On Measuring Multiobjective Evolutionary Algorithm Performance

David A. Van Veldhuizen
Dept. of Electrical and Computer Engineering
Graduate School of Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433-7765
david.vanveldhuizen@hedo.brooks.af.mil

Gary B. Lamont
Dept. of Electrical and Computer Engineering
Graduate School of Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433-7765
gary.lamont@afit.af.mil

Abstract- Solving optimization problems with multiple (often conflicting) objectives is generally a quite difficult goal. Evolutionary Algorithms (EAs) were initially extended and applied during the mid-eighties in an attempt to stochastically solve problems of this generic class. During the past decade a variety of Multiobjective EA (MOEA) techniques have been proposed and applied to many scientific and engineering applications. Our discussion's intent is to rigorously define and execute a *quantitative* MOEA performance comparison methodology. Almost all comparisons cited in the current literature visually compare algorithmic results, resulting in only relative conclusions. Our methodology gives a basis for absolute conclusions regarding MOEA performance. Selected results from its execution with four MOEAs are presented and described.

1 Introduction

Multiobjective Evolutionary Algorithms (MOEAs) are now a well-established field within Evolutionary Computation, initially developed in the mid-eighties when the first MOEA designed to solve Multiobjective Optimization Problems (MOPs) was implemented. Over 450 publications (Coello, 1999b) have since proposed various MOEA implementations and applications, and to a much lesser extent, underlying MOEA theory.

Solving MOPs with MOEAs presents several unique challenges (e.g., vector-valued fitness, *sets* of solutions, evolutionary operator complexity). Space does not permit a detailed presentation of all material necessary to understand this dynamic and rapidly growing research field. For a good introduction to the relevant issues and past EA-based approaches, see Van Veldhuizen (Van Veldhuizen, 1999), Coello Coello (Coello, 1999a), and Fonseca and Fleming (Fonseca and Fleming, 1995). This paper focuses on rigorously defining and executing a *quantitative* MOEA performance comparison methodology.

The remainder of this paper is organized as follows. Section 2 presents relevant MOP concepts while Section 3 outlines a quantitative MOEA experimental methodology. Sections 4 and 5 discuss experimental results and observations, followed by our conclusions in Section 6.

2 MOP Background

Although single-objective optimization problems may have a unique optimal solution, MOPs (as a rule) present a possibly uncountable set of solutions, which when evaluated produce vectors whose components represent trade-offs in decision space. A decision maker then implicitly chooses an acceptable solution by selecting one of these vectors.

We have defined an MOP's global optimum to be the *Pareto front* determined by evaluating each member of the *Pareto optimal solution set* (Van Veldhuizen, 1999). Since Pareto concepts are crucial components of the MOP domain and MOEA implementations, their incarnations are now briefly described.

During MOEA execution, a "local" set of Pareto optimal solutions (with respect to the *current* MOEA generational population) is determined at each EA generation and termed $P_{current}(t)$, where t represents the generation number. Many MOEA implementations also use a secondary population, storing all/some Pareto optimal solutions found through the generations (Van Veldhuizen, 1999).

This secondary population is termed $P_{known}(t)$, also annotated with t (representing completion of t generations) to reflect possible changes in its membership during MOEA execution. $P_{known}(0)$ is defined as $\emptyset$ (the empty set) and P_{known} alone as the *final*, overall set of Pareto optimal solutions returned by an MOEA. Of course, the *true* Pareto optimal solution set (termed P_{true}) is not explicitly known for MOPs of any difficulty. P_{true} is defined by the functions composing an MOP; it is fixed and does not change. Because of the manner in which Pareto optimality is defined, $P_{current}(t)$ is always a non-empty solution set (Van Veldhuizen, 1999).

$P_{current}(t)$, P_{known}, and P_{true} are sets of MOEA genotypes where each set's phenotypes form a Pareto front. We term the associated Pareto front for each of these solution sets as $PF_{current}(t)$, PF_{known}, and PF_{true}.[1]

[1] Note that Horn (Horn, 1997) uses P_{online}, $P_{offline}$, and P_{actual}

Thus, when using an MOEA to solve MOPs, the implicit assumption is that one of the following conditions hold: $PF_{known} = PF_{true}$, $PF_{known} \subset PF_{true}$, or that over some norm (Euclidean, RMS, etc.), $PF_{known} \in [PF_{true}, PF_{true} + \epsilon]$.

3 MOEA Experiments

The major goal of these experiments is to compare well-engineered MOEAs in terms of effectiveness and efficiency as regards *carefully selected test problems* from the same class. Jackson et al. (Jackson et al., 1991) imply this should suffice to show MOEA feasibility and promise. We also wish to determine how well the test problems and proposed metrics capture and report essential MOP and MOEA characteristics and performance, as well as analyzing any interesting observations resulting from experiment execution and result analysis.

We report relevant *quantitative* MOEA performance based on appropriate experiments. Almost all comparisons cited in the current literature *visually* compare algorithmic results. As experimental numeric MOPs' P_{true} and PF_{true} are often not known (and almost never presented) these conclusions are then relative. The methodology described in the next section gives a basis for *absolute* conclusions regarding MOEA performance – a goal actively pursued by few other MOEA researchers (e.g., Zitzler and Thiele (1999) and Shaw et al. (1999)).

