The Value of Online Adaptive Search: A Performance Comparison of NSGAII, ε-NSGAII and εMOEA

Joshua B. Kollat[1] and Patrick M. Reed[2]

[1] Department of Civil and Environmental Engineering,
The Pennsylvania State University,
406B Sackett Building, University Park,
PA 16802-1408
juk124@psu.edu
[2] Department of Civil and Environmental Engineering,
The Pennsylvania State University,
212 Sackett Building, University Park,
PA 16802-1408
preed@engr.psu.edu

Abstract. This paper demonstrates how adaptive population-sizing and epsilon-dominance archiving can be combined with the Nondominated Sorted Genetic Algorithm-II (NSGAII) to enhance the algorithm's efficiency, reliability, and ease-of-use. Four versions of the enhanced Epsilon Dominance NSGA-II (ε-NSGAII) are tested on a standard suite of evolutionary multiobjective optimization test problems. Comparative results for the four variants of the ε-NSGAII demonstrate that adapting population size based on online changes in the epsilon dominance archive size can enhance performance. The best performing version of the ε-NSGAII is also compared to the original NSGAII and the εMOEA on the same suite of test problems. The performance of each algorithm is measured using three running performance metrics, two of which have been previously published, and one new metric proposed by the authors. Results of the study indicate that the new version of the NSGAII proposed in this paper demonstrates improved performance on the majority of two-objective test problems studied.

1 Introduction

Deb *et al.* [1] identified three primary goals in multiobjective (MO) optimization using evolutionary algorithms (EAs): (1) to obtain good convergence toward the Pareto-optimal solution set, (2) to develop a diverse, or evenly distributed set of non-dominated solutions and to maintain this diversity throughout the entire run of the algorithm, and (3) to achieve the first two goals at the lowest computational cost and in the most efficient manner possible. The third goal can be realized through the development of new techniques to achieve convergence and diversity at the lowest possible computational cost and to develop online adaptive EAs that can assess on-line performance and modify key algorithm parameters throughout a run. The ultimate goal of online adaptation is to enhance algorithmic efficiency, reliability, and ease-of-use.

This study presents alternative techniques by which epsilon-dominance archiving [2], the Nondominated Sorted Genetic Algorithm-II (NSGAII) [3], and parameter adaptation [4] can be combined. These techniques use online performance assessment to adapt population size and to automatically terminate search based on minimal user input and can potentially be integrated into any multiobjective evolutionary algorithm (MOEA) to improve algorithm efficiency, reliability, and ease-of-use. In this paper, section 2 provides a description of the algorithms being compared as well as justification for their selection. Sections 3 and 4 discuss the performance metrics and test problems used in the study. Section 5 presents the results of the simulation in two parts: (1) a comparison of four versions of the ε-NSGAII and (2) a comparison of the best version of the ε-NSGAII identified in the first half of the study to the NSGAII [3] and the εMOEA [1]. Conclusions and potential future research is provided in section 6.

2 Tested Algorithms

The current study is conducted in two parts. The first compares the performance of four versions of the ε-NSGAII, and the second compares the best version of the ε-NSGAII with the NSGAII and the εMOEA.

2.1 Overview of ε-NSGAII

The primary objective of this study is to demonstrate the ε-NSGAII's efficacy at solving multiobjective optimization problems quickly, efficiently, and reliably. The ε-NSGAII is based on the NSGAII, which uses a fast non-dominated sorting approach to classify solutions according to level of non-domination and a crowding distance operator to preserve solution diversity [3]. The ε-NSGAII extends these concepts by adding ε-dominance [2], adaptive population sizing, and self termination to minimize the need for parameter calibration as demonstrated by Reed *et al.* [4].

ε-dominance is a concept whereby the user is able to specify the precision with which they want to obtain the Pareto-optimal solutions to a multiobjective problem, in essence giving them the ability to assign a relative importance to each objective. This is accomplished by applying a grid (sized by user specified ε values) to the search space of the problem. Larger ε values result in a courser grid (and ultimately fewer solutions) while smaller ε values produce a finer grid. The fitness of each solution is then mapped to a box fitness based on the specified ε values. Non-domination sorting is then conducted using each solution's box fitness, and solutions with identical box fitness (i.e., solutions that occur in the same grid block) are compared and those that are dominated within the grid block are eliminated. This results in no more than one non-dominated solution existing in any one grid block, preventing clustering of solutions and promoting a more even search of the objective space. The interested reader can refer to prior work by Laumanns *et al.* [2] and Deb *et al.* [1] for a more detailed description of ε-dominance.

