3 OPTIMIZATION IN PARETO SPACE

3.1 Introduction

Chapters 1 and 2 showed that LTM design often requires the exploration of vast decision spaces while balancing conflicting design objectives (e.g., reducing sampling costs while maintaining an acceptable picture of the groundwater contamination plume). The size and complexity of these decision spaces motivated the use of multiobjective GAs, primarily in this thesis because their population-based approach to multiobjective search and optimization has been shown to be more efficient than other more traditional optimization techniques (see *Deb* 2001). The purpose of this chapter is to demonstrate that multiobjective genetic algorithms are capable of navigating LTM decision spaces while accurately quantifying design tradeoffs. Moreover, this chapter demonstrates how theoretical work from the genetic and evolutionary field can be used to improve real-world applications of multiobjective GAs by following the recommendations of *Goldberg* (1998).

Goldberg (1998) compels GA users to seek "…GAs that solve hard problems quickly, reliably, and accurately—through a combination of effective (1) *design methodology*, (2) *design theory*, and (3) *design*." *Goldberg* (1998) also describes the *control map* concept for GAs where theoretical relationships for population sizing, selection pressure, crossover, and mutation are used to find values for the control parameters that identify a performance "sweet spot" where efficient search and optimization is enabled. The *design theory* for efficient GAs and the concept of the control map for GA performance are largely derived from *Goldberg et al*. (1992a) and *Thierens* (1995), respectively. Although these studies provided a theoretical basis for the design of GAs, practitioners still face the difficulty of using relationships that require the specification of parameters that are not readily identifiable in practice such as building block (BB) size and

order. BB's blocks are highly relevant subsets of the binary digits representing designs and are used by the GA to construct optimal solutions.

This difficulty was first addressed by the simple GA *design methodology* presented by *Reed et al*. (2000b) for single objective applications. This chapter extends *Reed et al.* (2000b) to multiobjective applications, presenting a *design methodology* for the Nondominated Sorted Genetic Algorithm (NSGA) that enabled the algorithm to accurately quantify 2 dimensional tradeoffs. This chapter has been excerpted from *Reed et al.* (2001a).

3.1.1 Previous Work

Evolution-based multiobjective optimization (EMO) methods are pragmatic tools for solving problems with large decision spaces and conflicting objectives. *Schaffer* (1984) provided the seminal work within the EMO field in which a vector evaluated genetic algorithm (VEGA) was designed to search decision spaces for the optimal tradeoffs among a vector of objectives. Subsequent innovations in EMO have resulted in a rapidly growing field with a variety of solution methods that have been used successfully in a wide range of applications (for reviews see *Fonseca & Fleming* 1995, *Coello* 1999, *Van Veldhuizen* 1999). These solution methods have garnered increased attention over the past decade and have been applied in a variety of contexts within the water resources field.

Cieniawski (1993) is one of the earliest studies in water resources to utilize EMO methods. The study is an empirical comparison of the performance of VEGA relative to niching-based techniques from *Goldberg & Richardson* (1987) for identifying a monitoring network to detect potential contaminant leaks from a hazardous waste landfill. *Cieniawski* (1993) and *Cieniawski et al.* (1995) clearly espouse the efficiency of EMO methods in quantifying tradeoffs between maximizing a groundwater-monitoring network's reliability in detecting contaminants

and minimizing the area of contaminated aquifer at the time of first detection. Ritzel *et al*. (1994) compared the performances of VEGA, a domination ranking-based genetic algorithm (Pareto GA), and mixed integer chance constrained programming (MICCP) in solving a multiobjective, groundwater pollution containment application. *Halhal et al.* (1997) successfully incorporated Pareto domination ranking into the messy genetic algorithm (*Goldberg et al.* 1989) to quantify the tradeoffs in rehabilitating water distribution networks. *Gupta et al.* (1998) combined a downhill simplex method with an evolutionary search strategy implementing Pareto ranking to seek tradeoff solutions when calibrating hydrologic models. These studies show how EMO methods have been adapted to solve a variety of water resource applications. This chapter focuses on the Nondominated Sorted Genetic Algorithm (NSGA) because *Zitzler et al.* (2000) showed that the NSGA performed as well or better than a representative sampling of EMO methods on a suite of test problems with properties similar to our application.

3.1.2 Motivation and Scope

One of the difficulties in applying EMO methods is identifying parameter settings that ensure efficient navigation of the decision space and adequate coverage of the Pareto frontier (*Van Veldhuizen & Lamont* 2000, *Cieniawski* 1993). Most practitioners use trial-and-error runs to identify the best parameter settings, but this approach is quite time consuming, particularly for applications with computationally intensive fitness functions. A major goal of this chapter is to develop guidelines for using theoretical relationships from the genetic and evolutionary computation literature to ensure that the NSGA efficiently navigates the problem's decision space. These guidelines are then applied to quantify the tradeoffs implicit in designing sampling strategies for a long-term groundwater-monitoring network. Additionally, a niching-based elitist

enhancement of the NSGA is also presented, which substantially improves coverage of the Pareto frontier.

3.2 Monitoring Application

The test case developed in this chapter uses data drawn from a 38 million-node flow-and-transport simulation performed by *Maxwell et al.* (2000). The simulation provided realistic historical data for the migration of a hypothetical perchloroethylene (PCE) plume in a highly heterogeneous alluvial aquifer.

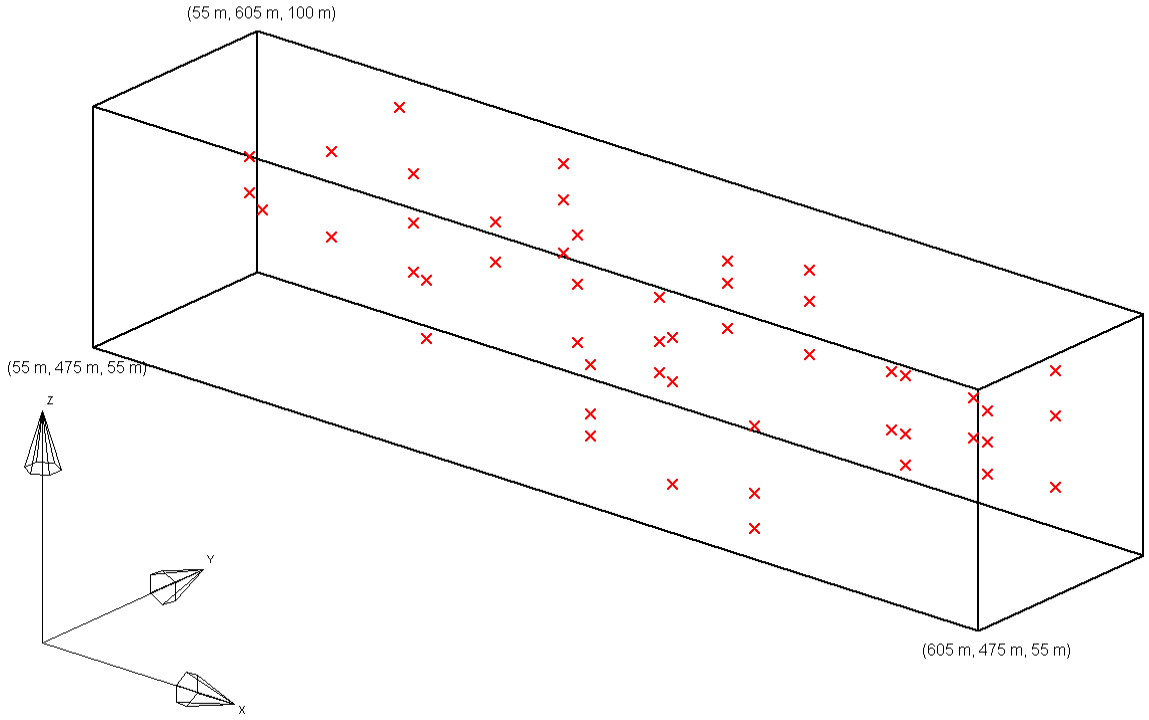


Figure 3.1 The 50 potential sampling locations (designated by the x's above) within a 20 well multi-level monitoring network

The hydrogeology of the test case is based on an actual site located at the Lawrence Livermore National Laboratory in Livermore, California, currently being managed under the United States' Comprehensive Environmental Response, Compensation and Liability Act (CERCLA) program.

Data were provided for a total of 50 hypothetical sampling locations within the 20-well multi-level monitoring network shown in Figure (3.1). The data represent a snapshot in time, 10 years after a continuous point source began contaminating the aquifer system. The monitoring wells can sample from 1 to 3 locations along their vertical axis and have a minimum spacing of 60-m between wells in the horizontal plane.

The site is assumed to be undergoing long-term monitoring, in which groundwater samples are used to assess the effectiveness of current remediation strategies. During this long-term monitoring phase of a remediation, sampling and laboratory analysis can be a controlling factor in the costs of remediating a site. Quarterly sampling of the entire network in Figure (3.1) has a potential cost of over $70,000 annually for PCE testing alone, which could translate into millions of dollars if the site had a typical life span of 20 to 30 years.

The significance of these costs has motivated the development of several approaches for reducing the fiscal burden posed by long term monitoring by identifying redundant wells in groundwater monitoring networks that can be omitted from future sampling periods (*Cameron & Hunter* 2000, *Aziz et al.* 2000, *Reed et al.* 2000a, *Rizzo et al.* 2000). These methods define sampling points to be redundant when they minimally affect interpolation-based plume estimates. They employ a variety of single objective optimization techniques ranging from a simple genetic algorithm to trial-and-error heuristics. The objective of these methods is to minimize sampling costs while incorporating performance objectives associated with plume estimates as constraints. The management model presented in this work builds on these previous methods by introducing a sampling design methodology that explicitly identifies the tradeoffs encountered when reducing monitoring costs.

3.2.1 Problem Formulation

To identify which wells are redundant, this chapter employs a local concentration approach with the intention of attaining the best-interpolated picture of the PCE plume for the least cost. Equation (3.1) gives the multiobjective problem formulation for quantifying the tradeoff between sampling costs and maintenance of a high quality interpolated picture of the plume.

$$Minimize\ F(\bar{x}_\kappa) = \left[f_1(\bar{x}_\kappa), f_2(\bar{x}_\kappa)\right], \quad \forall \kappa \in \Omega$$

$$f_1(\bar{x}_\kappa) = \sum_{i=1}^{nwell} C_S(i) x_{\kappa i}$$

$$f_2(\bar{x}_\kappa) = \sum_{j=1}^{n_{est}} \left(c^*_{all}(\bar{u}_j) - c^\kappa_{est}(\bar{u}_j)\right)^2 \tag{3.1}$$

$F(\bar{x}_\kappa)$ is a vector valued objective function whose components $[f_1(\bar{x}_\kappa), f_2(\bar{x}_\kappa)]$ represent the cost and squared relative estimation error (SREE), respectively, for the κ^{th} monitoring scheme $\bar{x}_\kappa$ taken from the collection of all possible sampling designs Ω. Equation (3.2) defines the binary decision variables representing the κ^{th} monitoring scheme.

$$x_{\kappa i} = \begin{cases} 1, \text{ if the } i^{th} \text{ well is sampled} \\ 0, \quad \text{otherwise} \end{cases}, \quad \forall \kappa, i \tag{3.2}$$

If the i^{th} well is sampled it is assumed that all available locations along the vertical axis of that well will be sampled at a cost of $C_S(i)$. $C_S(i)$ ranged from \$365 to \$1095 for 1 to 3 samples analyzed for PCE solely (*Rast* 1997). Sampling all available levels within each well reduces the size of Ω from 2^{50} to 2^{20} where 50 and 20 represent the total number of sampling locations and monitoring wells (*nwell*), respectively. Reducing the size of Ω enabled the entire

decision space of this application to be enumerated. Enumeration was employed to identify the true Pareto frontier so that the performance of the NSGA under different parameter settings could be rigorously tested.

