities for much more involved interaction between the two methods. We therefore spend the remainder of the report on the latter combination.

3.1 MAUA $\Rightarrow$ GA (Black Box Objective Function)

The more straightforward and obvious combination of the two algorithms is to use MAUA to map a multicriteria problem into a single-attribute (i.e., a scalar utility function $U[\vec{a}]$). A GA can then optimize this function, exploring a large search space in the process. Although this technique is simple, apparently it has never been implemented. Yet a quick look at published GA applications reveals many candidates for the MAUA$\Rightarrow$GA combined algorithm. Typically, two or three attributes are combined into a single scalar fitness function for the GA using a weighted sum. Another familiar technique in designing fitness functions for GAs is to choose thresholds for minimum and maximum values of an attribute, and then assign penalties to the fitness function when these thresholds are crossed. An interesting study would involve going back and redoing some of these applications by first conducting an MAUA to determine the decision makers' utility function, running the GA on the resulting $U[\vec{x}]$, and comparing the new solution and its utility to those found previously with the GA alone.

3.2 GA $\Rightarrow$ MAUA (Pareto Optimal Set)

Our second strategy for combining GAs with MAUA is to use the GA to find the Pareto optimal set P for a multicriteria problem, and then, if necessary, use MAUA to find the prefered solution from this set. This technique, GA$\Rightarrow$MAUA, is more complicated, subtle, and interactive than the MAUA$\Rightarrow$GA approach. Finding P is a challenging problem. In Section 4, we develop a GA that is capable of finding P: the *Niched Pareto GA*. Here we discuss how P can help MAUA, and how MAUA can help the GA find P.

3.2.1 Usefulness of P

As de Neufville points out in Chapter 20 of (de Neufville, 1990), "...procedures... for multiobjective analysis...have the great advantage of not requiring any explicit measure of preference. They are therefore much simpler [than MAUA] and should be used whenever possible." P can be of greater general use than the best solution (highest $U[\vec{x}]$) for a particular decision maker, since P is independent of any decision maker and his or her utility functions. The best solution for each decision maker might be different, but all best solutions will be in P. Therefore, P can be reused with each new decision maker, saving analysis time.

More specifically, knowing the Pareto optimal set P prior to performing MAUA could be useful in a number of ways:

- For small P, make MAUA unnecessary
- A check on MAUA
- Only need $U[\vec{a}]$ on P
- Only need utility and preferential independence on P
- Only need partial MAUA
- Help with group decision making

If P is small enough, the decision makers might be able to directly choose a solution by inspection. Even when P is very large, the shape of the curve (i.e., the front) might make the region of interest obvious. For example, de Neufville (1990) notes how decision makers can often agree that solutions at the "knee" of the front, if such a region exists, represent the best compromise. Choice by direct inspection of P eliminates the need for MAUA.

Even if P is too large or the best choice is otherwise unclear, and MAUA is required, P can help the MAUA process. P can be a simple check on the regular MAUA procedure, since any solution obtained by MAUA must be in P. It might also be possible for P to improve on the MAUA in a more interactive manner. For example, it might be easier to determine the utility function $U[\vec{a}]$ for $\vec{a} \in P$. Also, we now require utility and preferential independence on the Pareto frontier only. The Niched Pareto GA does require monotonicity of preference for each attribute, but not preferential or utility independence. This should allow us to use the GA$\Rightarrow$MAUA method on a wider array of problems than with MAUA alone.

In addition, a partial MAUA could be performed to reduce the number of attributes such that the Pareto set P' of the reduced problem is amendable to choice by inspection. Thus, MAUA would be performed on just a subset of the attributes, perhaps just two, replacing them with a single attribute (the multiattribute utility function of those attributes for that decision maker).

If the set P is too large to enumerate, MAUA could utilize a regular (single-attribute) GA to search P using the scalar-valued $U[\vec{a}]$.

In this report, we do not treat explicitly the issue of group decision making. But we note here that a small P might alleviate the known difficulty of decision analysis with multiple decision makers with very different utility functions. In particular, Arrow has shown that it is in general impossible to find a single utility function $U[\vec{a}]$ for a group. Nor can we even find, in general, a single partial order of preference. But here P can help. If P is small enough that all of it can be viewed, or we can visualize the surface and focus in on the most promising choices, then our group of decision makers might be able to come to a consensus on the order of preference *for just that set P.*

4 A New Algorithm: The Niched Pareto GA

The specifics of the *Niched Pareto GA* are localized to implementation of selection for the genetic algorithm. Selection is used in GAs to determine which individuals of the current population will have their genotypic information passed to the next generation. 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 chosen to be represented in the next population. This allows the best individuals to have a high probability of being represented in future generations. Also, 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. Here we implement a form of selection that will allow us to have multiple answers to a problem with multiple attributes. In addition, we want these answers to be representative of the Pareto optimal front. To accomplish this 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.

4.1 Pareto domination tournaments

In order to obtain a Pareto optimal surface, tournament selection must be altered to use multiple attributes. What we have implemented is Pareto domination tournaments. Note that the domination relation establishes a *partial order* on the set of all phenotypes (i.e., attribute vectors)[5]. An individual with an attribute

[5]In the single-attribute case (normal function optimization), this ordering becomes a *total ordering*.

vector higher in the partial order than that of another individual is preferred (i.e., the first individual dominates the second).

The binary relation of domination leads naturally to a binary tournament (tournament size 2) 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. Our resulting fronts seemed too fuzzy. 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[6]".

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.

This simple scheme might be improved in any number of ways, such as by performing some kind of non-domination ranking (Goldberg, 1989) on the comparison set and the candidates and using the ranks as fitnesses. But our scheme was quick to implement and should give roughly the same performance as any more sophisticated procedure. Most important, the sample size t_{dom} (size of comparison set) gives us control over selection pressure, or what we call *domination pressure*. It turns out, empirically, that the performance of the Niched Pareto GA is somewhat sensitive to the amount of domination versus sharing pressure applied.

Below we present pseudocode for Pareto domination tournaments assuming that all of the attributes are to be maximized. S is an array of the N individuals in the current population. *random_pop_index* is an array holding the N indices of S, in a random order.

```
function selection                 /* Returns an individual from the current population S */
begin
    shuffle(random_pop_index);              /* Re-randomize random index array */
    candidate_1 = random_pop_index[1];
    candidate_2 = random_pop_index[2];
    candidate_1_dominated = false;
    candidate_2_dominated = false;
    for comparison_set_index = 3 to t_dom + 3 do        /* Select t_dom individuals randomly from S */
        begin
            comparison_individual = random_pop_index[comparison_set_index];
            if S[comparison_individual] dominates S[candidate_1]
                then candidate_1_dominated = true;
            if S[comparison_individual] dominates S[candidate_2]
                then candidate_2_dominated = true;
        end        /* end for loop */
    if ( candidate_1_dominated AND ¬ candidate_2_dominated )
        then return candidate_2;
    else if ( ¬ candidate_1_dominated AND candidate_2_dominated )
        then return candidate_1;
    else
        do sharing;          /* Insert sharing code from below */

  end
```

[6] 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 (1989).

It will not always be the case that there will be a clear winner among any given set of candidates. 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 sample size t_{dom} could mean that neither is dominated. And of course both could be dominated. How is a winner then chosen in such a "tie"? If we just randomly allow one of the candidates to be picked, 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. This form of sharing should maintain the genetic diversity along the population fronts and allow the GA to develop a reasonable representation of the Pareto optimal front.

4.2 Sharing on the Non-dominated Frontier

We apply the established technique of fitness sharing (Goldberg & Richardson, 1987) to maintain a population distributed along the Pareto optimal frontier. Such application raises some interesting questions, however. In what space should distance be measured? What metric should we use? How do we degrade fitness when fitness is a vector, not a scalar? How do we estimate niche size? Below we address each of these questions.

4.2.1 Background: Normal Fitness Sharing

Fitness sharing was introduced by Goldberg and Richardson (1987), analyzed in detail by Deb (1989), 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[7].

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 desireable peaks. 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. Goldberg and Richardson (1987) point out that an equilibrium should exist when the shared fitness of all niches are equal:

$$\frac{f_i}{m_i} = \frac{f_j}{m_j} \quad \forall\; i,j \;\; niches$$

Later we discuss the use of this equation for sizing populations and niches in the Niched Pareto GA.

One last important piece of background information on sharing concerns the combination of tournament selection and sharing. Fitness sharing is more or less independent of the selection method used, but was originally combined with fitness proportionate selection (a.k.a., weighted roulette wheel). When sharing is combined with the more popular tournament selection, however, the niched GA exhibits chaotic behaviour (Oei, Goldberg, and Chang, 1991). The wild fluctuations in niche subpopulations induced by the "naive"

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

combination of sharing and tournament selection can be avoided. Oei, Goldberg, and Chang 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 (1992) on a "niching-difficult" problem. Also in (Goldberg, Deb, and Horn, 1992), 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.

4.2.2 Sharing in Attribute Space

In any application of sharing, we can always implement genotypic sharing, since we always have a genotype (the encoding). Typically, however, we know more about the problem, such as the decoded decision variables, which represent a phenotype. Deb's work (1989) indicated that in general, phenotypic sharing was superior to genotypic sharing. Intuitively, we want to perform sharing in a space we "care more about", that is, some phenotypic space.

In a multiobjective problem, we have two candidate phenotypes: the decision variables and the attributes. We might be interested in finding different combinations of decision variables that yield similar attribute vectors, but our primary concern is in finding diverse, but non-dominated, attribute vectors. It therefore makes sense to perform our sharing in *attribute space*. We are interested in maintaining diversity along the phenotypic Pareto optimal front, which exists only in *attribute space*. Although it is possible that genotypic sharing[8], would give us the same diversity along the front, we run across the problem that more than one set of genotypes will give us the same attribute vector. This might lead to large "gaps" in our distribution along the front.

Later, we discuss the possibility of using *nested sharing* to achieve diversity in both the attribute vectors (i.e., on the front) and decision variable vectors of the solutions.

4.2.3 Equivalence Class Sharing

Equivalence class sharing is implemented when there is no clear winner among the prospective candidates. In the case of the Niched Pareto GA, 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.