The adaptive population sizing scheme used in the original form of the ε-NSGAII is based on the population sizing theory of Harik *et al.* [5] and the automatic parameterization methodology proposed by Reed *et al.* [4]. The ε-NSGAII uses a series of "connected runs" where small populations are exploited to pre-condition search with successively doubled population sizes. Pre-conditioning occurs by injecting current solutions within the epsilon-dominance archive into the initial generations of larger population runs. For example, the initial population (usually five individuals) is evolved until it is no longer making significant progress. When this occurs, the population size is increased, a subset of archived solutions are injected into the next population, and the search continues. Under the current design of the algorithm, two injection scenarios exist. If the archive size is smaller then the population into which it will be injected, the remaining individuals needed to fill the population are randomly generated. However, if the archive is larger than the subsequent population, then individuals are randomly selected from the archive to fill the population.

The search is terminated using two user-specified criteria: (1) the *intra-run* criterion and (2) the *inter-run* criterion. The intra-run criterion defines the two cases when the current population N will be doubled: (1) if search within *w* generations (termed the lag window) fails to yield a specified percentage increase in the number of archived solutions or (2) the maximum run duration has been reached. The termination of search across all runs (i.e., across all populations used) compares how the archive size changes at the end of two successive runs of the ε-NSGAII. For example, a run that uses a population of N to evolve an ε-nondominated set composed of A individuals will be compared to a second run that used a population of 2N to evolve an ε-nondominated set of K individuals. The results of these runs are used in equation (1), to define which of the two following courses of action will be taken: (1) population size is again doubled, resulting in 4N individuals to be used in an additional run of the ε-NSGA-II or (2) the algorithm stops to allow the user to assess if the ε-nondominated set has been quantified to sufficient accuracy. Δ was set to 10-percent for this study as recommended by Reed et al. [4].

$$\textit{if } \Delta < \left(\frac{|K - A|}{A} \right) 100 \textit{ then double N and continue search} \qquad (1)$$

else stop search

The solutions obtained in the archive at the end of the final run represent a sufficient approximation of the true Pareto front based on user defined accuracy goals.

2.1.1 Improving Termination. This study investigates improvements in the way the ε-NSGAII adaptively sizes its population and self-terminates. The initial version of the algorithm based its inter- and intra-run termination strictly on changes in the number of solutions stored in the epsilon dominance archive. However, a shortcoming to this method exists when the algorithm finds a number of new solutions dominating the same number of solutions in the archive. The algorithm could terminate in this case because the quantity of solutions in the archive has remained constant. By this scenario the algorithm may be making significant progress while the archive size remains constant.

To ameliorate this issue, a second version of the algorithm has been developed which accounts for solution quality in the termination criteria. This is accomplished by monitoring both the archive size and the number of solutions that have been replaced when the archive is updated. The solutions that differ will be improved solutions since a new solution will only be accepted into the archive if it ε-dominates one or more existing solutions. In addition, since the computational cost of a quality comparison is higher than simply checking for a change in solution quantity, the quality comparison is only conducted if the quantity comparison results in insufficient improvement. With the improved termination criteria, the ε-NSGAII will continue the search and seek better convergence to the true Pareto set.

2.1.2 Improving Population Sizing and Injection. The current population doubling scheme used by the ε-NSGAII has a possible flaw in that it lacks a bound on population growth. This issue is particularly important for high-dimensional, difficult problems with large Pareto optimal sets. This study investigates this issue by analyzing two alternative population sizing schemes and by comparing their performance to the prior population doubling methodology.

The first scheme bases the size of the population on the size of the epsilon dominance archive by maintaining a 25% injection rate. For example, if 25 ε-nondominated solutions exist in the archive at the end of a run, the subsequent population size will be four times the archive size, or 100 individuals. This scheme bounds the maximum size of the population to four times the number of solutions that exist at the user specified ε resolution. Theoretically, this approach allows population sizes to increase or decrease, and in the limit when the epsilon dominance archive size stabilizes, the ε-NSGAII's "connected runs" are equivalent to time continuation [6]. Since this method guarantees that new individuals will be introduced at the end of each run by 25% injection, the chances of escaping local nondominated fronts are greatly improved.

The second proposed population sizing scheme attempts to merge both the old doubling scheme and the new scheme based on 25% injection. This scheme, hereafter referred to as the adaptive scheme, bases the subsequent population size on the smaller of either population doubling, or 25% injection. This method typically provides a slower population growth rate while also bounding the size of the population using the 25% injection scheme.