The SREE provides a measure of how the interpolated picture of the plume using data only from wells included in the κ^{th} sampling plan compares to the result attained using data from all available sampling locations. The measure is computed by summing the squared deviations between the PCE estimates using data from all available sampling locations, $c^{*}_{all}(\overline{u}_j)$, and the estimates based on the κ^{th} sampling plan $c^{\kappa}_{est}(\overline{u}_j)$ at each location $\overline{u}_j$ in the interpolation domain. Each $\overline{u}_j$ specifies the coordinates for the j^{th} grid point in the interpolation domain. The interpolation domain consisted of a total of 3300 grid points (n_{est} in equation (3.1)). The PCE estimates used in the calculation of the SREE for each of the sampling designs were attained using the nonlinear spatial interpolation method described below.

3.2.2 Nonlinear Spatial Interpolation

The interpolation method used in this chapter is a variant of the scheme used by *Barry and Sposito* (1990) in their analysis of tracer plumes at the Borden site located in Ontario, Canada. The *Barry and Sposito* (1990) interpolation method was selected because it requires minimal modeling assumptions and it has been successfully applied to three-dimensional historical data. *Barry and Sposito* (1990) interpolated the Borden tracer data using 7 fitting parameters with nonlinear least squares. This chapter simplified the interpolation scheme to use 3 fitting parameters in a nonlinear least squares version of inverse distance weighting. Neglecting 4 of the 7 fitting parameters did not appreciably affect the cross-validation residuals and improved the algorithm's stability. Additionally, using fewer fitting parameters reduced the size of the

Jacobian matrix and the number of iterations required to evaluate each sampling design, enabling Ω to be enumerated.

Equation (3.3) shows the interpolation function used in this chapter, which estimates of PCE concentration $c_{est}^{\kappa}(\overline{u}_j)$ at each unknown location $\overline{u}_j$ by the weighted sum of the $nsamp(\kappa)$ total samples $c(\overline{u}_\psi)$ taken in the κ^{th} sampling scheme.

$$c_{est}^{\kappa}(\overline{u}_j) = \sum_{\psi=1}^{nsamp(\kappa)} w(\overline{u}_j, \overline{u}_\psi)\, c(\overline{u}_\psi) \big/ w_t \tag{3.3}$$

The weights for each of the samples in the κ^{th} sampling plan are calculated as a function of the distance between the ψ^{th} sample and the j^{th} grid point in the interpolation domain as shown in equation (3.4).

$$w(\overline{u}_j, \overline{u}_\psi) = 1/D_\kappa(\overline{u}_j, \overline{u}_\psi)$$

$$D_\kappa(\overline{u}_j, \overline{u}_\psi) = \left[\left(u_{j1} - u_{\psi 1}\right)^2/\alpha_1 + \left(u_{j2} - u_{\psi 2}\right)^2/\alpha_2 + \left(u_{j3} - u_{\psi 3}\right)^2/\alpha_3\right]^{0.5} \tag{3.4}$$

The w_t factor is the sum of the $nsamp(\kappa)$ weights calculated using equation (3.4) and serves to scale the system such that the weights sum to one. The parameters α_1, α_2, and α_3 are fitting parameters that were used to minimize the cross-validation residuals for each of the sampling designs considered in this chapter.

Equation (3.5) gives the cross-validation estimation method used in this chapter to fit the α-fitting parameters shown in equation (3.4) to the data provided by each sampling plan (*Barry and Sposito* 1990).

$$Minimize\ S(\kappa) = \sum_{\psi=1}^{nsamp(\kappa)} \left\{ c(\overline{u}_\psi) - \sum_{h=1, h\neq\psi}^{nsamp(\kappa)} w(\overline{u}_\psi, \overline{u}_h)\, c(\overline{u}_h) \big/ \left(w_t - w(\overline{u}_\psi, \overline{u}_h)\right) \right\}^2 \tag{3.5}$$

The cross-validation method minimizes the sum of the squared residuals $S(\kappa)$ between each of the actual PCE concentrations $c(\overline{u}_{\psi})$ and their interpolated estimates based on the ($nsamp(\kappa)$ – 1) remaining data $c(\overline{u}_{h})$ in the κ^{th} sampling plan. Equation (3.5) was solved for every sampling plan considered in this chapter using the Levenberg-Marquardt nonlinear least squares solution method (Moré 1977). The Levenberg-Marquardt solution technique implicitly requires that the number of data points in the system must exceed the number of fitting parameters. This requirement necessitated that the monitoring designs considered in this chapter have a minimum of four PCE samples.

3.3 The Basics of the NSGA

The NSGA utilizes the Darwinian process of natural selection to effectively search for solutions that are optimal across a vector of objectives. The algorithm is very similar in form to the simple genetic algorithm (sGA), in that it exploits the operators of selection, crossover, and mutation when building a set of optimal solutions. The performance of both algorithms can be described using the building block (BB) theory presented by *Holland* (1975) and *Goldberg* (1989). For both the NSGA and sGA, highly fit designs have a higher probability of being selected to mate and pass their traits (or BBs) to succeeding generations. Stochastic remainder selection was used in this chapter as recommended in *Srinivas & Deb* (1995). The operators of crossover and mutation are identical for the two algorithms. Crossover (occurring with probability p_c) exerts an innovative force on the system by allowing favorable traits from parent designs to be juxtaposed in offspring that possess higher relative fitness values (*Thierens & Goldberg* 1993). Mutation locally refines solutions by randomly changing bit values (with a probability of p_m) from 0 to 1 or vice versa within a design's genotype. The difference between the NSGA and the sGA lies in how fitness is assigned. Unlike the sGA, the NSGA evaluates

sampling designs in terms of a vector of objectives. A sampling design cannot be assessed in terms of its performance in any single objective because it may perform poorly with respect to the remaining objectives. The NSGA employs the concepts of *Pareto dominance* and *niching* to assign fitness values to sampling designs in two steps described below (*Srinivas & Deb* 1995).

3.3.1 Pareto Dominance

The first step in fitness assignment employs the *Pareto dominance* concept defined in equation (3.6), using notation adapted from *Van Veldhuizen & Lamont* (2000).

$$F(\bar{x}) \prec F(\bar{x}'), \textit{iff } \left(\forall \beta, f_\beta(\bar{x}) \le f_\beta(\bar{x}') \ \wedge \exists\, \beta \in (1, n_{obj}) : f_\beta(\bar{x}) < f_\beta(\bar{x}')\right) \tag{3.6}$$

Equation (3.6) states that a design $\bar{x}$ dominates another design $\bar{x}'$ (represented by $F(\bar{x}) \prec F(\bar{x}')$) if and only if it performs as well as $\bar{x}'$ in all n_{obj} objectives and better in at least one (assuming minimization of all objectives). The NSGA identifies nondominated individual designs within the current population and assigns an arbitrary dummy fitness value to each of them. These individuals are all initially assigned the same fitness value to ensure that they have an equal likelihood of being selected to pass their traits unto latter generations; these designs compose the first nondominated front.

3.3.2 Niching & Sharing

The second step in fitness assignments for the individuals in the first front utilizes the concept of *niching* (for more details see *Goldberg & Richardson* 1987, *Deb & Goldberg* 1989, *Mahfoud* 1995, *Horn* 1997) to ensure that the NSGA finds a diverse set of solutions defining the entire extent of the tradeoff (or Pareto front) among the objectives. Extending the "natural selection" analogy to include the phenomena of *niching* and *speciation* helps to satisfy this goal. *Horn* (1997) defines niching as a "form of cooperation" where there is "… the localization of competition around finite, limited resources (niches), resulting in the lack of competition

between such areas, and causing the formation of species for each niche." Niching allows the NSGA to form stable subpopulations of sampling designs (species) each of which are well adapted to search for nondominated solutions specific to subspaces (niches) in Ω. The NSGA elicits niche formation by treating dummy fitness as a limited resource, which is shared by sampling designs using the relationship given in equation (3.7) (*Goldberg & Richardson* 1987, *Srinivas & Deb* 1995).

$$Sh\left[d(\kappa,\kappa')\right]=\begin{cases}1-\left(\dfrac{d(\kappa,\kappa')}{\sigma_{share}}\right)^2, & if\ d(\kappa,\kappa')<\sigma_{share}\\ 0, & otherwise\end{cases} \tag{3.7}$$

The value of the fitness sharing function $Sh\left[d(\kappa,\kappa')\right]$ ranges between 0 and 1 for two sampling designs κ and κ' depending on the ratio of their distance from one another $d(\kappa,\kappa')$ and the niche size defined by the niche radius σ_{share}. The sharing function measures the similarity between designs ranging from completely dissimilar (*Sh* = 0) to being identical (*Sh* = 1). The distance term $d(\kappa,\kappa')$ can measure similarity in several ways that are discussed in the next section.

The third step in assigning the fitness values to the nondominated set identified in step 1 consists of dividing each individual's dummy fitness by the sum of all of its sharing function values (termed the niche count) as shown in equation (3.8).

$$niche\ count(\kappa)=\sum_{\kappa'=1}^{N} Sh[d(\kappa,\kappa')],\ \forall\kappa \tag{3.8}$$

When these three steps are completed, the individuals in the first front are removed from consideration and the remaining population members undergo nondomination ranking and sharing to yield the second nondominated front. These individuals are assigned the minimum

shared fitness value of the first front decremented by a value ΔD_f. This process continues until the entire population is assigned to the n_f th front and given a shared dummy fitness value. After shared dummy fitness values have been assigned, the NSGA is operated the same as the sGA, constructing the Pareto front using the traditional operators of selection, crossover, and mutation. See *Goldberg* (1989) for more background on these operators.

