Solving Expensive Multiobjective Optimization Problems: A Fast Pareto Genetic Algorithm Approach

Hamidreza Eskandari
Department of Industrial Engineering and Management Systems
University of Central Florida
4000 Central Florida Blvd, Orlando, FL 32816
eskandar@mail.ucf.edu

Christopher D. Geiger
Department of Industrial Engineering and Management Systems
University of Central Florida
4000 Central Florida Blvd, Orlando, FL 32816
cdgeiger@mail.ucf.edu

ABSTRACT
We present a new multiobjective evolutionary algorithm (MOEA), called fast Pareto genetic algorithm (FPGA). FPGA uses a new ranking strategy for the simultaneous optimization of multiple objectives where each solution evaluation is computationally expensive. New genetic operators are employed to enhance the algorithm's performance in terms of convergence behavior and computational effort. Computational results for a number of benchmark test problems indicate that FPGA is a promising approach and it outperforms the improved nondominated sorting genetic algorithm (NSGA-II), which can be considered a widely-accepted benchmark in the MOEA research community, within a relatively small number of solution evaluations.

Categories and Subject Descriptors
G.1.6 [**Numerical Analysis**] Optimization – evolutionary algorithms, unconstrained optimization.

General Terms
Algorithms, Performance, Experimentation.

Keywords
Multiobjective optimization, Genetic algorithms, Pareto optimality.

1. INTRODUCTION
Most real-world problems often involve multiple, conflicting objectives, where improving one objective may degrade the performance of one or more of the other objectives. Multiobjective optimization problems (MOPs) have the following general form:

$$\min \text{ (max) } z = \mathbf{f}(\mathbf{x}), \tag{1}$$

where $\mathbf{f}(\mathbf{x})$ is vector of m objective functions to be optimized, *i.e.*, $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))$, and solution x is a n-dimensional vector of decision variables that are continuous or discrete or both. Eq. 1, which can be converted to a maximization problem with no loss of generality, is typically subject to the constraints

$$\mathbf{g}_j(\mathbf{x}) \le \mathbf{b}_j, \quad j = 1,2,\ldots,k \text{ , and} \tag{2}$$

$$a_i \le x_i \le b_i, \quad i = 1,2,\ldots,n, \tag{3}$$

where $\mathbf{b}$ is a k-dimensional vector of inequality constraints. Eq. 3 restricts the values of each decision variable x_i between an upper and a lower bound.

Traditional approaches for solving MOPs typically try to scalarize the multiple objectives into a single objective using a vector of user-defined weights. This transforms the original multiple optimization problem formulation into a single objective optimization problem with a single solution. Several drawbacks of using such traditional methods include:

- The weight (or priority) vector used in the scalarization process greatly influences the final solution;
- Some optimal solutions may never be found if the objective space is not convex (or concave, for maximization problems); and
- Traditional approaches may not work effectively if objectives have discontinuous variable space.

However, these and other known drawbacks to traditional approaches have motivated researchers and practitioners to seek alternative techniques to find a set of Pareto optimal solutions rather than just a single solution [*e.g.*, 3, 4]. A solution is Pareto optimal if there exists no feasible solution for which an improvement in one objective does not lead to a simultaneous degradation in one or more of the remaining objectives. In other words, the solution is a nondominated solution.

1.1 Evolutionary Algorithms for Multiobjective Optimization
Many heuristic search algorithms have been developed to solve MOPs including simulated annealing, tabu search and evolutionary algorithms (EAs). EAs, the focus of this study, are population-based search algorithms inspired by Darwinian evolutionary theory (*i.e.*, the survival of the fittest). It has been shown that EAs are intelligent optimization algorithms that are able to balance exploration and exploitation of the solution search space [8]. Other major advantages of EAs for MOPs include:

- EA-based approaches are capable of finding a set of good solutions rather than a single solution [4].
- EA-based approaches are capable of exploring the search space more thoroughly within a smaller number of function evaluations than other point-to-point local search procedures such as simulated annealing and tabu search [1].
- EA-based approaches are less dependent on the selection of the starting solutions, and they do not require definition of a neighborhood [1].

In recent years, several variations of MOEAs have been developed to handle MOPs [*e.g.*, 3, 4]. Many of the suggested MOEAs have been employed in a variety of real-world applications [2]. Among the existing algorithms, an improved

version of the nondominated sorting genetic algorithm (NSGA-II) of Deb *et al.* [6], a newer version of strength Pareto EA (SPEA2) of Zitzler *et al.* [12], Pareto-archived evolution strategy (PAES) of Knowles and Corne [9] are the more popular MOEAs.

1.2 Purpose of Research

In many previous applications of MOEAs, the time to perform a single solution evaluation is of the order of minutes or even hours restricting the total number of solution evaluations. Additionally, many real-world problems involve complicated, "black-box" objective functions making a large number of solutions evaluations computationally- and/or financially-prohibitive. Therefore, a fast multiobjective optimization algorithm capable of rapidly finding a diverse set of Pareto optimal solutions would be greatly beneficial. The purpose of this research is to propose such an EA-based multiobjective optimization methodology that finds evenly-distributed Pareto optimal solutions in an acceptable period of time.

The remainder of this paper is organized as follows. Section 2 describes the proposed MOEA that uses a new solution ranking strategy. The proposed MOEA and a widely-accepted benchmark MOEA are used to solve a suite of published test problems. Section 3 and Section 4 present the experimental design and computational results, respectively. This paper is concluded and future research directions are given in Section 5.

