AMGA: An Archive-based Micro Genetic Algorithm for Multi-objective Optimization

Santosh Tiwari*
Department of Mechanical Engineering
Clemson University, Clemson, SC, USA
stiwari@clemson.edu

Patrick Koch
Engineous Software Inc.
2000 CentreGreen Way, Cary, NC 27516, USA
patrick.koch@engineous.com

Georges Fadel
Department of Mechanical Engineering
Clemson University, Clemson, SC, USA
fgeorge@clemson.edu

Kalyanmoy Deb
Department of Mechanical Engineering
Indian Institute of Technology
Kanpur, India
deb@iitk.ac.in

ABSTRACT

In this paper, we propose a new evolutionary algorithm for multi-objective optimization. The proposed algorithm benefits from the existing literature and borrows several concepts from existing multi-objective optimization algorithms. The proposed algorithm employs a new kind of selection procedure which benefits from the search history of the algorithm and attempts to minimize the number of function evaluations required to achieve the desired convergence. The proposed algorithm works with a very small population size and maintains an archive of best and diverse solutions obtained so as to report a large number of non-dominated solutions at the end of the simulation. Improved formulation for some of the existing diversity preservation techniques is also proposed. Certain implementation aspects that facilitate better performance of the algorithm are discussed. Comprehensive benchmarking and comparison of the proposed algorithm with some of the state-of-the-art multi-objective evolutionary algorithms demonstrate the improved search capability of the proposed algorithm.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristics

General Terms

Algorithms

*Address all correspondence to this author

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

Keywords

Multi-objective optimization, evolutionary algorithms, micro-genetic algorithm, diversity preservation

1. INTRODUCTION

Multi-objective optimization has become main-stream in recent years and many algorithms to solve multi-objective optimization problems have been suggested. The use of multi-objective optimization in industry has been accelerated by the availability of faster processing units and the computational analysis models for various engineering problems and disciplines. Multi-objective optimization algorithms, especially those based on evolutionary principles, have seen wide acceptability because most engineering problems are NP-hard [1] and therefore a quick computation of approximate solutions is often desirable. Evolutionary algorithms (EAs) are adaptive search techniques inspired from nature and their working principle is based on Darwin's theory of survival-of-the-fittest [2, 3]. The adaptive nature of EAs can be exploited to design optimization algorithms by designing suitable variation operators and an appropriate fitness function. The Genetic algorithm (GA) [4, 5, 6, 7] is one of the evolutionary techniques that has been successfully used as an optimization tool. Typically, a GA works with a population (a set of solutions) instead of a single solution (individual). This property of a GA makes it an ideal candidate for solving multi-objective optimization problems where the outcome (in most cases) is a set of solutions rather than a single solution. The population approach of a GA also makes it resilient to premature convergence, thereby making it a powerful tool for handling highly non-linear and multi-modal functions. Some of the notable efforts in designing multi-objective evolutionary algorithms (MOEAs) are Strength Pareto Evolutionary Algorithm (SPEA2) [8], Pareto-Envelope Based Selection Algorithm (PESA-II) [9], Non-dominated Sorting Genetic Algorithm (NSGA-II) [10], Neighborhood Cultivation Genetic Algorithm (NCGA) [11], Dynamic Multi-objective Evolutionary Algorithm (DMOEA) [12], Improved Strength Pareto

Evolutionary Algorithm 2 (SPEA2+) [13], Intelligent Multi-Objective Evolutionary Algorithm (IMOEA) [14], ϵ-Multi-objective Evolutionary Algorithm (ϵ-MOEA) [15], OmniOptimizer [16], and Fast Pareto Genetic Algorith (FastPGA) [17]. A historical and comprehensive survey of MOEAs can be found in [7].

In this paper, we propose a novel multi-objective evolutionary algorithm geared towards solving large optimization problems. The design of the proposed algorithm has been motivated primarily by the fact that in most optimization scenarios; almost the entire time is spent by the analysis routines and the user does not have the computational resources to perform a large number of function evaluations. The actual time spent by the optimization algorithm in performing selection, crossover, mutation, and diversity preservation is often negligible compared to the total simulation time. Another important guiding principle that has shaped the design of the proposed algorithm is the fact that, the user is often satisfied if a good enough non-dominated solution set is obtained. In most engineering applications, once a reasonable solution quality is achieved (often within 1% of the desired function value), a global optimizer (e.g. a genetic algorithm) is replaced by a local optimizer (e.g. a gradient based method) which has a faster convergence rate and can generate solutions with higher accuracy. Often the desired function value is not known; in such cases, the optimizer is terminated when the convergence rate falls below a certain threshold. The proposed algorithm also benefits from the existing literature in that it borrows several concepts like formulation for crossover, mutation, two-tier fitness assignment mechanism, ranking strategy, preservation of elite and diverse solutions etc from the existing MOEAs. Unless otherwise stated, the concept of Pareto domination [18] has been used for comparing two solutions. The optimization problem that the proposed algorithm attempts to solve is formally stated in equation 1.