3.4 Multiobjective Search & Optimization

3.4.1 Performance Considerations

Construction of the Pareto frontier requires that the NSGA be effectively designed to navigate a problem's decision space. Effective design of the NSGA entails careful consideration of the factors controlling the algorithm's performance. *Zitzler et al*. (2000) concluded that population sizing and elitism are the most influential factors influencing the performance of EMO methods. Elitist operators provide a means of ensuring that the best individuals are identified and allowed to pass their traits to latter generations. Unlike sGA applications, EMO methods cannot simply pass a single individual with the current best objective function value into the next generation. Multiobjective optimization requires that some fraction of the solutions along the current nondominated front be passed on to the next generation. The next four sections of this chapter describe an NSGA design methodology that integrates several relationships from the genetic and evolutionary literature to identify effective population sizes and determine the niching parameters.

This methodology is meant to serve as a guide for applying the NSGA to other applications. Additionally, an elitist strategy is presented with the aim of using niching to guide the selection of elite population members and maximize *online* performance (the term online performance means that only the nondominated individuals within a single generation are

considered when assessing the NSGA's performance). Performance is finally discussed in the context of selection pressure in the fourth and final section of the NSGA design methodology.

3.4.1.1 Population Sizing: Ensuring Genetic Diversity

The first step in designing an efficient NSGA is to perform initial problem analysis to determine a range of population sizes. The goal of this initial step is to provide a means of selecting the best population size for our application with respect to a single random seed. Computationally intensive objective functions that occur in many water resources applications require that effective parameters settings for the NSGA be identified using a minimum number of runs, because a single run can take days or even weeks. The range of population sizes considered in this chapter were attained using relationships from *Mahfoud* (1995), who derived population-sizing relationships for genetic algorithms employing niching to solve multimodal problems. The relationships used in this chapter were designed for applications with multiple optima that have *identical fitness values*. Recall, the NSGA employs niching using members of the same nondominated set with *identical dummy fitness values*, which makes the population-sizing relationships in *Mahfoud* (1995) relevant to NSGA applications.

Equation (3.9) gives population size estimates assuming that crossover will not disrupt the traits (or BBs) required to assemble optimal solutions.

$$N \geq q\left[-\ln\left(\frac{1-\gamma^{\frac{1}{g}}}{q}\right)\right] \tag{3.9}$$

The relationship was derived as the minimum population size N required to maintain q niches for g generations with reliability γ for the special case when the niches have identical fitness values. Disruption adversely affects the NSGA's performance when crossover between a pair of parent strings destroys traits that are essential to the evolution of nondominated individuals in

subsequent generations (for more details see *Mahfoud* 1995, *Zitzler et al.* 2000, *Van Veldhuizen & Lamont* 2000). Equation (3.10) accounts for the potential disruptive effects of crossover in population size estimates (*Mahfoud* 1995).

$$N \geq 2\ln\left(\frac{1-\gamma^{\frac{1}{g}}}{q}\right) \Bigg/ \ln\left[(1-p_c)\left(1-\frac{1}{q}\right)^2 + p_c\left(1-\frac{1}{q^2}\right) - 2p_c(1-p_d)\frac{1}{q}\left(1-\frac{1}{q}\right)\right] \tag{3.10}$$

Equation (3.10) represents the minimum population size N required to maintain q niches for g generations with reliability γ given a probability of crossover p_c and probability of disruption p_d. The probability of disruption p_d can be conservatively estimated using equation (3.11), which assumes the maximally disruptive operator of uniform crossover. In uniform crossover, the binary values at each bit position in two individual strings is swapped with probability p_c.

$$p_d = 1-\left(\frac{1}{2}\right)^{O(m)} \tag{3.11}$$

For the binary-coded NSGA, the decision variables ($\bar{x}_\kappa$ in equation (3.1)) representing a potential design (termed its *phenotype*) are converted to binary representations and concatenated into a string of binary variables (termed the design's *genotype*) of length l. The traits or (BBs) are actually small subsets of binary digits from each design's genotype. Equation (3.11) computes the probability that the BBs will be disrupted as a function of their order $O(m)$, which is the smallest number of fixed value digits in a design's genotype that are relevant to the final solutions of the problem.

As was the case in the sGA design methodology presented by *Reed et al*. (2000b), the actual size of BBs in an engineering application, $O(m)$ are unknown, requiring the practitioner to employ conservative assumptions and problem-specific information to calculate a potential range of population sizes from equations (3.9) and (3.10). The p_d was set by assuming that the BB

order $O(m)$ may range from 1 to 5 because higher order BBs will be disrupted by the operators of selection, crossover, and mutation (*Goldberg* 1989), following a similar approach presented by *Reed et al.* (2000b) for designing sGAs.

The remaining parameters in equations (3.9) and (3.10) were set as follows. The population size calculations assumed a reliability of 85 percent ($\gamma = 0.85$). The number of generations until convergence g can be estimated to fall within the range $[2l, l\ln l]$ where l is the binary string length of each design (*Thierens et al.* 1998, *Thierens & Goldberg* 1994). *Mahfoud* (1995) states that equations (3.9) and (3.10) are not sensitive to the parameter value g and recommends that the user set the value of this parameter to be greater than or equal to the number of generations they expect to run the algorithm. For these reasons, g was set equal to $2l$ (40 generations) for this chapter. A majority of EMO applications use a value for p_c falling within the range [0.6, 0.9] (*Fonseca & Fleming* 1995, *Coello* 1999, *Van Veldhuizen* 1999, *Horn* 1997). In this application, p_c was set equal to 0.6 to reduce the potential for disruption and the resulting population size estimates.

The number of niches q was set using information specific to the tradeoff being sought between cost and SREE in this application. The discussion of the spatial interpolation stated that the number of PCE samples $c(\overline{u}_{\psi})$ ranged between a minimum value of 4 and a maximum of 50, yielding a tradeoff with a maximum of 46 discreet cost levels. The goal of this application is to find the minimum SREE for each of these discreet costs levels. The number of niches was set equal to 40, which represents a goal of attaining over 85 percent of the points in the Pareto frontier.

Using these parameters in equations (3.9) – (3.11) yielded six population size estimates ranging from a minimum value of 370 to a maximum of 870. The lower bound population size

estimate of 370 assumes that crossover will not disrupt BBs. The remaining five population size estimates account for disruption of BBs with orders, $O(m)$, ranging from 1 to 5. Given this range of population sizes, the computational complexity of using the NSGA to solve this problem was estimated by multiplying N by the number of generations g to attain the total number of function evaluations. Function evaluations took an average time of 0.04 s on a Dell XPS T800r running Windows NT, yielding estimated total run times between 10 and 25 minutes.

Finally, it should be noted that both equations (3.9) and (3.10) assume that mutation is minimally disruptive. This assumption was accounted for by setting p_m to be equal to $1/N$ (*DeJong* 1975, *Schaffer et al.* 1989), which limits the number of mutations to l and reduces the operator's influence as N increases. In this application, the optimal population was determined by running the NSGA for each of the six estimates and choosing the run that best defined the full extent of the tradeoff between cost and SREE.

3.4.1.2 Sharing: Sizing of the Niche

An important component of ensuring that the NSGA converges to the Pareto frontier is properly setting the size of niches represented by σ_{share} in equation (3.7). *Deb & Goldberg* (1989) provide guidance for setting this parameter in both phenotypic and genotypic space. Recall that a sampling design's phenotype is the floating-point representation of each of its decision variables. Additionally, the design's genotype is its concatenated binary representation (or its chromosome). Equation (3.7) requires that the distance $d(\kappa, \kappa')$ between 2 individuals be compared to σ_{share} in order to compute the sharing function values used to assign fitness. If the individuals are within σ_{share} distance of each other, their fitness is penalized to encourage future generations to spread along the entire Pareto frontier.

Two different distance metrics are used to compare the relative magnitudes of $d(\kappa, \kappa')$ and σ_{share}. In genotypic space, $d(\kappa, \kappa')$ is equal to the Hamming distance between the chromosomes representing individuals κ and κ'. The Hamming distance is simply the total number of positions in the binary strings that have different values. In phenotypic space, the Euclidean distance metric shown in equation (3.12) is used to calculate the distance between individuals.

$$d(\kappa, \kappa') = \sqrt{\sum_{i=1}^{npar} \left(x_{i,\kappa} - x_{i,\kappa'}\right)^2}, \forall \kappa \tag{3.12}$$

For this application, the total number of decision variables (*npar*) is equal to the number of monitoring wells, *nwell* in equation (3.1).

To determine the appropriate size of σ_{share} for a particular problem, three approaches can be taken. These approaches are described below in order of increasing domain-specific knowledge required. *Deb & Goldberg* (1989) derived equation (3.13) as a guide for practitioners for sizing niches in genotypic space by deriving σ_{share} to represent "…the maximum bits of difference allowed between the strings to make *q*-subspaces [or niches] in the solution space".

$$\frac{1}{2^l} \sum_{b=0}^{\sigma_{share}} \binom{l}{b} = \frac{1}{q} \tag{3.13}$$

Equation (3.13) assumes that the niches are uniformly spaced with each niche apportioned 1/*q* of the decision space (*Deb & Goldberg* 1989). The equation represents a binomial distribution with a probability of 0.5, which can be solved for σ_{share} using the cumulative binomial distribution tables when string lengths *l* are less than or equal to 25 bits (indexed by *b* above). In this chapter, the chromosomes have a length of *l* = 20 and the number of niches *q* = 40, which resulted in a genotypic σ_{share} equal to 5. For longer string lengths, the binomial distribution can

be approximated with the normal distribution. In this case, equation (3.14) from *Deb & Goldberg* (1989) can be used, where the "… z^* corresponding to the fraction $1/q$ may be found from a cumulative normal distribution chart".

$$\sigma_{share} = \frac{1}{2}\left(l + z^* \sqrt{l}\right) \tag{3.14}$$

The second approach for selecting σ_{share} is for phenotypic space. Deb & Goldberg (1989) derive σ_{share} in phenotypic space where "… each niche is enclosed in a p-dimensional hypersphere of radius σ_{share} such that each sphere encloses $1/q$ of the volume of the space". Equation (3.15) is the resulting expression for the phenotypic niche radius, which was found to have a value of 1.859 in this chapter.

$$\sigma_{share} = \sqrt{\sum_{i=1}^{npar}\left(x_{i,\max} - x_{i,\min}\right)^2} \Bigg/ 2\,q^{\frac{1}{npar}} \tag{3.15}$$

Phenotypic sharing has been shown to outperform genotypic sharing in most previous applications (*Fonseca & Fleming* 1995, *Coello* 1999, *Van Veldhuizen* 1999, *Horn* 1997, *Mahfoud* 1995).

Finally, an additional form of phenotypic sharing has been employed successfully in other applications (*Fonseca & Fleming* 1995, *Horn* 1997) where the distance metric between individuals is calculated in objective space. *Horn* (1997) provides guidance to practitioners for sizing niches in objective space using a geometric approach similar to the *Deb & Goldberg* (1989) phenotypic niche sizing method given in equation (3.15). The method requires that the maximum and minimum values of the objectives be known a priori. This approach was not used in this chapter because of the discrete nature of the sampling costs. Moreover, the maximum value of the SREE could not be calculated a priori.