3.1 Experimental Methodology

Although test suite or benchmark functions do provide a common basis for MOEA comparisons, results are empirical unless the global optima are known. We note that finding a general MOP's Pareto optimal solution set is NP-Complete (Bäck, 1996, pg. 56). However, there is a way to approximate P_{true} for specific problem cases. Teaming this data with appropriate metrics then allows desired quantitative MOEA comparisons.

When the real- (continuous) world is modeled (e.g., via objective functions) on a computer (a discrete machine), there is a fidelity loss between the (possibly) continuous mathematical model and its discrete representation. Any formalized MOP being computationally solved suffers this fate. However, at a "standardized" computational resolution and representation, MOEA results can be quantitatively compared not only against each other but against certain MOPs' PF_{true}. Thus, whether or not these selected MOPs' PF_{true} is actually continuous or discrete is not an experimental concern, as the representable P_{true} and PF_{true} are fixed based on certain assumptions.

instead of the terms $P_{current}(t)$, P_{known}, and P_{true}. Our notation is more precise as it allows for generational specification and encompasses each set's corresponding Pareto front.

For purposes of these experiments we define a *computational grid* by placing an equidistantly spaced grid over decision variable space, allowing a uniform sampling of possible solutions. Each grid intersection point (computable solution) is then assigned successive numbers using a binary representation. For example, given a fixed length binary string, decision variable values are determined by mapping the binary (sub)string to an integer.

Even though a binary representation restricts a search space's size, it does allow for a "fair" and quantitative MOEA comparison, determination of an MOP's PF_{true} (at some resolution), and an enumeration method for deterministically searching a solution space. The underlying resolution may be increased/decreased as desired, at least up to some point where the enumerative computation becomes impractical or intractable. This methodology is designed for experimentation and used to make judgments about proposed MOEAs and their implementations.

We constructed an enumerative search program executing on parallel high-powered computers whose purpose is to find P_{true} and PF_{true} for several selected numeric MOPs. The resulting sets are still only a discrete representation of their continuous counterparts, but are the "best possible" at a given computational resolution. The program is written in "C" and uses the Message Passing Interface (MPI) to distribute function evaluations among many processors. A parallel implementation is selected to efficiently process large solution spaces and construct a database containing P_{true} for certain problem instantiations.

Using the resultant database, various MOEA results may be compared not only against each other, but also against the *true* MOP optimum (at some resolution). At least for selected MOPs a true quantitative comparison is then possible, thus allowing for absolute performance observations.

3.2 MOEA Test Algorithms

Four MOEAs were selected for testing. These algorithms and their original *raisons d'etre* are:

1. **MOGA**. Implemented by Fonseca and Fleming (1998). Initially used to explore incorporation of DM goals and priorities in the multiobjective search process. Employs an original Pareto ranking scheme and incorporates fitness sharing.

2. **MOMGA**. Implemented by Van Veldhuizen and Lamont (2000), (Van Veldhuizen, 1999). Initially used to explore the relationship between MOP solution building blocks and their use in MOEA search. Incorporates fitness sharing and Horn et al.'s (1994) tournament selection.

3. **NPGA**. Implemented by Horn et al. (1994). Initially used to explore benefits of providing P_{known} as input to a decision analysis technique. Uses tournament selection based on Pareto optimality. Incorporates fitness sharing.

4. **NSGA**. Implemented by Srinivas and Deb (1994). Initially used to explore bias prevention towards certain regions of the Pareto front. Employs Goldberg's (Goldberg, 1989) Pareto ranking scheme and incorporates fitness sharing.

We note here that these algorithms were selected because they specifically incorporate what appear to be key theoretical problem/algorithm domain aspects such as Pareto ranking, niching, and fitness sharing. Other researchers appear to share these thoughts as the MOGA, NPGA, and NSGA (or variants thereof) are the literature's most cited and imitated (Van Veldhuizen, 1999).

The MOGA, NPGA, and NSGA are based on "traditional" GAs; the MOMGA is based on the mGA and can be considered non-standard. However, the conceptual evolutionary process modeled by each algorithm is the same and gives the basis for their direct comparison. Other algorithms were considered but not included in these experiments (for various reasons); those and other alternative MOEAs may be considered in later experiments.

Although the NFL theorems (Wolpert and Macready, 1997) show there is no "best" EA, certain EAs have been experimentally shown to be more likely effective than others for some real-world problems – attempts to experimentally examine MOEA effectiveness are few. Additionally, no studies have been performed showing which parameters (or parameter values) are key to good performance.

3.3 MOEA Key Parameters

Many EA experiments vary key algorithmic parameters in an attempt to determine the most effective and efficient implementation for a particular problem instantiation or class. A parameter analysis investigating effects of differing parameter values is beyond the scope of these experiments; we do not wish to "tune" MOEAs for good performance on some problem class. These algorithms are then executed with default parameter values as reported in the literature, implementing each MOEA "out of the box" as it were. However, the term "default" is somewhat of a misnomer as no formal MOEA parameter value studies are known.