$$\begin{array}{ll} \text{Minimize} & (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})) \\ \text{Subject to} & g_j(\mathbf{x}) \leq 0, \quad j = 1, 2, \ldots, J \\ & h_k(\mathbf{x}) = 0, \quad k = 1, 2, \ldots, K \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, \quad i = 1, 2, \ldots, N \end{array} \tag{1}$$

The usual definition of Pareto-domination [18] that is used in the present context is as follows: A feasible solution a dominates another feasible solution b for a M-objective minimization problem, if the following conditions are met:

1. $f_i^{\mathsf{a}} \leq f_i^{\mathsf{b}}$ for all $i = 1, 2, \ldots, M$,
2. $f_i^{\mathsf{a}} < f_i^{\mathsf{b}}$ for at least one $i \in \{1, M\}$.

The remainder of this paper is organized as follows. In section II, the proposed optimization algorithm is described. In section III, benchmark problems, algorithm tuning parameters, and simulation results are presented. In section IV, inferences from the simulation run are presented. Finally in section V, a brief conclusion of the study is presented.

2. DESCRIPTION OF THE OPTIMIZATION ALGORITHM

The proposed algorithm is an evolutionary optimization algorithm and relies on genetic variation operators for creating new solutions. The generation scheme deployed in the proposed algorithm can be classified as generational since during a particular iteration (generation), only solutions created before that iteration take part in the selection process. The algorithm however generates a small number of new solutions at every iteration and therefore it can also be classified as an almost steady-state genetic algorithm. The algorithm works with a small population size and maintains an external archive of *good* solutions obtained. At every iteration, a small number of solutions are created using the genetic variation operators. The newly created solutions are then used to update the archive. We refer to the proposed algorithm as *Archive based Micro Genetic Algorithm* (AMGA) owing to the fact that it works with a very small population size and uses an external archive to maintain its search history. It is recommended to use a large size for the archive to obtain a large number of non-dominated solutions. The size of the archive determines the computational complexity of the proposed algorithm, however for computationally expensive optimization problems, the actual time taken by the algorithm is negligible as compared to the time taken by the analysis routines. The parent population is created from the archive and binary tournament selection is performed on the parent population to create the mating population. The design of the algorithm is independent of the encoding of the variables and thus the proposed algorithm can work with almost any kind of encoding (so long as suitable genetic variation operators are provided to the algorithm). The algorithm uses the concept of Pareto ranking borrowed from NSGA-II [10] and is based on a two-tier fitness mechanism. The pseudo-code of the proposed algorithm (AMGA) is as follows.

```
The AMGA pseudo-code:
1  Begin
2    Generate initial population.
3    Evaluate initial population.
4    Update the archive (using the
       initial population).
5    repeat
6      Create parent population from
         the archive.
7      Create mating pool from the
         parent population.
8      Create off-spring population from
         the mating pool.
9      Evaluate the off-spring population.
10     Update the archive (using the
         off-spring population).
11   until (termination)
12   Report desired number of solutions
       from the archive.
13 End
```

Although the above mentioned pseudo-code for AMGA is very simple, it clearly separates all the conceptual steps of the algorithm. The above pseudo-code encapsulates the working of the algorithms like NSGA-II [10] and SPEA2 [8] in that, both the algorithms obey the above mentioned pseudo-code. The parent population is created from the archive using a strategy similar to environmental selection (used in SPEA2). The creation of the mating pool is based on binary tournament selection and is similar (but not identical) to the one used in NSGA-II. Any genetic variation operator can be used to create the off-spring population. The

strategy used to update the elite population (archive) relies on the domination level of the solutions, diversity of the solutions, and the current size of the archive and is based on the non-dominated sorting concept borrowed from NSGA-II. In order to reduce the number of function evaluations per generation, AMGA uses a small size for the parent population and the mating pool. The parent population is created from the archive using only the diversity information of genotypes (variables). Using an external archive that stores a large number of solutions provides useful information about the search space as well as tends to generate a large number of non-dominated points at the end of the simulation. We now discuss each step in the pseudo-code of the AMGA in greater detail.

2.1 Generation of the initial population

The initial population (P_0) can be generated in multiple ways. It can be either generated randomly such that all the variables are inside the search space or can be uniformly sampled. We choose to create the initial population using Latin hypercube (LH) sampling [19] coupled with unbiased Knuth shuffling since it gives a good overall random (unbiased) distribution of the population in the genotypic (variable) space and does not require any objective or constraint function evaluation. Let the size of the parent population be N and the number of variables be n. Let the lower bound of variable i be l_i and the upper bound be u_i. To generate a LH sample, the variable range is divided into N equal segments of size $\frac{u_i - l_i}{N}$ each, and a real random number is generated in each segment. Then a random permutation of integers from 1 to N is generated and the individual with index i is assigned a value located at $\pi(i)^{th}$ position in the permutation. This process is repeated for all the variables. This ensures that the resultant population spans the entire genotypic space, is sufficiently random and is free from any biases. Further details of this procedure can be found in [16].