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. When $f_i = f_j$ (i.e., the fitness function is flat for individuals i and j), then their shared fitnesses, f_i/m_i and f_j/m_j respectively, are ordered solely by their niche counts m_i and m_j. This situation would arise on a "plateau" of constant fitness, where the radius of the plateau was much larger than σ_{share}. Although we found no studies that analyzed or reported the behaviour of fitness sharing in such regions, we speculate that the population would spread out across the plateau. We should expect to find individuals σ_{share} units apart from one another, assuming a sufficiently large population to cover the plateau, and assuming crossover and mutation do not remove excessive numbers of individuals from the plateau each generation.

For our multiattribute application, it is hoped that using this form of sharing will encourage individuals to spread out along the front, rather than accumulate in a small region and eventually converge to a single individual.

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

[8] Or phenotypic sharing using the decision variables as the phenotype.

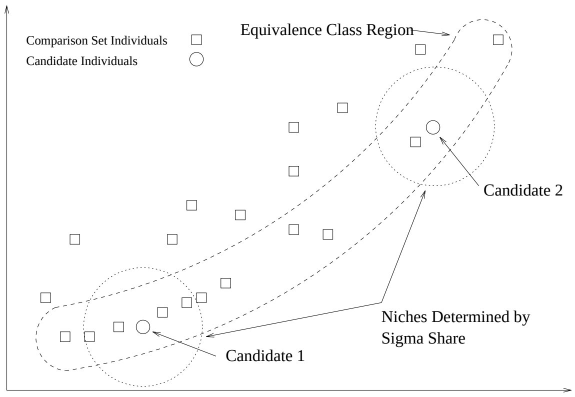

Figure 1: Equivalence class sharing.

subset of the *union* of the comparison set and the candidates (the dashed region in Figure 1). 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.

4.2.4 Implementation of Equivalence Class Sharing

We have to add three lines to our psuedocode for Pareto domination tournament selection given above, to implement equivalence class sharing.

```
function selection
begin
    ⋮
        else if nichecount[candidate_1] > nichecount[candidate_2]
            then return candidate_2;
            else  return candidate_1;
end
```

4.2.5 Sizing Niches on the Pareto Frontier

The performance of fitness sharing is sensitive to the setting of the niche radius σ_{share} and the population size N. In single-objective, or scalar, fitness sharing, Deb (1989) gives some rough guidelines for setting these parameters. Such guidelines were put to a severe test in Goldberg, Deb, and Horn (1992), and succeeded in predicting niching success and failure. Briefly, Deb suggests fixing the σ_{share} at some known minimal separation between desired optima, then sizing N for the equilibrium state $f_i/m_i = f_j/m_j$, where all shared fitnesses are equal, and i and j represent two different, desired peaks. The idea is to size N large enough so that the subpopulations m_i are sufficient to withstand the noise of the selection method during "steady state" (Horn, 1993).

Although Deb's sizing does not take into account search, it does provide us with an estimated lower bound on the population size given some idea of the minimal separation of good niches. To adapt Deb's approach to niching on the Pareto front, we also need to have some idea of the separation of our desired "optima". Fortunately, we do have some information in that respect.

Again, we will leave search out of our analysis for now and size only for the steady state. This will give us a lower bound. The desired steady state for the Niched Pareto GA is a population distributed evenly over the entire non-dominated frontier. There are no local optima on the frontier. Instead, we want a more or less uniform distribution to *represent* the possibly continuous frontier. Therefore we can simply take our population size N, divide it into the total surface area of the frontier, $Area_{pareto}$, and extract the required σ_{share} (or we could derive N for a *desired* separation σ_{share}):

$$Area_{niche}[\sigma_{share}] \approx \frac{Area_{pareto}}{N}$$

Here we use the term *Area* since the Pareto front is a surface of dimension $n-1$ in a search space with volume of dimension n, where n is the number of attributes. Similarly, niche volume is of dimension n, but the intersection of a niche hypersphere with the frontier is an area of dimension $n-1$.

For the $Area_{niche}$ term[9], we can try to calculate exactly the $n-1$ dimensional slice of the n dimensional hypersphere that would be on the frontier. Or we can use the approximation $(\sigma_{share})^{n-1}$, so that $(\sigma_{share})^{n-1} \approx \frac{Area_{pareto}}{N}$. Note that for the two-dimensional case, $\sigma_{share} = \frac{Area_{pareto}}{N}$.

As for the term $Area_{pareto}$, we might have some idea of the size of the Pareto front. Such is the case with the three applications presented later in this report. But if we don't, we still have upper and lower bounds on $Area_{pareto}$, assuming that we have bounds on our attribute values. Assume we know the extreme values (i.e., best and worst values) of our search space along the attribute axes: $\{(a_1^{best}, a_1^{worst}), (a_2^{best}, a_2^{worst}), ..., (a_n^{best}, a_n^{worst})\}$. The minimum length Pareto frontier is the hyperplane connecting these extremes. For example, in the two-dimensional case using Euclidean distance, the minimum Pareto front is simply the hypotenuse of the two attribute axes:

$$min(Area_{pareto}) = \sqrt{|a_1^{best} - a_1^{worst}|^2 + |a_2^{best} - a_2^{worst}|^2}$$

We can also find an upper bound on the area of the Pareto surface. In general, the Pareto surface will be more convoluted than the minimal surface described above. However, the requirement of non-domination limits these convolutions. The surface must be monotonic (i.e., all first-order partial derivatives never change signs). Therefore an asymptotic upper bound on the area of the Pareto surface for the two-dimensional, Euclidean distance case is simply the sum of the attribute ranges:

$$max(Area_{pareto}) < |a_1^{best} - a_1^{worst}| + |a_2^{best} - a_2^{worst}|$$

This is an asymptotic bound since no Pareto surface could actually have this area, although there are allowable surfaces arbitrarily close to it in size. Together, these two bounds should be of some use in sizing σ_{share} and population N for the Niched Pareto GA when we have no other information or estimate of $Area_{pareto}$.

4.3 Scaling Attributes and Shaping Niches

Niching along the Pareto optimal front is similar to tiling a surface with equal-size niches. Thus we should consider the shape of such tiles (niches) since those shapes will effect the distribution of individuals on and near the frontier. Affecting niche shape are relative attribute scales and the metric used to measure distance between individuals.

As S. Ranjithan pointed out to us, vastly different scales for the attributes can severely affect the distribution of individuals induced by sharing. Most metrics treat each component of a spatial coordinate vector the same. If we use such metrics for sharing in raw, unscaled attribute space, we might end up comparing apples to oranges. For example, in a two attribute problem we might be trading off cost in the range of \$0 to \$1 million, for reliability ranging from 0 to 1. Our niche sizing method suggested above would be dominated by the largest ranging attribute (cost, in our example). Sizing σ_{share} for the "larger" attribute means that we will have sparse sampling of the smaller attribute, since σ_{share}, and hence a

[9]This term actually refers to the area of the niche that does not overlap with other niches and so is a hypersphere of diameter σ_{share}. The *total* area of a niche is a hypersphere of *radius* σ_{share}.

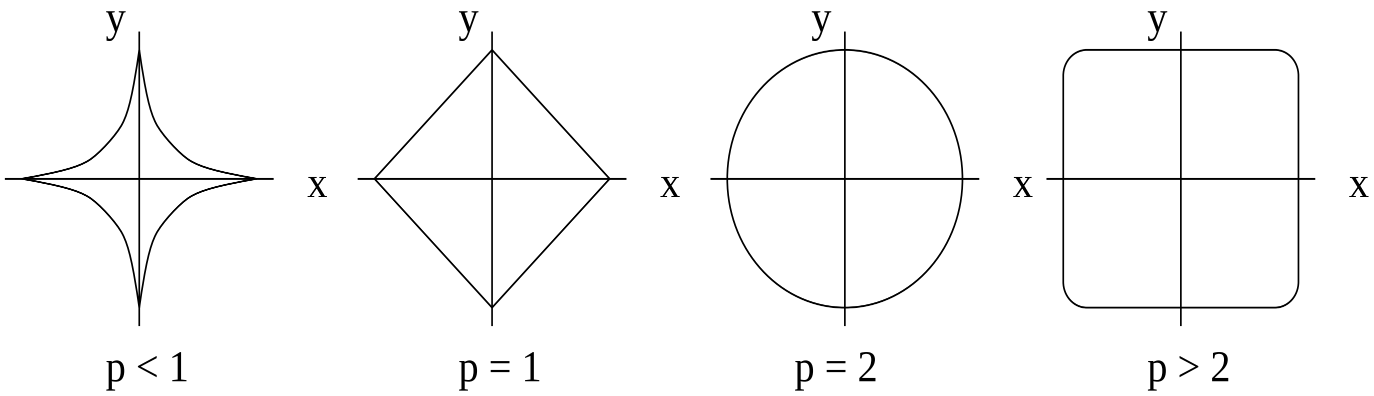

Figure 2: Niche shapes in two dimensions, as the degree p of the Holder metric varies.

single niche, could cover the entire range of the smaller attribute. Conversely, sizing niches for the smaller attribute would lead to convergence of the population to a small subrange of the larger attribute, since many more than N niches could fit into the larger attribute range.

The way to avoid this unintentional niching bias is to scale the attributes so that they all range over the same set of values, say 0 to 1. Note that this is only possible if we have some idea of a maximum and mininum value for each attribute, but this is often the case. Note also that after normalizing all attribute ranges to the range $\{0..1\}$, the above niche sizing method[10] leads to an upper bound of $2/N$ and a lower bound of $\sqrt{2}/N$ for σ_{share}.

We might actually want to bias our niching in favor of one attribute or the other. With care, differential scaling of attribute ranges might be a useful tool in controlling the distribution of niches along the Pareto optimal front. But another method of tuning niche distribution is easily visualized. This is the method of varying the degree of the Holder metric in order to change the shape of the niche.

In regular fitness sharing, we can change the parameters σ_{share} and α_{share} to affect the *radial extent* of the niche (Goldberg and Richardson, 1987). But varying the degree of the Holder metric,

$$distance(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

changes the *shape* of the niche, such that for $p \neq 2$, the niche is not radially symmetric. For example, in two dimensions ($n = 2$), Figure 2 shows how the shape of the niche varies with p. As $p \to \infty$, we approach the maximum holder metric, which is simply the absolute value of the largest component of the sum above. Thus, for large p and $n = 2$, we essentially have a square niche.