2. PROPOSED METHODOLOGY – FAST PARETO GENETIC ALGORITHM

The proposed framework named fast Pareto genetic algorithm (FPGA) utilizes a population-based evolutionary algorithm. However, more importantly, this framework incorporates a new solution ranking strategy into a MOEA. A real-coded GA is implemented to avoid the difficulties associated with binary representation and bit operations, particularly when dealing with continuous search spaces with large dimension. Recall that each solution to a MOP is represented by an n-dimensional vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where a decision variable x_i is a real number bounded by a lower limit a_i and upper limit b_i, *i.e.*, $x_i \in [a_i, b_i]$. The dimension of the vector is equal to the number of decision variables of the problem under study.

New search operators are also introduced to improve the proposed algorithm's convergence behavior and to reduce the required computational effort. An elitism operator is implemented to ensure the propagation of the Pareto optimal solution set. A regulation operator is introduced to dynamically adapt the population size of the GA as needed up to a user-specified maximum population size, which is the size of the set of nondominated solutions in this study. Figure 1 gives the pseudocode for FPGA.

The primary steps of FPGA are the following:

1. Initialize all decision parameters to user-specified values;
2. Create an initial population of candidate solution vectors randomly at the first generation; however, FPGA can be easily modified to generate the initial population heuristically, seeded with user-defined solution vectors, or using a combination of these approaches;
3. If it is the first generation, go to Step 5; otherwise, increment the generation number and select pairs of solutions as parents from the previous population in the reproduction operation using binary tournament selection;
4. Perform the crossover and mutation operations to generate candidate solutions (offspring);
5. Evaluate the candidate solution vectors for the m objective functions and record them;
6. Combine generated candidate solutions with the previous population to form a composite population;
7. Rank the composite population of solutions based on the new ranking strategy using their fitness values;
8. Regulate the population size according to the number of nondominated solutions and generate a new population from the composite population by discarding the inferior (dominated) solutions; and
9. Terminate the algorithm if the stopping criterion is met; otherwise, return to Step 3.

A detailed discussion of the primary features of FPGA is provided in the sections that follow.

```
initialize user decision parameters (numvars, numobjs,
maxpopsize, maxsolevals, p_c, p_m, …)
t := 0
create initial random population P_t = {x_1^t, x_2^t, x_3^t, …}
evaluate(P_t)
do while (stopping criteria is not met)
{
    t := t +1
    P'_t := select(P_{t-1})        // select pairs of solutions for
                                      reproduction
    O_t := crossover( P'_t )
    O_t := mutate(O_t)
    evaluate(O_t)
    CP_t := P_{t-1} ⋃ O_t          // form composite population
    rank(CP_t)
    regulate(CP_t)
    P_t := generate(CP_t)
}end do
```

Figure 1. Pseudocode of the proposed fast Pareto genetic algorithm.

2.1 Initialization and Solution Evaluation

After initializing the user-specified parameter settings (*e.g.*, number of decision variables, number of objectives, maximum population size, maximum number of solution evaluations, *etc.*), the initial population is created by random sampling each decision variable within its defined range of variation. The evaluation of new solutions in terms of the objective functions is accomplished by calculating the complicated mathematical, closed-form expressions specified by the published benchmark problems, which are discussed later. At each generation, the obtained solutions with their corresponding objective values are all recorded.

2.2 Ranking and Fitness Assignment

In the FPGA, before ranking and fitness assignment is performed, a new solution set $\mathbf{O}_t$ generated by crossover and mutation operations are combined with previous population $\mathbf{P}_{t-1}$ to form the composite population $\mathbf{CP}_t$, *i.e.*, $\mathbf{CP}_t = \mathbf{P}_{t-1} \bigcup \mathbf{O}_t$, where $\bigcup$ denotes the union of the two sets. The new ranking strategy is based on the classification of candidate solutions of the composite population

$\mathbf{CP}_t$ into two different categories (ranks) according to solution dominance. These ranks are used to evaluate solution fitness for the purpose of solution reproduction for subsequent generations. Combination of previous generation $\mathbf{P}_{t\text{-}1}$ with generated offspring $\mathbf{O}_t$ provides an opportunity to preserve the superior solutions in the next generation and discard the inferior solutions depending on the number of nondominated solutions obtained in the composite population.

Firstly, all nondominated solutions are identified as the first rank, which implies that there is no solution that is better than these solutions with respect to all objectives simultaneously. The fitness of the nondominated solutions in the first rank is calculated by comparing each nondominated solution with one another and assigning a fitness value. These values are computed using the crowding distance approach suggested by Deb *et al.* [6], which has been shown to help maintain diversity among the nondominated solutions in the Pareto optimal front. It is important to note that a solution with a large fitness value is preferred. The larger a solution's fitness value, the greater the distance that solution is from its neighboring nondominated solutions along the Pareto front.