2.2 Approach used for updating the archive

The archive maintains a pool of good solutions obtained during the search process. Let the maximum allowed size for the archive be A and let the size of the parent population be P and the size of the mating pool be M. The size of the archive is taken to be much larger than the size of the parent population or the size of the mating pool ($A \gg P, M$). Initially the archive is empty and hence in step 4 of the AMGA pseudo-code, the contents of the initial population are simply copied to the archive. During the main iteration loop of the AMGA (steps 5 through 11), the archive is filled with the new solutions created at every iteration unless the size of the mating pool is greater than the remaining slots in the archive. At this stage, not all new solutions generated can be accommodated in the archive. The current archive population A_t (at iteration t) and the off-spring population C_t are combined into U_t. Then A_{t+1} is created from U_t using the non-dominated sorting procedure employed in NSGA-II. It has been shown that the crowding distance operator used in NSGA-II does not work well with more than two objectives [20, 21]. The diversity preservation technique used in NSGA-II is replaced by an improved method proposed by Kukkonen et al. [21]. We propose further refinements to the crowding distance computation and the diversity preservation technique proposed by Kukkonen et al. in the next subsection. AMGA uses the diversity information at two places, which are as follows.

1. *Non-dominated sorting*: During non-dominated sorting, when selection is to be performed between two solutions having the same rank, the diversity information is used. In Kukkonen et al. [21], a novel method based on efficient nearest neighbor search has been proposed to perform the pruning of crowded solutions. We use this method during the non-dominated sorting process.

2. *Binary tournament selection*: During the binary tournament selection, when the primary fitness (the domination level or the rank) is the same for the two solutions, secondary fitness based on diversity is used. The method based on efficient nearest neighbor search cannot be used, since it does not assign a numerical value to a solution that quantifies its diversity. The crowding distance operator used in NSGA-II assigns a diversity metric to every solution in the non-dominated set and hence this operator (with improvements) is used to assign the secondary fitness.

2.2.1 *Diversity preservation used in non-dominated sorting*

For the case of non-dominated sorting, we refer to Kukkonen et al. [21]. The authors have proposed a novel method for pruning of crowded solutions based on the efficient nearest neighbor search. At every iteration, two nearest neighbors are found. From the two nearest neighbors found, the one with the smaller value for the Euclidean distance to the second nearest neighbor is deleted. If the second nearest neighbors also have the same Euclidean distance, the third nearest neighbors (and so on) are searched. If one of the solutions from the nearest neighbor pair happens to be an extremal point of the Pareto-optimal frontier, then the other one is deleted from the set. We propose a slight modification to the main algorithm. Consider the Pareto-optimal front in Figure 1. O is the origin and minimization for all three objectives is assumed. In the algorithm proposed by Kukkonen et. al. [21], all the solutions having a minimum or maximum value for any objective are assigned an infinite value for the diversity. All the solutions that lie on the line segment AB have the minimum value of the objective function corresponding to the Z axis. Similarly all the solutions that lie on the line segments AC and BC have minimum values for the objectives corresponding to Y and X axes respectively. Ideally only points A, B and C (or the solution closest to these points) should get an infinite value for the diversity. We therefore propose that only the solutions with the maximum (or worst) objective value be assigned an infinite value for the diversity (minimization for all the objectives is assumed). This modification would not change the distribution of the solutions on the Pareto-optimal front, rather it would ensure the maximum spread of the obtained solutions.

2.2.2 *Diversity computation used in fitness assignment*

For the case of fitness assignment, we refer to the crowding distance operator proposed in NSGA-II [10]. We propose two refinements to the algorithm proposed in [10].

Consider Figure 2. Suppose that the solution B is not an extreme solution and has both left and right neighbors

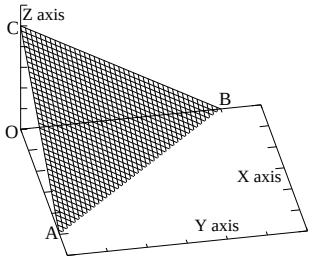

Figure 1: Pareto-optimal front in 3 dimensions

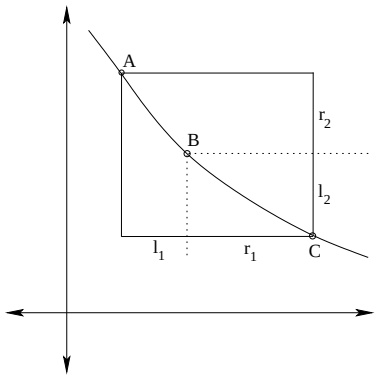

Figure 2: Crowding distance computation

A and C respectively. The usual formulation (proposed in NSGA-II) for the crowding distance (CD) computation gives $\text{CD}(B) = l_1 + r_1 + l_2 + r_2$. Larger the value of $\text{CD}(B)$, more diverse is the solution. It should be noted that $\text{CD}(B)$ depends only on the location of A and C and not on the location of B (so long as B is inside the bounding box defined by A and C). Ideally, it is desirable that solution B lies at the center of the bounding box for good diversity. It is also desirable that, the larger the dimension of the bounding box, the larger the value of $\text{CD}(B)$. We thus need a formulation for CD which is maximized if the dimension of the bounding box is maximized or for a given size of the bounding box, CD is maximized if the solution B lies at the center of the bounding box. We thus propose the modified formulation for crowding distance computation given in equation 2.