We are particularly interested in $p < 2$. Under such metrics, niches along diagonal lines would be more densely packed than niches along lines parallel to an attribute axis. If we used such a metric for sharing in the Niched Pareto GA, we would hope to see a higher distribution of individuals along portions of the front where the tradeoff is far from parallel to any attribute axis. Such regions are of particular interest to us since they represent sets of non-dominated solutions where the tradeoff is "best". Where the front is more nearly parallel with an attribute axis, we are trading off a lot of one attribute for a slight gain in another. But at "knees" in the surface, the tradeoff is closer to one-to-one. Thus we might wish to place more individuals in such regions to better explore and interpolate them.

In our applications below, we use the metropolitan, or city-block, metric. This is the Holder metric of degree $p = 1$.

[10] Assuming 2 attributes and Euclidean distance.

```
Unitation
12 |    P
11 |        -   P
10 |        -   -   -   P
9  |        -   -   -   -   -   P
8  |        -   -   -   -   -   -   -   P
7  |        -   -   -   -   -   -   -   -   -   P
6  |        -   -   -   -   -   -   -   -   -   -   P
5  |        -   -   -   -   -   -   -   -   -   -
4  |        -   -   -   -   -   -   -   -
3  |        -   -   -   -   -   -
2  |        -   -   -   -
1  |        -   -
0  |    -
----------------------------------------------------------
        0   1   2   3   4   5   6   7   8   9  10  11  12     Pairs
```

Figure 3: Problem 1's discrete, two dimensional attribute space, with feasible (-) and Pareto (P) points indicated.

5 Application to Three Problems

The interaction of the niching and dominance parameters (σ_{share} and Pareto tournament size, respectively) calls for further analysis of the behaviour of this new algorithm. But preliminary empirical results are promising. We find the Pareto frontier for an artificial, two-attribute function over bit strings (number of ones versus number of adjacent "01" or "10" pairs). We also apply our Niched Pareto GA to Schaffer's F2, a simple two-attribute function used to test Schaffer's multicriteria GA algorithm (VEGA) in his 1984 dissertation. The Niched Pareto GA demonstrates performance superior to VEGA in finding and maintaining more of the Pareto set. Finally, we test the algorithm on a real world, open problem whose non-dominated frontier is not known[11]

5.1 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. Both attributes are simple functions of finite length l bit strings. Unitation $Unit[s]$ is simply the number of ones in the 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). At unitation $\geq l/2$ a tradeoff exists, since adding more ones increases $Unit$ but must decrease Prs.

In Figure 3, we plot the feasible region of the two-dimensional attribute space $Unit$ versus Prs for a 12-bit problem ($l = 12$). P indicates a point on the Pareto front, while "-" indicates a feasible point that is dominated by some member(s) of the Pareto set. Note that much of the attribute space is infeasible (e.g., a string cannot have any adjacent complementary pairs when it has unitation 12, since it has no zeroes). Note also that a single point in attribute space can represent as few as one individual string (or zero, for infeasible points!) or many strings (e.g., $(unit, prs) = (1, 2)$).

[11]These applications are early tests of the Niched Pareto GA. Many of the insights and guidelines discussed in the previous section were not available at the time of these runs. Consequently, we did not always scale attributes or properly size niches, population, etc., as we have since learned to do.

```
Unitation                             Generation 0
12 |    0
11 |        0   2
10 |        0   0   1   4
9  |        0   0   1   2   1   2
8  |        0   0   0   4   8   1   2   1
7  |        0   0   1   2   5   3   6   1   0   0
6  |        0   0   2   2   6   3   5   6   0   0   0
5  |        0   0   6   2   2   2   1   6   0   0
4  |        0   2   0   1   1   3   0   1
3  |        0   0   0   0   0   1
2  |        0   1   0   0
1  |        0   0
0  |    0
----------------------------------------------------------
        0   1   2   3   4   5   6   7   8   9   10  11  12     Pairs
```

Figure 4: Distribution of the randomly generated initial population.

Although the unitation versus pairs problem is deliberately abstract, one can imagine a realistic analogy. For example, the strings might encode the types of structural elements in some design, with a zero representing an element of high cost and a one representing a less expensive element. Imagine further that adjacent, different elements complement each other and increase the overall strength of the design. Thus our two objectives are to minimize cost (maximize unitation) while maximizing strength (adjacent complementary pairs).

Next we test the Niched Pareto GA on the unitation versus pairs problem. In Figure 4, we plot the initial population (generation 0) of 100 randomly generated 12-bit individuals[12]. The numbers plotted in attribute space are the numbers of individuals (strings) with those attribute values. Notice how we obtain a spread of individuals (except at the "less likely" attribute coordinates, such as the all-ones or all-zeroes positions), with no bias towards or away from the Pareto frontier.

Figure 5 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 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. Figure 6 shows the population distribution in the 200th generation of a 28-bit problem [13], $N = 400$. In this run (Figure 6), our niche size σ_{share} was 5.0 and our tournament size t_{dom} was 10.

To illustrate the importance of sharing, we present in Figure 7 the results of running the Niched Pareto GA with sharing turned off (call it the *Pareto GA*). That is, when two competitors in a domination tournament tie (i.e., neither or both are dominated by the comparison set), we randomly choose one as the winner. Figure 7 shows that after 100 generations, the population is distributed over only a portion

[12] Our population size $N = 100$, tournament size $t_{dom} = 10$, niche size $\sigma_{share} = 2.0$, and we are using a single-point crossover probability $p_c = 0.9$ and mutation rate $p_m = 0.01$.

[13] Again, the GA misses one extreme point on the front, namely the strings with the highest number of pairs, and unitation of $l/2$. However, we have observed several runs where the GA did maintain an apparently stable subpopulation at this extreme point as well.

```
Unitation                                 Generation 100
12 |    26
11 |         0   11
10 |         0    0    0   14
9  |         0    0    0    0    0   12
8  |         0    0    0    0    0    0    0   16
7  |         0    0    0    0    0    0    0    0    0   21
6  |         0    0    0    0    0    0    0    0    0    0    0
5  |         0    0    0    0    0    0    0    0    0    0
4  |         0    0    0    0    0    0    0    0
3  |         0    0    0    0    0    0
2  |         0    0    0    0
1  |         0    0
0  |    0
-----------------------------------------------------------------
        0    1    2    3    4    5    6    7    8    9   10   11   12        Pairs
```

Figure 5: Stable subpopulations on the Pareto front.

```
Unitation                                 Generation 200
28 —  14
27 —      0 29
26 —      0  0  0 29
25 —      0  0  0  0  0 30
24 —      0  0  0  0  0  0  1 25
23 —      0  0  0  0  0  0  0  0  0 28
22 —      0  0  0  0  0  0  0  0  0  0  0 32
21 —      0  0  0  0  0  0  0  0  0  0  0  0  0 30
20 —      0  0  0  0  0  0  0  0  0  0  0  0  0  1  0 30
19 —      0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 32
18 —      0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1 32
17 —      0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1 27
16 —      0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  2 26
15 —      0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 27
14 —      0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
13 —      0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
12 —      0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
11 —      0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
10 —      0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
9  —      0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
8  —      0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
7  —      0  0  0  0  0  0  0  0  0  0  0  0  0  0
6  —      0  0  0  0  0  0  0  0  0  0  0  0
5  —      0  0  0  0  0  0  0  0  0  0
4  —      0  0  0  0  0  0  0  0
3  —      0  0  0  0  0  0
2  —      0  0  0  0
1  —      0  0
0  —   0
————————————————————————————————————————————————————————————————————————————
       0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28   Pairs
```

Figure 6: Success on a much larger problem, $l = 28$.

```
Unitation                                 Generation 100
28 —     0
27 —       0  0
26 —       0  0  0  0
25 —       0  0  0  0  0  0
24 —       0  0  0  0  0  0  0  1
23 —       0  0  0  0  0  0  0  1  0 12
22 —       0  0  0  0  0  0  0  0  0  2  1 26
21 —       0  0  0  0  0  0  0  0  0  0  0  2  1 58
20 —       0  0  0  0  0  0  0  0  0  0  0  0  1  2  1 77
19 —       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  5  1 86
18 —       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  8  0 67
17 —       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  5  2 29
16 —       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  5  0  3
15 —       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2
14 —       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
13 —       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
12 —       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
11 —       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
10 —       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
9  —       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
8  —       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
7  —       0  0  0  0  0  0  0  0  0  0  0  0  0  0
6  —       0  0  0  0  0  0  0  0  0  0  0  0
5  —       0  0  0  0  0  0  0  0  0  0
4  —       0  0  0  0  0  0  0  0
3  —       0  0  0  0  0  0
2  —       0  0  0  0
1  —       0  0
0  —     0
--------------------------------------------------------------------------------------
        0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28   Pairs
```

Figure 7: Premature convergence when sharing is turned off.

of the Pareto frontier. Not only is the Pareto GA not *maintaining* the whole front, it never even *finds* the other portions of the tradeoff curve in the first place. There is no sharing to spread out the population of the early generations. This search failure leads us to describe the Pareto GA behaviour as *premature convergence*. Eventually, the entire population should converge to a single point on the front, due to drift on the frontier.

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

5.2 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 (Schaffer, 1984). This is the simple function F2, with a single decision variable, the real-valued x, and two attributes, $f21$ and$f22$ to be minimized:

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

The decision variable is mapped to a 14-bit string as a binary-coded integer. Thus $00000000000000 = x_{min}$ and $11111111111111 = x_{max}$, where $x_{min} = -6.00$ and $x_{max} = 6.00$. We plot $f21$ and $f22$ over this range of x in Figure 8. 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.

For our purpose of comparison, we reproduce some of Schaffer's graphs illustrating VEGA's performance on F2. Figure 9 shows a plot of the population distribution at generation 0, taken from (Schaffer, 1985), and at generation 3, to be found in either (Schaffer, 1984) or (Schaffer, 1985). The population size is rathter small, $N = 30$. Schaffer used the results shown in Figure 9 to demonstrate that VEGA's selection scheme was moving the population towards points on the front, and that he was getting a distribution on that front. However, as can be seen on the right of Figure 9, the distribution on the front is rather clumpy.