The niche sizing relationships given by *Deb & Goldberg* (1989) and described above have been shown to be robust for a variety of problems (*Goldberg et al.* 1992b, *Horn* 1997) and will be analyzed in greater detail in the results section of this chapter.

3.4.1.3 Elitism: Seeking the King of the Niche

Zitzler et al. (2000) showed that elitism is one of the most important factors affecting the performance of EMO methods. A variety of elitist strategies have been used previously, usually consisting of maintenance of a population of nondominated solutions outside of the normal operators of the given EMO method being employed (for more details see *Ishibuchi & Murata* 1996, *Bäck* 1996, *Parks & Miller* 1998, *Zitzler & Thiele* 1999). *Zitzler et al*. (2000) state the primary question practitioners must answer when using elitist strategies as: "When and how are which members of the elite set re-inserted into the population?". The elitist strategy employed in this chapter was designed to use previously derived niching relationships to answer this question and maximize the online performance of the NSGA. In an elitist sGA, the best member in the population at generation t, if not present in the new population resulting from selection, crossover, and mutation at generation (t+1), randomly replaces one member of the population. For the NSGA, sharing provides niches that represent stable subpopulations that search for nondominated solutions in subspaces of Ω. Conceptually, the elitist strategy proposed in this chapter is very similar to the sGA, in that the current best individual in a given niche at generation t, if not present in generation (t+1), is inserted into that subpopulation, ensuring that its traits are available for subsequent search for the Pareto front.

This strategy was implemented by defining σ_{elite} or the elite radius, which is a parameter that allows the user to easily manipulate the amount of elitism. The elite radius defines the distance (either genotypic or phenotypic) beyond which members of the current nondominated

set are considered independent from one another. Only independent members of the nondominated set are considered for insertion in the next generation. For this application, $\sigma_{elite} \approx \sigma_{share}$ which means that only one representative of each niche in the current nondominated set is considered for elitist reproduction into the next generation. The elitist solutions were selected in the four steps shown below from the nondominated set (or first front) at each generation *t*.

Step 1: Randomly select an objective f_β for β equal 1 to n_{obj}

Step 2: Flip a coin to determine whether to start with either the member in the current nondominated set with the maximum value of f_β or the member with the minimum value.

Step 3: Identify the next point in the nondominated set that satisfies the following conditions:

(1) Is a distance greater than σ_{elite} from the current solution

(2) Is the closest member of the nondominated set to the current position

If none exist, then elitist reproduction is ceased or not performed at all.

Step 4: Repeat *Step 3* until elitist reproduction is ceased.

This approach identifies a niched elitist set by systematically stepping through the current nondominated front from one end to the other. After the elitist set of solutions is selected using the above steps, those members who are not represented in generation (*t*+1) randomly replace individuals within that population. Setting the elite radius equal to the niche radius worked well for this application, but the elite radius parameter allows the practitioner to directly manipulate the elitist selection pressure for other applications where this rule-of-thumb may not work as effectively.

The importance of elitism in the performance of the NSGA *Zitzler et al.* (2000) relates directly to the concepts of *genetic drift* and *selection pressure*. Genetic drift occurs when promising solutions in the population do not experience sufficient selection pressure and converge to non-optimal values (drift stall) under the influence of crossover and mutation. Elitism increases selection pressure by ensuring that the traits of nondominated individuals remain in the population for use in later generations.

3.4.1.4 Selection Pressure: Avoiding Drift Stall

The importance of elitism in the performance of the NSGA motivated a further analysis of the role that selection pressure has on the algorithm's performance. Stochastic remainder selection selects a particular individual using the ratio of the individual's fitness and the average fitness of the current population in generation *t*. Recall from the introduction to the NSGA above that selection is based on dummy fitness values, which are decremented by a constant value ΔD_f for each of the n_f successive fronts within the population. For example, analysis of the initial random population of 760 individuals used in the elitist runs discussed in the results section, shows that the expected number of copies of members in the nondominated set in the next generation had an average value of 1.7. The expected number of members from the 10^{th} and 20^{th} fronts in the next generation was 1.45 and 1.25, respectively. Although members of dominated fronts are expected to receive fewer copies than the nondominated front, the relative difference between 1.7 and 1.25 is small, which gives rise to an increased potential for drift stall to occur.

This increased potential for drift stall motivated an analysis of the effect of scaling the fitness of successive fronts on the NSGA's performance. The dummy fitness values for successive fronts are scaled using a scaling coefficient S_c whose value is less than one, such that

the minimum fitness in each front is guaranteed to be at least (1-S_c) percent higher than the maximum fitness in the front that immediately succeeds it. The scaling-based fitness assignments replace the constant decrement ΔD_f assignments used previously in this work. S_c was set to be equal to 0.9, which ensures that the minimum fitness for front ($n_f - 1$) is at least 10 percent greater than the maximum fitness in the $n_f{}^{th}$ front. For the random population of 760 designs discussed above, the expected numbers of individuals in the next generation from the 1st, 10th, and 20th fronts will then be 6.2, 2.27, and 0.78 respectively. Note that scaling the system in this manner exponentially decreases the fitness of the members of the fronts succeeding the nondominated set. Setting S_c requires striking a balance between maintaining a diverse population and ensuring that selection pressure is sufficient to prevent drift stall. Further discussion on this issue is given in the results section below.

3.4.2 Defining a Measure of Relative Performance

To compare performance of the NSGA under different parameter settings, a measure of the algorithm's performance must be defined. In this work, the performance of the NSGA as a function of its parameters was measured relative to the Pareto frontier for cost and SREE. The frontier is shown in Figure (3.2), which shows the 36 sampling designs that compose the Pareto optimal set identified using enumeration of the more than 1 million potential designs in Ω. The performance of the NSGA was quantified using a relative scoring metric (RSM) that measures the deviation of the nondominated set in generation *t* from the true front using equation (3.16).

$$Deviation(\eta) = \begin{cases} \left| SREE_{true}(\bar{x}_\eta) - SREE(\phi(t)) \right| \Big/ SREE_{\max}, & if\ f_1(\bar{x}_\eta) = f_1(\phi(t))\ \forall \eta, \phi(t) \\ 1, & otherwise \end{cases}$$

$$RSM = 1 - \left(\sum_{\eta=1}^{36} Deviation(\eta) \Big/ 36 \right) \tag{3.16}$$

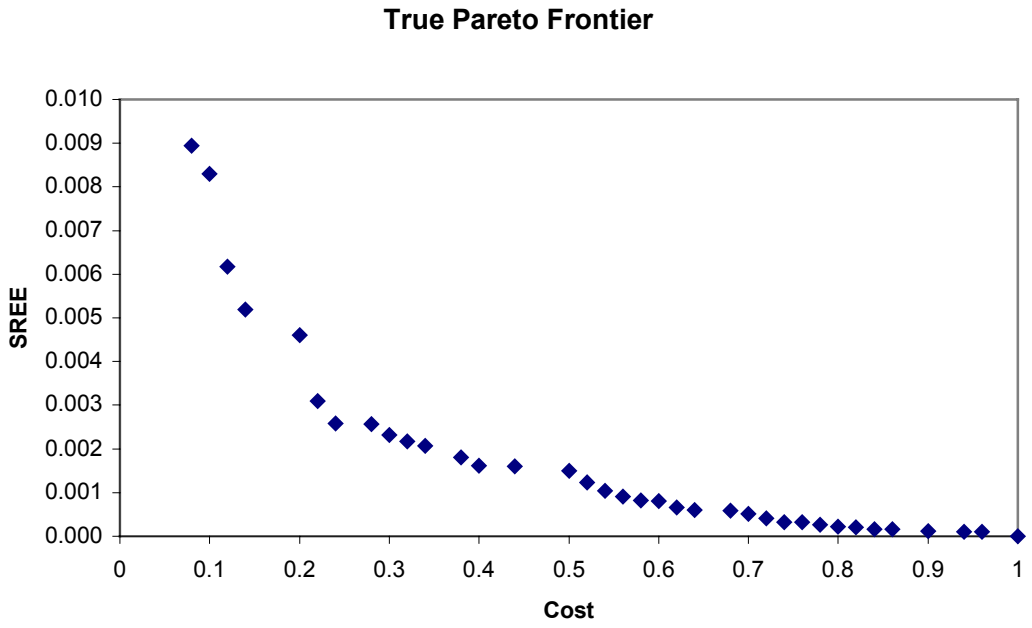


Figure 3.2 Actual tradeoff between the squared relative estimation error and cost. Each objective has been scaled to fall within the interval [0, 1]. The maximum and minimum SREE values found in Ω were 0 and 13,000,000, respectively. The maximum and minimum cost values found in Ω were $18,420 and $1473, respectively.

Although the maximum SREE value ($SREE_{\max}$) cannot be calculated a priori, its maximum value was recorded while performing the enumeration of the decision space. Equation (3.16) requires the maximum value because the SREE values have to be scaled such that they fall within the interval [0,1]. Equation (3.16) defines the deviation between the η^{th} member of the enumerated Pareto optimal set and the ϕ^{th} member of the current nondominated set in generation t to be equal to the absolute difference of their SREE values ($SREE_{true}(\bar{x}_\eta) - SREE(\phi(t))$ shown above) if the designs have the same cost. If a cost level present in the Pareto optimal set is not represented in the nondominated set at generation t then equation (3.16) assumes a maximum deviation of one. The RSM was used to monitor the performance of the NSGA in order to evaluate the effectiveness of the guidelines presented in this section for a realistic application.

3.5 Results & Discussion

The guidelines presented in the previous section identify a range of potential population sizes and compute the niching parameters used by the NSGA to search a diverse set of stable subpopulations for the Pareto frontier. Additionally, the niching parameters are integral to properly setting the elitist selection pressure. The subsequent sections analyze the effectiveness of these guidelines by presenting the results of over 400 trial runs elucidating the performances of both the NSGA and the Elitist NSGA as a function of their parameters for the long-term monitoring design application.