2.2 Other Algorithms

The ε-NSGAII's performance has been tested in this study relative to the NSGAII [3] and the εMOEA [1]. The NSGAII was chosen for comparison since it is the original algorithm from which the ε-NSGAII was derived. The εMOEA is a steady state MOEA that co-evolves both an evolutionary algorithm population and an archive population by randomly mating individuals from the population and the archive to generate new solutions [1]. The εMOEA was chosen for comparison because it uses the concept of ε-dominance to preserve solution diversity and its effectiveness on the test problems examined in this study has been demonstrated previously [1].

3 Metrics Used to Assess Performance

The performance of the algorithms tested in this study is assessed using a convergence metric and a diversity metric proposed by Deb and Jain [7] as well as a combined convergence/diversity metric proposed by the authors. The convergence metric proposed by Deb and Jain [7] measures the average Euclidean distance between the non-dominated solutions generated by an algorithm and the set of Pareto-optimal solutions. Deb and Jain's diversity metric attempts to alleviate many of the shortcomings associated with previously proposed diversity metrics. This metric measures diversity by projecting algorithm-based solutions and reference solutions onto an M-1 dimensional plane where M is the number of objectives. The projected solutions are then compared to the projected reference solutions in terms of their distribution across the objective space. The metric value is determined by favoring well distributed solutions with a high weight and assigning low weights to clustered solutions. In this study, a new metric is proposed that effectively measures both convergence and diversity. This metric uses the concept of ε-dominance to measure the percentage of solutions that have been found within a user specified ε distance of a reference set. Steps in calculating the metric are as follows:

1. Apply ε-dominance to a reference solution set according to user specified ε values.
2. At each generation, match each solution generated by the algorithm to its corresponding ε-nondominated reference set solution. Each reference solution can only have one algorithm solution associated with it. If more than one solution exists within ε of the reference solution, then the closest solution in terms of Euclidean distance is chosen. This accounts for overlapping ε regions in the reference set and frees up the additional solutions to be matched with other ε-nondominated reference solutions.
3. Each ε-nondominated reference solution that has a corresponding algorithm solution receives a score of one, while each reference solution that has no corresponding algorithm solution receives a score of zero. The metric is then calculated according to equation (2).

$$\varepsilon(P^{(t)}) = \sum_{i=1}^{n} h_i \Big/ n \tag{2}$$

   where h_i is one or zero for each solution i in the ε-nondominated reference set and n is the total number of solutions in the reference set.

This metric measures convergence by accounting for solutions that have converged to within ε of a reference set. Diversity is accounted for by including only one solution for each ε-nondominated reference solution, regardless of the existence of additional solutions in that ε-block. This ensures that clustered solutions do not contribute to the calculation of the metric.

4 Test Problem Suite

The relative performance of the algorithms' was assessed using a suite of two-objective test problems that have been commonly employed in prior literature. Each of the test problems are discussed briefly below. The naming convention used in referring to the test problems is adopted from [8]. A more detailed description of the construction of these test problems can be found in [8] and [9].

- T_1: a convex, thirty variable test problem (m = 30 *and* $x_i \in [0, 1]$) with the Pareto front formed at $g(\mathbf{x}) = 1$.
- T_2: a 30 variable test problem (m = 30 *and* $x_i \in [0, 1]$) with a concave Pareto front formed at $g(\mathbf{x}) = 1$.
- T_3: a 30 variable test problem (m = 30 *and* $x_i \in [0, 1]$) with a discontinuous Pareto front formed at $g(\mathbf{x}) = 1$.
- T_4: a multi-modal (21^9 local fronts), convex, ten variable test problem (m = 10, $x_1 \in [0, 1]$, *and* $x_2,\ldots,x_m \in [-5, 5]$) with the Pareto front formed at $g(\mathbf{x}) = 1$.
- T_6: a non-uniform, non-convex, ten variable test problem (m = 10 *and* $x_i \in [0, 1]$) with the Pareto front formed at $g(\mathbf{x}) = 1$.