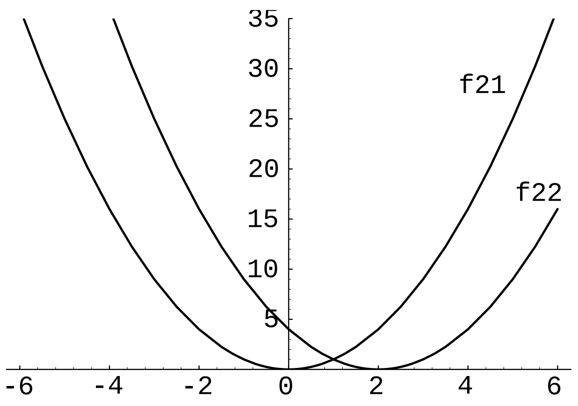

Figure 8: Schaffer's function F2, $P = \{x \mid 0 \leq x \leq 2\}$

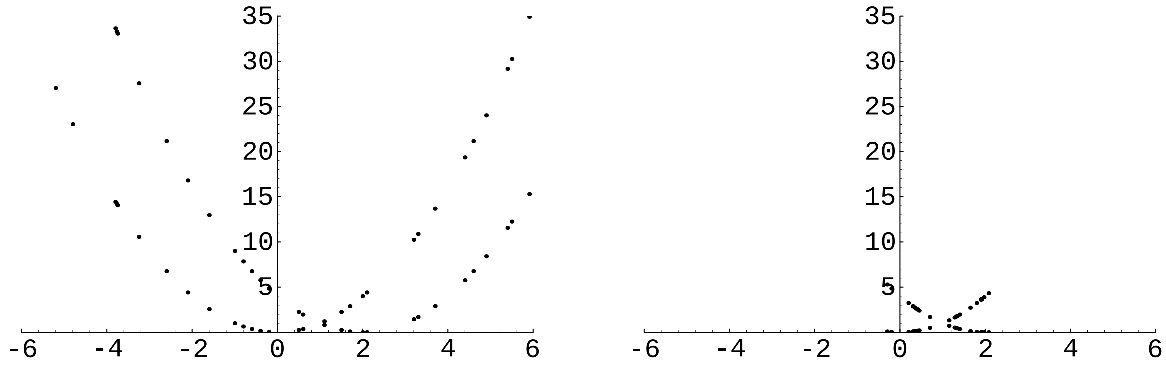

Figure 9: VEGA on Schaffer's function F2, generations 0 (left) and 3 (right).

We might hope that subsequent generations would show a better spread, as little convergence has taken place in most GAs after only two generations. However, Schaffer gives no further results on F2. This leads us to speculate that VEGA, with no explicit niching mechanism, is likely converging to a single point or small region on the front rather rapidly.

Next we try the Niched Pareto GA on F2. 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 10 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[14]. Most importantly, the Niched Pareto GA exhibits stability in this population distribution for many more generations than were indicated for VEGA ($200 > 3$).

We should point out that F2 is clearly 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.

[14] Larger population sizes result in much better coverage (smaller gaps) of the Pareto front. But we use 30 as our population size here for a fair comparison with the figure from Schaffer (1984).

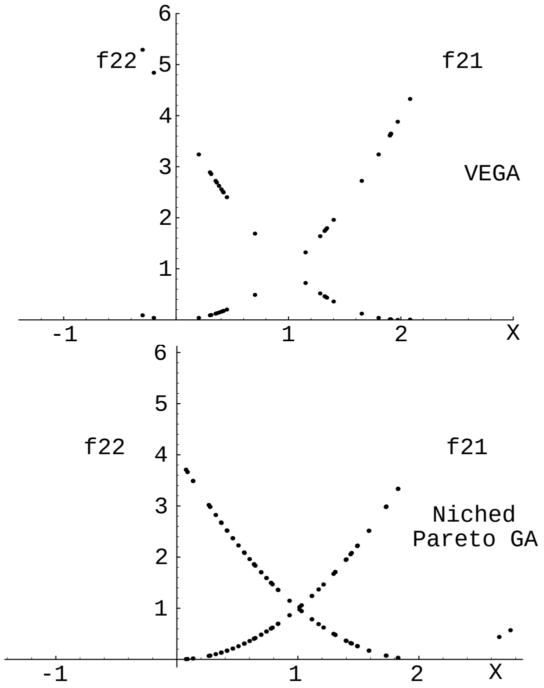

Figure 10: VEGA versus the Niched Pareto GA. Top: Expanded view of VEGA's performance on F2's Pareto frontier, generation 3. Bottom: The Niched Pareto GA's performance on F2, generation 200.

5.3 Problem 3: Open Problem in Hydrosystems

Neither of these two abstract problems provides a significant challenge to the Niched Pareto GA. Thus we chose a larger, real-world (i.e., unsolved) application for our third test problem: optimal well placement for contaminant monitoring[15]. Although we have not yet had time to experiment with various parameter settings, nor have we compared our solutions to those found via other algorithms, our first few runs indicate that the Niched Pareto GA is working. That is, as the generations go by, the population moves toward the Pareto frontier (i.e., in the right direction!) while spreading out in attribute space (i.e., maintaining high quality diversity). Thus the GA is finding and maintaining new and improved tradeoffs among the two attributes.

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 of $\{0, 0\}$ where we detect no plumes and therefore have no volume of contaminant to clean up. If we maximize the number of detected plumes (which would also be the environmentally correct thing to do) our volume of cleanup would skyrocket (which would displease the businessman). Therefore, there is a tradeoff involved. This is a perfect example where finding the Pareto optimal front is desired. By finding the front, we can give the decision maker a set of possible choices. He or she can then decide which choice is best.

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 and it makes apparent the need for a front end to MAUA.

[15] This open problem was developed and explored 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 for providing us with this challenging, multiattribute design problem.

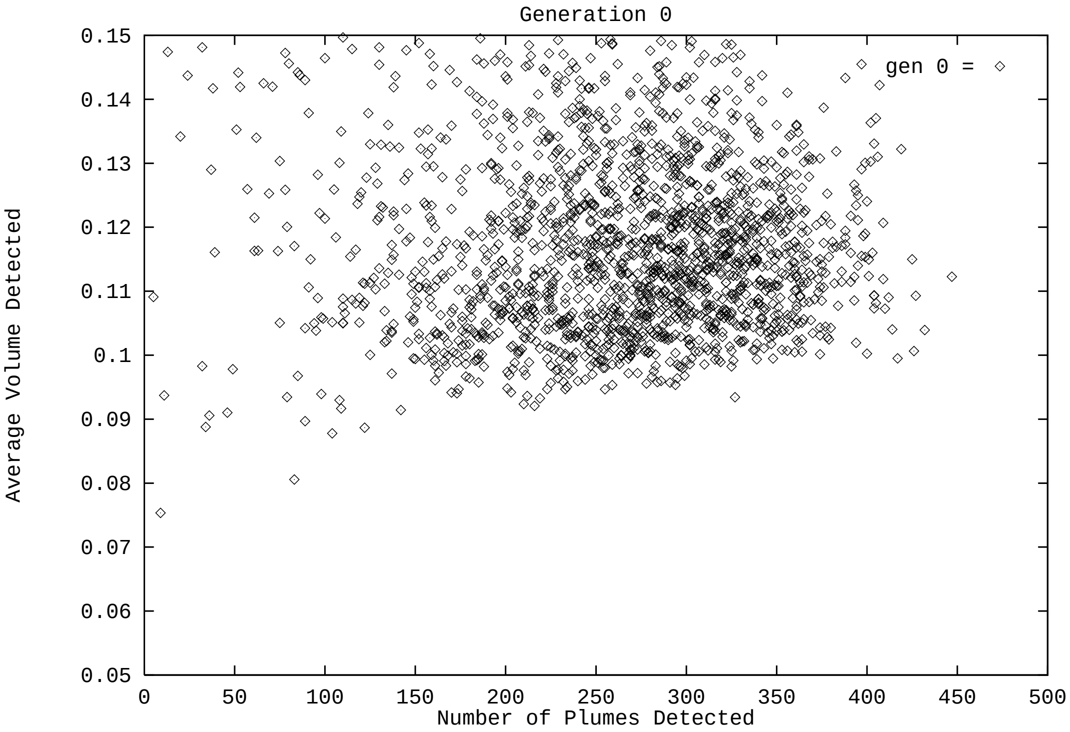

Figure 11: The initial population distribution

In this application we were given a set of simulations of leaks from a land fill site. 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 fitness function to return the number of plumes and average volume detected by any set of wells given to the function. Several preliminary runs were performed to determine whether or not the Niched Pareto GA could find Pareto fronts for this real world problem.

For our specific problem we applied a Niched Pareto GA to pick a set of $k = 20$ wells from a possible set of $w = 396$ wells. Our values for σ_{share} and the dominance tournament size t_{dom} were 40 and 40, respectively[16]. Looking at Figure 11 one can see that our random initial population is distributed throughout the search space. It is assumed that the Pareto optimal front will lie below and to the right of these initial values. Figure 12 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. Figure 13 shows both the initial population (indicated by diamonds) and the 230th generation population (indicated by crosses). There is definite improvement as to the location of the front and the decrease in the number of dominated individuals in the population.

These results show that our phenotypic sharing and dominance tournaments are working together. We do not know yet whether this is the actual Pareto optimal front or a sub-optimal front of solutions. To test for this, a more rigorous study of the function with many variations of σ_{share} and t_{dom} could be conducted. Below, we discuss other ways of testing the front found by the Niched Pareto GA. We have shown, however, that a tradeoff curve better than a random sampling can be developed by the Niched Pareto GA on an open problem.

[16] Because we had not yet developed any guidelines for setting these two parameters at the time of the runs, we simply experimented with combinations over broad ranges. We have since developed some insights into the proper settings for σ_{share} and dominance tournament size t_{dom}. These are discussed elsewhere in the report.