$$\text{CD}(B) = \sum_{i=1}^{M} l_i r_i \tag{2}$$

where M is the number of objectives. The product $l_i r_i$ is maximized for a constant size of the bounding box if $l_i = r_i$; i.e. if B lies at the center of its neighbors. Also, as the size of the bounding box grows, the value of the product increases, thus the formulation given in equation 2 accounts for the size of the bounding box as well as the location of B inside the bounding box. The extreme solutions are assigned a value of infinity as in the original case.

The above formulation has a potential shortcoming. Suppose there is a solution B' identical to B. In that case, the formulation described by equation 2 would give a value of zero for diversity for both B and B' since distance to the nearest neighbor is zero in all the directions (since at least one of l_i or r_i would evaluate to zero). The original formulation (as proposed in NSGA-II) would give non-negative values to both B and B'. Depending upon their actual position in the sorted array, the crowding distance for solutions B and B' would evaluate to one of the values from $(l_1 + l_2, l_1 + r_2, r_1 + l_1, r_1 + r_2)$. This situation also is not desirable, since the obtained values do not accurately reflect the diversity of the two solutions. We therefore propose further modification which can be applied to the original as well as proposed crowding distance formulation. We suggest that all (but one) copies of an identical solution be removed and assigned a value of zero for the crowding distance before applying the formulation given in equation 2. All identical copies can be removed in $\text{O}(N \log N)$ time if there are N solutions in a non-dominated set.

2.3 Scheme for generating the parent population from the archive

This is the most important step of AMGA. The size of the parent population significantly affects the performance of the algorithm. To illustrate a key concept used in the generation of parent population from the archive, we refer to the two-objective problem ZDT4 [6]. This problem has 100 distinct Pareto-optimal fronts out of which only one is globally Pareto-optimal. The plot of the objective space for the ZDT4 function after the first generation is shown in Figure 3. Only 6 solutions out of the 100 belong to the first rank. A significant number of function evaluations can be saved if only the solutions in the first rank are chosen for the parent population, which in turn creates the off-spring solutions. It is thus noted that a reduction in number of function evaluations can be obtained if only the best solutions (belonging to rank 1) are included in the parent population. This however is a greedy scheme and therefore may give premature convergence if the best solutions happen to be near a locally optimal basin. It is therefore desirable to include few more solutions in the parent population and probably not all the solutions in rank 1. The extra solutions can be chosen such that they are diverse. Several schemes can be used to generate the parent population. If only the solutions from rank 1 are included in the parent population, fast convergence can be obtained, but it will often result in a local Pareto-optimal front (such as in the case of ZDT4). Test problems ZDT4 and ZDT6 were used as the benchmark problems to fine-tune the selection scheme used to generate the parent population. Whilst ZDT4 has a large number of local Pareto-optimal fronts, ZDT6 has a highly non-uniform distribution of points in the genotypic space corresponding to a uniform distribution of points on the Pareto-optimal frontier in the phenotypic space. In AMGA, the parent population is generated from the archive based on the diversity information alone. The diversity is computed in the genotypic space for the creation of the parent population. The pruning method (based on efficient nearest neighbor search) used during the archive update is used to generate the parent population. This step is the most time consuming since a large population is pruned to a very small population.

The diversity based selection gives preference only to those solutions that are less crowded and very few new solutions are created. Since parent-centric recombination (simulated binary crossover [22]) is used with AMGA, the empty spaces (near diverse solutions) fill up relatively quickly which im-

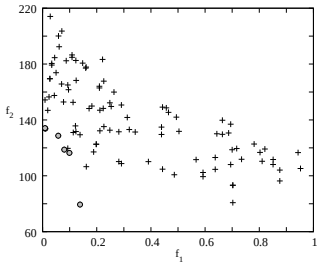

Figure 3: Objective space plot for the zdt4 problem

proves the distribution of solutions on the Pareto-optimal front. A potential limitation of the proposed approach is exposed when all the solutions in the archive occupy a region in the search space which points to a locally optimal front. The phenomenon of genetic drift will then guide the search towards that locally optimal frontier. Unless the crossover and mutation operators create a solution that lies inside the globally optimal basin, the search would lead to premature convergence. It is practically impossible to detect this phenomenon since the only information available about the search space is contained in the archive which now has all the solutions that belong to a locally optimal basin. Further, it can never be guaranteed that with the proposed (or any other optimization) algorithm, global convergence would be achieved. The proposed scheme relies on the discovery of at least one solution in the globally optimal basin which also happens to belong to the first rank (if it does not belong to the first rank, it may be removed whilst updating the archive). There always exists a tradeoff between the selection pressure and the diversity required by an evolutionary optimization algorithm. From the rigorous benchmarking conducted, it is concluded that the method proposed in this paper performs better overall on the problems chosen for the benchmark study.

