ASMiGA: An Archive-Based Steady-State Micro Genetic Algorithm

Kaustuv Nag, Tandra Pal, *Member, IEEE*, and Nikhil R. Pal, *Fellow, IEEE*

***Abstract*—We propose a new archive-based steady-state micro genetic algorithm (ASMiGA). In this context, a new archive maintenance strategy is proposed, which maintains a set of nondominated solutions in the archive unless the archive size falls below a minimum allowable size. It makes the archive size adaptive and dynamic. We have proposed a new environmental selection strategy and a new mating selection strategy. The environmental selection strategy reduces the exploration in less probable objective spaces. The mating selection increases searching in more probable search regions by enhancing the exploitation of existing solutions. A new crossover strategy DE-3 is proposed here. ASMiGA is compared with five well-known multiobjective optimization algorithms of different types—generational evolutionary algorithms (SPEA2 and NSGA-II), archive-based hybrid scatter search, decomposition-based evolutionary approach, and archive-based micro genetic algorithm. For comparison purposes, four performance measures (HV, GD, IGD, and GS) are used on 33 test problems, of which seven problems are constrained. The proposed algorithm outperforms the other five algorithms.**

***Index Terms*—Archive-based algorithm, genetic algorithms, multiobjective evolutionary optimization, Pareto front.**

I. Introduction

DEVELOPING methodologies for solving multiobjective optimization problems (MOPs) [37] has become an important area of investigation in the recent past. The main reason behind this is the multiobjective nature of various real-world problems for which often the objectives are conflicting in nature and often are required to satisfy a set of constraints. Consequently, simultaneous optimization of each objective is not possible. The goal of solving MOPs is not to find an optimal solution; rather to find a set of nondominated solutions.

Without loss of generality, we can describe a constrained MOP as

$$\begin{aligned} &\text{Minimize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}),\ f_2(\mathbf{x}),\ \ldots,\ f_M(\mathbf{x})) \in \Re^M \\ &\text{Subject to} \\ &g_j(\mathbf{x}) \le 0,\ j = 1,\ 2,\ \ldots,\ J \\ &h_k(\mathbf{x}) = 0,\ k = 1,\ 2,\ \ldots,\ K \\ &x_i^{(L)} < x_i < x_i^{(U)},\ i = 1,\ 2,\ \ldots,\ N \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1,\ x_2,\ \ldots,\ x_N)^T \in \Re^N$, $\Re^N$ is the decision variable (genotypic) space, $\Re^M$ is the objective (phenotypic) space, and $\mathbf{f}: \Re^N \to \Re^M$ consists of M objective functions.

Let $\mathbf{p} = (p_1, p_2, \ldots, p_M)^T,\ \mathbf{q} = (q_1, q_2, \ldots, q_M)^T \in \Re^M$ be two objective vectors. $\mathbf{p}$ is said to *dominate* $\mathbf{q}$ or $\mathbf{p} \prec \mathbf{q}$ if $p_i \le q_i$ for all $i = 1,\ 2,\ \ldots,\ M$ and $\mathbf{p} \ne \mathbf{q}$. A solution is feasible if it satisfies all the constraints; otherwise, it is infeasible. There exists several definitions of constraint-domination; we use the one given in [1]. According to [1], a solution $\mathbf{p}$ constraint-dominates another solution $\mathbf{q}$ or $\mathbf{p} \prec_c \mathbf{q}$ for an M objective minimization problem, if any one of the following conditions is satisfied.

1) Solution $\mathbf{p}$ and $\mathbf{q}$ are both feasible and solution $\mathbf{p}$ dominates solution $\mathbf{q}$, *i.e.*, $\mathbf{p} \prec \mathbf{q}$.

2) Solution $\mathbf{p}$ is feasible and solution $\mathbf{q}$ is not.

3) Solution $\mathbf{p}$ and $\mathbf{q}$ are both infeasible, but solution $\mathbf{p}$ has a smaller "overall constraint violation."