Existing empirical experimental results sometimes indicate mating restriction is necessary for good performance – at other times various MOEA implementations seem to operate well without it. These empirical results indicate the NFL theorems are alive and well (Wolpert and Macready, 1997). As incorporating mating restriction in some experimental software required major code modifications, and because of its uncertain usefulness in the MOP domain, mating restriction is *not* incorporated in these experimental MOEAs.

The mGA's "cut and splice" EVOPs' effect (when both are used) is intended to be similar to recombination's (Goldberg et al., 1989). The MOMGA used mGA default parameters for these operators, namely $p_{cut} = 0.2$ (only one cut allowed per string) and $p_{splice} = 1.0$. There is not yet a "default" MOEA crossover rate but past experiments used crossover probabilities in the range $p_c \in [0.7, 1.0]$ (Van Veldhuizen, 1999). Thus, the other experimental MOEAs used single-point crossover with $p_c = 1.0$. All but the MOMGA used a mutation rate of $p_m = \frac{1}{l}$ where l is the number of binary digits. The MOMGA did not employ mutation (i.e., $p_m = 0$). As in the original mGA presentation (Goldberg et al., 1989), this results in the most stringent possible testing. As mutation is not available to provide diversity and "recreate" BBs, losing a BB from the population means it is gone forever.

We compare experimental MOEA results derived after an identical number of solution evaluations are performed, using that factor as a measure of common computational effort. However, the number of executed solution evaluations differs between MOMGA runs (even those solving the same MOP) because of internal parameters dynamically governing its operation. In these experiments, the MOMGA is set to execute for three eras and to contain 100 individuals in each juxtapositional population. All MOMGA runs are terminated before the total number of solution evaluations for a run exceeds 65,536 (2^{16}). The total fraction of explored search space is then bounded above by $\frac{2^{16}}{2^{24}} = 0.39\%$ (the computational grid used for all MOPs results in 2^{24} possible solutions). Historically, EAs often execute at most tens of thousands of fitness evaluations and this experimental limit is within that range. As it explores only a small fraction of the search space an MOEA's effectiveness should be readily apparent in how well its results (P_{known} and PF_{known}) compare to P_{true} and PF_{true} (if known).

Thus, for all test MOPs, the MOMGA was executed first and the number of executed solution evaluations per run determined. The other MOEAs (each with population size $N = 50$) were then set to execute almost the same number of evaluations (N multiplied by the number of generations), ensuring a very nearly equivalent computational effort for each tested MOEA. Again, please note these experiments' purpose is to explore MOEA performance and not to determine ideal parameter settings over the test functions.

3.4 MOEA Experimental Metrics

What metrics might adequately measure an MOEA's results or allow meaningful comparisons of specific MOEA implementations? Appropriate metrics must be selected upon which to base MOEA performance claims, and as the literature offers few quantitative MOEA metrics, proposed metrics must be carefully defined to be useful. Additionally, no single metric can entirely capture total MOEA performance, as some measure algorithm effectiveness and others efficiency. Temporal effectiveness and efficiency may also be judged, e.g., measuring an MOEA's progress each generation. All may be considered when judging an MOEA.

The metrics identified in this section measure performance in the phenotype domain. Whereas many operations researchers attempt to generate P_{true} (and thus implicitly measure performance in genotype space) (Van Veldhuizen, 1999), MOEA researchers have mainly focused on generating PF_{true} (and thus measure performance in phenotype space). As there is a direct correspondence between solutions in P_{true} and vectors in PF_{true} one method may not be "better" than another. However, note that multiple solutions may map to an identical vector. Also, we here discuss only primary metrics used in our experiments – a more complete discussion is found elsewhere (Van Veldhuizen, 1999).

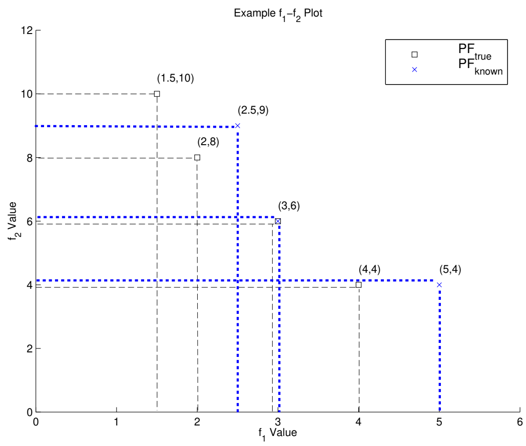

Figure 1: PF_{known} / PF_{true} Example

3.4.1 Generational Distance

Used in other experiments (Van Veldhuizen and Lamont, 1998) this metric is a value representing how "far" PF_{known} is from PF_{true} and is defined as:

$$G \triangleq \frac{(\sum_{i=1}^{n} d_i^p)^{1/p}}{n}, \qquad (1)$$

where n is the number of vectors in PF_{known}, $p = 2$, and d_i is the Euclidean distance (in objective space) between each vector and the *nearest* member of PF_{true}. A result of 0 indicates $PF_{true} = PF_{known}$; any other value indicates PF_{known} deviates from PF_{true}. The example in Figure 1 has $d_1 = \sqrt{(2.5-2)^2 + (9-8)^2}$, $d_2 = \sqrt{(3-3)^2 + (6-6)^2}$, and $d_3 = \sqrt{(5-4)^2 + (4-4)^2}$, and $G = \sqrt{1.118^2 + 0^2 + 1^2}/3 = 0.5$.