2.4 Creation of the off-spring population

The binary tournament selection operator (without replacement) used in NSGA-II is used to generate the mating population. Unlike NSGA-II, the size of the mating population is half the size of the parent population. Since SBX crossover [22] takes two parents and produces two children, and we want every parent in the mating pool to participate exactly once in the crossover operation, the size of the mating pool must be an even number (a multiple of 2). Since the size of the mating pool is half the size of the parent population, the size of the parent population must be a multiple of 4 (which is also the minimum value for the size of the parent population). The size of the off-spring population is the same as the size of the mating pool. In NSGA-II, every solution in the parent population participates in exactly two tournaments, whereas in AMGA, every solution participates in exactly one binary tournament selection. Since every solution participates in the binary tournament selection in AMGA, it has a chance of being included in the mating pool. Also, this modification does not introduce extra copies of good solutions in the mating pool (better diversity). This modification favors diversity over selection pressure and experimental results demonstrated that it leads to faster convergence (since the number of function evaluations per generation is reduced by half, the number of generations can be doubled for the same number of function evaluations).

SBX crossover [22] and polynomial mutation [23] have been used as the genetic variation operators. In Deb et al. [16], it was shown that a modification to the polynomial mutation operation can improve the resilience to premature convergence. The modification increases the disruptiveness of the mutation operator. The original polynomial mutation operator [23] does not perturb a variable if it is at the boundary and uses the same probability distribution function for the entire range of the variable space. The modified polynomial mutation operator perturbs a solution with 50% probability if it is on the boundary and uses two different probability distributions for the two regions in which a variable divides its search range. For more details of the modified polynomial mutation operator, the reader is referred to the original study [16]. Both the crossover and mutation operators used in this study deploy a distribution index η which is a user defined parameter. The smaller the value of η, the larger the perturbation in the design variables (and vice versa). The value of η used for the crossover and mutation has noticeable impact on the performance of the algorithm. The suggested values of η are often determined empirically and may not work on all the problems. One of the goals of this research is to reduce the impact of tuning parameters set by the user on the performance of the algorithm. To reduce the sensitivity of the algorithm to changes in the distribution index, randomization is introduced in the value of η. A uniformly distributed random number is generated in the range [0, 1]. If the random number lies in [0, 0.3), 0.1η is used as the distribution index. If the random number lies in [0.3, 0.7], η (user supplied value) is used as the distribution index. If the random number lies in (0.7, 1.0], 10η is used as the distribution index. This modification reduces (but does not eliminate) the dependence on the user supplied value of η. The random number generator is invoked every time the crossover or mutation routine is called and the η for that operation is determined using the random number obtained.

2.5 Worst case complexity of the proposed algorithm

Let the size of the parent population be N, and the size of the archive be A. The size of the mating population would be $N/2$. Let the total number of function evaluations be T and the number of objectives be M. Step 2 of AMGA takes $O(N)$ time. Step 3 of AMGA also takes $O(N)$ time. In step 4, the initial population is copied to the archive which can be done in $O(N)$ time. Steps 5 through 11 form the main iteration loop of the AMGA. The iteration loop consumes maximum time when the archive is filled with all the solutions belonging to the first rank (i.e. the diversity based selection is imposed on the complete archive). The worst case complexity of the diversity operator used with the AMGA is $O(MA^2 \log(A))$. The non-domination ranking procedure used consumes $O(MA^2)$ time (although an asymptotically faster method [24] exists). The number of iterations performed depends on the number of function evaluations, and the size of the parent population. For T function evaluations, the number of generations of the algorithm is $\frac{2(T-N)}{N}$. In general, $N \ll T$, hence the number of generations performed is $\frac{2T}{N}$. Thus, the overall worst-case complexity of AMGA is $O(\frac{TMA^2 \log(A)}{N})$.

3. SIMULATION RESULTS

To assess the relative performance of AMGA, it is benchmarked against two state-of-the-art multi-objective evolutionary algorithms. The performance of AMGA is compared with Non-dominated Sorting Genetic Algorithm II [10] and Fast Pareto Genetic Algorithm [17]. We choose FastPGA as one of the algorithms for benchmark study since it is very recent high performing algorithm. NSGA-II is chosen because it is a de-facto standard against which the performance of other algorithms is compared. It has been shown in several studies that NSGA-II and SPEA2 have similar performance characteristics [10, 8, 15] and therefore SPEA2 is not included in this benchmark study. jMetal [25] is a JAVA based library that provides implementations for NSGA-II and FastPGA. For FastPGA, the reference implementation provided by jMetal is used in this study. The implementation of NSGA-II is provided by jMetal as well as iSIGHT-FD. In our tests, we found the iSIGHT-FD implementation of NSGA-II to be slightly better than the jMetal implementation. In this study, the simulation results of both the implementations are reported. The simulation results of NSGA-II obtained using the jMetal implementation are denoted with NSGA-II-1 and the simulation results obtained using iSIGHT-FD implementation are denoted with NSGA-II-2. For the purpose of the benchmark study; two-objective ZDT problems 1, 2, 3, 4, and 6; and three-objective DTLZ problems 1 and 2 are used. The description of the ZDT and DTLZ test problems is omitted and can be found in [6] and [26] respectively.