5 Simulation Results

5.1 ε-NSGAII Version Comparison

The first portion of the study compared four versions of the ε-NSGAII to identify the impacts of population sizing and termination criteria on solving the two-objective test problem suite. The versions of the ε-NSGAII compared are identified as the ε-NSGAIIv1 (the original version of the algorithm) and the ε-NSGAIIv2 (a version in which the termination criteria have been changed to account for solution quality). The ε-NSGAIIv2 was further tested using population doubling, 25% injection, and the adaptive population sizing scheme identified previously in section 2.1.2. Each version of the algorithm used an initial population size of n = 5. Each population was run for a maximum of 250 generations. Based on Deb's recommendations, the recombination scaling factor η_c was set equal to 15 and the probability of crossover was set equal to 1.0. The polynomial mutation operator used a scaling factor η_m = 20 and the probability of mutation was set equal to $1/i$, where i is the number of real-coded decision variables. The inter- and intra-run termination criteria were each set to 10%. The ε-NSGAIIv1 and the ε-NSGAIIv2 using population doubling and adaptive population sizing used an initial lag window of 50 generations for the first three runs and a final lag window of 25 thereafter (lag window is the number of generations across which the algorithm is checked for termination conditions). The ε-NSGAIIv2 that used 25% injection used an initial lag window of 50 for the first run and a lag window of 10 thereafter since the archive tends to grow more quickly by this approach. Each

of the performance metrics used a reference set of solutions containing approximately 1000 points. ε values were chosen for each test problem to result in approximately 100 solutions, and the grid sizes, G, used to compute the diversity metric, were obtained using 1/ε (Table 1). The analysis was conducted using 50 random seeds and all results reflect all random seeds.

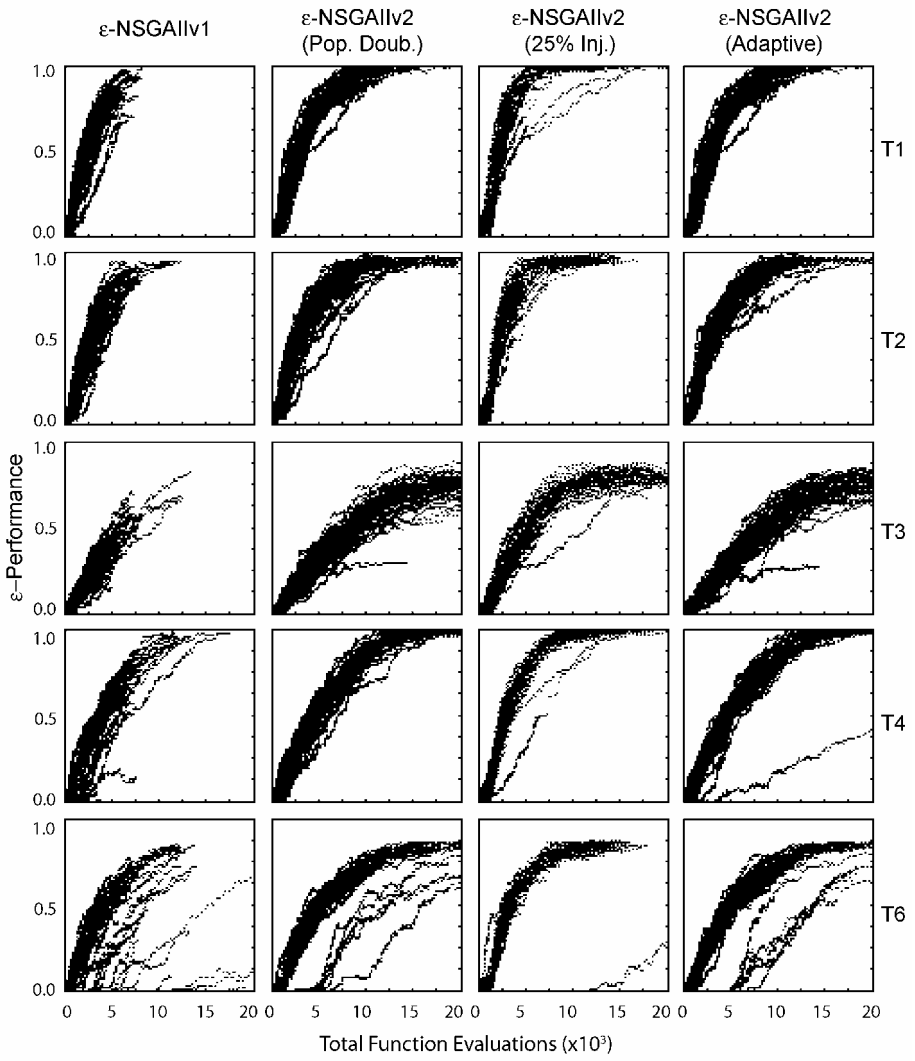

Fig. 1. Running ε-performance metric results for each version of the ε-NSGAII examined on test problems T_1, T_2, T_3, T_4, and T_6. Results represent 50 random seeds plotted as ε-performance versus total function evaluations in thousands. This figure can be cross-referenced with Table 2 to determine final mean ε-performance