A solution $\mathbf{x}^* \in \Re^N$ is said to be (globally or true) Pareto optimal if there exists no solution $\mathbf{x} \in \Re^N$ such that $\mathbf{f}(\mathbf{x})$ constraint-dominates $\mathbf{f}(\mathbf{x}^*)$. The set of all Pareto optimal points is called Pareto Set and is denoted by PS. The set of all Pareto objective vectors is called (true) Pareto Front, $PF = \{\mathbf{f}(\mathbf{x}) \in \Re^M | \mathbf{x} \in PS\}$. For many problems the number of Pareto optimal solutions is very large, may even be infinite.

In the last few decades, many multiobjective evolutionary algorithms (MOEAs) [1]–[19], [71]–[73] have been proposed to overcome the difficulties associated with MOPs. The output of any multiobjective optimization approach is a set of solutions. A good solution set should be:

1) as close as possible to the true Pareto front (PF);

2) uniformly distributed over the entire PF.

For a given computational effort (maximum number of objective function evaluations), a search algorithm should intensify the search on a particular region of the search space to satisfy the first property of a good output solution set. On the contrary, for the second property, it needs to search the whole search region uniformly. Thus, our targets are conflicting because we have fixed number of function evaluations. A good multiobjective optimizer (MOO) must realize a good trade-off between exploration and exploitation to satisfy these properties.

Manuscript received May 9, 2013; revised November 7, 2013 and April 4, 2014; accepted April 10, 2014. Date of publication May 7, 2014; date of current version December 15, 2014. This paper was recommended by Associate Editor S. Mostaghim.

K. Nag is with the Department of Instrumentation and Electronics Engineering, Jadavpur University, Kolkata 700 108, India (e-mail: kaustuv.nag@gmail.com).

T. Pal is with the Department of CSE, National Institute of Technology, Durgapur 713209, India (e-mail: tandra.pal@gmail.com).

N. R. Pal is with the Electronics and Communication Sciences Unit (ECSU), Indian Statistical Institute, Kolkata 700 108, India (e-mail: nikhil@isical.ac.in).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2014.2317693

Some of the popular MOOs are: generational genetic algorithms (like NSGA-II [1], SPEA2 [2]), hybrid scatter search (like AbYSS [16]), decomposition-based multiobjective evolutionary approaches (like MOEA/D-DE [17]–[19]), and archive-based micro genetic algorithms (like AMGA2 [5]). But algorithms like NSGA-II [1], SPEA2 [2], and AbYSS [16] cannot satisfactorily solve complicated test problems like DTLZ3, DTLZ6 [27] even after significant number (25 000) of objective function evaluations [16]. In [5] the authors have shown that AMGA2 [5], GDE3 [3], and MOEA/D in general have superior performance as compared to NSGA-II [1], FastPGA [9], and AMGA [4]. They [5] have also discussed that among these algorithms, MOEA/D outperforms others when the number of function evaluations is significantly large (approximately 1 00 000). This could be a limitation of real-world MOPs that require high computational time for a single function evaluation. AMGA2 has used a modified definition of crowding distance for the generation of mating pool. But, crowding distance has a shortcoming regarding the choice of neighbors in more than bi-objective space [32]. Consequently, AMGA2 may not be effective for problems with more than two objectives. Moreover, Knowels in [59] has shown that several real world MOPs [60]–[65] have the following properties.

1) One evaluation needs time in order of minutes or hours.
2) Parallelism is not possible, i.e., only one evaluation can be performed at a time.
3) Number of function evaluations is fixed by some financial, resource, and time constraints. (In this paper, we consider fixed number of function evaluations).

Most of the algorithms, mentioned above, have not properly addressed all these issues. The main objective of this paper is to develop a suitable MOO to address all the issues discussed above satisfactorily.

Our contributions in this paper are as follows.