3.1 Performance Indicators

Four unary performance indicators are used to compare the three algorithms. The performance indicators used in this study are taken from [17]. The performance indicators are Delineation, Distance, Diversity and Hypervolume. In order to use these performance indicators, the true Pareto-optimal front must be known. Smaller value for a performance indicator means a better solution set. Ideally if the original Pareto optimal front is used as the solution set, all the performance indicators should evaluate to zero. Since a finite number of points (1,000 points for the case of two objectives and roughly 10,000 points for the case of three objectives) are used to represent the true Pareto-optimal frontier, a value of 0.01 or less for a performance indicator implies that the obtained solution set is virtually indistinguishable from the Pareto optimal front. If the value of the performance indicator is 0.5 or more, this implies that an acceptable solution set was not obtained. All the objectives are normalized (the Pareto-optimal set is mapped to the range [0, 1]) before the performance indicators are computed. Only the non-dominated solutions belonging to rank 1 are considered for computing the performance indicators. A brief description of each performance indicator follows next.

- *Delineation Metric*: It measures "how much of the true Pareto-optimal front is represented by the obtained solution set".
- *Distance Metric*: It measures the average Euclidean distance between the true Pareto-optimal front and the obtained solution set.
- *Diversity Metric*: It measures the uniformity of distribution and the spread of obtained solution set.
- *Hypervolume Metric*: In the formulation described in [17], it measures the fraction of search space not dominated by the obtained solution set in comparison to the true Pareto-optimal set.

Detailed description of the four performance indicators can be found in [17]. In [17], performance indicators are only formulated for the case of two objectives. The definition of delineation, distance, and hypervolume as described in [17] can be directly extended to three objectives. For the case of diversity, it is non-trivial to define the spread (since the obtained solution set is a surface in 3D). For the case of three objectives, the diversity metric is computed using the Euclidean distance to the nearest neighbors. For solution i, let the distance to its nearest neighbor be d_i. Let the average of all d_is be d_m. Let the size of the obtained solution set be S. Then, the diversity for the case of three objectives is given by Equation 3.

$$\text{Diversity Metric} = \sqrt{\frac{1}{S}\sum_{i=1}^{S}(d_i - d_m)^2} \tag{3}$$

The formulation given in Equation 3 only accounts for the uniformity of distribution. It does not account for the spread; since in three dimensions, the extremal points of the Pareto optimal set form a curve and it is non-trivial to compute the spread for the general case. The nadir objective vector is taken as [1.1, 1.1] for the purpose of computing the hypervolume metric. Any solution that is dominated by the nadir objective vector is discarded from the obtained solution set for the purpose of computing the hypervolume metric.

3.2 Simulation Parameters

Identical parameter settings (wherever possible) are used for all the algorithms. The identical parameter settings for NSGA-II, FastPGA, and AMGA are: size of the initial population = 100, crossover probability = 1.0, mutation probability = $1/n$, where n is the number of variables, crossover distribution index = 15.0, and mutation distribution index = 20.0. For FastPGA, maximum population size is equal to 100. For AMGA, maximum allowed size for the archive = 100, size of the parent population = 8, and number of Pareto-optimal solutions desired = 100. Number of function evaluations for ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, DTLZ1, and DTLZ2 are 6500, 6500, 6000, 10000, 10000, 20000, and 10000 respectively.

3.3 Results in tabulated format

30 random simulations each are performed for NSGA-II, FastPGA and AMGA. The mean and median values for all performance indicators is summarized in Tables 1 and 2. Delineation, distance, diversity, and hypervolume are abbreviated as Del., Dis., Div., and Hyp. in the tables respectively.

4. INFERENCES

As is evident from Tables 1 and 2, the performance of AMGA is better than NSGA-II and FastPGA on the two-objective test problems. On the two three-objective test