All dominated solutions are identified as the second rank. Each dominated solution in the second rank is compared with all other solutions and assigned a fitness value depending on the number of solutions they dominate. The idea here is similar to the strength concept employed in SPEA [13] and SPEA2 [12]; however, it has been generalized. The fitness assignment takes into account both dominating and dominated solutions for any dominated solution. Here, each solution in the composite population $\mathbf{CP}_t$ is assigned a net strength value $S(\mathbf{x}_i)$, indicating the number of solutions it dominates, where

$$S(\mathbf{x}_i) = \left|\left\{\mathbf{x}_j \mid \forall \mathbf{x}_j \in \mathbf{CP}_t \wedge \mathbf{x}_i \succ \mathbf{x}_j \wedge j \neq i\right\}\right|. \quad (4)$$

The cardinality of a set is denoted as $|\cdot|$ and the expression $\mathbf{x}_i \succ \mathbf{x}_j$ means that solution $\mathbf{x}_i$ dominates solution $\mathbf{x}_j$. Then, the fitness value of each dominated solution is calculated by

$$F(\mathbf{x}_i) = \sum_{\mathbf{x}_i \succ \mathbf{x}_j} S(\mathbf{x}_j) - \sum_{\mathbf{x}_j \succ \mathbf{x}_i} S(\mathbf{x}_j), \ \forall \mathbf{x}_j \in \mathbf{CP}_t \wedge j \neq i. \quad (5)$$

In other words, the fitness value assigned to each dominated solution $\mathbf{x}_i$ in the second rank is equal to the summation of the strength values of all solutions it dominates minus the summation of the strength values of all solutions by which it is dominated. In contrast to SPEA and SPEA2 where the strength values of only solutions that $\mathbf{x}_i$ is dominated by (second term in Eq. 5) is considered, FPGA takes into account both dominating and dominated solutions with respect to solution $\mathbf{x}_i$. This strategy provides more information on Pareto dominance and niching relations among solutions in the composite population and reduces the chance that two solutions have the same fitness value. Therefore, no additional diversity preservation mechanism is used among the dominated solutions in the second rank thus requiring less computation (unlike SPEA2, which requires much higher computation for the density estimator). It is interesting to note that if most solutions do not dominate one another, it is implied that they belong to the first rank where crowding distance operator is invoked to maintain the diversity among them.

After the fitness values of all candidate solutions in $\mathbf{CP}_t$ are calculated, the solutions are compared where three different scenarios might occur. In the first scenario, two selected solutions have different ranks in which the solution with the better rank is preferred. In the second scenario, two solutions have the same rank but different fitness values. In such cases, the solution with larger fitness value is preferred. In the last scenario, two solutions have the same rank and fitness value. In the last case, one of solutions is randomly preferred.

2.3 Elitism and Population Regulation

An elitism operator with relatively high intensity is implemented to ensure propagation of the nondominated solutions (*i.e.*, elite solutions) to subsequent generations. This is accomplished by copying all solutions in the previous generation $\mathbf{P}_{t\text{-}1}$ to the composite population $\mathbf{CP}_t$.

The number of nondominated solutions usually increases over generations resulting in low elitism intensity in early generations if the population size is quite large and kept fixed. Moreover, the fluctuations of the number of nondominated solutions over generations demand an adaptive population sizing strategy to place appropriate emphasis of elitism intensity on nondominated solutions. If elitism intensity is too high, premature convergence might occur and if elitism intensity is too low, convergence might be too slow and computationally-expensive. Therefore, FPGA employs a regulation operator to dynamically adjust the population size until it reaches a user-specified maximum population size as calculated by:

$$\left|\mathbf{P}_t\right| = \min\left\{a_t + \left\lceil b_t \times \left|\left\{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbf{CP}_t \wedge \mathbf{x}_i \text{ is nondominated}\right\}\right|\right\rceil, \mathit{maxpopsize}\right\} \quad (6)$$

,

where $\left|\mathbf{P}_t\right|$ is the population size at generation t, a_t is a positive integer variable that might change over generations, b_t is a positive real variable that might change over generations, $\lceil x \rceil$ is the smallest integer that is greater than or equal to the real number x, and *maxpopsize* is the user-specified maximum population size. In this study, we set $a_t = 20$, $b_t = 1$ and *maxpopsize* = 100. Thus, we get

$$\left|\mathbf{P}_t\right| = \min\left\{20 + \left|\left\{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbf{CP}_t \wedge \mathbf{x}_i \text{ is nondominated}\right\}\right|, 100\right\}.$$

In other words, the population size at generation t is 20 plus the number of nondominated solutions in the composite population if it is not larger than the pre-specified maximum population size. Otherwise, it is kept (truncated) equal to the maximum population size.

FPGA, unlike many of the proposed MOEAs, benefits the small number of offspring created by crossover and mutation operations over generations. In this study, we set the number of offspring created at each generation $|\mathbf{O}_t| = 20$. It is interesting to note that this feature makes FPGA capable of saving significant number of solution evaluations during the search and utilizes the exploitation in a more efficient manner at later generations. Bear in mind that in expensive MOPs where only small number of solution evaluations is desired, more emphasis on exploitation and less emphasis on exploration can be extremely beneficial. Creating a large number of offspring at early generations consumes considerable number of solution evaluations limiting the total number of generations which consequently results in no extensive utilization of exploitation. FPGA requires less number of solution

evaluations per generation providing more generations for more search exploitation.

The suggested values for a_t, b_t and $|\mathbf{O}_t|$ are obtained by performing several pilot runs. As the intent of this research is to introduce a novel strategy that addresses adaptive population sizing and conservative offspring generation in order to improve the efficiency and effectiveness of the search, the attempt to determine optimal settings for these parameters is left for future study.

2.4 Termination Criterion

Different approaches have been used to stop the search process of EAs including those that consider the landscape of the response surface, the desired solutions quality, the specific number of solution evaluations and the required computation time. Originally designed for dealing with expensive MOPs, FPGA uses a new stopping criterion that considers the convergence speed towards the true Pareto optimal front. Here, when the number of nondominated solutions reaches the pre-specified maximum population size and thereafter no changes are made in the number of nondominated solutions within a certain number of solution evaluations, the search stops. For better understanding of the suggested stopping criterion for expensive MOPs, a convergence velocity measure is defined.