1) We propose a new archive-based steady-state micro genetic genetic algorithm (ASMiGA), which combines a set of old and new evolutionary algorithmic components.
2) We propose a new environmental selection or archive truncation strategy, which reduces the chance of exploring the less desirable objective space. The environmental selection strategy keeps the elite solutions in the archive. The archive truncation strategy prunes solutions efficiently maintaining the diversity among the solutions in the archive. This strategy also makes the archive adaptive and dynamic in size as it keeps only the non-dominated solutions in the archive unless its cardinality falls below a minimum allowable size.
3) A new mating selection mechanism is proposed. It increases the chances of exploiting desirable solutions.
4) Based on this mating selection, a new crossover mechanism, DE-3, is proposed, which is suitable for steady-state genetic algorithms.
5) We provide many useful experimental results, which reveal that ASMiGA significantly outperforms five well-known existing algorithms (SPEA2 [2], NSGA-II [1], AbYSS [16], MOEA/D [19], and AMGA2 [5]) for a wide range of test problems. Four performance measures, hypervolume (HV) [47], generational distance (GD) [30], inverted generational distance (IGD) [30], and generalized spread (GS) [16], on 33 test problems, are used for comparison. Although primarily we focus on unconstrained problems, we test our algorithm on seven constrained problems of moderate size and for these problems too our algorithm performs much better than others suggesting that ASMiGA is equally effective for constrained problem particularly when the allowed computational effort (number of function evaluations) is fixed. This paper is an extended version of [48] where some preliminary results of this investigation are reported.

II. Multiobjective Evolutionary Algorithms (MOEAs)

The literature on MOEAs [1]–[19], [57], [58], [67], [68], [71]–[73] is quite rich. Comprehensive surveys and additional references can be found in [20]–[23] and [66]. A survey on constrained handling (CH) strategies in nature inspired algorithms (NIAs) is present in [49]. In this review, we only discuss the differences between different relevant algorithmic components and their impacts on performance. There are algorithms, which are specifically designed for environments having limited processing/memory power [69], [70]. We do not consider these algorithms as in this paper, we address only those three issues stated in Section I.

An MOO should: 1) provide a good solution set; 2) be easy to implement; and 3) be computationally less expensive. But, it is difficult to devise an MOO which will balance between computational cost, performance, and complexity (regarding implementation) of the algorithm. Usually, MOEAs use some measures for archive truncation. For example, NSGA-II [1] assigns crowding distance before archive truncation, while SPEA2 [2], GDE3 [3] assign the measure before archive truncation and reassign it after deleting each solution. The later approach is better than the former but it requires more computational overhead. Some of the computationally efficient algorithms are difficult to implement. For example, DMOEA [13] is computationally efficient having computational complexity O(MN), but it has been criticized for its difficulty to implement [22]. We like to mention that RDGA [12] is computationally expensive [complexity is $O(MN^3)$] as well as difficult to implement [22].

In generational evolutionary algorithms, either all or most of the solutions of the parent population (archive) are replaced in each generation. Whereas, in steady state evolutionary algorithms, only a few (mostly one or two) solutions of the parent population (archive) are replaced in each generation. Several existing algorithms perform very well for bi-objective problems, but may not perform well for problems with more than two objectives. As an illustration, NSGA-II uses crowding distance to consider diversity. But as reported in [32] crowding distance has some limitations regarding choice of neighbors when the number of objectives is more than two. There are several attempts to modify crowding distance [4], [5], [8], but these modifications are not free from the drawback about the choice of neighbors. In [72], two different crowding

estimation operators for objective space and variable space are introduced and successfully used with multiple selection schemes.

Some MOEAs are generational, e.g., NSGA-II, SPEA [7], SPEA2, GDE3, and Omni-Optimizer [8]; while algorithm like ssNSGA-II [6] is steady-state. Again, a few algorithms (such as AMGA, AMGA2) have characteristics in between steady-state and generational. An adaptive strategy is used by FastPGA [9], which dynamically adapts the archive size.

While pruning the archive, algorithms like NSGA-II [1], SPEA2 [2], and AMGA2 [5] do not truncate boundary solutions but PESA [10] and SPEA [7] may prune boundary solutions.

There is enough variation among the selection strategies used by different algorithms. For example, NSGA-II uses fast-nondominated-sort, while SPEA uses strength Pareto evolutionary approach, which is extended in SPEA2 and FastPGA. We note here that usually the domination level is emphasized over the diversity, which is not necessarily a good choice for multimodal problems [5]. This may not also be a good choice for constrained problems. Again, the fast-nondominated-sort [1] is computationally costlier than the SNOV [67], [68] method. But, SNOV may allow some bad individuals to survive. To overcome this in [68] a preselection is used to filter bad solutions. This process uses a reference point, which starts as the center of the normalized objective space and then it moves to the origin with generations. In each generation, all solutions, which are dominated by the solution closest to the reference point, are deleted. This strategy may not be a good choice when the maximum allowable size of the archive is limited (fixed) and the archive tries to maintain only nondominated solutions.