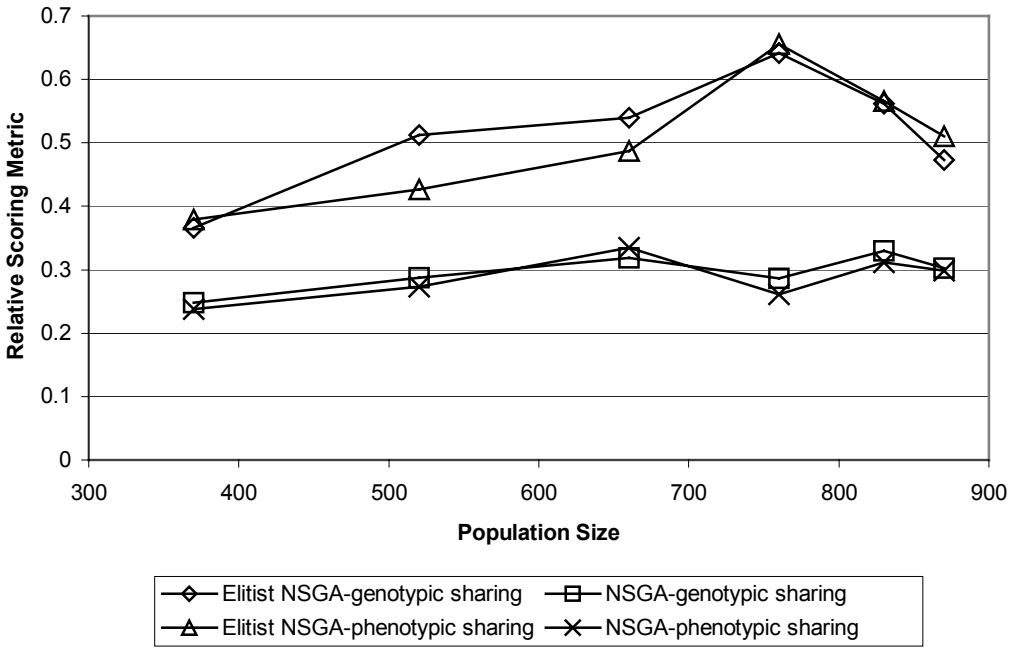


Figure 3.3 Selection of the proper population size for both the NSGA and the Elitist NSGA using the best RSM value attained from a single generation

3.5.1 Performance of the NSGA & the Elitist NSGA

Figure (3.3) shows the performances of both the NSGA and the Elitist NSGA for the range of population sizes attained from equations (3.9) – (3.11) when implementing either

genotypic or phenotypic sharing. It is readily apparent from the figure that elitism greatly improves the online performance of the NSGA regardless of the population size used. Figure (3.3) was used to select the optimal population size for each of the four forms of the NSGA being considered. The performance trends for the NSGA and the Elitist NSGA are nearly identical for both sharing methods. The NSGA's population sizes were set equal to 830 and 660 for genotypic and phenotypic sharing, respectively. These population sizes are consistent with previous studies, which have found that phenotypic sharing outperforms genotypic sharing. *Deb & Goldberg* (1989) argue that the reduced performance of genotypic sharing is caused by increased sensitivity of Hamming distance calculations to the assumption that each niche is uniformly apportioned $1/q$ of the decision space, as required by equation (3.13). *Mahfoud* (1995) states that in addition to the sensitivity of genotypic sharing to the uniformity assumption, genetic drift and population sizing are influential in the performance differences between the two sharing schemes. Larger population sizes are required to ensure that important subpopulations (or niches) receive sufficient selection pressure and are not lost to the "noisy discrimination of genotypic sharing" (*Mahfoud* 1995).

Figure (3.3) and Figure (3.4) confirm that for the NSGA to have comparable performance under both sharing schemes the algorithm requires larger population sizes for genotypic sharing. Figure (3.4) shows that the NSGA has relatively poor online performance for discerning the true tradeoffs between costs and SREE for the monitoring application presented in this chapter, regardless of the sharing scheme invoked. It is important to note that genotypic sharing requires the least amount of problem-specific information of the sharing methods discussed in this chapter, but the inherent tradeoff of the NSGA using this form of sharing lies in the increased computational demands required due to the increased population size necessary for sharing in

genotypic space. This tradeoff is not present for the Elitist NSGA where Figure (3.3) clearly shows that the optimal population size is clearly defined by the peak when N is equal to 760 regardless of the sharing method used.

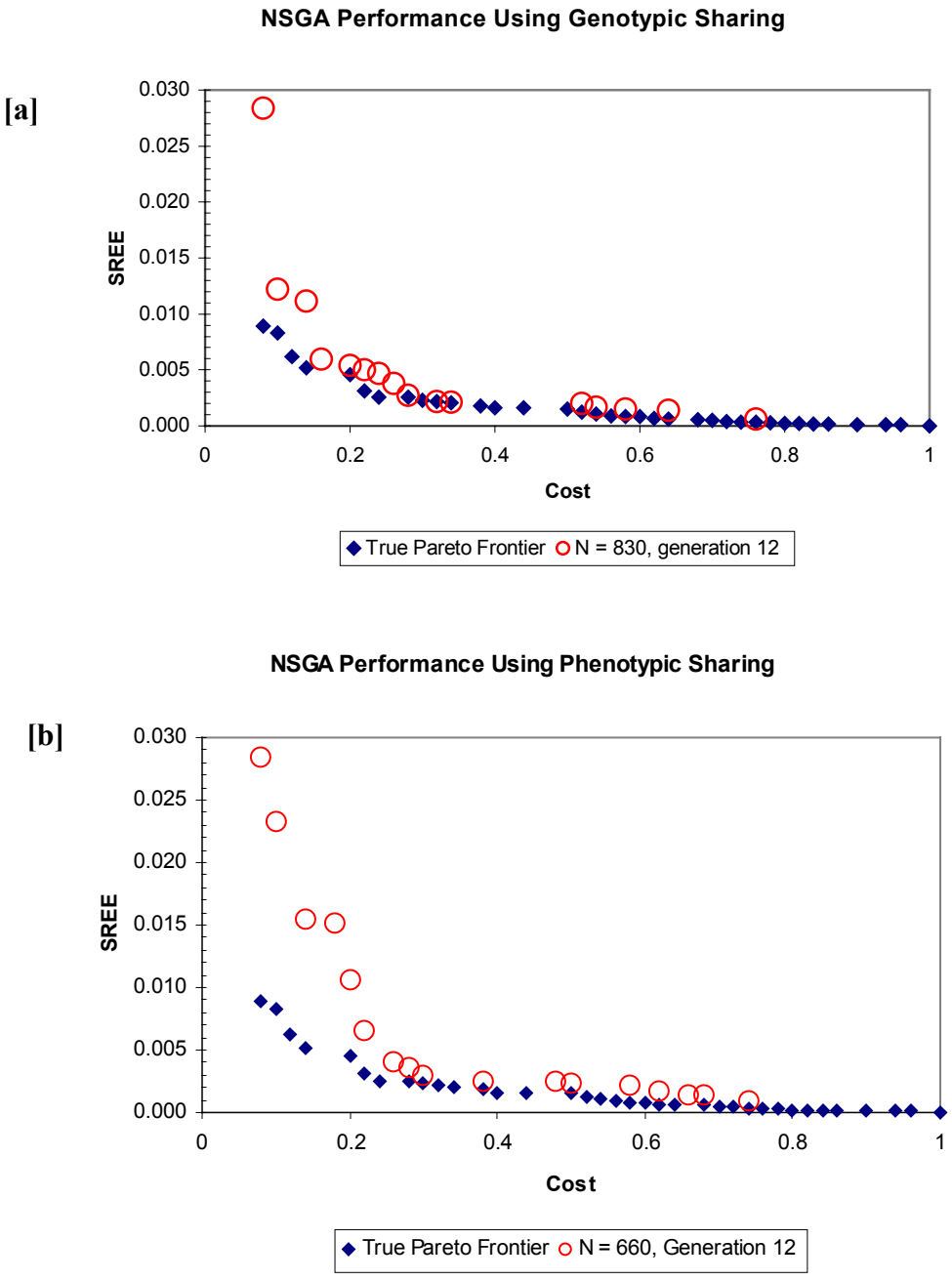


Figure 3.4 NSGA's performance relative to the true Pareto front using both (a) genotypic and (b) phenotypic sharing

Figure (3.5) shows that the niching-based elitist strategy presented in this chapter greatly improved the performance of the NSGA. The algorithm found solutions along the full extent of the Pareto front and was able to identify 17 of 36 members of the Pareto optimal set exactly, regardless of the sharing method used.

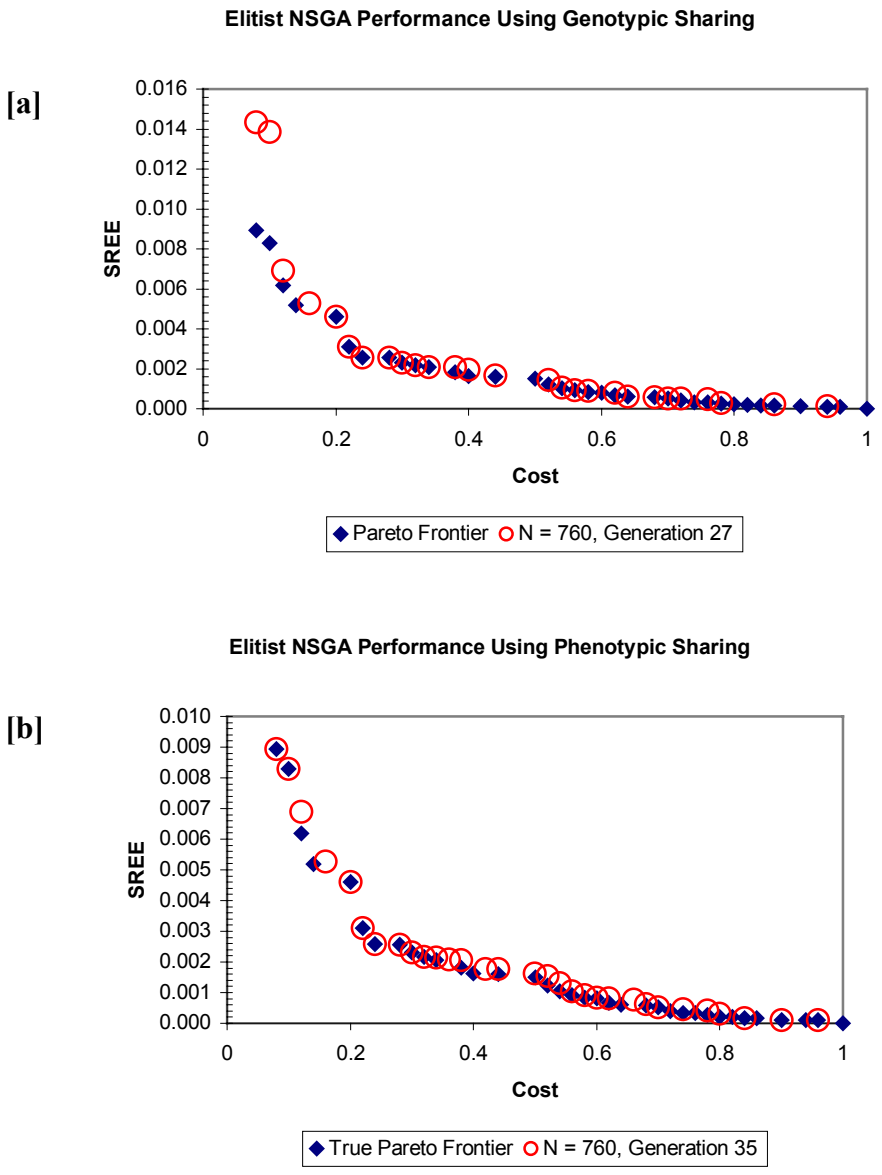


Figure 3.5 Elitist NSGA's performance relative to the true Pareto front using both (a) genotypic and (b) phenotypic sharing

Figure (3.3) and Figure (3.5) show that elitism served to both improve the online performance of the algorithm and help to overcome the "noisy discrimination of genotypic sharing" relative to sharing in phenotypic space.