Definition 1: The Pareto production rate (*PPR*) is the rate at which a particular algorithm produces nondominated solutions per population and is calculated as

$$PPR_t = \frac{|\mathbf{NP}_t|}{|\mathbf{P}_t|}, \tag{7}$$

where $\mathbf{P}_t$ is the population at generation t, and $|\mathbf{NP}_t|$ denotes the number of nondominated solutions belonging to population $\mathbf{P}_t$. When PPR_t reaches one (*i.e.*, all solutions in the population are nondominated) and it does not make any changes over a pre-specified number of solution evaluations implying no promising nondominated solutions are found within this period, the search stops.

2.5 A Discussion on Computational Complexity of FPGA

To determine the computational complexity of FPGA, consider the worst case complexity at generation t of the search process. The key operations of FPGA with respect to complexity include the new ranking strategy, fitness assignment, and crowding distance computation. The complexity of FPGA's ranking strategy that determines the nondominated solutions and dominated solutions is $O(mN_t\log N_t)$ for $m = 2$ and 3 and $O(mN_t\log N_t^{m-2})$ for $m \geq 4$. The complexity of the crowding distance computation performed for fitness assignment of the nondominated solutions is $O(mN_t\log(N_t))$. Sorting of the nondominated solutions based on their fitness assignments obtained from crowding distance needs $O(mN_t\log N_t)$ computations. Fitness assignment of dominated solutions requires $O(m\,N_t^2)$ computations. Thus, the overall complexity of FPGA is at most $O(m\,N_t^2)$. If the maximum population size of FPGA is the static population size of most other MOEAs, *i.e.*, N, the overall complexity of FPGA is $O(mN^2)$, which is no more than that of other popular MOEAs such as NSGA-II, SPEA2, and PAES.

3. EXPERIMENTAL STUDY

In this section, we evaluate the performance of FPGA on a suite of published benchmark test problems having two objectives and no coupled constraints. In all test problems, the functions are to be minimized. The results of FPGA are also benchmarked against one of the state-of-the-art MOEAs – the real-coded NSGA-II of Deb *et al.* [6]. It has been reported that NSGA-II outperforms most of its competitors including SPEA and PISA, and it competes closely with SPEA2 in terms of convergence to the true Pareto optimal front while maintaining the diversity [*e.g.*, 6, 7, 12]. However, SPEA2 requires higher computational complexity of $O(mN^2\log N)$ [12] compared to that of NSGA-II, $O(mN^2)$, raising the question of whether the computationally-intensive fitness assignment strategy and truncation operator in SPEA2 pays off for expensive MOPs. Some studies report that there is no significant difference between the performance of SPEA2 and NSGA-II, although SPEA2 requires significantly higher computational time [5, 7, 12].

3.1 Benchmark Test Problems

The suite of test problems consists of four well-known ZDTs real-variable benchmark problems that have been selected from the literature [11]. The test problems ZDT1 and ZDT2 have 30 decision variables each and the former has a convex Pareto optimal front and the latter has five discontinuous Pareto optimal fronts. The 10-decision variable test problem ZDT4 is a multi-frontal (multi-modal) problem having a large number of local Pareto optimal fronts and a single global Pareto optimal front. The test problem ZDT6 has 10 decision variables and a nonconvex Pareto optimal front. Moreover, the density of solutions across its Pareto optimal front is non-uniform and the density towards the Pareto optimal front gets thin. The selected suite of test problems include quite challenging Pareto optimality characteristics. Many researchers have used these problems as benchmarks for evaluating their proposed algorithms [*e.g.*, 6, 12].

3.2 Algorithm Parameter Settings

For both FPGA and NSGA-II, all of the user-specified parameter settings (except the maximum number of solution evaluations) are used according to the suggested values in the original study of Deb *et al.* [6]. These settings are summarized in Table 1. In order to make better comparisons, the maximum population size for FPGA is set to the suggested population size used by Deb *et al.* [6]. The number of solution evaluations shown in Table 1 depends on the characteristics and complexity of the underlying problem. The number of solution evaluations is kept relatively small to evaluate the performance of each algorithm more effectively for expensive, real-world MOPs that may only allow a small number of solution evaluations.

3.3 Performance Metrics

In MOPs, there are three primary goals: 1) *fast convergence* to the true Pareto frontier solution set in the objective space, 2) *close proximity* to the true Pareto frontier solution set, and 3) *diversity and even dispersion* of the obtained nondominated solutions along the true Pareto optimal front. Many performance metrics have been introduced within the last decade [*e.g.*, 3, 4, 6, 7, 13]. Few performance metrics have been suggested to simultaneously consider the above goals. Most previous studies emphasize only the closeness and diversity measures. Fast convergence to optimal solutions for expensive MOPs is very important. This is especially the case in real-world problems where finding the optimal or even

near-optimal solutions is often computationally-prohibitive. In this study, four performance metrics are used to assess the convergence behavior and diversity of FPGA and NSGA-II, two of which are newly introduced. They are the diversity metric and the delineation metric. Two of the four metrics, delineation and hypervolume, are employed for simultaneous evaluation of closeness and diversity of the obtained solutions to gain a more thorough overall evaluation. For each test problem, each algorithm is run with 30 different seed values and the mean, standard deviation and 95% confidence interval are computed. The lower and upper bounds of the 95% confidence interval are calculated by $\bar{x} \pm t_{\alpha/2,n-1}\, s/\sqrt{n}$, where $\bar{x}$ is the sample mean, s is sample standard deviation, α is the significance level and is equal to 0.05 and n is the sample size and is equal to 30. Given the fact that in expensive MOPs the time required for solution evaluations significantly dominates the actual CPU time of any approach, no attempt is made to measure the computation time needed to run each algorithm. However, equality of the computational complexity of FPGA and NSGA-II indicates that there should be no appreciable difference between their computation times.