As discussed in [49], there exists several ways of constraint handling (CH) in single objective constrained optimization. These CH techniques are often extended to solve constrained MOPs. Yet, there are some CH techniques specifically designed to solve constrained MOPs [50]–[56]. Although these CH strategies are particularly designed for constrained MOPs, we use the popular modified feasibility rules [1] (an extension of single objective CH technique) in ASMiGA.

Different MOEAs use different variation operators. For real-coded chromosomes, the popular choices are: uni-modal normal distribution crossover [34], simulated binary crossover (SBX) [35], and parent centric crossover [36]. These variation operators suffer from lack of self-adaptability [5]. DE crossover [24], on the other hand, has the self-adaptability. Tiwari *et al*. [5] introduced a modified version of DE crossover, which is known as DE-2 crossover operator. In DE-2, the primary parent is selected based on domination level and diversity, and auxiliary parents are selected randomly. The appropriate choice of primary parent helps to improve the performance of the operator especially on multimodal test problems [5].

Most of the algorithms use only one population (archive). However, Zhan *et al.* [71] authors co-evolve multiple populations for multiple objectives and use an external shared archive for exchange of search information.

The proposed algorithm, ASMiGA, attempts to exploit the best features of different MOEAs. Our algorithm is easy to implement and has a computational complexity of $O(MN^2)$. ASMiGA uses a new archive truncation strategy, which always preserves boundary solutions. It preserves diversity only in phenotypic space, which is discussed in detail in Section III. ASMiGA does not always emphasize on the domination level over diversity. In case of mating selection, we emphasize on diversity and in case of environmental selection we emphasize on domination level. When solutions of the first nondominated front are not adequate to fill the minimum required archive size, ASMiGA picks dominated solutions using a two-tier fitness scheme. Depending on the problem, our algorithm uses either SBX or the proposed DE-3 crossover. In this paper, we have used random initialization for comparing our algorithm with other algorithms. However, users can initialize the population using domain knowledge, if available.

III. Archive-Based Steady-State Micro Genetic Algorithm (ASMiGA)

ASMiGA is a steady-state algorithm. There are mainly two reasons to opt for a steady-state algorithm.

1) We try to use as many generations as possible, and, if the number of function evaluations is fixed, steady-state algorithm can maximize the number of generations.
2) When parallelism is not allowed and each function evaluation takes a long time, steady-state algorithms are preferred.

In ASMiGA, in each generation we evaluate only one or two offspring(s). Only four (for DE-3) or two (for SBX) parents are sufficient for that. An algorithm is micro genetic, if its working population size is quite small compared to its population (archive) size. We make our algorithm micro genetic keeping the working population size as low as possible; i.e., four for DE-3 crossover and two for SBX crossover.

The archive of ASMiGA always tries to store only nondominated set of solutions. It bounds the archive size between a minimum ($N_{\min}$) and a maximum ($N_{\max}$). $N_{\min}$ must be at least two to use SBX crossover, and it must be at least four, to use DE-3 crossover. To increase the diversity among the solutions, however, one can increase the value of $N_{\min}$. The pseudocode of ASMiGA is depicted in Fig. 1.

A. Population Initialization and Choice of Crossover

If the user has any prior knowledge, it can be incorporated during the initialization of population. In absence of prior knowledge, LH sampling [25] can also be useful in the initialization process. However, in this paper, we have taken $N_{\max}$ number of random initial solutions.

ASMiGA uses DE-3 crossover, if any of the following is more than three: 1) the number of objectives; 2) the number of variables; or 3) the number of constraints. Otherwise, SBX crossover is used. If DE-3 crossover is used then a flag *DEChosen* is turned ON, otherwise, it is turned OFF.