Table 1. ε values and G (the number of grid blocks used in the diversity metric) for each of the two-objective test problems used in the study. ε values were chosen to generate approximately 100 solutions for each test problem

Test Problem	ε_1	ε_2	G
T1	0.0075	0.0075	133
T2	0.0076	0.0076	131
T3	0.00261	0.00261	383
T4	0.0074	0.0074	135
T6	0.0067	0.0067	149

Table 2. Final results comparing four versions of the ε-NSGAII (ε-NSGAIIv1, ε-NSGAIIv2 using population doubling, ε-NSGAIIv2 using 25% injection, and ε-NSGAIIv2 using adaptive population sizing). Included are the means and standard deviations for total solutions found, total function evaluations, ε-performance, convergence, diversity, and CPU time in seconds for 50 random seeds. The best result in each category is indicated by bold-underlined text

	MOEA	Sol.		NFE		ε-Perf ($x10^{-4}$)		Conv. ($x10^{-6}$)		Diver. ($x10^{-4}$)		Time (s) ($x10^{-3}$)	
		Avg.	σ	Avg.	σ	Avg.	σ	Avg.	σ	Avg.	σ	Avg.	σ
T1	v1	90	15	**5550**	**1416**	8212	1536	401	65	7581	1193	**213**	**64**
	v2 - Doub.	96	3	10540	3270	9272	501	**378**	**24**	8030	202	500	220
	v2 - 25% Inj.	**99**	**0**	12738	2081	**9855**	**71**	389	25	**8226**	**31**	1137	183
	v2 - Adapt.	96	3	10540	3270	9273	501	**378**	**24**	8030	202	492	224
T2	v1	88	20	**6085**	**2183**	7648	1937	**397**	75	6927	1524	**233**	**117**
	v2 - Doub.	98	2	14425	5819	9147	514	402	**21**	7606	208	859	665
	v2 - 25% Inj.	**99**	**0**	12585	1400	**9451**	**142**	406	22	**7683**	**88**	1153	163
	v2 - Adapt.	97	3	12538	5415	8986	578	406	25	7517	235	667	458
T3	v1	80	18	**6067**	**2227**	4612	1342	608	1393	6783	1502	**209**	**112**
	v2 - Doub.	96	11	55010	27249	7392	740	**269**	23	8170	943	10714	12166
	v2 - 25% Inj.	**98**	**5**	19259	3894	**7832**	**377**	271	**18**	**8390**	**405**	1809	449
	v2 - Adapt.	97	11	39677	9084	7598	769	270	23	8308	902	3364	1043
T4	v1	70	38	**6813**	**4181**	5758	3735	11998	49703	5884	3126	**221**	**182**
	v2 - Doub.	97	4	18270	9266	9266	952	**383**	**24**	8042	246	1086	865
	v2 - 25% Inj.	**100**	**1**	14392	9796	**9796**	**97**	384	24	**8175**	**43**	1214	225
	v2 - Adapt.	97	4	18878	9246	9246	963	399	77	8035	268	1099	869
T6	v1	93	6	**9926**	**4720**	5102	2744	189	250	6385	342	**394**	340
	v2 - Doub.	96	5	18225	11707	7752	794	454	**112**	6620	280	1293	2430
	v2 - 25% Inj.	**100**	**1**	14295	3501	**8363**	**258**	**447**	133	**6809**	**46**	1175	**284**
	v2 - Adapt.	97	4	16532	7069	7704	845	471	160	6612	275	797	671

The results of the ε-performance metric versus total function evaluations in thousands for 50 random seeds and each test problem are shown in Figure 1. These plots reflect the spread of results for all random seeds across all generations. Based on the scatter plots in Figure 1, the ε-NSGAIIv1 performs the poorest in terms of ε-performance as is evidenced by its premature termination for many test problems and

random seeds. In addition, there is a large spread in ε-performance across random seeds for many of the test problems. Examining the scatter plots of the three versions of the ε-NSGAIIv2 for test problem T_1, the version using 25% injection converges the fastest with high reliability across random seeds. The reliability of the algorithm can be assessed by examining the spread of the scatter plot (e.g., a tighter plot indicates high reliability). This is also the case for test problems T_2, T_3, and T_4. For test problem T_6, the ε-NSGAIIv2 using 25% injection clearly outperforms the other three versions of the algorithm.