Table 1. Parameter settings for FPGA and NSGA-II.

Algorithm Parameter	**FPGA and Real-Coded NSGA-II**			
Test Problem	ZDT1	ZDT3	ZDT4	ZDT6
No. of Solution Evaluations	6500	6000	10000	10000
Initial Population Size	100			
Maximum Population Size	100			
Crossover Probability	1			
Mutation Probability	1/*n* (where *n* is number of variables)			
Crossover Type	Simulated Binary Crossover (η_c=15)			
Mutation Type	Polynomial Mutation (η_m=20)			
Selection Scheme	Binary Tournament			

3.3.1 *Distance from the Pareto Optimal Front*

Deb *et al.* [5] suggest the distance metric ϒ, which evaluates the convergence to a known Pareto optimal frontier set. The goal of this metric is to identify how close a set of obtained solutions are to the true Pareto optimal set. The smaller the value of this metric, the closer the solutions are to the true Pareto optimal frontier set. Ideally, this metric is zero, where each obtained solution falls exactly on one of the selected H uniformly-spaced solutions from the true Pareto optimal set in the objective space. However, the likelihood of this happening is rare.

3.3.2 *Diversity of Nondominated Solutions*

We define the diversity metric Δ to evaluate the extent of dispersion of the obtained nondominated solutions in the objective space. Here, the goal is to obtain a set of nondominated solutions that are both widely- and uniformly-distributed along the Pareto optimal front at the end of the search. To compute the diversity metric Δ, the Euclidean distance d_i between consecutive nondominated solutions is calculated in the objective space, as shown in Figure 2, where i = 1, ..., $|\mathbf{NP}_t|$-1 and $|\mathbf{NP}_t|$ is the number of nondominated solutions at the end of the search. Then, the standard deviation of these distances σ_d is calculated representing the degree of non-uniformity of the nondominated solutions. The minimum Euclidean distance of the two extreme Pareto solutions of the true Pareto optimal set from the nondominated solutions, denoted by d_p and d_q, is calculated. Note that the distances d_p and d_q are the distances from the closest nondominated solutions, not necessarily the endpoints of nondominated solutions, to the two extreme Pareto solutions (shown in Figure 2). Finally, the diversity of the set of nondominated solutions is

$$\Delta(\mathbf{NP}_t) = d_p + d_q + \sqrt{\frac{1}{|\mathbf{NP}_t|-1}\sum_{i=1}^{|\mathbf{NP}_t|-1}\left(d_i - \bar{d}\right)^2}\,. \tag{8}$$

The first two terms of Eq. 8 measure the spread of the nondominated solutions and the last term measures their uniform spacing. The diversity metric Δ returns a value in the range of [0, ∞). Small values of this metric mean the nondominated solutions are well-spread and well-distributed. Ideally, this metric takes a value of zero. This happens when each end nondominated solution falls exactly on the extreme Pareto optimal solutions and all Euclidean distance d_i between consecutive nondominated solutions are equal in the objective space. However, similar to the distance metric ϒ, this rarely happens.

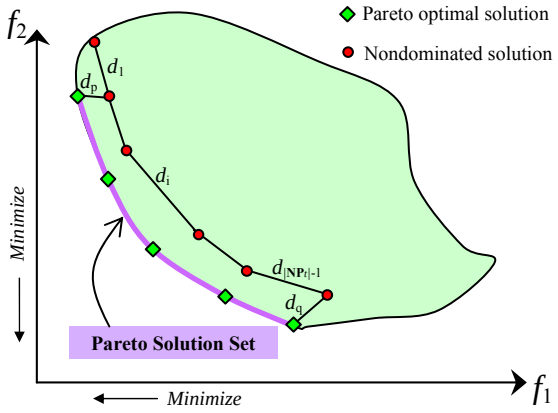

Figure 2. Diversity metric Δ.

3.3.3 *Delineation of Pareto Optimal Front*

The delineation metric Φ is proposed to evaluate simultaneously the extent of both convergence and diversity to a known Pareto optimal front. The goal of this study is to identify a set of solutions that well represent the Pareto optimal set. The idea behind this metric is how well each solution on the Pareto optimal front is represented by the obtained solutions. To calculate the delineation metric Φ, a large set of H uniformly-spaced solutions from the Pareto optimal set that well reflects the true Pareto optimal front must be known. The same set of H solutions used in calculating the distance metric ϒ are used here. The minimum Euclidean distance from each Pareto optimal solution to the obtained solutions l_i is calculated and the average of these distances is used as the delineation metric Φ, *i.e.,*

$$\Phi(\mathbf{P}_t) = \frac{1}{H}\sum_{i=1}^{H} l_i\,. \tag{9}$$

Figure 3 presents the calculation procedure of this metric. It is important to note that all solutions obtained by an algorithm including those that are dominated are considered for the calculation of this metric. The delineation metric Φ returns a value in the range of [0, ∞). The smaller the value of this metric, the better the Pareto optimal solutions are represented by the obtained solutions. Ideally, this metric is zero, where population size is

adequately large ($\geq H$) and each H selected Pareto solution is exactly overlapped by one of the nondominated solutions. The likelihood of this happening is zero, especially when population size is smaller than H, which is the case in most applications.