Table 1: The Mean value for all the performance indicators

Problem	Algorithm	Del.	Dis.	Div.	Hyp.
ZDT1	NSGA-II-1	0.0530	0.0546	0.0861	0.0873
	NSGA-II-2	0.0480	0.0495	0.0753	0.0794
	FastPGA	0.0230	0.0225	0.0408	0.0762
	AMGA	0.0045	0.0017	0.0062	0.0075
ZDT2	NSGA-II-1	0.0973	0.0987	0.3278	0.2967
	NSGA-II-2	0.0967	0.0952	0.3318	0.2766
	FastPGA	0.0394	0.0388	0.1441	0.1610
	AMGA	0.0044	0.0012	0.0061	0.0115
ZDT3	NSGA-II-1	0.0308	0.0286	0.1126	0.0912
	NSGA-II-2	0.0322	0.0311	0.1151	0.1005
	FastPGA	0.0126	0.0114	0.0628	0.0785
	AMGA	0.0033	0.0007	0.0404	0.0050
ZDT4	NSGA-II-1	1.0352	1.4101	2.2804	0.9398
	NSGA-II-2	0.8208	0.8133	1.7373	0.6947
	FastPGA	0.7278	0.8806	1.4569	0.8335
	AMGA	0.1560	0.1650	0.2630	0.2533
ZDT6	NSGA-II-1	0.2695	0.2744	0.7337	0.5992
	NSGA-II-2	0.1209	0.1271	0.3388	0.3107
	FastPGA	0.0463	0.0484	0.1345	0.1315
	AMGA	0.0048	0.0015	0.0178	0.0136
DTLZ1	NSGA-II-1	0.5966	1.2715	0.0057	0.6618
	NSGA-II-2	0.7384	3.1760	0.0071	0.6549
	FastPGA	0.3755	1.6488	0.0051	0.4325
	AMGA	0.4590	0.9150	0.0035	0.4477
DTLZ2	NSGA-II-1	0.0748	0.0386	0.0030	0.1526
	NSGA-II-2	0.0708	0.0161	0.0054	0.1406
	FastPGA	0.0728	0.0357	0.0029	0.1493
	AMGA	0.0725	0.0264	0.0035	0.1127

Table 2: The Median value for all the performance indicators

Problem	Algorithm	Del.	Dis.	Div.	Hyp.
ZDT1	NSGA-II-1	0.0521	0.0532	0.0873	0.0895
	NSGA-II-2	0.0488	0.0494	0.0720	0.0802
	FastPGA	0.0237	0.0230	0.0388	0.0454
	AMGA	0.0045	0.0016	0.0062	0.0074
ZDT2	NSGA-II-1	0.0914	0.0934	0.3115	0.2868
	NSGA-II-2	0.0887	0.0874	0.2981	0.2617
	FastPGA	0.0392	0.0391	0.1434	0.1320
	AMGA	0.0044	0.0011	0.0059	0.0114
ZDT3	NSGA-II-1	0.0311	0.0296	0.1120	0.0911
	NSGA-II-2	0.0308	0.0280	0.1130	0.0978
	FastPGA	0.0128	0.0116	0.0619	0.0478
	AMGA	0.0033	0.0007	0.0406	0.0050
ZDT4	NSGA-II-1	0.8963	1.4332	2.0251	0.9815
	NSGA-II-2	0.7863	0.7822	1.5788	0.8684
	FastPGA	0.7171	0.7698	1.3636	0.9082
	AMGA	0.1355	0.1416	0.2207	0.2190
ZDT6	NSGA-II-1	0.2733	0.2756	0.7498	0.6060
	NSGA-II-2	0.1194	0.1244	0.3378	0.3115
	FastPGA	0.0467	0.0488	0.1363	0.1334
	AMGA	0.0047	0.0015	0.0167	0.0133
DTLZ1	NSGA-II-1	0.6569	1.1633	0.0027	0.8794
	NSGA-II-2	0.7525	2.0924	0.0035	0.9427
	FastPGA	0.1492	0.8567	0.0021	0.1806
	AMGA	0.1276	0.0809	0.0029	0.1277
DTLZ2	NSGA-II-1	0.0739	0.0372	0.0024	0.1533
	NSGA-II-2	0.0711	0.0164	0.0029	0.1399
	FastPGA	0.0725	0.0374	0.0026	0.1486
	AMGA	0.0703	0.0269	0.0029	0.1108

problems, AMGA and FastPGA have similar performance; both of which outperform NSGA-II. The performance of the two implementations of NSGA-II is different. It can be attributed to the fact that, the performance of a non-deterministic algorithm using probabilistic operators depends on the i) method used for generating the initial population, ii) random number generator used, and iii) specific implementation of the probabilistic operators. The test problems used for this study are unconstrained and form a very small set. To gain more confidence in the performance of AMGA and to assess its usefulness, it is therefore imperative to benchmark more test problems especially the problems with constraints. Test problems ZDT4 and DTLZ1 have 99 and 161,050 local Pareto optimal fronts respectively and therefore challenge the capability of an algorithm to converge to the global Pareto optimal frontier.

AMGA places much larger emphasis on the diversity aspect as compared to NSGA-II because, for highly multi-modal problems or for problems with an uneven distribution of points on the Pareto-optimal frontier, maintaining diversity helps in achieving good convergence. The diversity operator used in AMGA is of $O(A^2 \log(A))$ complexity and thus increasing the size of the archive significantly increases the required computation time. A faster yet equally efficient diversity preservation technique can help in reducing the computational complexity of AMGA. In most MOEAs, increasing the population size while keeping the number of generations constant, increases the total number of function evaluations. AMGA removes this coupling and thus facilitates independent tuning of the population size (size of the archive) and the number of function evaluations.