Table 2 presents a summary of the mean results and standard deviations for the performance of each version of the ε-NSGAII on each test problem using 50 random seeds. Results displayed include total solutions found, total function evaluations, ε-performance, convergence, diversity, and the CPU time in seconds (i.e. wall clock time). The best performance in each category is indicated by bold-underlined text. The ε-NSGAIIv2 using 25% injection found the most solutions, achieved the highest ε-performance and the highest diversity with the lowest standard deviations for all test problems. In terms of total function evaluations, the ε-NSGAIIv1 outperformed the other versions on all test problems, but as a result of premature termination. When examining the convergence metric, the ε-NSGAIIv2 using 25% injection outperformed the ε-NSGAIIv1 in all test problems except T_2. In terms of CPU time, but neglecting results from the ε-NSGAIIv1 due to premature termination, CPU time was sacrificed for improved performance in test problems T_1, T_2, T_4, and T_6. However, there is a large improvement in CPU time for test problem T_3. On average, the ε-NSGAIIv2 using 25% injection was the most robust at solving the two-objective test problems examined in this study. Based on these findings, the performance of the ε-NSGAII using 25% injection was subsequently compared to NSGAII and εMOEA.

5.2 Comparing NSGAII, ε-NSGAII, and εMOEA

For the second half of the study, the second version of ε-NSGAII that used 25% injection (hereafter referred to as ε-NSGAII for simplicity) as its population sizing scheme was chosen for comparison to the NSGAII and the εMOEA, as it was found to be the most robust version of the algorithm. The same two-objective test problems (T_1, T_2, T_3, T_4, and T_6) are again used for comparison. Again, the analysis was performed using 50 random seeds and all results shown reflect all 50 random seeds. However, since the ε-NSGAII self-terminates, in order to make a fair algorithmic comparison, the NSGAII and the εMOEA were set to run for the same mean number of function evaluations that the ε-NSGAII required to completely solve each test problem and self terminate. This methodology provides a snapshot in time of the performance of the other algorithms at the time that the ε-NSGAII terminates on average. The NSGAII was parameterized using a population size of 100 individuals, uniform crossover with probability $P_c = 1.0$, and probability of mutation, $P_m = 1/i$ where i is the number of real coded variables in the test problem. The εMOEA was parameterized using the same settings as previously mentioned for NSGAII and the ε settings and grid resolution, G, were set identically to those values contained in Table 1 for each test problem. However, as noted above, the number of generations that each algorithm was set