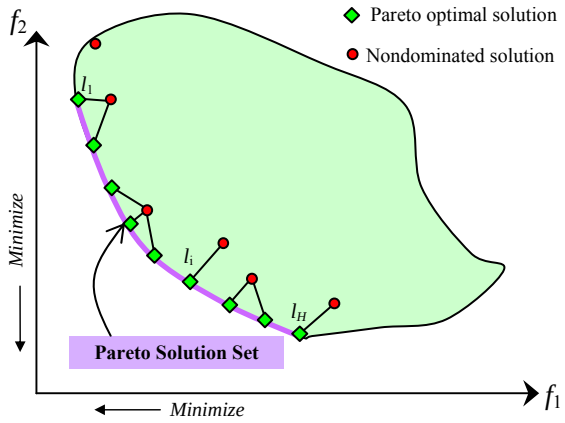

Figure 3. Delineation metric Φ.

3.3.4 Hypervolume

The hypervolume metric *HV*, originally suggested by Zitzler and Thiele [13], calculates the volume of the objective space dominated by the nondominated solutions having the reference point **R**. The goal of this measure is to identify the proportion of the volume enclosed by reference point and Pareto optimal front covered by the nondominated solutions obtained at the end of the search. To be consistent with other performance metrics used in this study (*i.e.*, the smaller value of the metric, the better), a modification of hypervolume ratio *HVR* is employed. Here, the proportion of the volume enclosed by reference point **R** and the true Pareto optimal front that is not covered by the nondominated solutions is of interest.

4. DISCUSSION OF THE RESULTS

Table 2 and Table 3 show the output statistics including mean, standard deviation and 95% confidence interval (CI) of the four performance metrics obtained from generating 30 random runs for each test problem using FPGA and NSGA-II. Distance ϒ and diversity Δ metrics are shown in Table 2 and delineation metric Φ and hypervolume ratio HVR are given in Table 3. Recall that lower values are preferred for all four metrics.

The results shown in Table 3 indicate that FPGA significantly outperforms NSGA-II with respect to the convergence to the Pareto optimal front. There is no overlap between the confidence intervals of the distance metric ϒ for FPGA and NSGA-II in all problems. Compared with FPGA, NSGA-II exhibits poor convergence in the ZDT4 and ZDT6 test problems. Both MOEAs have acceptable standard deviations for ϒ-metric on most problems. An exception occurs on ZDT4, where NSGA-II has very high standard deviation for ϒ-metric. To illustrate the convergence behavior of FPGA and NSGA-II, the sample obtained populations at the end of the search together with the Pareto optimal front for ZDT1, ZDT3, ZDT4 and ZDT6 are shown in Figure 4, Figure 5, Figure 6 and Figure 7, respectively. These figures show the superiority of FPGA over NSGA-II in rapidly converging to the true Pareto optimal solution set while preserving a diverse set of nondominated solutions. Within the fixed number of solution evaluations, FPGA obtains the population of nondominated solutions while a significant proportion of solutions in NSGA-II are dominated solutions, indicating that FPGA has much faster convergence. It is interesting to note that all obtained nondominated solutions yielded by NSGA-II at the end of the search are dominated by the nondominated solutions of FPGA in most problems. The favorable performance of FPGA is most likely due to high elitism intensity and regulation operator employment. These settings help to improve search space exploitation and to save a considerable number of solution valuations for further investigation at later generations.

Table 2 shows that FPGA has significantly better performance than NSGA-II in terms of the diversity metric Δ for most problems. There is no overlap of confidence intervals for Δ for FPGA and NSGA-II in ZDT1, ZDT4 and ZDT6 problems. NSGA-II performs slightly better than FPGA on ZDT3 with respect to this metric. It is surprising to note that FPGA has better Δ-metric than NSGA-II in many runs on ZDT3, but its performance is actually poor in few runs. The reason for this happening is most likely due to the employment of high elitism intensity resulting in biasness towards some particular regions of the Pareto front in few runs. This undesired biasness with FPGA is also realized on ZDT3 and ZDT4 problems having relatively large standard deviation. NSGA-II has good standard deviations for Δ-metric on all problems, except in ZDT4, where it has very high standard deviation.

Table 3 indicates that FPGA has better performance than NSGA-II in terms of the delineation metric Φ for most problems. There is no overlap between the confidence intervals of the Φ-metric for FPGA and NSGA-II in FON, KUR, ZDT1, ZDT4 and ZDT6 problems. FPGA has a little better average performance than NSGA-II on ZDT3, but there is a considerable overlap of their confidence intervals. The standard deviations of the Φ-metric across all problems for both MOEAs are small, except for FPGA on ZDT3 (due to the poor diversity in few runs) and for NSGA-II on ZDT4.

For *HVR*, the reference point **R** is set at (1, 1.1) for all test problems. The results shown in Table 3 indicate that FPGA outperforms NSGA-II with respect to *HVR*. There is no overlap of confidence intervals for *HVR* for FPGA and NSGA-II in all problems. It is interesting to note that although there is considerable overlap between the confidence intervals of both algorithms on ZDT3 with respect to delineation metric Φ, FPGA outperforms NSGA-II with respect to *HVR*. Regarding the obtained results, it is implied that although nondominated solutions with FPGA in few runs do not represent the Pareto fronts of ZDT3 pretty well, they dominate a considerable portion of the hypervolumes enclosed by the Pareto fronts and reference point **R**. The standard deviations of *HVR* for NSGA-II is small on all problems, except in ZDT4.