3.4.2 Spacing

This metric is a value measuring the spread (distribution) of vectors throughout PF_{known}. The experimental MOEAs perform fitness sharing in an attempt to "evenly sample" PF_{true} as reflected in each generational population ($PF_{current}(t)$). Because PF_{known}'s "beginning" and "end" are known, a suitably defined metric judges how well PF_{known} is distributed. Schott (Schott, 1995) proposes such a metric measuring the range (distance) variance of neighboring vectors in PF_{known}. Called *spacing*, he defines this metric as:

$$S \triangleq \sqrt{\frac{1}{n-1} \sum_{i+1}^{n} (\overline{d} - d_i)^2}, \qquad (2)$$

where $d_i = \min_j (\mid f_1^i(\vec{x}) - f_1^j(\vec{x}) \mid + \mid f_2^i(\vec{x}) - f_2^j(\vec{x}) \mid)$, $i, j = 1, \ldots, n$, $\overline{d}$ is the mean of all d_i, and n is the number of vectors in PF_{known}. A value of zero for this metric indicates all members of PF_{known} are equidistantly spaced. However, note that vectors composing PF_{true} in objective space are not necessarily uniformly spaced. The example in Figure 1 has an $S = 0.25$. Srinivas and Deb (Srinivas and Deb, 1994) define a similar measure expressing how well an MOEA has distributed Pareto optimal solutions over the Pareto optimal set (genotype space).

3.4.3 Overall Nondominated Vector Generation and Ratio

The tested MOEAs add $P_{current}$ to P_{known} each generation, most probably resulting in different cardinalities for P_{known}. This metric then measures the total number of nondominated vectors found during MOEA execution and is defined as:

$$ONVG \triangleq \mid PF_{known} \mid. \qquad (3)$$

Schott (Schott, 1995) uses a like metric (although defined over the Pareto optimal set, i.e., $\mid P_{known} \mid$). Genotypically or phenotypically defining this metric is probably a matter of preference, but we again note multiple solutions may map to an identical vector, i.e., $\mid P_{known} \mid \geq \mid PF_{known} \mid$. Although counting the number of nondominated solutions gives some feeling for how effective the MOEA is in generating desired solutions, it does not reflect on how "far" from PF_{true} the vectors in PF_{known} are. Additionally, too few vectors and

PF_{known}'s representation may be poor; too many vectors may overwhelm the DM.

It is difficult to determine what good values for $\mid ONVG \mid$ might be. PF_{known}'s cardinality may change at various computational resolutions as well as differing (perhaps radically) between MOPs. Reporting the ratio of PF_{known}'s cardinality to the discretized P_{true}'s gives some feeling for the number of nondominated vectors found versus how many exist to be found. This metric is then defined as:

$$ONVGR \triangleq \frac{\mid PF_{known} \mid}{\mid PF_{true} \mid} . \quad (4)$$

A value of 1 indicates the MOEA has found the same number of nondominated vectors as exists in PF_{true}. The example in Figure 1 has an $ONVG = 3$ and an $ONVGR = 0.75$.

3.5 MOEA Computational Environment

All MOEAs are executed on the same computational platform for consistency. The host is a Sun Ultra 60 workstation with dual 300 MHz processors and 512 MB RAM, running Solaris 2.5.1. Many other computational platforms would suffice but this high-end host offers exclusive access. The MOMGA and NPGA are extensions of existing algorithms and specific software, are written in "C" and compiled using the Sun WorkShop Compiler version C 4.2. Much of our associated research and related experimentation employs a specialized *MATLAB* EA-based toolbox; the MOGA and NSGA are written as self-contained "m-files" leveraging pre-defined toolbox routines. These MOEAs are also executed on the Sun platform described previously but within the *MATLAB* 5.2 environment. Timing results are not of specific experimental concern here. However, for all experimental MOPs, each MOEA run executes in a matter of minutes.

3.6 MOEA Experimental MOPs and Approach

Several MOPs are elsewhere substantiated, formulated, and proposed for use in an MOEA test function suite (Van Veldhuizen, 1999). For these experiments' test functions we select the following MOPs: MOP1, MOP2, MOP3, MOP4, and MOP6. These are all bi-objective MOPs exhibiting key characteristics of the MOP domain.

The purpose of these experiments is to compare well-engineered algorithms in terms of effectiveness as regards *carefully selected test problems.* We wish to determine selected MOEA performance over the MOP domain class, to evaluate the usefulness of proposed test functions and metrics, and to record other germane observations arising during experiment execution and result analysis. Thus, the metrics described in Section 3.4 are selected for use because they initially appear to be the most appropriate indicators of MOEA performance and thus provide a validated basis for MOEA comparison.