Algorithm ASMiGA
1. ***Begin***
2. *Initialize population of size N_{max} and set all parameters.*
3. *Select SBX or DE-3 crossover.*
4. *Evaluate the initial solutions.*
5. *Initialize the archive using initial population.*
6. ***While*** *(termination condition is not satisfied)*
7. ***Do***
8. *Perform mating selection.*
9. *Perform crossover to create offspring solution(s).*
10. *Perform polynomial mutation on the offspring(s).*
11. *Evaluate the mutated offspring(s).*
12. *Update the archive with the offspring(s).*
//Environmental selection.
13. ***End***
14. ***Return*** *desired number of solutions from the archive.*
15. ***End***

Fig. 1. Pseudocode of ASMiGA.

B. Initializing Archive Using Initial Population

The fast-nondominated-sort [1] is applied on the initial population. After obtaining $N_{\min}$ solutions, we stop the sorting. Let F_l be the lth sub-front and $|F_l|$ be the number of solutions in F_l. Let this procedure produces a set of sub-fronts $\{F_1, F_2, \ldots, F_i\}$ such that, $(|F_1| + |F_2| + \ldots |F_i|) = N_{\min}$ and $(|F_1| + |F_2| + \ldots |F_{i-1}|) < N_{\min}$ are satisfied. If i is greater than one, then a flag named *fastNonDominatedSortRequired* is turned ON. This suggests that the archive is not a set of nondominated solutions, and in future fast-nondominated-sort has to be performed on this archive. This procedure makes sure that the size of the archive will not become less than $N_{\min}$.

C. Mating Selection

Unlike many archive-based algorithms, our mating pool and working population are the same. Here our objectives are to: 1) intensify the search in a region near the PF and 2) search the whole region uniformly to obtain diversity or good spread. To search near the true PF region, we should concentrate on solutions having lower rank. Most MOEAs use a two-tier fitness mechanism where first priority goes to domination level and second priority goes to diversity. But, it is not always good to emphasize on domination level (rank) over diversity—particularly for multimodal and constrained problems because a solution may have poor domination level or larger constraint violation, but excellent genotype mixed with collateral noise [29]. In this case, the collateral noise is responsible for its poor domination level or larger constraint violation. In ASMiGA, we emphasize on domination level in environmental selection. And in mating selection either we randomly pick the parents, or we pick them based on diversity. We never use domination level in mating selection. It may make our algorithm more explorative in a few initial generations (until $N_{\min}$ solutions appear in the first front), and more exploitative after that. Thus, we search both toward PF and along the PF.

Next, we need to determine a measure for diversity. Crowding distance [1] is a very popular measure used by many MOEAs. There are different variations of crowding distance [1], [4], [5], [8], but all these definitions have the same shortcoming regarding choice of neighbors when we have more than two-objectives [32]. Consequently, we use crowding distance for bi-objective problems only. For more than two objectives, we use nearest Euclidean distance in the normalized objective space. If the objective space is not normalized, some objectives may automatically get higher priorities. Though some algorithms like—SPEA2+ [11], Omni-Optimizer [8], and AMGA [4] preserve diversity both in objective space (phenotypic space) and in variable space (genotypic space), keeping diversity only in the objective space is desired. If we try to keep diversity in the variable space, we may include some solutions having poor domination level. This may make the algorithm unnecessarily more explorative.

The mating algorithm proceeds as follows. If *DEChosen* flag is ON then we select one primary parent and three auxiliary parents for the proposed DE-3 crossover. To select the primary parent, a random number r is generated in [0, 1]. If r is greater than the selection ratio S_r, then a solution is chosen randomly from the archive as the primary parent. Otherwise, the objective values of all solutions in the archive are normalized.[1] For each solution, we then compute its distances (in the normalized objective space) from its first and second nearest neighbors. If there is no tie, the solution with the highest first nearest distance is selected as the primary parent; otherwise, from the set of tied solutions, we select the one with the highest second nearest distance. If tie occurs even for the highest second nearest distance, we pick one of the tied solutions randomly. We randomly choose three auxiliary parents from the archive. Selection process ensures that all these four parents are distinct.

In mating selection, if *DEChosen* flag is OFF, then we need to select two parents to perform the SBX crossover. The first parent is selected by the same procedure used for selection of the primary parent of DE-3 crossover. The other parent is selected randomly from the archive. The restriction remains the same—the two parents must be distinct.