Table 2. Mean, standard deviation and 95% confidence interval of distance and diversity metrics for FPGA and NSGA-II over 30 random runs.

Test Problem	Algorithm	Distance ϒ			Diversity Δ		
		Avg.	Std. Dev.	95% CI	Avg.	Std. Dev.	95% CI
ZDT1	FPGA	0.0210	0.0110	[0.0169, 0.0251]	0.0769	0.0296	[0.0659, 0.0879]
	NSGA-II	0.0659	0.0128	[0.0612, 0.0707]	0.1324	0.0220	[0.1242, 0.1406]
ZDT3	FPGA	0.0200	0.0092	[0.0166, 0.0235]	0.2017	0.1036	[0.1631, 0.2403]
	NSGA-II	0.0297	0.0091	[0.0263, 0.0331]	0.1968	0.0233	[0.1882, 0.2055]
ZDT4	FPGA	0.0332	0.0262	[0.0234, 0.0430]	0.3812	0.1804	[0.3140, 0.4485]
	NSGA-II	0.7677	0.3414	[0.6404, 0.8950]	1.5111	0.5797	[1.2950, 1.7273]
ZDT6	FPGA	0.0445	0.0082	[0.0414, 0.0475]	0.1393	0.0256	[0.1297, 0.1488]
	NSGA-II	0.2647	0.0380	[0.2506, 0.2789]	0.7239	0.1063	[0.6843, 0.7636]

Table 3. Mean, standard deviation and 95% confidence interval of delineation and hypervolume ratio metrics for FPGA and NSGA-II over 30 random runs.

Test Problem	Algorithm	Delineation Φ			Hypervolume Ratio *HVR*		
		Avg.	Std. Dev.	95% CI	Avg.	Std. Dev.	95% CI
ZDT1	FPGA	0.0208	0.0097	[0.0172, 0.0244]	0.0443	0.0198	[0.0369, 0.0517]
	NSGA-II	0.0599	0.0111	[0.0557, 0.0640]	0.1259	0.0226	[0.1175, 0.1343]
ZDT3	FPGA	0.0269	0.0255	[0.0174, 0.0364]	0.0850	0.0345	[0.0722, 0.0979]
	NSGA-II	0.0286	0.0084	[0.0255, 0.0318]	0.1086	0.0252	[0.0992, 0.1180]
ZDT4	FPGA	0.0701	0.0457	[0.0531, 0.0872]	0.0910	0.0479	[0.0732, 0.1089]
	NSGA-II	0.6557	0.3128	[0.5391, 0.7724]	0.8173	0.2123	[0.7381, 0.8964]
ZDT6	FPGA	0.0415	0.0079	[0.0385, 0.0444]	0.1083	0.0190	[0.1012, 0.1154]
	NSGA-II	0.2538	0.0396	[0.2391, 0.2686]	0.5731	0.0690	[0.5473, 0.5988]

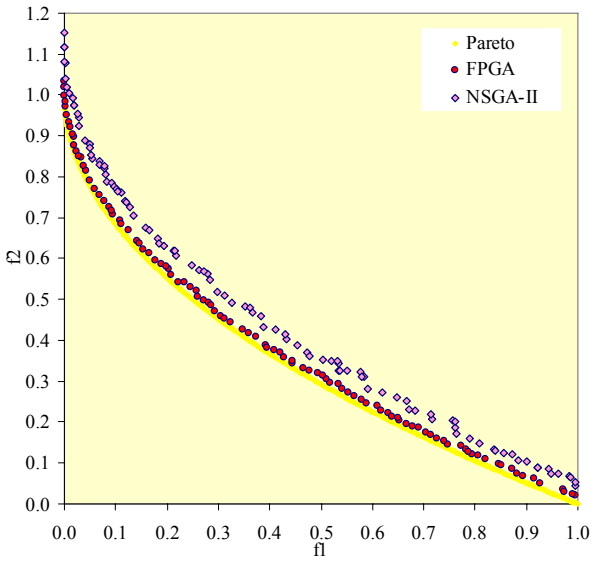

Figure 4. The populations with FPGA and NSGA-II on ZDT1.

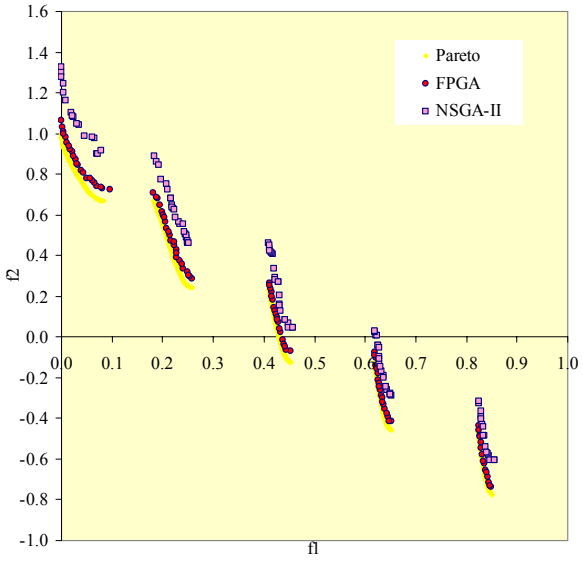

Figure 5. The populations with FPGA and NSGA-II on ZDT3.