4 MOEA Experimental Analyses

All MOP test functions used are formulated as stated elsewhere (Van Veldhuizen, 1999, pp. 5-13 – 5-14); other experimental and algorithmic parameters are as discussed in Sections 3.1 to 3.3. For statistical comparison purposes the four experimental MOEAs were each executed ten times for each MOP, providing a statistical sample with which to derive metric values. Each MOEA's results for each MOP are separately analyzed followed by a discussion of their performance across all tested MOPs.

4.1 Experimental Metrics and MOPs

Although the experimental metrics and MOPs were previously theoretically validated, these experiments highlight practical implementation difficulties. We discuss elsewhere that some MOEA metrics appear not as valuable as others (Van Veldhuizen, 1999). Therefore, using their current definitions, we consider spacing, generational distance, and $ONVG$ as the most meaningful metrics for analysis. Although generational distance is dependent upon objective space size, spacing and $ONVG$ values may not be as much so.

Taken overall, the selected test suite MOPs appear useful in practice as well as in theory. We do make recommendations to modify some formulations, however (Van Veldhuizen, 1999).

4.2 Experimental Statistical Analyses

Preliminary results imply each algorithm's observations are *not* normally distributed, and that the variance is noticeably different for different MOEAs (Van Veldhuizen, 1999). This data may not then satisfy necessary assumptions for parametric mean comparisons and we thus consider non-parametric statistical techniques for analyzing these experimental results.

The Kruskal-Wallis H-Test requires no assumptions about the probability distributions being compared (McClave et al., 1998). However, other assumptions must be satisfied in order to apply this test: that five or more measurements are in each sample and that the samples are random and independent; and that the probability distributions from which the samples are drawn are continuous. Our initial results meet this criteria so we test the following hypotheses:

> H_0: The probability distributions of MOGA, MOMGA, NPGA, and NSGA results applied to MOP$\mathbf{X}$ are identical.

H_a: At least two of the experimental MOEAs' result distributions differ.

Kruskal-Wallis H-Test results are termed p-values, also called the observed significance levels. We reject the null hypothesis whenever $p \leq \alpha$. Using a significance level $\alpha = 0.1$, we in all cases see there is enough evidence to support the alternative hypothesis and conclude that for each MOP and recorded metrics, at least two MOEAs' results' distributions differ (Van Veldhuizen, 1999).

This result allows use of the Wilcoxon rank sum test in comparing the results of MOEA "pairs," attempting now to determine which of a given two MOEAs does "better." This test assumes the sample of differences is randomly selected and that the probability distributions from which the sample of paired differences is drawn is continuous (McClave et al., 1998). Our initial results meet this criteria so we test the following hypotheses:

H_0: The probability distributions of MOEA_1 and MOEA_2 results applied to MOP**X** are identical.

H_a: The results' distributions differ for the two MOEAs.

There are six possible MOEA pairings: MOGA and MOMGA, MOGA and NPGA, MOGA and NSGA, MOMGA and NPGA, MOMGA and NSGA, and NPGA and NSGA. If c Wilcoxon rank sum tests are performed with an overall level of significance α, the Bonnferroni technique (Neter et al., 1990) allows us to conduct each individual test at a level of significance $\alpha^* = \alpha/c$ (McClave et al., 1998). For these tests we select an overall significance level $\alpha = 0.2$ due to the fact we have only 10 data points per MOEA. Thus, $\alpha^* = 0.2/6 = 0.0\overline{3333}$.

Wilcoxon rank sum test results are termed p-values, also called the observed significance levels. We reject the null hypothesis whenever $p \leq \alpha^*$. These tests show the majority of pairwise MOEA comparisons provide enough evidence to support the alternative hypothesis, and we can thus conclude in those cases that there is a significant statistical difference between the MOEAs (Van Veldhuizen, 1999). We are then able to make conclusions about each MOEA's results as regards each of the three metrics. For each metric, a figure presenting mean metric performance (μ) is plotted for each MOP by algorithm with error bars 2σ in length ($\mu + \sigma$, $\mu - \sigma$, with σ the standard deviation).

4.2.1 Generational Distance Statistical Analysis

Figure 2 presents MOEA performance as regards generational distance. The MOP1 values for both the NPGA and NSGA were large enough to skew the results when viewed in this format – the graph truncates those two bars (their respective results are $G \approx 66$ and $G \approx 11$). The pairwise Wilcoxon rank sum tests allow us to state that in general, when considering generational distance the MOGA, MOMGA, and NPGA gave better results than the NSGA over the test suite problems. This is certainly true for MOP4; the MOGA and MOMGA perform much better than the other two MOEAs on MOP6.