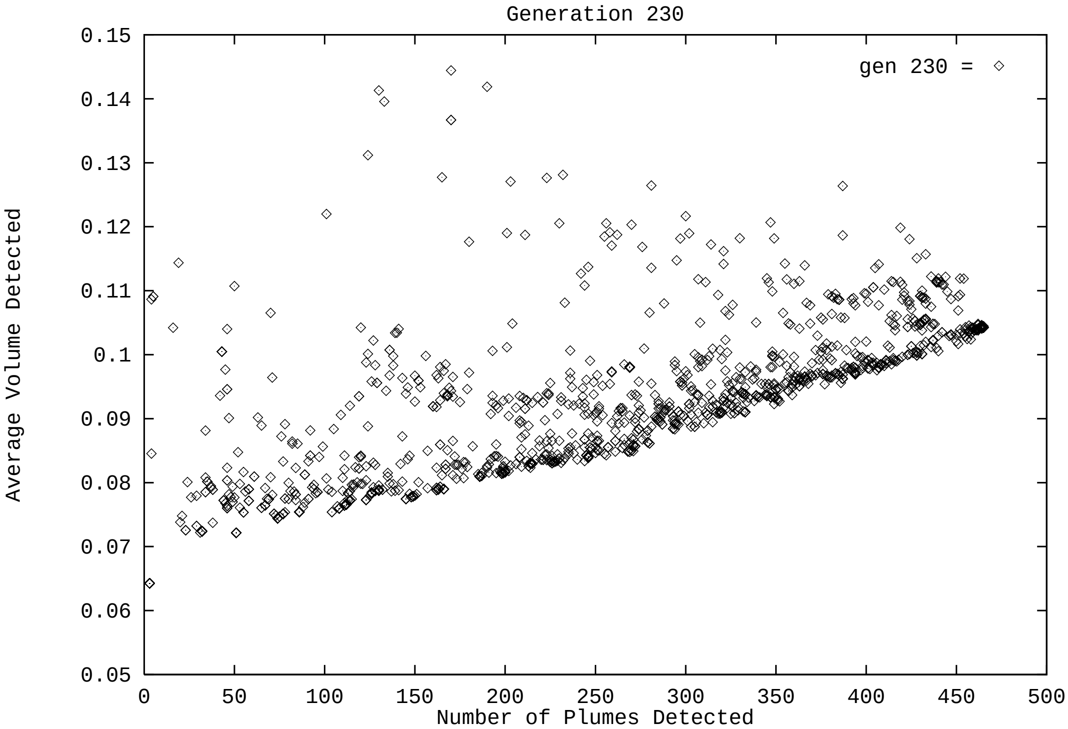

Figure 12: The population distribution after 230 generations

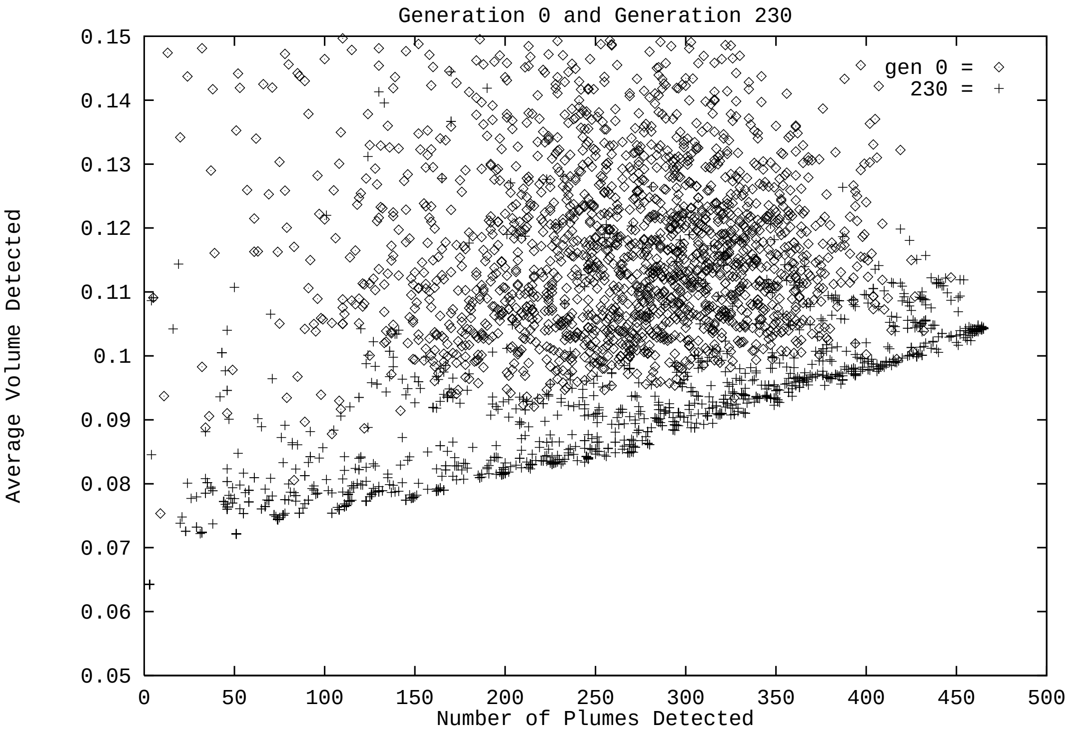

Figure 13: Results from a single run for $k = 20$ and $w = 396$

```
Unitation                    Generation 100
28 —   0
27 —     0 3
26 —     0 1 3 3
25 —     0 0 2 3 5 2
24 —     0 1 3 4 2 3 2 5
23 —     0 0 1 0 4 2 1 4 4 3
22 —     0 0 0 4 2 2 4 1 6 3 4 2
21 —     0 0 1 1 0 1 2 5 4 6 5 3 1
20 —     0 0 1 0 0 2 0 3 1 4 5 4 5 5 3
19 —     0 0 0 0 0 0 1 1 4 3 2 5 4 6 6 4 3 4
18 —     0 0 0 0 0 0 0 2 0 2 4 2 4 4 5 6 3 5 2 0
17 —     0 0 0 0 0 1 1 1 1 1 0 1 4 4 5 5 8 4 6 3 4 1
16 —     0 0 0 0 0 0 0 1 0 1 0 3 2 5 5 1 3 4 3 4 5 5 3 1
15 —     0 0 0 0 0 0 0 0 0 1 0 0 0 1 3 1 4 2 6 1 4 2 4 0 1 0
14 —     0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 1 3 1 3 3 0 4 4 0 1 0 0
13 —     0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 2 2 1 0 2 0 1 1 1 1 0
12 —     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0
11 —     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
10 —     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
9  —     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
8  —     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7  —     0 0 0 0 0 0 0 0 0 0 0 0 0 0
6  —     0 0 0 0 0 0 0 0 0 0 0 0
5  —     0 0 0 0 0 0 0 0 0 0
4  —     0 0 0 0 0 0 0 0
3  —     0 0 0 0 0 0
2  —     0 0 0 0
1  —     0 0
0  —   0
--------------------------------------------------------------
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28   Pairs
```

Figure 14: Population spread on problem 1 when domination pressure is too low ($t_{dom} = 2$).

6 Discussion

The initial implementation and testing of the Niched Pareto GA leads to some intriguing ideas for immediate and long term extensions. In particular, we consider how to properly adjust the domination pressure, how to extract a closed form description of our estimated front, how to evaluate the Niched Pareto GA's performance, how to find multiple solutions at a single point on the front, how to incorporate uncertainty, and the critical issue of search.

6.1 Setting Selection Pressure

Earlier we showed how to adapt Deb's sizing estimates for σ_{share} and population size N. Our experimental runs showed that the performance of the Niched Pareto GA was robust with respect to σ_{share}, with similar performance over a range of values. But so far we have not yet developed any guidelines for selection pressure (i.e., the size of the domination tournaments t_{dom}). Our experience with the three applications above has given us some empirical insight into the importance of this parameter. Although we found that the algorithm was fairly robust with respect to t_{dom}, we found significantly different behaviour once t_{dom} exceeded this large range of values.

In general, increasing the size of the dominance tournament seemed to increase pressure towards the front, as we would expect. The higher domination pressure caused the population to move more rapidly in the direction of the front (that is, improvement in both attributes). But it also seemed to curtail exploration. Too high a value for the dominance tournament size (e.g., $t_{dom} = 200$ for the hydrosystems application above), and the population "prematurely converged" to only a portion of the front, usually near the middle. The effect was the same as when sharing is turned off, as in Figure 7. Conversely, with too small a dominance pressure (e.g., $t_{dom} = 2$), the final population consisted of many dominated solutions. That is, with too little selection pressure with respect to sharing, the population failed to achieve a tight distribution on the front, and was distributed well behind, as well as on, the front. For an example of this behaviour, compare Figure 14, in which $t_{dom} = 2$, to Figure 6.

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. Therefore, this value must be chosen carefully for each problem that uses the Niched Pareto GA. From our limited experiments, we have order of magnitude guidelines:

- When $t_{dom} \approx 1\%$ of N, we saw too many dominated solutions.
- When $t_{dom} \approx 10\%$ of N, we witnessed a tight and complete distribution.
- When $t_{dom} \gg 20\%$ of N, the algorithm prematurely converged.

Until analytical guidelines are developed, we will continue to explore the algorithm's behaviour over a range of t_{dom} on each new problem.

6.2 Extracting a Front

In this study we were more concerned with our algorithm's on-line performance than with its off-line, best-of-run statistics. But if we were asked to present our "best front" to the decision maker, it is obvious, looking at Figure 13, that simply presenting the final generation will not do. We should first find the off-line, best-of-run results, and then possibly interpolate a tradeoff surface for presentation to the decision maker.

6.2.1 Non-domination Sorting: Best-of-Run

Finding the best-of-run simply requires a non-domination sort of all individuals generated over all generations. Equivalently, we could simply update the best-so-far Pareto set each generation by sorting the union of the best-so-far with the current generation, thanks to the transitivity of the domination relation. The final best-so-far set will then be the best-of-run. We hope that the best-so-far set differs little from the current generation (and that the best-of-run differs little from the final population). However, the noise inherent in our genetic operators, and the small niche counts of equivalence class fitness sharing, imply that our best-of-run/best-so-far sets in general should be larger (and therefore be a denser representation of the front) than any single population.

6.2.2 Interpolating a Function to Describe the Front

When our Pareto front is continuous, or otherwise contains many points, we will probably want to represent our "answer" as a tradeoff surface rather than a collection of points. Several standard methods exist for interpolating a surface from a set of points. Whatever method is used, the resulting surface must meet the Pareto non-domination criterion, by having all first-order partial derivatives of constant sign. This constraint can be enforced explicitly, or the user can experiment with low-order interpolating functions, such as polynomials of small degree. Figure 15 shows a possible quadratic interpolation of a Pareto optimal set.

6.2.3 Tweaking the Front

Upon presentation to the decision maker, two problems could arise:

1. The decision maker picks a point on the tradeoff curve that is not represented in the best-of-run set.
2. The decision maker wants a point that is in front of the interpolated Pareto optimal front.

Both situations are illustrated in Figure 16.

If a specific point is desired whose attributes are on or very near the interpolated Pareto optimal front but which is not represented in the final population of our GA, the situation might be resolved by creating a new single-attribute fitness function for the regular, single-objective GA. The fitness function would be to minimize the difference between our desired attribute values and any single individual's attribute values. The best-of-run set could provide the initial population for such targeted refinement.