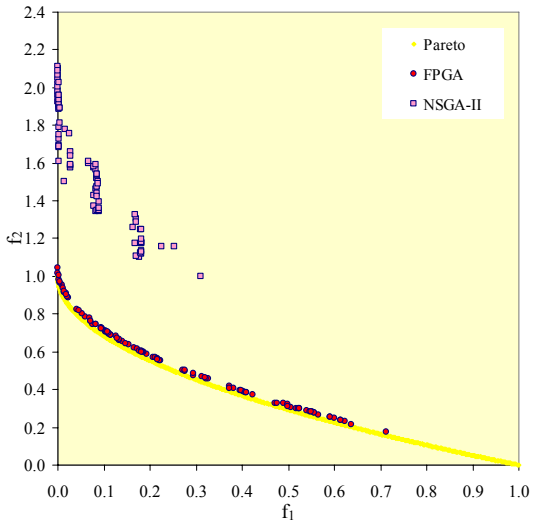

Figure 6. The populations with FPGA and NSGA-II on ZDT4.

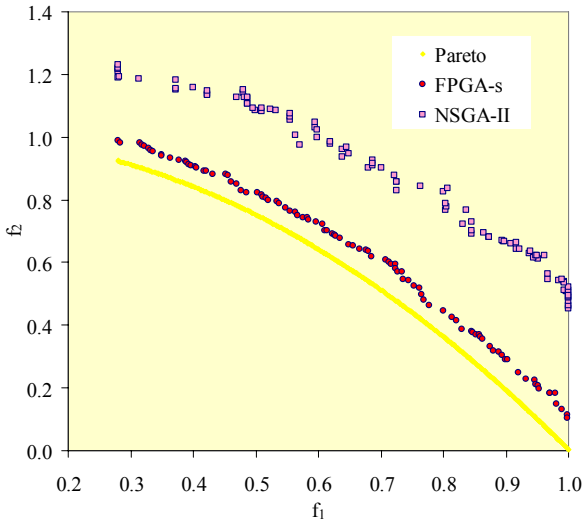

Figure 7. The populations with FPGA and NSGA-II on ZDT6.

5. SUMMARY AND CONCLUSIONS

This research presents a MOEA, called FPGA, for dealing with MOPs where each solution evaluation is computationally- and/or financially-expensive. This approach incorporates a Pareto-based multiobjective optimization method into a genetic algorithm. New genetic operators are introduced to enhance the algorithm's performance in finding Pareto optimal solutions minimizing computational effort. In addition to distance and hypervolume metrics, two new metrics, called diversity and delineation, are defined to better discriminate among the MOEAs. Computational results for a number of well-known test problems with different Pareto optimality characteristics indicate that FPGA is capable of efficiently and effectively direct the search toward Pareto optimal front. Statistical analyses show that, within a relatively small number of solution evaluations, FPGA outperforms NSGA-II in most problems in terms of rapidly converging to the true Pareto optimal solution set while preserving a diverse, evenly-distributed set of nondominated solutions. Adaptive population sizing is most likely one of the main factors resulting in the superiority of FPGA over NSGA-II in this benchmark environment.

Future research includes additional testing and benchmarking FPGA on several other MOPs higher in dimension of objective space. The attempt to find the best FPGA parameter settings will also be pursued in the next step of this study. Finally, more precise statistical analyses of the results will be performed.

6. ACKNOWLEDGMENTS

The authors are extremely grateful to Mark P. Kleeman and Gary B. Lamont of the Air Force Institute of Technology for their thorough review of a related work that helped to strengthen this paper. The authors would also like to thank Luis C. Rabelo of the University of Central Florida for his financial support during the early stages of this research investigation.

7. REFERENCES

[1] April, J., F. Glover, J. Kelly and M. Laguna. 2003. Practical introduction to simulation optimization. In *Proceedings of the 2003 Winter Simulation Conference*. eds. S. Chick, T. Sanchez, D. Ferrin and D. Morrice, 71-78. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

[2] Coello, C.A.C., and G.B. Lamont. 2004. Applications of Multi-Objective Evolutionary Algorithms. Singapore: World Scientific.

[3] Coello, C.A.C., D.A. Van Veldhuizen and G.B. Lamont. 2002. *Evolutionary Algorithms for Solving Multi-Objective Problems*. 1st Ed. New York: Kluwer Academic Publishers.

[4] Deb, K. 2001. *Multi-Objective Optimization using Evolutionary Algorithms*. 1st Ed. Chichester, UK: John Wiley & Sons.

[5] Deb, K., M. Mohan and S. Mishra. 2005. Evaluating the epsilon-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions. *Evolutionary Computation* 13(4), 501–525.

[6] Deb, K., A. Pratap, S. Agarval, and T.A. Meyarivan. 2002. Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computing*, 6, 182-197.

[7] Erbas, C., S. Cerav-Erbas and A. D. Pimentel. 2006. Multiobjective Optimization and Evolutionary Algorithms for the Application Mapping Problem in Multiprocessor System-on-Chip Design. To appear in *IEEE Transactions on Evolutionary Computation*.

[8] Goldberg, D.E. 1989. *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison Wesley.

[9] Knowles, J. and D. Corne. 1999. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation*.

[10] Srinivas N. and K. Deb 1994. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *International Journal of Evolutionary Computation*, 2(3), 221-248.

[11] Zitzler, E., K. Deb and L. Thiele. 2000. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation* 8(2), 173195.

[12] Zitzler, E., M. Laumanns and L. Thiele. 2001. SPEA2: Improving the Strength Pareto Evolutionary Algorithm, Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland.

[13] Zitzler, E. and L. Thiele. 1999. Multiobjective Evolutionary Algorithms: A Comparative Study and Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation* 83-98.