5. CONCLUSION

In this paper, a new multi-objective optimization algorithm namely AMGA is proposed. AMGA is designed to work with a very small population size and maintains an external archive of good solutions obtained. AMGA also benefits from the existing literature in that it borrows several concepts from existing algorithms. Improved formulation for diversity preservation techniques is proposed and is incorporated into the AMGA. AMGA has been designed to facilitate independent tuning of algorithm parameters in that the size of the initial population, the size of the archive, and the size of the parent population (working population) can be independently fine-tuned. Such a design allows for choosing a small size for the parent population which significantly speeds up the search process. AMGA also attempts to minimize the impact of variation in distribution index for crossover and mutation on the performance of the algorithm. The performance comparison of AMGA with NSGA-II and FastPGA demonstrates the superior performance of AMGA on the two objective test problems. For the case of three

objectives, AMGA and FastPGA have similar performance. The research and development of AMGA can be perceived as an exercise in combining and improving the best features of different algorithms and best practices into a unified optimization procedure. The two guiding principles that have shaped the design of AMGA are i) focus on reducing the number of function evaluations required for the same degree of convergence, and ii) making the algorithm immune to changes in tuning parameters. AMGA has partially achieved both of these goals. Some resources on AMGA are available at http://people.clemson.edu/∼stiwari/amga.html.

6. REFERENCES

[1] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[2] R. Dawkins. *The Selfish Gene*. Oxford University Press, New York, 1976.

[3] N. Eldredge. *Macro-Evolutionary Dynamics: Species, Niches and Adaptive Peaks*. McGraw-Hill, New York, 1989.

[4] J. Holland. Adaptation in natural and artificial systems. Technical report, University of Michigan, 1975.

[5] D. E. Goldberg. *Genetic Algorithms for Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[6] K. Deb. *Multi-objective Optimization Using Evolutionary Algorithms*. Chichester, UK: Wiley, 2001.

[7] C. A. Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge Information Systems*, 1(3):269–308, 1999.

[8] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Proceedings of the EUROGEN2001 Conference*, pages 95–100, 2001.

[9] D. W. Corne, J. D. Knowles, and M. J. Oates. The pareto envelope-based selection algorithm for multi-objective optimization. In *Parallel Problem Solving from Nature*, pages 839–848, Berlin, 2000. Springer.

[10] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[11] S. Watanabe, T. Hiroyasu, and M. Miki. NCGA: Neighborhood cultivation genetic algorithm for multi-objective optimization problems. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2002*, pages 458–465, 2002.

[12] G. G. Yen and L. Haiming. Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation. *IEEE Transactions on Evolutionary Computation*, 7(3):253–274, June 2003.

[13] M. Kim, T. Hiroyasu, and M. Mikiand S. Watanabe. SPEA2+: Improving the performance of the strength pareto evolutionary algorithm 2. In *Parallel Problem Solving from Nature PPSN VIII*, pages 742–751, 2004.

[14] S. Y. Ho, L. S. Shu, and J. H. Chen. Intelligent evolutionary algorithms for large parameter optimization problems. *IEEE Transactions on Evolutionary Computation*, 8(6):522–541, 2004.

[15] K. Deb, M. Mohan, and S. Mishra. Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. *Evolutionary Computation Journal*, 13(4):501–525, 2005.

[16] K. Deb and S. Tiwari. Omni-optimizer, a generic evolutionary algorithm for single and multi-objective optimization. *European Journal of Operational Research*, 185(3):1062–1087, 2008.

[17] H. Eskandari, C. D. Geiger, and G. B. Lamont. Fastpga: A dynamic population sizing approach for solving expensive multiobjective optimization problems. In *Evolutionary Multiobjective Optimization Conference EMO-2007*, pages 141–155. LNCS 4403 Springer-Verlag Berlin Heidelberg, 2007.

[18] Y. Sensor. Pareto optimality in multi-objective problems. *Applied Mathematics and Optimization*, 4(1):41–59, March 1977.

[19] W. L. Loh. On latin hypercube sampling. *Annals of Statistics*, 33(6):2058–2080, 2005.

[20] S. Kukkonen and K. Deb. Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems. Technical Report 7, Indian Institute of Technology Kanpur, India, 2006.

[21] S. Kukkonen and K. Deb. A fast and effective method for pruning of non-dominated solutions in many-objective problems. Technical Report 4, Indian Institute of Technology Kanpur, India, 2007.

[22] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148, 1995.

[23] K. Deb and M. Goyal. A combined genetic adaptive search (geneas) for engineering design. *Computer Science and Informatics*, 26(4):30–45, 1996.

[24] M. T. Jenson. Reducing the run-time complexity of multiobjective eas: The nsga-ii and other algorithms. *IEEE Transactions on Evolutionary Computation*, 7(5):503–515, 2003.

[25] Juan J. Durillo, Antonio J. Nebro, Francisco Luna, Bernabe Dorronsoro, and Enrique Alba. jmetal: a java framework for developing multi-objective optimization metaheuristics. Technical report, E. T. s. Ingenieria Informatica Campus de Teatinos 29071 Malaga Spain, 2006.

[26] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization problems. In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 1, pages 825–830, Honolulu, HI, USA, 12–17 May 2002.