Figure (3.6) shows the performance of the Elitist NSGA for the full range of possible values that both σ_{elite} and σ_{share} can be assigned under both sharing schemes (when N equals 760). Several observations are apparent from analysis of these plots. First, note that decreasing σ_{elite} from its maximum to its minimum value shows the NSGA's behavior between the extremes of having no elitism (at the maximum value) to the case when all nondominated individuals are selected for elitist reproduction (at the minimum value). Both plots show that the NSGA's performance improves with increasing elitist selection pressure. Figure (3.6) also confirms the noisy nature of sharing in genotypic versus phenotypic space. Under genotypic sharing, Figure (3.6a) shows several small peaks in performance for relatively sporadic combinations of niching and elitist parameter settings. In contrast, Figure (3.6b) shows that sharing in phenotypic space yields a smoother surface with a well-defined peak relative to the surface in (3.6a). Both plots serve to verify the guiding relationships discussed in the methods sections for population sizing, niching, and elitist selection. The arrows in the figure designate the algorithm's performance when both σ_{elite} and σ_{share} are set using equations (3.12) and (3.14) taken from *Deb & Goldberg* (1989).

Figure (3.4) and Figure (3.5) show that both the NSGA and the Elitist NSGA solutions have gaps in the extreme regions of the Pareto frontier. These gaps occur because the subpopulations or niches representing these portions of the frontier were not able to survive for successive generations and are symptomatic of genetic drift stall.

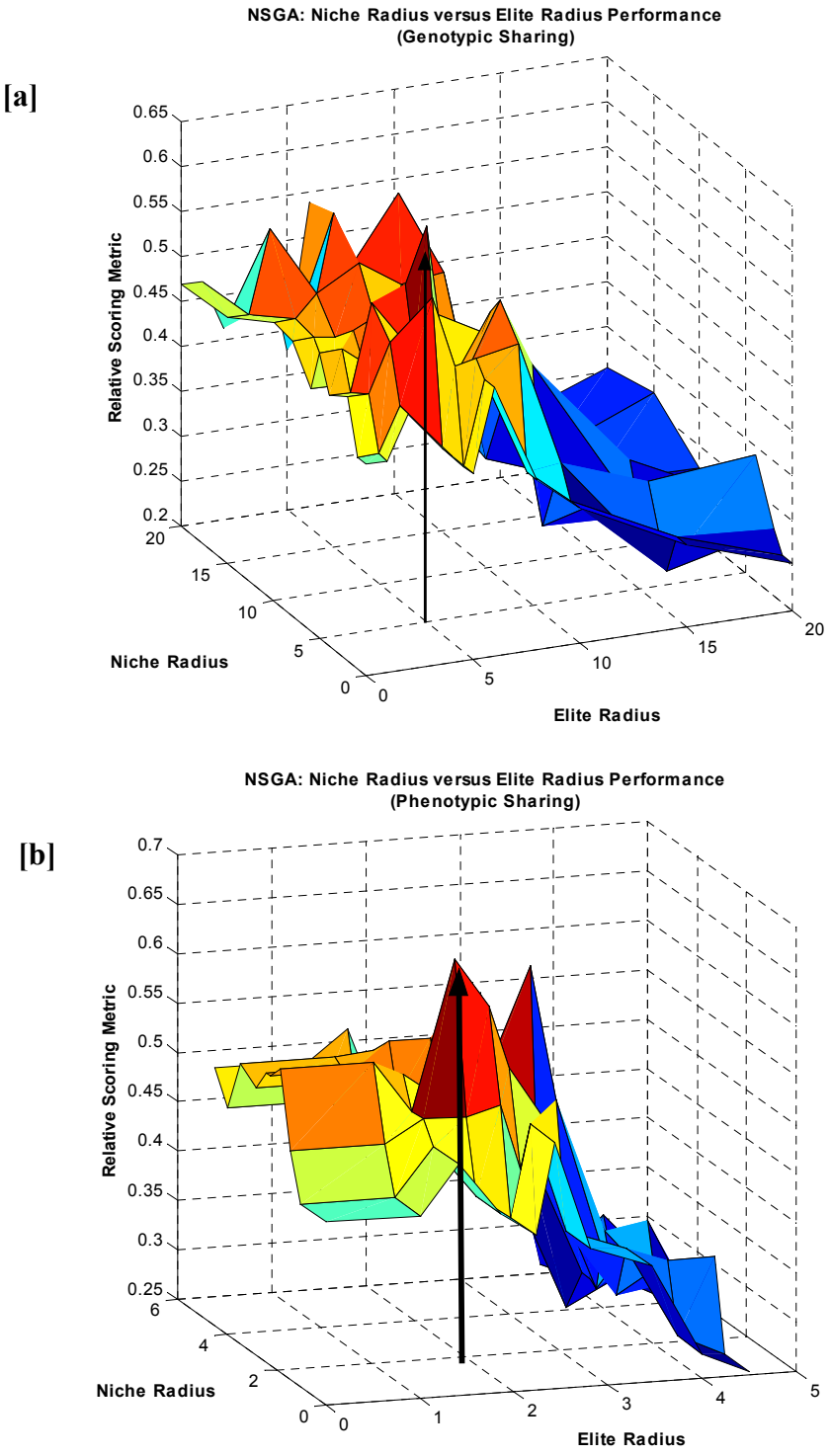


Figure 3.6 Analysis of the NSGA's performance for the full range of possible values that the parameters controlling niching and elitism can be assigned when using (a) genotypic sharing and (b) phenotypic sharing. The arrows designate the algorithm's performance when these parameters are set equal to the recommended niche radius attained from the relationships presented by Deb & Goldberg (1989).

Genetic drift also appears to be prevalent in Figure (3.4), which shows that the NSGA (in the absence of elitism) is unable to identify more than 3 members of the Pareto optimal set regardless of the sharing method invoked. Figure (3.7) shows the performance of all forms of the NSGA as a function of time.

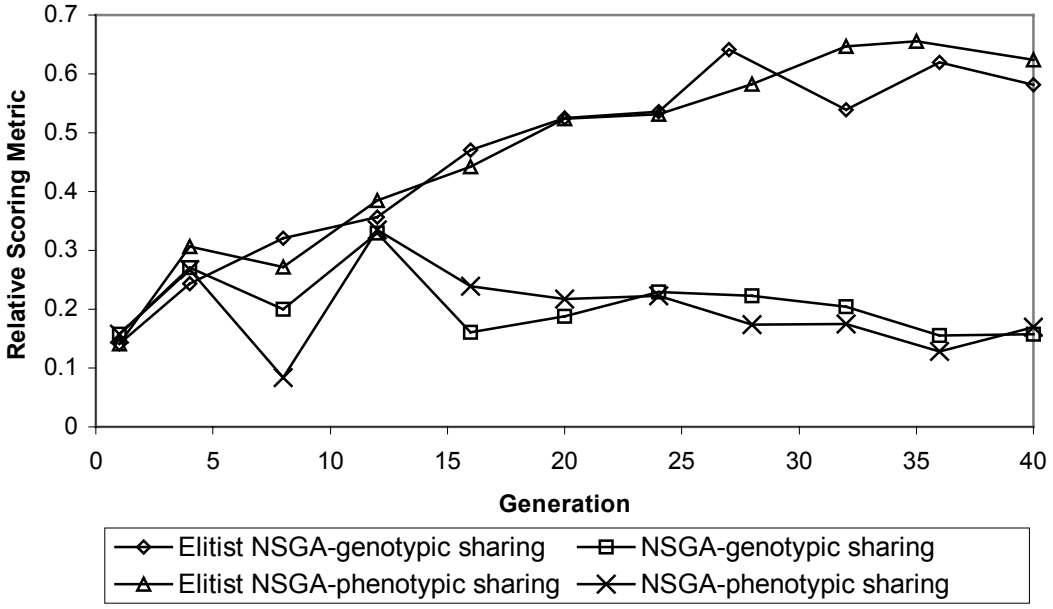


Figure 3.7 Performance of both the Elitist NSGA and the NSGA as functions of generation

The figure shows that elitism allows the NSGA to steadily improve performance over the duration of the run. In the absence of elitism, the sporadic peaks in performance are steadily degraded over the course of the runs, confirming that drift stall is occurring. Recall that the NSGA decrements the dummy fitness values for successive fronts by a constant value ΔD_f and that stochastic remainder selection selects population members based on the ratio of their dummy fitness relative to the population's average dummy fitness. Drift stall is occurring because the constant decrement ΔD_f does not properly scale the ratio of the nondominated individuals' fitness values relative to the population's average fitness, making it impossible for the NSGA to

distinguish and propagate these individuals into successive generations. These observations motivated further analysis of the influences of selection pressure and drift stall on the algorithm's performance in the subsequent section.

3.5.2 Performance Under Increased Selection Pressure

The results of the previous section showed that both the NSGA and the Elitist NSGA converged to nondominated fronts with gaps in the extreme portions of the Pareto frontier. Additionally, Figure (3.7) shows that the NSGA's performance only sporadically improves and generally degrades over the duration of the runs regardless of the sharing method considered. These observations confirm that several of the niches are either being lost or converging to non-optimal values due to the absence of sufficient selection pressure. Furthermore, it is readily apparent that elitism significantly improves the algorithm's performance confirming the results of *Zitzler et al.* (1999). Elitism serves to increase the selection pressure on the niches and reduce the influence of genetic drift on the system, which in effect is analogous to rescaling the relative fitness values of the population. Modifying the NSGA and Elitist NSGA such that successive fronts have exponentially decreasing fitness using the scaling coefficient S_c (as described in the methods section) enabled further analysis of the influence of increased selection pressure on performance. For the monitoring application, analysis of the NSGA's performance for scaling coefficient values ranging from 0.7 to 0.95 showed that S_c should be set to approximately 0.9. Values for S_c lower than 0.9 greatly reduced the diversity of the populations and caused premature convergence (due to excessive selection pressure). Scaling coefficient values above 0.9 resulted in significant gaps in the extreme regions of the Pareto frontier due to insufficient selection pressure and genetic drift.