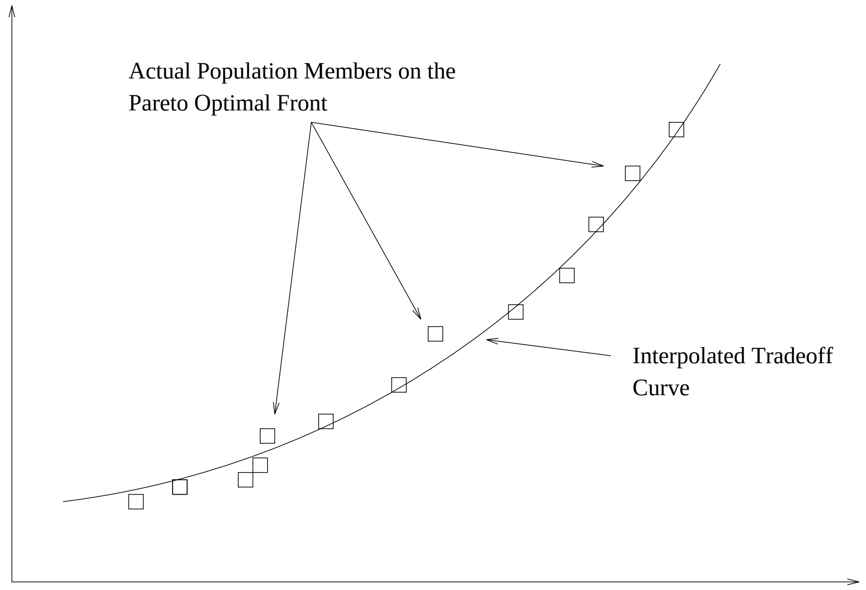

Figure 15: An example tradeoff curve interpolated from a Pareto optimal set.

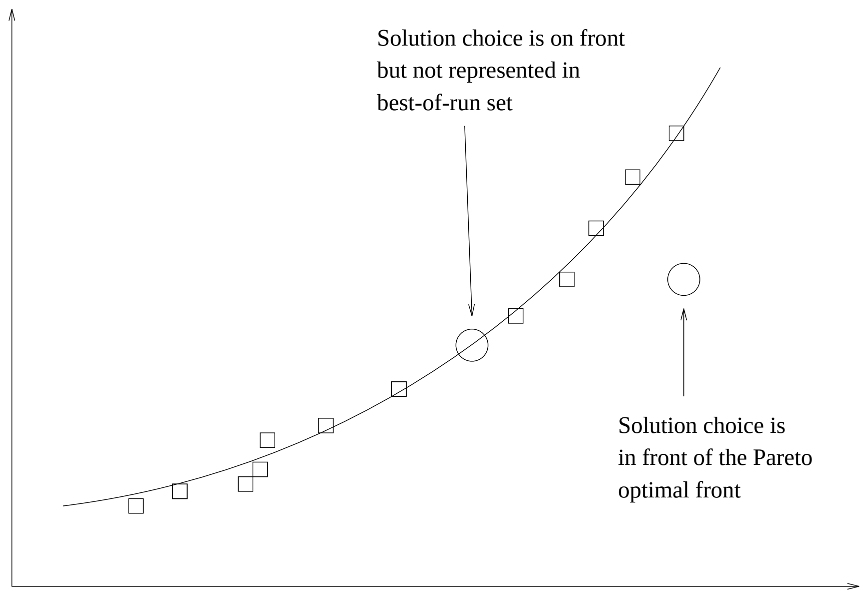

Figure 16: Possible problems with a decision maker's solution choice.

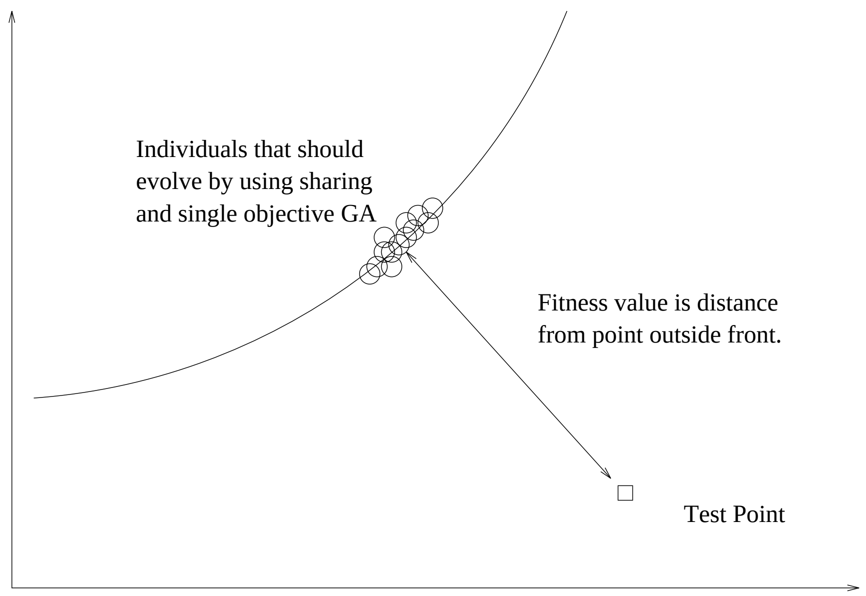

Figure 17: Using a single-attribute GA to find portions of the Pareto optimal front.

6.3 How to Test the Niched Pareto GA

Testing the Niched Pareto GA is a difficult task on some problems. For our first two test functions, problems 1 and 2, we knew exactly what the Pareto optimal set would be and could easily verify our results. However, the Niched Pareto GA would be most useful on applications such as our third test function, problem 3. Such problems are intractable; there is no known Pareto optimal front and we don't know how the Niched Pareto GA should perform.

A simple, though not complete test, would be to check for the endpoints of the front. These endpoints should be the result of optimizing for each single attribute. Sometimes, we know these endpoints apriori. For example, in problem 3 above the lowest cost solution is any set of wells that detect no plumes. Typically, outlying wells detect nothing. If we drill only at such locations, we will have no cleanup costs (and a 0% reliability!). When we don't know the endpoints, such as the maximum *possible* reliability (using just k out of the w wells) in problem 3, we could turn to scalar optimization techniques, such as simulated annealing.

Perhaps most intriguing is the idea of using a regular, single-objective GA to check the performance of a multiattribute GA. We could apply a regular GA to the optimization of each attribute separately. If the best the regular GA (or any other scalar optimizer) could do matches the corresponding endpoint of the found Pareto optimal front, then at least we know that we are on the right track. We could also use a regular GA, or other scalar optimizer, on various linear combinations of the attributes (e.g., optimize a*COST + b*RELIABILITY to get some points in between the endpoints). The regular GA might have better luck finding each point, since it is concentrating all of its population resources on optimizing a particular attribute, or scalar combination of attributes.

Another possible way to check or improve the performance of the Niched Pareto GA is to use a single-attribute GA with sharing and attempt to develop subsections of the Pareto optimal front. Here we could implement as a fitness function the distance to a point that lies ahead of what is believed to be the Pareto optimal front. Individuals should emerge in the final population that define a small subsection of the Pareto optimal front, as suggested by Figure 17. Perhaps one would be able to define the front in a piecewise manner using this method as well. Though it would eventually become computationally prohibitive, such a piecewise approach might be a worthwhile way to check for portions of the Pareto optimal front.

The methods described in this section can indicate, but not guarantee, that the Niched Pareto GA is coming close to the actual Pareto optimal front.

6.4 Genotypic versus Phenotypic Diversity: Nested Sharing

The Niched Pareto GA implements phenotypic sharing, where the attribute vector is the phenotype. We are first and foremost interested in maintaining diversity along the tradeoff surface. But if we achieve such diversity, we might then also be interested in finding diverse solutions at given tradeoff points. In other words, we might want to know about the several different ways of achieving the same set of attribute values. We would then be spreading out our solutions in both attribute space and the space of decision variables[17].

Simultaneously searching for two kinds of diversity is a goal that trancends the multiattribute focus of this report. We present some possible techniques which, if successful, could be applied to sharing in regular, scalar GAs as well:

- **Pointwise expansion -** We pick a point on the Pareto front (i.e., in attribute space) and search for diversity in decision variable space. We first run the Niched Pareto GA to find a front. We then run a niched GA in which the objective fitness function is distance, measured in attribute space, to the chosen point on the front. This distance is to be minimized. Sharing is performed in decision variable space. This method only explores one point, or small region, of the front at a time, but brings the entire population to bear on that search.

- **Threshold sharing -** If two individuals are in the same equivalence class, and we have calculated their niche counts (in attribute space) and their niche counts are within some fixed δ of each other, then we perform the second level of sharing (in decision variable/genotype space). We do this by simply calculating and comparing their *genotypic* niche counts. This approach introduces a new parameter, δ.

- **Sequential sharing -** Generate a new population as before (with attribute sharing). From this intermediate population, generate the next population by performing sharing in decision variable space. In other words, perform selection using the Niched Pareto GA with sharing in attribute space. Then perform selection using the Niched Pareto GA with sharing in decision variable space. Finally, perform crossover and mutation.

- **Simultaneous sharing, multiplicative -** (This algorithm was suggested by Georges Harik.) Multiply the niche count in attribute space by the niche count in decision variable space and use this product as the overall niche count for comparing two individuals in the same equivalence class. It might be necessary to raise the attribute niche count to some power to reflect the relative importance of maintaining diversity along the Pareto front first and foremost.

- **Simultaneous sharing, additive -** Similar to the algorithm above, but add the two different niche counts instead of multiplying them. This is similar to combining the attribute and decision variable spaces into a single space (which is yet another approach to multi-level or nested sharing!). Or we could simply change the metric to encompass both spaces. But in all cases, we would likely have to use weights, or else scale attributes and decision variables, to emphasize the primary importance of attribute space diversity.

All of these ideas could be adapted for multi-level sharing in a scalar search space, and are thus of general interest. In the case of the Niched Pareto GA, we should first make sure we can find and maintain the Pareto front before we expand our search in other dimensions. However, we are ultimately very interested in decision variable diversity because it would add robustness to our solutions[18].

[17] The decision variables are often considered to be the genotype, even though there might be an extra step involved in decoding from a binary string, for example, to real-valued decision variables. For the purposes of this discussion, we do not distinguish between the genotype of the low level encoding and the phenotype of the decoded decision variables. Both are much more similar to each other than to the "high level" phenotype of the attribute vector.