D. DE-3 Crossover, SBX Crossover, and Polynomial Mutation

If the flag *DEChosen* is OFF then SBX crossover [29], [35] is performed; otherwise, we perform DE-3 crossover to generate 2 or 1 off-springs, respectively.

Let $\mathbf{p}$ be the primary parent and $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$ be the auxiliary parents; N be the number of variables and j_r be a random integer uniformly distributed between 1 and N. Let r_j be another uniformly distributed random number in [0, 1]. DE crossover [3] operator uses two tuning parameters F and CR. Let, a_{ij} (p_j) be the jth component of $\mathbf{a}_i$ ($\mathbf{p}$), $\mathbf{o}$ be the offspring, and o_j be the jth variable of $\mathbf{o}$, then

$$o_j = \begin{cases} a_{3j}+F(a_{1j}-a_{2j}), \text{ if } r_j < CR \text{ or } j=j_r \\ p_j, \text{ otherwise} \end{cases} \quad (2)$$

For the DE crossover [3] all parents are selected randomly. A modified form of the DE crossover, known as DE-2 crossover [5], selects auxiliary parents randomly, while

[1] In this paper, whenever we have used normalization, we mean a linear transformation to [0, 1] where the minimum value is normalized to 0 and the maximum value is normalized to 1.

primary parent is selected based on the domination level (measured by rank) and diversity (measured by crowding distance) before selection of auxiliary parents. In our DE-3 crossover, we select the primary parent based on the diversity in normalized objective space, while the auxiliary parents are selected randomly. DE-3 and DE crossover use the same mathematical formulation as in (2).

Detailed description of SBX crossover and polynomial mutation can be found in [35] and [8], respectively.

E. Updating Archive Using Off-Spring(s)

In many algorithms (e.g., GDE3) the off-spring is compared with its parent first. If it dominates its parent, then only it is considered further for environmental selection. This strategy may cause problem if the off-spring does not dominate its parent but it dominates many other solutions of the archive. For this, we compare the off-spring with the entire archive in ASMiGA.

We use an adaptive sized archive for the following reasons.

1) For quick convergence, exploitation is preferred over exploration and to increase the exploitation we need to concentrate on domination level. But, as mentioned earlier, concentrating on domination level in mating selection may lead to poor outcome for constrained and multimodal test problems. One plausible solution may be to keep the archive size smaller when the archive is full of poor solutions, i.e., the archive is composed of solutions from many fronts.
2) We are using the modified feasibility rules defined by Deb *et al*. [1] to handle constraints, which may lead to premature convergence. One reasonable solution to this is to keep the archive size sufficiently large. Therefore, we keep at least $N_{\min}$ solutions in the archive. At the end we need to provide $N_{\max}$ nondominated solutions to the user. These motivate us to make our archive size adaptive. We always keep at least $N_{\min}$ solutions to maintain genotypic diversity. Initially when the solutions in the archive are far away from the true PF and forms more than one front, we keep only $N_{\min}$ solutions, otherwise the archive will be full of poor solutions.

Now, we explain the exact archive updating strategy. If DE-3 crossover is used then one offspring is generated in each evaluation, otherwise, two off-springs are generated by SBX crossover. For each off-spring we proceed as follows to update the archive.

If the flag *fastNonDominatedSortRequired* is ON, we add the offspring solution to the archive. Then the fast-nondominated-sort is performed on the archive. This sorting is stopped after getting $N_{\min}$ solutions. It produces a set of sub-fronts $\{F_1, F_2, \ldots, F_i\}$ such that, $(|F_1| + |F_2| + \ldots |F_i|) = N_{\min}$ and $(|F_1| + |F_2| + \ldots |F_{i-1}|) < N_{\min}$ are satisfied. If i is one, *fastNonDominatedSortRequired* flag is turned OFF. During this procedure a special case may arise: the total number of solutions up to and including $(i-1)$th sub-fronts is less than $N_{\min}$ and the total number of solutions up to and including the ith sub-front is $N_{\max}+1$. In this case, a solution from the ith sub-front is pruned using the *pruneOneSolution* procedure described in Section III-F and the remaining solutions are added to the archive. The archive size then becomes $N_{\max}$ and the *fastNonDominatedSortRequired* flag is turned ON.