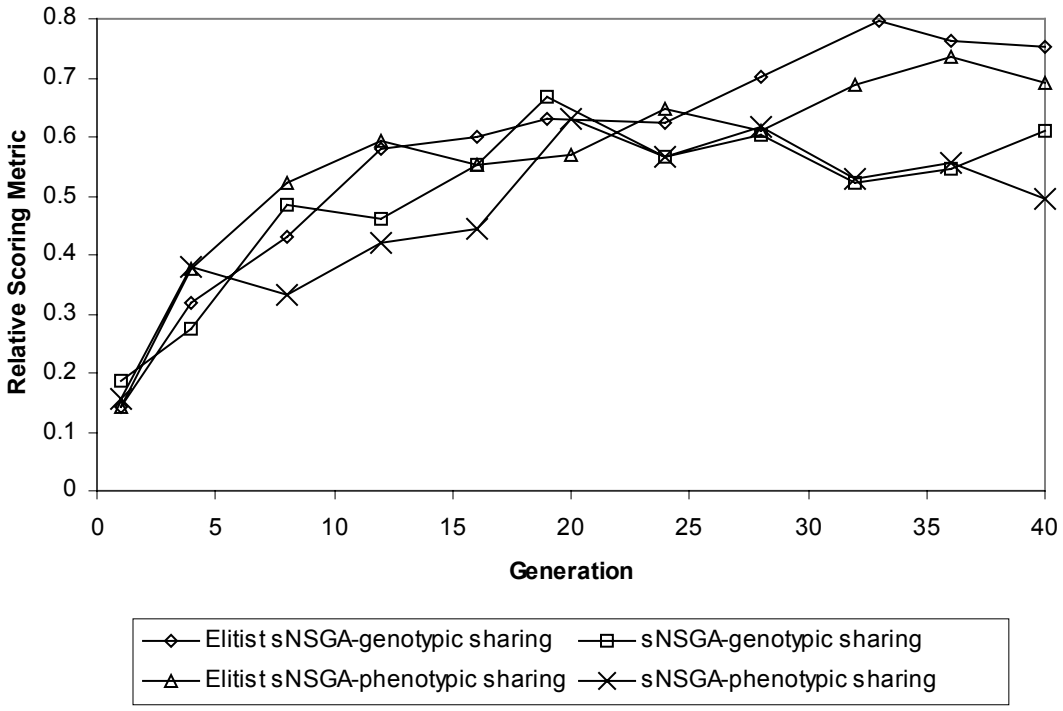


Figure 3.8 Performance of both the Elitist NSGA and the NSGA as a function of generation after rescaling the fitness assignments

Figure (3.8) shows the performances of the scaled NSGA (sNSGA) and the Elitist sNSGA as a function of time when S_c equals 0.9. The population sizes used in these runs remained the same as those in the previous section for all forms of the sNSGA except the non-elitist, genotypic sNSGA where a population size of 870 was found to be optimal, which represents only a slight increase from the previous section. Rescaling was not able to reduce the sensitivity of genotypic sharing to non-uniformities in the niche spacing, although elitism successfully closed the performance gap between the two sharing methods for the Elitist sNSGA. It is immediately obvious from Figure (3.8) that rescaling the fitness assignments greatly improved the performance of the sNSGA, which no longer shows degradation in performance with time and instead steadily improves towards an upper bound in performance. Note that the combination of rescaling and elitism resulted in the Elitist sNSGA exceeding the previous results

shown in Figure (3.7) for the Elitist NSGA. The phenotypic, Elitist NSGA was able to reach a maximum RSM value of 0.66 whereas the genotypic, Elitist sNSGA attained a maximum RSM value of 0.8.

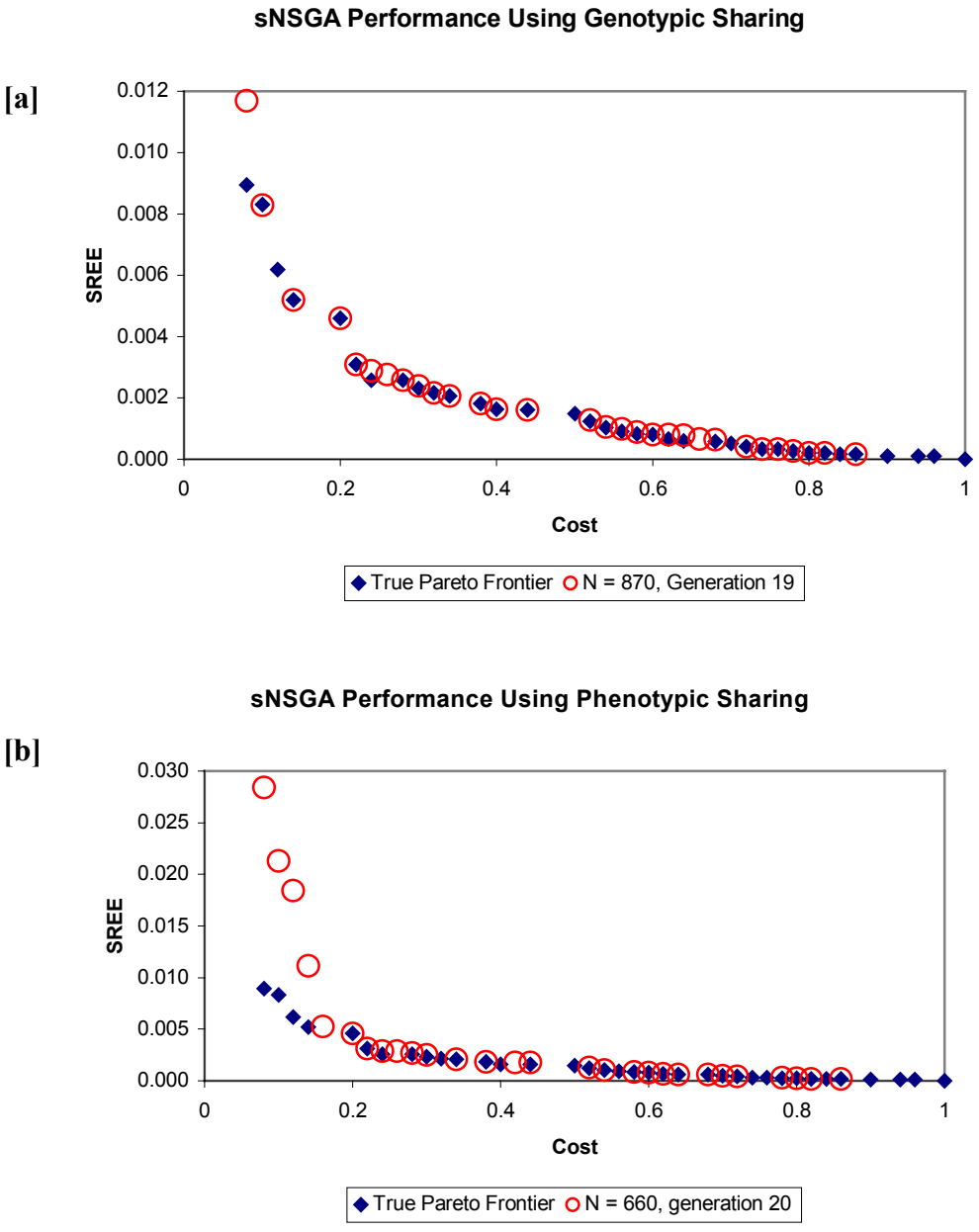
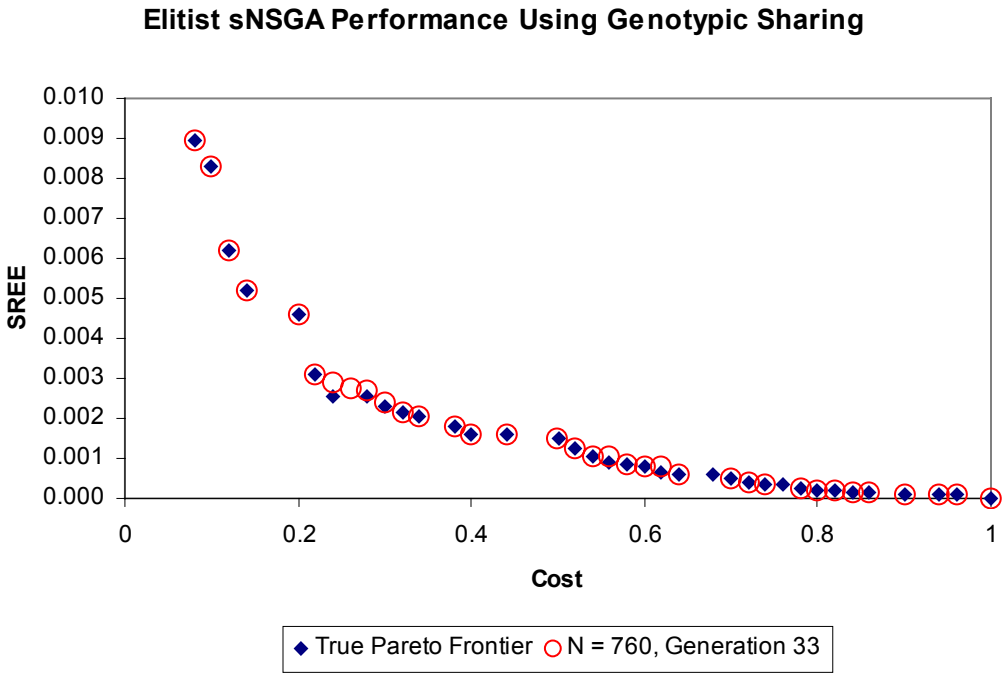


Figure 3.9 sNSGA's performance relative to the true Pareto front using both (a) genotypic and (b) phenotypic sharing

[a]



[b]

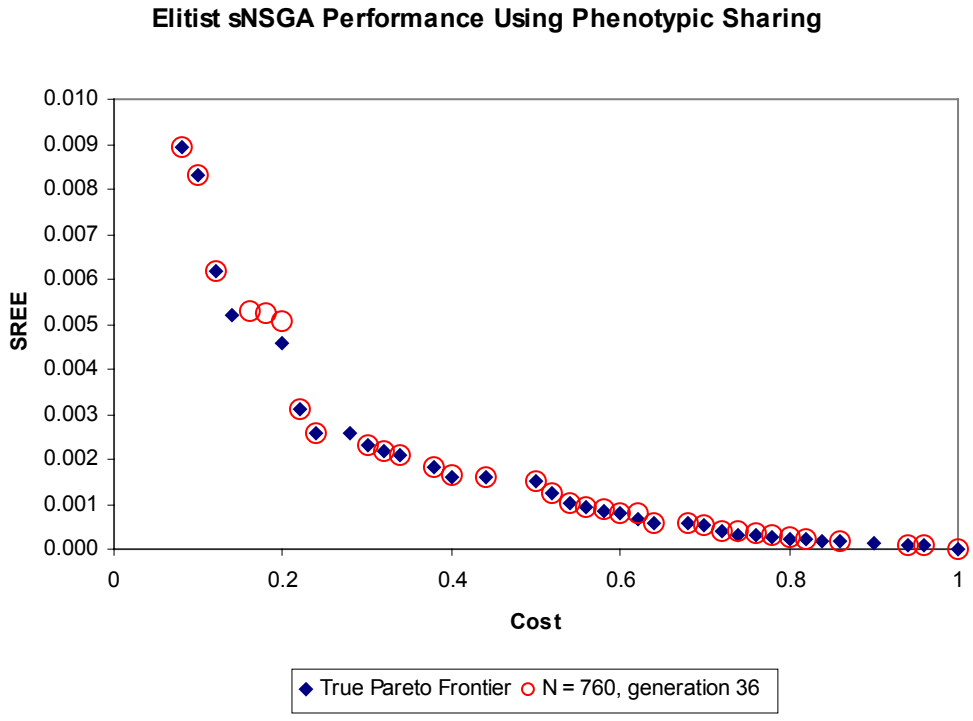


Figure 3.10 Elitist sNSGA's performance relative to the true Pareto front using both (a) genotypic and (b) phenotypic sharing