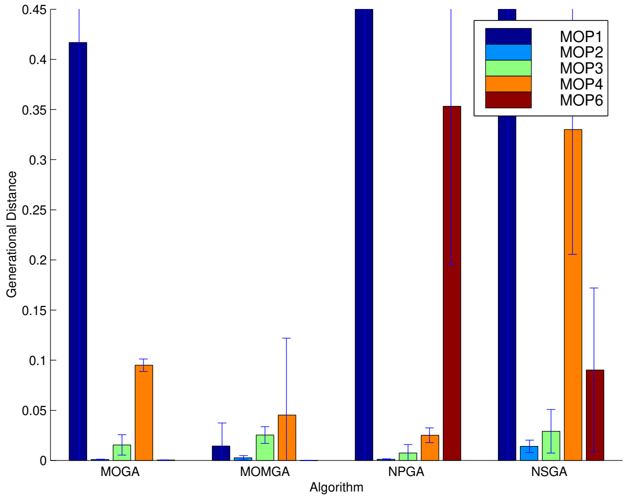

Figure 2: Overall Generational Distance Performance

4.2.2 Spacing Statistical Analysis

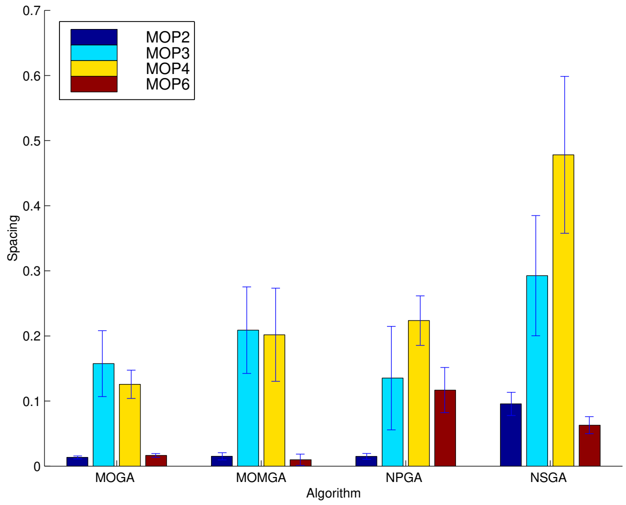

Figure 3: Overall Spacing Performance

Figure 3 presents overall MOEA performance as regards spacing. This graph does not report MOP1 results as they are (possibly) somewhat misleading (Van Veldhuizen, 1999). The pairwise Wilcoxon rank sum tests allow us to state that in general, when considering spacing the NSGA gave worse results over the test suite problems. This is certainly true for MOP2 and

MOP4; the MOGA and MOMGA perform much better than the other two MOEAs on MOP6.

4.2.3 $ONVG$ Statistical Analysis

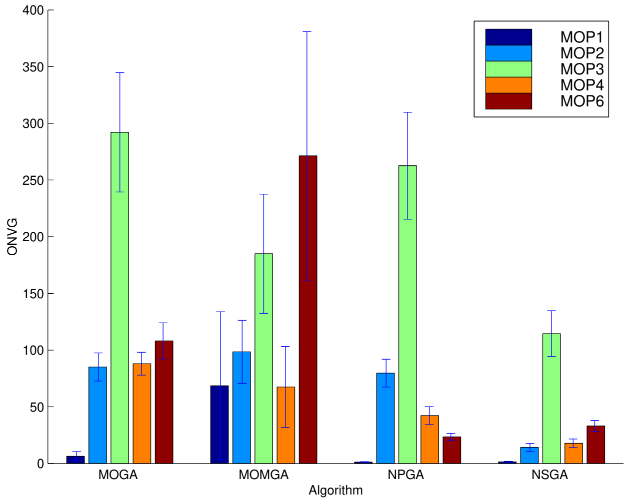

Figure 4: Overall $ONVG$ Performance

Figure 4 presents overall MOEA performance as regards $ONVG$. The pairwise Wilcoxon rank sum tests allow us to state that in general, when considering $ONVG$ the NSGA again gave worse results over the test suite problems. Disregarding MOP1 allows us to state the these three algorithms *always* outperformed the NSGA. This is a surprising result. Although sometimes performing worse (when considering spacing and generational distance), the NSGA generally returned values "close" to the other algorithms. In this case the NSGA consistently returns fewer (often less than half) the number of nondominated vectors than the other algorithms. We highlight this result as we are attempting to provide a DM with a number of choices represented by the nondominated vectors composing PF_{known} .

5 MOEA Experimental Observations

A number of related experiments are executed in support of those analyzed in this chapter. Selected results and observations gleaned through the experimental process are reported in this section.

5.1 Experimental Timing Analysis

Section 3.5 intimates that the MOMGA and NPGA execute more quickly than the MOGA and NSGA. Although probably still true, the following explains the primary cause of these observations. As previously indicated, the MOGA and NSGA are implemented within a $MATLAB$ toolbox. Part of the toolbox's default output includes CPU and wall-clock execution time. Figure 5 reports

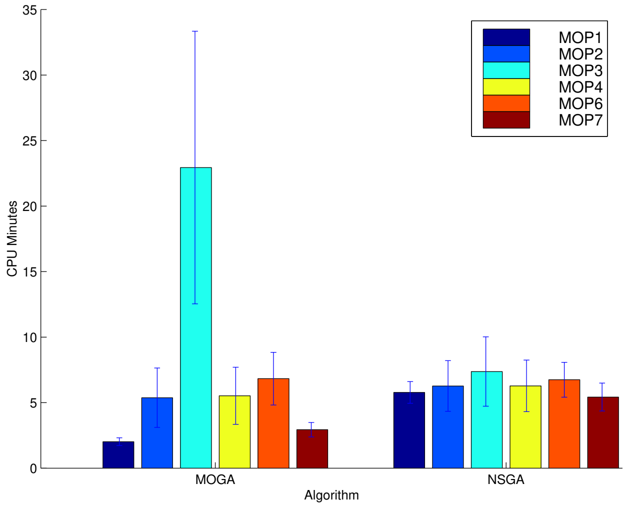

Figure 5: MOEA Timing

mean CPU timing results (μ) for each MOP tested with error bars as above.