[18] If for some reason one of the given solutions becomes infeasible at a later date, we would have an alternative solution with the same performance.

6.5 Incorporating Uncertainty

We have presented two alternatives for dividing up the challenges of multiobjective optimization between the GA and MAUA. In the MAUA$\Rightarrow$GA alternative, MAUA handles the challenge of uncertainty in the attribute values. In the GA$\Rightarrow$MAUA alternative, it is not clear how we should handle uncertainty. We might rely wholly or partially on MAUA, and we might ask whether or not we can use GAs to help deal with uncertainty. Two methods immediately come to mind:

- Calculate vectors of expected attribute values
- Let the GA handle uncertainty as noise in the fitness function

While we do not have the time or space here to fully explore these ideas, we feel it is important to present our preliminarny results in order to emphasize, to the GA community in particular, the difficulty of uncertainty and hence the importance of MAUA. First, we review the fundamental problem of uncertainty.

6.5.1 The problem of uncertain attribute values in decision analysis

So far we have assumed a deterministic evaluation function $A : \vec{x} \to \vec{a}$, where A maps a vector of decision variable settings $\vec{x}$ into a vector $\vec{a}$ of n attribute values. In general, in MAUA, we assume that A is non-deterministic, so that $A(\vec{x})$ can take on a range of attribute vectors for each $\vec{x}$. We do assume, however, knowledge of the probability distribution. For example, in the discrete case, we would know every probability $P(\vec{a}_i|\vec{x})$ for every possible outcome $\vec{a}_i$ of $A(\vec{x})$. Note that $\sum_{\forall i} P(\vec{a}_i|\vec{x}) = 1$. Thus, an evaluation of A for $\vec{x}$ would yield a set of pairs $(\vec{a}_i, p_i)$, one for each possible outcome. For the rest of this report, we use only discrete probability distributions in our examples.

We note here also that having objective or subjective knowledge of the probability distribution of the possible outcomes is sometimes said to be decision making *under risk*, with the term *under uncertainty* reserved for the more general case of having little or no knowledge of the probability distribution (Sawaragi, Nakayama, & Tanino, 1985)[19]. However, for our discussions we will assume knowledge of the probability distributions over the possible outcomes when we talk about decision making under uncertainty.

The problem is how to compare these sets of possible outcomes with each other. A naive, but "pure", approach would be to again use the idea of non-domination ranking and preference. We could define a partial order on all such sets, with one set preferred to another if and only if each pair $(\vec{a}_i, p_i)$ of one set "dominated" every pair in the other set. Here domination could mean that the vector of attribute values $\vec{a}$ from the first pair Pareto dominates the attribute vector from the second pair, AND that the probability p from the first pair is greater than that of the second pair. Weaker domination relations could also be defined. It seems likely, however, that such partial orders would be quite "flat", with many more unordered than ordered pairs. Thus a search based on such a partial order would probably lead to a Pareto optimal set not much smaller than the size of the entire search space.

Below we present somewhat more thoughtful candidates for handling uncertainty during multiattribute GA search.

6.5.2 Calculate vectors of expected attribute values

The general idea of this approach is to reduce the set of possible outcome vectors to a single outcome vector. This vector of attribute values might in some sense represent the "expected" vector, much as an expected value is defined for scalars. We discuss the meaningfulness of such a concept for vectors at the end of this subsection. First we present three candidate methods for calculating the "expected attribute vector":

- Multiply attribute vectors by probabilities

[19] It is not clear how one could handle such general uncertainty without making *some* assumptions about the probability distribution(s).

- Use uncertainties as separate attributes
- Use single-attribute utilities as attributes

Multiply attribute vectors by probabilities One of the first possible methods that comes to mind is to multiply each possible outcome by the probability of that outcome, and then to combine the results into an *expected outcome*, which is a vector of expected attribute values, $\vec{E}[a_i]$ or $\vec{E}[\vec{a}]$. The expected outcome procedure is inspired by MAUA, which multiplies scalar utility values of outcomes by their probabilities of occurence and combines them into an expected utility $E[U(\vec{x})]$ for the setting $\vec{x}$ of decision variables. But utility theory relies on the axiom of *substitution* or *independence* (de Neufville, 1990). Do we have such axiomatic justification for the expected outcome vector $\vec{E}[\vec{a}]$?

The answer is no. The substitution axiom only applies to *utility* functions. It does not apply to the attribute values themselves. The substitution axiom assumes that a person's *preferences* (i.e., relative utility values) are linear in probability. The utility values themselves are not, in general, linear in the attribute values, nor in the probabilities of such values.

Another way to think about the meaninglessness of expected attribute values is in terms of the information lost in their calculation. Calculating an expected value of a single attribute ignores the decision maker's attitude towards uncertainty. An expected value of a_i tells the decision maker nothing at all about the uncertainty in that value. There might be none at all, for example. A utility function, in contrast, includes the decision maker's attitude towards uncertainty (e.g., risk aversion for a certain range of the attribute value). Thus an expected utility value, although not telling us anything about the uncertainty in that value, already incorporates the decision maker's *tolerance* of uncertainty in that a_i. If we choose not to address the uncertainty at the leaves of the decision tree (i.e., at each possible outcome), then we must somehow preserve the information about uncertainty as we work backwards in the decision tree. This leads us to our second idea for calculating expected attribute vectors: uncertainty as a separate attribute.

Uncertainties as separate attributes Rather than using the uncertainty (as measured by probability in the above paragraph) of an *outcome*, we could use uncertainty of an *attribute* as an attribute itself. Suppose we calculate a vector of expected attribute values for an alternative, as described above. We then also calculate the variance σ^2 for each of these expected attribute values. We include these n variances as an additional n attributes, thus doubling the number of attributes in the problem from the original[20] n. The new vector, although twice the dimension of the original, gives us information about the uncertainty in the expected attribute vector. This information can be used in the search to allow "more certain" (lower σ^2) solutions to dominate less certain alternatives, all other attributes being equal (or dominated by the more certain individual). The variances can also be used by the decision makers in choosing among the alternatives in the final Pareto front found by the GA.

This method needs to be explored. There are several possible drawbacks. If the probability distributions are very different from a Gaussian, σ^2 might not be a very meaningful measure of uncertainty. In addition, doubling the dimension of the search space might render the problem too large and difficult even for the Niched Pareto GA. Finally, the Pareto optimal set for the expanded problem might be orders of magnitude larger than for the original problem.

Single-attribute utilities as attributes So far, we have resisted bringing in MAUA to help our GA deal with uncertainty. But we could bring in *part* of MAUA specifically to help us deal with uncertainty. MAUA's treatment of uncertainty is embodied in the single-attribute utility functions u_i. The estimation and calculation of the multiattribute tradeoff parameters, namely the k_i and big-K scaling factors, do not incorporate uncertainty and so are not needed here.

[20] Assuming that all n original attributes are uncertain. For attributes with no uncertainty, we do not need to calculate variances, nor expected values for that matter.

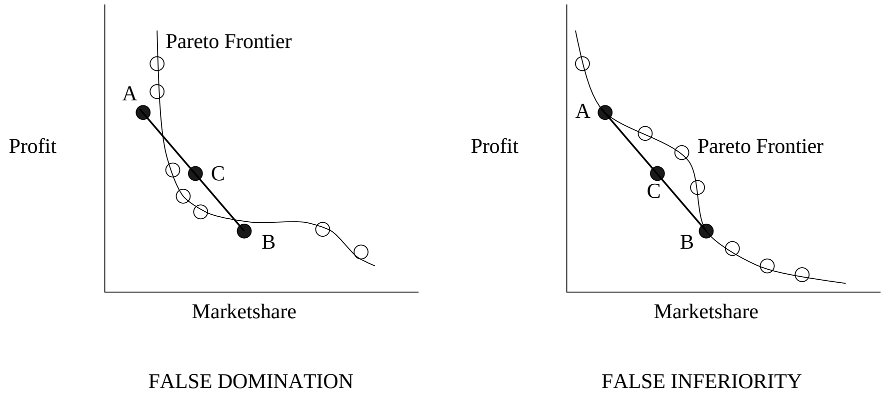

Figure 18: Domination Aliasing.

If we assess the u_i, then we can evaluate each possible outcome as a vector of u_i values[21]. This allows us to multiply each such vector by its probability of occurrence, and add the results to calculate a more meaningful expected vector. Rather than an expected attribute vector, we will have an expected utility vector, or vector of expected utilities. The expected utilities are meaningful, whereas the expected attribute values were not. But we are still left with the question: how meaningful is the *vector* of expected utilities? Can we compare such vectors in terms of the domination relation? We try to answer such questions below.

Meaningfulness of expected vectors The meaningfulness or meaninglessness of expected vectors and vectors of expected component values is treated in the literature. Here we simply point out one problem with using such expected vectors, whether an expected attribute vector or expected utility vector. This is the problem of *domination aliasing* and it leads to *false domination* or *false inferiority*. Figure 18 illustrates these two phenomema.

Suppose that in a two-attribute problem, as shown in Figure 18, one alternative solution has two possible outcomes $\vec{A}$ and $\vec{B}$, whose coordinates in attribute space are shown. Suppose that $\vec{A}$ is the actual outcome with probability p and $\vec{B}$ is the outcome otherwise $(1-p)$. Then the expected outcome vector, $\vec{C}$, is the linear combination $p\vec{A}+(1-p)\vec{B}$, with $0 \leq p \leq 1$. So $\vec{C}$ lies somewhere on the straight line connecting $\vec{A}$ and $\vec{B}$. Assume we are maximizing both attributes. On the left of Figure 18 we show how $\vec{C}$ might dominate some portion of the Pareto front, even though neither $\vec{A}$ nor $\vec{B}$ are on the front themselves. Thus $\vec{C}$ can dominate other alternatives that are not only on the front, but have no uncertainty in their outcomes. This can be described as "false domination".

The opposite situation can also occur, as shown on the right of Figure 18. Although $\vec{A}$ and $\vec{B}$ both lie on the front, their expected vector $\vec{C}$ is dominated, not only by other members of the front, but by points between $\vec{C}$ and the front as well. This phenomenon might be called "false inferiority".