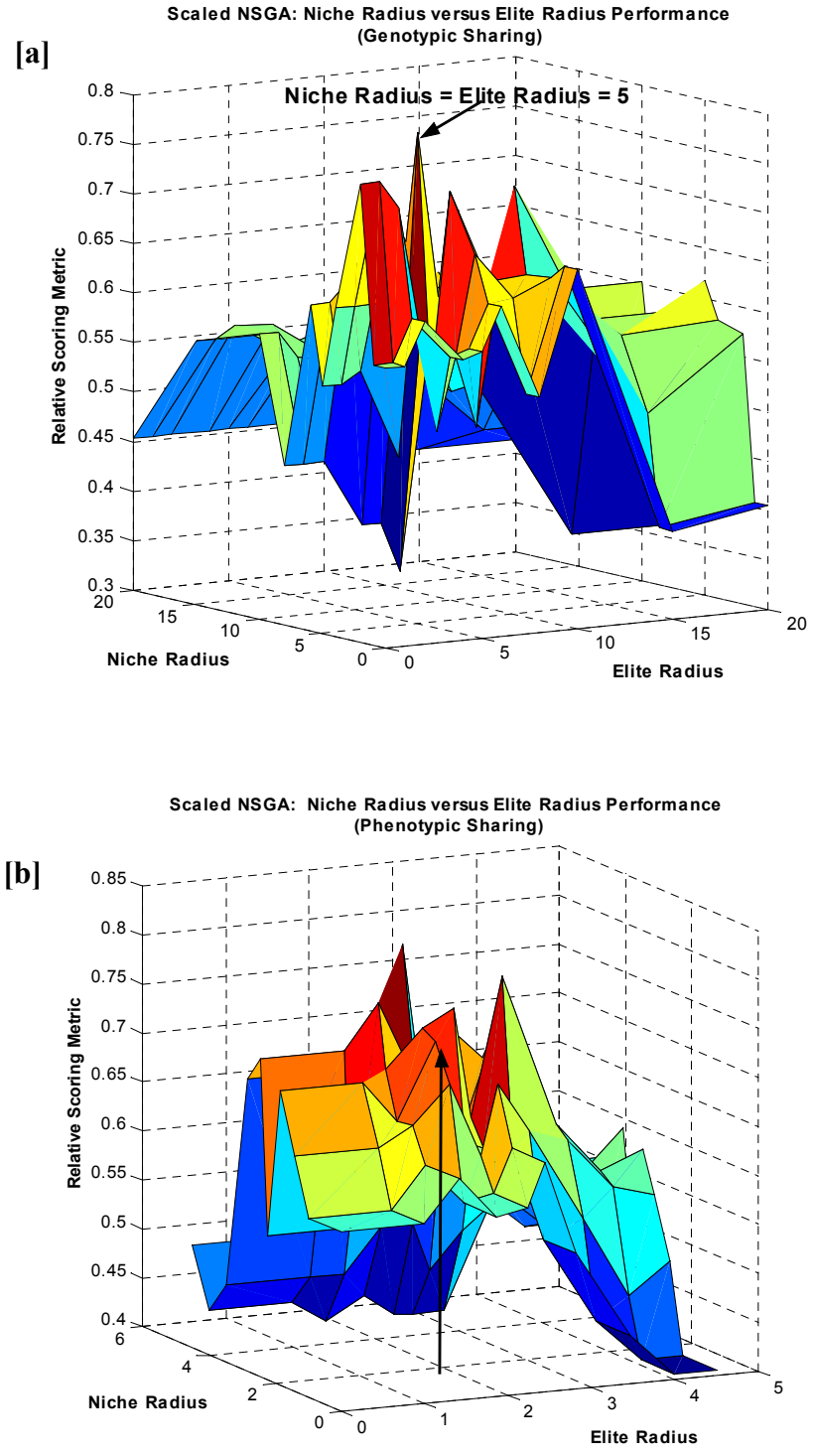


Figure 3.11 Analysis of the sNSGA's performance for the full range of possible values that the parameters controlling niching and elitism can be assigned when using (a) genotypic and (b) phenotypic sharing. The arrows designate the algorithm's performance when these parameters are set equal to the recommended niche radius attained from the relationships presented by Deb & Goldberg (1989).

Figure (3.9) and Figure (3.10) show the improved performances of all forms of the NSGA that result from rescaling the fitness assignments. Recall that previously the non-elitist NSGA was able to find a maximum of 3 members of the Pareto optimal set. Rescaling the shared dummy fitness values resulted in finding 21 and 17 members of the Pareto optimal set when implementing genotypic and phenotypic sharing, respectively. The additional use of elitism enabled the sNSGA to almost replicate the Pareto frontier for both sharing methods, as shown in Figure (3.10). The genotypic, Elitist sNSGA provided the most complete representation of the frontier by exactly finding 29 members of the Pareto optimal set while also finding close representations of all but 2 of the remaining 36 members. The phenotypic, Elitist sNSGA found either exact or close representations of 31 of the 36 members of the Pareto optimal set. The slight decrease in performance of the phenotypic, Elitist sNSGA can be explained with analysis of Figure (3.11).

Figure (3.11) shows the performance of the sNSGA using both sharing schemes for the full range of values σ_{elite} and σ_{share} that can be assigned. The plot again shows that increasing the elitist selection pressure by decreasing σ_{elite} generally improves performance. The slight difference in performance between genotypic and phenotypic sharing was caused by the assumption used in this chapter that $\sigma_{elite} \approx \sigma_{share}$. Figure (3.11a) shows that this assumption results in finding the highest peak in performance for the sNSGA under genotypic sharing, whereas Figure (3.11b) shows that this assumption was not able to exactly find the highest performance peak in phenotypic space. It should be noted from Figure (3.6) and Figure (3.11), however, that assuming $\sigma_{elite} \approx \sigma_{share}$ for the monitoring application generally resulted in finding peak or very near peak performances for the various forms of the NSGA in both sharing spaces, which is probably sufficient for most applications. Figure (3.11a) again shows that genotypic

space was generally very noisy relative to phenotypic space, which explains why the sNSGA still required larger population sizes to be used in genotypic space in order for the two sharing schemes to attain comparable results in the absence of elitism. Additionally, the increased noise present in genotypic space after rescaling, as shown in Figure (3.11a) relative to the previous case in Figure (3.6a), explains why N had to be increased from 830 to 870 as discussed above. Overall, the guiding relationships and considerations discussed in the methods section of this chapter were able to successfully guide the design of the NSGA. Figure (3.10) shows that the algorithm was able to find 95 percent of the Pareto optimal set when the influences of population sizing, elitism, and genetic drift were carefully considered in its design.

3.6 Conclusions

The two dimensional tradeoff between cost and SREE for the groundwater monitoring application presented in this work was accurately quantified by ensuring that the NSGA was properly designed to navigate the problem's decision space. Preliminary problem analysis identified a range of potential population sizes and the potential computational complexity of solving the problem. Additionally, relationships from *Deb & Goldberg* (1989) effectively sized the niches required to maintain stable subpopulations, each of which actively sought different sections of the Pareto frontier. Using the recommended niche radius from *Deb & Goldberg* (1989) also effectively identified the niched elitist set of solutions for successive generations in each of the runs performed in this chapter. Elitism greatly improved the efficacy of the NSGA and helped to reduce performance differences between genotypic and phenotypic sharing. Analysis of the algorithm's performance as a function of time showed the adverse effects of genetic drift and motivated further study of the role of selection pressure in the NSGA. Rescaling the shared dummy fitness assignments of the NSGA such that successive fronts had

exponentially decreasing fitness values greatly improved the algorithm's performance. Combining elitism with rescaled fitness assignments resulted in identifying all but 2 members of the Pareto optimal set.

This chapter provides practitioners with a methodology that identifies the proper parameter settings for the NSGA by: (1) identifying the most appropriate population size, (2) properly sizing niches for fitness-based sharing, (3) correctly setting the elitist selection pressure, and (4) careful performance analysis to avoid genetic drift stall. Parameter settings for the NSGA attained using a total of 10 runs resulted in peak or near peak performance of the algorithm. Six of the trial runs were necessary to select the proper population size. Elitism greatly improved the NSGA's performance and narrowed any performance differences between sharing in genotypic versus phenotypic space, a clear advantage when problem-specific information required for phenotypic sharing is not available. When such information is available, though, phenotypic sharing is generally preferred because of the increased noise associated with selection in genotypic space in most EMO applications *(Fonseca & Fleming* 1995, *Horn* 1997), as was observed in this application. Finally, the use of the niche radius to set the elitist selection pressure proved to be very effective in attaining peak or near peak performance from the NSGA when quantifying the 36 member Pareto optimal set of designs that compose the Cost—SREE tradeoff.

Rescaling the fitness assignments to further increase selection pressure and reduce the influence of genetic drift requires consideration of the potential tradeoff between improving the algorithm's performance and reducing diversity within the population. Additional analysis was necessary to properly address this tradeoff and select the appropriate scaling coefficient. The

final selection of a scaling coefficient required 4 additional trial runs after the six trial runs used to set the population and niching parameters.

Although the NSGA was able to successfully capture the Cost—SREE tradeoff, the user should note that this success required significant user interaction, a detailed analysis of 10 runs, and the evaluation of more than 280,000 sampling designs. The NSGA design methodology presented in this chapter requires significant user expertise and a tremendous investment of computational resources that precludes the solution of more challenging multiobjective problems (e.g., problems with more than 2 objectives). Chapter 4 overcomes these limitations by introducing a design methodology for the NSGA-II, a second generation EMO genetic algorithm (see *Deb et al.* 2000) that improves upon the NSGA. The NSGA-II design methodology in the next chapter simplifies Pareto optimization by automating parameter specification and reducing the computational effort required to solve multiobjective applications. The design methodology enables the automatic solution of a new class of high order multiobjective applications in which users can select, understand, and balance more than two performance criteria [see Chapters 4 and 6].