It is seen that the MOGA's mean values are generally less than those of the NSGA, although MOP3 is an exception. A straightforward reason exists for this. All MOEAs employed a secondary population. Whereas the MOMGA and NPGA append $P_{current}$ to a file containing P_{known} each generation, the MOGA and NSGA update P_{known} as part of their generational loop. As determination of Pareto optimality is $\mathcal{O}(n^2)$, and as the MOGA always returns more nondominated vectors than the NSGA (see Figure 4), its actual execution times are most likely actually much lower (as are the NSGA's). Coello Coello also notes the NSGA's large execution time (Coello, 1996, pp. 187-189). However, even if the MOGA's and NSGA's logic is changed to reflect that of the MOMGA and NPGA (or vice versa), we believe the MOMGA and NPGA are still faster running MOEAs because they are machine executables.

5.2 Additional Experimental Metrics

As discussed in Section 3.4, some experimental metrics may also be used to track MOEA generational performance. We provide examples here for further insight. These results are from an arbitrary MOP7 experimental run (MOP7 is defined elsewhere (Van Veldhuizen, 1999)); we note that the NPGA is not used in solving this MOP.

Generational Nondominated Vector Generation ($GNVG$) is a metric tracking the number of nondominated vectors produced each MOEA generation (Van Veldhuizen, 1999). Figure 6 presents $GNVG$ results. As the MOMGA records $P_{current}$ each *juxtapositional* generation, we see values for 13 generations each in eras 1 and 2, and 14 in era 3. Note that the number of

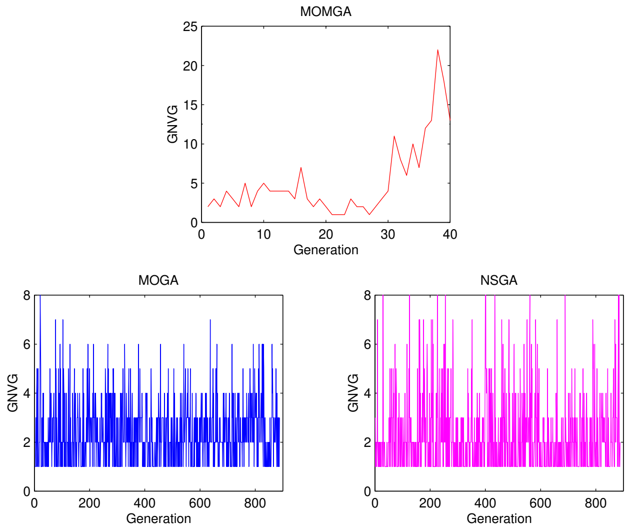

Figure 6: $GNVG$

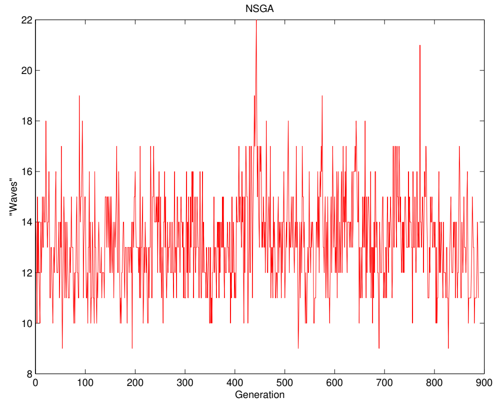

Figure 7: NSGA "Waves"

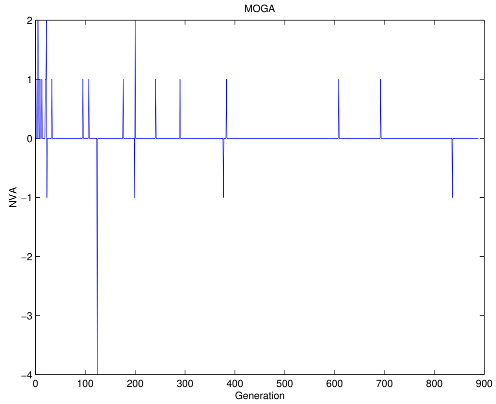

Figure 8: MOGA NVA

nondominated vectors is higher near the algorithm's end and that it may produce two to three times the number of vectors as the MOGA and NSGA. This is most likely due to the MOMGA's larger juxtapositional population size (100 individuals vs. 50). The latter MOEAs record results *every* generation. Their results are plotted using identical scales; only three NSGA generations reported a $GNVG$ above 8 (two of 9 and 1 of 11).