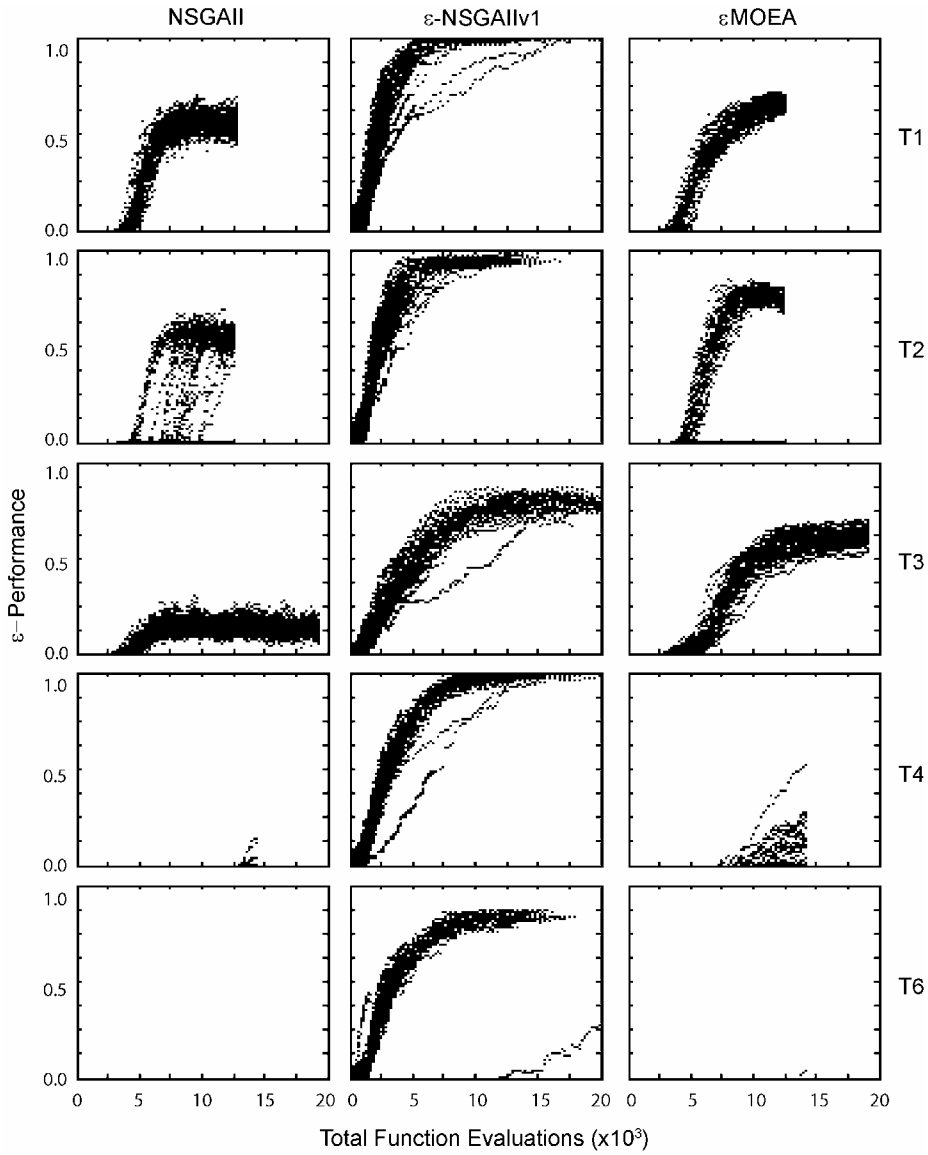

Fig. 2. Running ε-performance metric results for the NSGAII, the ε-NSGAII, and the εMOEA examined on test problems T_1, T_2, T_3, T_4, and T_6. Results represent 50 random seeds plotted as ε-performance versus total function evaluations in thousands. This figure can be cross-referenced with Table 4 to determine final mean ε-performance

Table 3. Average total function evaluations required by the ε-NSGAII to solve each test problem, and the corresponding number of generations that the NSGAII and the εMOEA were set to run for each test problem

Test Problem	ε-NSGAII Avg. NFE	NSGAII Gen.	εMOEA Gen.
T1	12738	128	6369
T2	12585	126	6293
T3	19259	193	9630
T4	14392	144	7196
T6	14295	143	7148

Table 4. Final results comparing the NSGAII, the ε-NSGAII and the εMOEA. Each algorithm was run for the same number of function evaluations. Results include mean and standard deviations for total solutions found, ε-performance, convergence, diversity, and CPU time in seconds for 50 random seeds. The best result in each category is indicated by bold-underlined text

	MOEA	Sol.		ε-Perf ($x10^{-4}$)		Conv. ($x10^{-6}$)		Diver. ($x10^{-4}$)		Time (s) ($x10^{-3}$)	
		Avg.	σ	Avg.	σ	Avg.	σ	Avg.	σ	Avg.	σ
T1	NSGAII	98	0	5455	462	561	79	7153	146	920	**46**
	ε-NSGAII	**99**	**0**	**9855**	**71**	**389**	**25**	**8226**	**31**	1137	183
	εMOEA	99	2	6659	251	543	65	8132	235	**846**	53
T2	NSGAII	62	45	3531	2618	**246**	187	4581	3389	982	**56**
	ε-NSGAII	**99**	**0**	**9451**	**142**	406	**22**	7683	**88**	1153	163
	εMOEA	98	14	7190	1071	698	127	**7820**	1112	**776**	100
T3	NSGAII	92	**2**	1301	365	659	2218	6936	**270**	1538	697
	ε-NSGAII	**98**	5	**7832**	**377**	271	**18**	**8390**	405	1809	449
	εMOEA	95	8	6172	363	**268**	19	8101	605	**1198**	**84**
T4	NSGAII	51	44	34	178	92637	113472	3086	273	1102	57
	ε-NSGAII	**100**	**1**	**9796**	**97**	**384**	**24**	**8175**	**43**	1214	225
	εMOEA	45	20	680	1135	20158	37345	3500	1665	**359**	**33**
T6	NSGAII	96	1	0	0	24429	3966	5703	130	1004	**54**
	ε-NSGAII	**100**	**1**	**8363**	**258**	**447**	**133**	**6809**	**46**	1175	284
	εMOEA	95	4	9	63	10074	1775	6497	192	**862**	76

to run differed depending on the results of the ε-NSGAII. For the εMOEA, the total iterations was determined by dividing the average total function evaluations required by the ε-NSGAII by two, since each iteration results in two function evaluations. Table 3 provides a summary the generational settings of each algorithm based on the results of the ε-NSGAII from part one of the study.