On the other hand, $\vec{C}$ cannot be dominated by any point dominated by $\vec{A}$ or $\vec{B}$. Thus transitivity is preserved, so we maintain a partial order. However, the ordering can be different from the original. A good question is how different. If, in general, the possible outcomes for a given alternative are relatively close in attribute space, then perhaps $\vec{C}$ can approximate the relative performance of that alternative over many evaluations.

[21] We are now using the single-attribute utilities as our attributes for the GA search, rather than the attributes themselves. Here we do this to take advantage of the u_i's incorporation of the decision maker's attitude towards uncertainty. In general, however, using single-attribute utilities might result in an easier search space than the original attribute space, due to the nonlinear emphasis of the u_i on certain ranges of an attribute's values. However, calculating u_i can be a lot of work, and can introduce errors from the assessment process. Also, u_i vary with each decision maker.

6.5.3 Let the GA handle uncertainty as noise in the fitness function

An approach quite different from the expected vector methods would be to treat uncertainties in the attributes as noisy fitness functions for the GA. It is well known in the GA community that GA search can tolerate certain types and amounts of noise in the fitness function, such as non-stationary functions and random non-deterministic function increments/decrements. Because the GA is itself a non-deterministic, stochastic algorithm, the noise in the function is often overcome. Specifically, the evolution of a building block (i.e., subset of allele settings) through the generations relies on many fitness evaluations of many individuals. The GA implicitly works with the expected fitness values of building blocks. Perhaps we can rely on GA processing to evaluate the expected performance of an alternative with uncertain outcomes, rather than trying to calculate an explicit expected vector or outcome to hand to the GA.

The implementation of this approach is simple. If we know the probability distribution of possible outcomes for a particular alternative, then when the GA calls for an evaluation of that alternative, we simply choose one of the possible outcomes according to the probability distribution. Using our example from Figure 18, we would choose outcome $\vec{A}$ with probability p and outcome $\vec{B}$ with probability $1-p$. We don't have to think hard to justify this approach, since we are giving the GA the expected performance of this alternative in reality. That is, after many evaluations M, we expect that pM times we get $\vec{A}$ and $(1-p)M$ times we get $\vec{B}$. How well the GA can work with this information and what kinds of Pareto sets it will present to the decision maker remains to be seen. We first need a better understanding of what the Niched Pareto GA is processing, and what *we want* to see on the Pareto frontier *under uncertainty*.

6.6 The Issue of Search

Throughout this report we have concentrated on the usefulness of the final solutions, maximum $U[\vec{x}]$ versus P. We have not discussed the critical issue of search[22]. Both the MAUA and Pareto approaches are searching the space of decision variable settings $\vec{x}$. But while the MAUA approach explores the behaviour of $U[\vec{x}]$ over that space, the Pareto approach looks at how the currently non-dominated set grows or shrinks over $\vec{x}$. Which type of search is easier? Which exploits the best information? The answers might be problem dependent. 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. Of course, spreading the population out all along the front limits the resources available to explore a particular region of the front. However, 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 (non-dominated) tradeoffs. Indeed, we see some evidence for this in application 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. In other words, 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.*

In the case of the MAUA approach, some intuitive arguments could be made that the composition of the function $U[\vec{a}]$ with the function $A[\vec{x}]$ to yield $U[\ A[\vec{x}]\ ]$ should help the GA search. Certainly any "scalarization" of a multiattribute problem should concentrate the GA search on a particular subspace of attributes, driving toward a single point on the Pareto front. The entire population can be used to explore this subset of attribute space. In addition, the nonlinearities introduced in the attribute utility functions

[22]Almost all analysis of niching methods, including this report on the Niched Pareto GA, has focused exclusively on the steady state behaviour of niching, rather than on the search for that steady state (e.g., Deb, 1989; Horn, 1993; Mahfoud, 1993). This exclusive focus can be justified, however, since we must be sure first that there exists a steady state before we worry about how a niched GA might reach it.

$u_i[x_i]$ and in the multiplicative pairs $u_i[x_i]u_j[x_j]$ might provide important information during GA search. Indeed, even a linear sum of attributes can give the GA useful search information. As a simple example, improvement in one attribute might be twice as important as in another attribute, a fact that cannot be used by a Pareto GA. Plainly stated, our intuition is that searching a total order should be easier than searching a partial order, in general.

7 Conclusions

It is not clear what is the best way to handle multiple objectives with a genetic algorithm. Indeed, different approaches might be called for by different situations (e.g., small Pareto set, group decision making). But it is clear to us that the various methods reviewed or introduced in this report represent some of the best techniques to date. We in the GA community have been ocassionally successful using an ad-hoc linear sum of attribute values or a constraint-based approach. Now that utility theory and MAUA have become rigorous, proven techniques in decision analysis, we can systematically incorporate nonlinear interactions among attributes in our scalar objective functions $U[\vec{x}]$. In addition, MAUA incorporates the decision maker's attitude towards uncertainty.

However, although it is superior to any ad-hoc linear or constraint-based approach, MAUA is limited to multiplicative pairwise interactions, and cannot incorporate other types of nonlinearities. In addition, MAUA requires a somewhat involved interview of the decision maker(s) by the decision analyst. This process produces at best an *estimate* of $U[\vec{a}]$, with significant errors introduced if care is not taken. Furthermore, this utility function is specific to a particular decision maker.

In contrast, the Pareto approach is independent of the decision maker. The Pareto optimal set P depends only on the objective function $A[\vec{x}]$, which maps decision variable vectors to attribute vectors. No matter what utility function a decision maker brings to the problem, the global optimum of $U[\vec{x}]$ will always be found in P. And if P is small enough or otherwise easily visualized, the decision maker might be able to choose the best alternative directly, without an assessment of $U[\vec{a}]$. We introduced the Niched Pareto GA as an algorithm for finding and maintaining solutions all along the Pareto front P. But P might be very large and/or convoluted, and therefore impossible for the Niched Pareto GA to maintain or even find.

The MAUA and Pareto approaches conduct GA searches of the same space $\vec{x}$, but in very different ways. Which approach has the advantage is likely to be problem specific but perhaps in very general ways. Further research might reveal some general guidelines for what kinds of problems are best treated by one method or the other. For now, we believe it is best to first conduct a Pareto search. That is, find the best non-dominated front using an algorithm like the Niched Pareto GA. Then, as time permits, test this solution using some of the methods suggested in this report. With sufficient evidence that the found frontier is a "good" one, it can then be examined by the decision maker. If a best choice is not obvious, then perhaps MAUA is called for. If there is evidence that the Niched Pareto GA, or similar algorithm, is failing to find a good tradeoff surface, the MAUA approach can be taken. Where time and resources allow, perhaps it is best to perform both MAUA and Pareto optimization, using each as a check against the other. In general, each method will tell us something different about the complex, multiattribute problem at hand. It might be worthwhile to redo some of the scalar GA applications in the literature as multiattribute problems using the MAUA or Pareto approaches, and see what other solutions can be found.

Finally, we have introduced several techniques that are applicable outside the set of multiobjective problems:

- Partial order optimization
- Equivalence class sharing
- Nested sharing

The domination tournament does not rely strictly on a domination relation, but rather on any antisymmetric, transitive relation. Thus it can be used to optimize any partially ordered space. Equivalence class sharing (or flat fitness sharing) is useful not only on the Pareto optimal frontier, but in any equivalence class induced by a partial order. Even in a totally ordered set (i.e., scalar function optimization), such sharing is useful in large niches, such as high plateaus, to obtain a representative sampling of the entire highly-fit region. Nested, or multilevel, sharing has only been discussed, not demonstrated, in this report. However, if successful, it would allow us to find diversity in several spaces simultaneously. Nested sharing represents a generalization of fitness sharing to multiple spaces akin to the generalization of the scalar GA to multiple objectives.

8 Acknowledgements

The authors would like to thank David E. Goldberg, with the General Engineering Department at the University of Illinois at Urbana-Champaign (UIUC), for his guidance and encouragement of our initial investigations of GAs and Pareto optimality. It was his insight that GAs, with niching in particular, are well-suited to finding the Pareto frontier. We also acknowledge the critical input of Deborah Thurston, also with UIUC's General Engineering Department, on the topics of decision analysis and MAUA. We are very grateful to Wayland Eheart and his students Sanmugavadivel Ranjithan, Pascal Stork, and Scott Cieniawski at UIUC's Department of Civil Engineering for helping us implement their ground water monitoring problem as our objective function in problem 3. Finally, the first author acknowledges support from NASA under contract number NGT-50873, while the second author acknowledges the support provided by the U.S. Army under Contract DASG60-90-C-0153.

References

de Neufville, R. (1990). *Applied Systems Analysis: Engineering Planning and Technology Management.* McGraw-Hill Publishing Company.

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

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

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

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.

Guddat, J., Vasquez, F. G., Tammer, K., & Wendler, K. (1985). *Multiobjective and Stochastic Optimization Based on Parametric Optimization.* Mathematical Research, Band 26. Berlin, DDR: Akademie-Verlag.

Horn, J., (1993). Finite Markov chain analysis of genetic algorithms with niching. To appear in *Proceedings of the Fifth International Conference on Genetic Algorithms.* Morgan-Kauffman.

Keeney, R. L., Lathrop, J. F., & Sicherman, A., (1986). An analysis of Baltimore Gas and Electric Company's technology choice. *Operations Research.* Volume 34, Number 1, January-February 1986, 18-39.

Keeney, R., & Raiffa, H., (1976). *Decisions with Multiple Objectives.* Wiley, New York.

Mahfoud, S. M., (1992). Crowding and preselection revisited. Parallel Problem Solving From Nature, 2. North-Holland.

Mahfoud, S. M., (1993). Simple analytical models of genetic algorithms for multimodal function optimization. To appear in *Proceedings of the Fifth International Conference on Genetic Algorithms.* Morgan-Kaufmann, San Mateo, CA.

Morgan, D. R., (1990). Decision making under uncertainty using a new chance constrained programming technique: a groundwater reclamation application. *Unpublished doctoral dissertation, University of Illinois, Department of Civil Engineering.*

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.

Ritzel, B., (1992). Genetic algorithms in multiple objective optimization. *Unpublished analytical class project for GE 493, University of Illinois at Urbana-Champaign, Spring 1992.*

Sawaragi, Y., Nakayama, H., & Tanino, T., (1985). *Theory of Multiobjective Optimization.* Mathematics in Science and Engineering, Volume 176. Academic Press, Incorporated (Harcourt Brace Jovanovich, publishers).

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

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.