If the *fastNonDominatedSortRequired* flag is OFF then to add the offspring to the archive, we compare it with all solutions in the archive. If any of the solutions in the archive, A, dominates the offspring, then it is discarded, and the archive remains the same as A. Otherwise, the off-spring will dominate a set of solutions, S (may be the null set). Then, if $N_{\min} = (|A| - |S| + 1) = N_{\max}$, we add the offspring to A, and delete solutions in S from A. Else if $(|A| - |S| + 1) < N_{\min}$, we use *pruneOneSolution* procedure to prune a solution from S, add the remaining solutions to A, and set *fastNonDominatedSortRequired* flag to ON. Else if $(|A| - |S|) = N_{\max}$ then the off-spring is added to A and a solution is pruned from A using *pruneOneSolution* procedure.

F. Procedure pruneOneSolution

It uses two different methods: one for bi-objective problems, and the other for problems with more than two objectives. For a bi-objective problem the crowding distance [1] is assigned to the solutions of the ith sub-front. Now, we delete the solution with the minimum crowding distance from the ith sub-front.

For problems with more than two objectives, for each solution in the ith sub-front, we compute its distances (in normalized objective space) from the first and second nearest neighbors in the ith sub front. For this, all solutions in the 1st to $(i-1)$th fronts are considered. If there is no tie, the solution having the minimum first nearest distance is pruned; otherwise, among the tied solutions, the one with the minimum second nearest distance is pruned. If tie continues, we randomly select one of the tied solutions and prune it.

IV. Results and Discussion

A. Comparing Algorithms

We have compared our algorithm with five algorithms: 1) strength Pareto evolutionary algorithm 2 (SPEA2) [2], an archive-based generational MOEA, which modifies SPEA [7]. SPEA2 uses binary tournament selection, SBX [35], and polynomial mutation [8] as default operators; 2) nondominated sorting genetic algorithm-II (NSGA-II) [1], which is an elitist and generational MOGA. NSGA-II uses binary tournament selection using a crowded-comparison operator, SBX [35], and polynomial mutation [8] as default genetic operators; 3) archive-based hybrid scatter search (AbYSS) [16], which is based on scatter search templates; 4) MOEA-based on decomposition (MOEA/D) [17]–[19], which decomposes a MOP into a number of scalar optimization sub-problems and optimizes the sub-problems simultaneously. MOEA/D-DE is a steady-state algorithm that uses differential evolutional (DE) crossover [3] and polynomial mutation [8]; and 5) AMGA2 [5], which uses an archive and a very small (micro) population.

B. Test Problems

We have used 33 test problems. Table I lists all these test problems and their characteristics [1], [2], [5], [26]–[28]

TABLE I
TEST PROBLEMS AND THEIR FEATURES

	Problem	n[a]	M[b]	C[c]	Features
1	Schaffer	1	2	0	Convex
	Fonseca	3	2	0	Non-convex
	Kurswae	3	2	0	Non-convex
2	ZDT1	30	2	0	Convex, uniformly distributed
	ZDT2	30	2	0	Non-convex
	ZDT3	30	2	0	Non-contiguous, convex, unbiased
	ZDT4	10	2	0	21^9 local Pareto fronts, non-convex
	ZDT6	10	2	0	Non-convex, Non-uniformly spread
3	WFG1	6	2	0	Separable, uni-modal, polynomial, flat, convex, mixed
	WFG2	6	2	0	Non-separable, uni-modal, and mult-modal, convex, disconnected
	WFG3	6	2	0	Non-separable, uni-modal, linear, degenerate
	WFG4	6	2	0	Separable, multi-modal, concave
	WFG5	6	2	0	Separable, deceptive, concave
	WFG6	6	2	0	Non-separable, uni-modal, concave
	WFG7	6	2	0	Separable, uni-modal, parameter dependant, concave
	WFG8	6	2	0	Non-separable, uni-modal, parameter dependant, concave
	WFG9	6	2	0	Non-separable, multi-modal, deceptive, parameter dependant, concave
4	ConstrEx	2	2	2	Constrained
	Srinivas	2	2	2	Constrained
	Tanaka	2	2	2	Constrained
	Osyczka2	6	2	6	Constrained
	Golinski	7	2	11	Constrained
5	Viennet2	2	3	0	Unconstrained
	Viennet3	2	3	0	Unconstrained
	Viennet4	2	3	3	Constrained
	Water	3	5	7	Constrained
6	DTLZ1	7	3	0	Linear hyper-plane, multi-modal
	DTLZ2	12	3	0	Hyper-spherical, convex
	DTLZ3	12	3	0	Many local Pareto fronts parallel to global Pareto front, multi-modal
	DTLZ4	12	3	0	Dense set of solutions exists near the f^M-f^1 plane
	DTLZ5	12	3	0	Curve, non-convex
	DTLZ6	12	3	0	Curve, non-convex
	DTLZ7	22	3	0	Disconnected Pareto optimal regions, Skewed