The results of the ε-performance metric (measured as the percentage of solutions that fall within ε of each objective) versus total function evaluations for all 50 random seeds and each test problem are shown in Figure 2. When the NSGAII and the εMOEA are given the same opportunity to solve each test problem (i.e., they are each

permitted the same number of function evaluations), the performance of the ε-NSGAII is superior for each test problem. In fact, for test problems T_4, and T_6, NSGAII and εMOEA achieve little to no measure of ε-performance. This is most likely the result of an insufficient number of function evaluations (based on the performance of the ε-NSGAII) for solving these particular test problems.

Table 4 shows the final mean results for the NSGAII, the ε-NSGAII, and the ε-MOEA. The ε-NSGAII finds the most solutions in all test problems except T_1. The ε-NSGAII achieves the highest ε-performance and the best convergence for all test problems except T_2, and T_3 where the NSGAII and the εMOEA are superior. The ε-NSGAII achieves the highest diversity in all test problems except T_2 where it is slightly outperformed by the εMOEA. Average CPU time is again sacrificed to achieve higher performance in all other measures by the ε-NSGAII. In summary, the ε-NSGAII showed improved performance above the NSGAII and the εMOEA in all respects except CPU time for test problems T_1, T_4, and T_6, and achieved comparable performance on test problems T_2 and T_3.

6 Conclusions and Future Work

Adaptive population sizing based on epsilon archive size produces the most robust and reliable performance for the ε-NSGAII algorithm in solving the two-objective test problem suite used in this study. In addition, the ε-NSGAII outperforms its predecessor, the NSGAII, and the steady state algorithm, the εMOEA, on test problems T_1, T_4, and T_6 and performed comparably to the NSGAII and the εMOEA on test problems T_2 and T_3 in terms of solution quality and reliability.

Future research in this area will include a comparative study using higher dimensional test problems. In addition, the affects of integrating adaptive population sizing and termination into other MOEAs will be explored. The authors are currently working on two MOEA comparative studies involving real-world applications to long-term groundwater monitoring and integrated hydrologic model calibration. The performance of the NSGAII, the ε-NSGAII, the εMOEA, the SPEA2, and the MOSCEM applied to these real-world problems is currently being explored. Reed and Devireddy [10, 11] have attained up to a 90% reduction in computational costs over the NSGAII when applying ε-dominance archiving and automatic parameterization to long-term groundwater monitoring problems. Preliminary results of our studies indicate that the ε-NSGAII's ability to adaptively size population and self-terminate while allowing the user to specify the precision of the resulting Pareto-optimal solutions eliminates much of the trial and error analysis associated with traditional MOEA parameterization.

References

1. K. Deb, M. Mohan, S. Mishra. A Fast Multi-objective Evolutionary Algorithm for Finding Well-Spread Pareto-Optimal Solutions. *KenGAL*, Report No. 2003002. Indian Institute of Technology, Kanpur, India, 2003.

2. M. Laumanns, L. Thiele, K. Deb, E. Zitzler, Combining Convergence and Diversity in Evolutionary Multiobjective Optimization. *Evolutionary Computation*, 10(3):263-282, 2002.
3. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Trans. Evol. Computation*, 6(2):182-197, 2002.
4. P. Reed, B.S. Minsker, D.E. Goldberg, Simplifying Multiobjective Optimization: An Automated Design Methodology for the Nondominated Sorted Genetic Algorithm-II. *Water Resources Research*, 39(7):1196-1201, 2003.
5. G.R. Harik, E. Cuantu-Paz, D.E. Goldberg, B.L. Miller. The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations. in *Proceedings of the 1997 IEEE Conference on Evolutionary Computation*, Piscataway, NJ, IEEE Press. pages 7-12, 1997.
6. D.E. Goldberg, *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA, 2002.
7. K. Deb, S. Jain. Running Performance Metrics for Evolutionary Multi-Objective Optimization. *KanGAL*, Report No. 2002004. Indian Institute of Technology, Kanpur, India, 2002.
8. E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):125-148, 2000.
9. K. Deb, Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. *Evolutionary Computation*, 7(3):205-230, 1999.
10. P. Reed, V. Devireddy, *Groundwater Monitoring Design: A Case Study Combining epsilon-Dominance Archiving and Automatic Parameterization for the NSGA-II*, in *Applications of Multi-Objective Evolutionary Algorithms*, C. Coello-Coello, Editor. World Scientific, New York. 2004 (In Press).
11. P. Reed, V. Devireddy. Using Interactive Archives in Evolutionary Multiobjective Optimization: Case Studies for Long-Term Groundwater Monitoring Design. in *The International Environmental Modeling and Software Society Conference*, Osnabruck, Germany. 2004 (In Press).