Only one citation in the known literature (Van Veldhuizen, 1999) reports any quantitative results concerning the number of "waves" or fronts produced (per generation) by implementing Goldberg's Pareto ranking scheme. We present a similar graph (Figure 7) showing the number of waves may vary significantly between generations. As this computation may be significant, it is most likely the main reason why the NSGA is the slowest MOEA tested (shown in Section 5.1). Figure 8 shows Nondominated Vector Addition (NVA) values recorded during a MOGA run. This metric is defined as the change in cardinality between $PF_{known}(t)$ and $PF_{known}(t+1)$ (Van Veldhuizen, 1999). This metric's value may remain steady for several generations and sometimes even shows a net loss of solutions from PF_{known}.

Figure 9 is reproduced from a previous publication investigating MOEA convergence (Van Veldhuizen and Lamont, 1998). The generational distance metric (see Section 3.4.1) is used to indicate an MOEA's convergence to PF_{true}. One expects G to generally decrease during execution if convergence is occurring, although the contextual nature of determining Pareto dominance is clearly reflected by the results in Figure 9.

6 Conclusion

This paper proposes and implements a quantitative MOEA performance comparison methodology, presenting experimental results and analyses. Using three practical MOP comparison metrics (generational distance, overall nondominated vector generation, and spacing), nonparametric statistical analyses then show NSGA performance to be statistically worse than the other tested MOEAs (over the test problems). It concludes with selected results and observations of related experiments.

References

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York.

Coello, C. A. C. (1996). *An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design*. PhD thesis, Department of Computer Science, Tulane University, New Orleans, LA.

Coello, C. A. C. (1999a). A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization

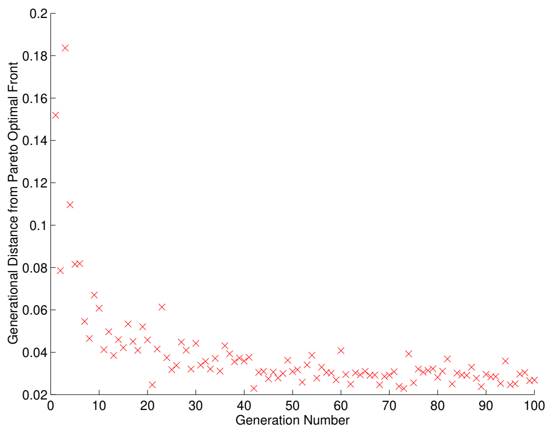

Figure 9: $PF_{current}(t)$'s G

Techniques. *Knowledge and Information Systems*, 1(3):269–308.

Coello, C. A. C. (1999b). List of References on Evolutionary Multiobjective Optimization. Online. Available: http://www.lania.mx/EMOO.

Fonseca, C. M. and Fleming, P. J. (1995). An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16.

Fonseca, C. M. and Fleming, P. J. (1998). Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms – Part I: A Unified Formulation. *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans*, 28(1):26–37.

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

Goldberg, D. E., Korb, B., and Deb, K. (1989). Messy Genetic Algorithms: Motivation, Analysis, and First Results. *Complex Systems*, 3:493–530.

Horn, J. (1997). *Handbook of Evolutionary Computation*, volume 1, chapter Multicriterion Decision Making, pages F1.9:1 – F1.9:15. IOP Publishing Ltd. and Oxford University Press.

Horn, J., Nafpliotis, N., and Goldberg, D. E. (1994). A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In Michalewicz, Z., editor, *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 82–87, Piscataway NJ. IEEE Service Center.

Jackson, R. H. F., Boggs, P. T., Nash, S. G., and Powell, S. (1991). Guidelines for Reporting Results of Computational Experiments – Report of the Ad Hoc Committee. *Mathematical Programming*, 49:413–425.

McClave, J. T., Benson, P. G., and Sincich, T. (1998). *Statistics for Business and Economics.* Prentice Hall, Upper Saddle River, NJ, 7th edition.

Neter, J., Wasserman, W., and Kutner, M. H. (1990). *Applied Linear Statistical Models: Regression, Analysis of Variance, and Experimental Designs.* Irwin, Boston, MA, 3rd edition.

Schott, J. R. (1995). Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Shaw, K. J., Nortcliffe, A. L., Thompson, M., Love, J., Fonseca, C. M., and Fleming, P. J. (1999). Assessing the Performance of Multiobjective Genetic Algorithms for Optimization of a Batch Process Scheduling Problem. In *1999 Congress on Evolutionary Computation*, pages 37–45, Washington, D.C. IEEE Service Center.

Srinivas, N. and Deb, K. (1994). Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248.

Van Veldhuizen, D. A. (1999). *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.* PhD Thesis, AFIT/DS/ENG/99-01, Air Force Institute of Technology, Wright-Patterson AFB.

Van Veldhuizen, D. A. and Lamont, G. B. (1998). Evolutionary Computation and Convergence to a Pareto Front. In Koza, J. R., editor, *Late Breaking Papers at the Genetic Programming 1998 Conference*, pages 221–228, Stanford, CA. Stanford University Bookstore.

Van Veldhuizen, D. A. and Lamont, G. B. (2000). Multiobjective Optimization with Messy Genetic Algorithms. In *To Appear in the Proceedings of the 2000 ACM Symposium on Applied Computing.* ACM.

Wolpert, D. H. and Macready, W. G. (1997). No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Zitzler, E. and Thiele, L. (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.