[a] Number of variables, [b] Number of objectives, [c] Number of constraints

and [38]–[46]. We have divided the test problems into six groups. The first group consists of three classical test problems—Schaffer [38], Fonseca [39], and Kursware [40]. The second group consists of five problems from the ZDT family [26]—ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. The third group has nine test problems of WFG family [41]—WFG1, WFG2, WFG3, WFG4, WFG5, WFG6, WFG7, WFG8, and WFG9. All problems in the first three groups are unconstrained and bi-objective. The fourth group consists of five constrained bi-objective problems—ConstrEx [1], Srinivas [44], Tanaka [43], Osyczka2 [42], and Golinski [2]. The fifth group contains some constrained and unconstrained problems with more than two objectives—Viennet2, Viennet3, Viennet4 [45], and Water [46]. The sixth and last group consists of seven unconstrained tri-objective problems from DTLZ family [27]: DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5, DTLZ6, and DTLZ7. For all the test problems, we have used the default problem definitions coded in jMetal 3.1 [28].

C. Performance Metrics

We have used four performance measures for comparing our algorithm with the other five algorithms. jMetal[2] 3.1 [28] is used to compute these metrics. The four performance metrics are: HV [16], [47], GD [16], [30], IGD [19], [30], and GS [1]. GS is a spread metric for bi-objective problems, which has been extended for more than two objectives as in [16] and [31]. This extended version of spread is called GS by the jMetal team. Among these four performance measures, both HV and IGD measure the distance to PF and diversity, GD measures distance to PF, and GS measures diversity. Both HV and IGD are generic indicators. However, no single performance metric can surely indicate superior performance of any algorithm. Even the outcomes from HV and IGD metrics differ. So, we choose to use both of the generic performance indicators.

D. Parameter Settings

We have used jMetal 3.1 [28] implementation of MOEA/D-DE [19], AbYSS [16], NSGA-II [1], and SPEA2 [2]. Since AMGA2 [5] is not present in jMetal 3.1, we have downloaded it from Kanpur genetic algorithm laboratory (KanGAL).[3] And as it has incorporated only 18 among the 33 test problems that we have considered, we compare AMGA2 only for those 18 test problems. Since, the jMetal implementation of MOEA/D-DE cannot handle constrained problems, we have considered only the unconstrained problems for comparing with it. We have considered 25 000 and 50 000 function evaluations for each run of each of the six algorithms: AMGA2, MOEA/D-DE, AbYSS, NSGA-II, SPEA2, and proposed ASMiGA. All these six algorithms are executed for 100 independent runs for each test problem. For NSGA-II, SPEA2, AbYSS, and AMGA2 we have used the same parameter settings as used by their respective authors. For MOEA/D-DE we have changed some of the parameter settings so that it can be applied with an archive of size 100. All other settings are the same as used by the authors of MOEA/D-DE.

The following experimental settings have been used for these six algorithms.

1) SPEA2: archive size = 100, distribution index of SBX crossover η_c = 20, probability of SBX crossover p_c = 0.9, distribution index of polynomial mutation η_m = 20, probability of mutation p_m = 1/number of variables.
2) NSGA-II: population size = 100, all other parameters are same as SPEA2.
3) AbYSS: population size P = 20, archive size = 100, size of *RefSet1* = 10, size of *RefSet2* = 10, distribution index of SBX crossover η_c = 20, probability of SBX crossover p_c = 1.0, number of improvement rounds = 1.

[2]jMetal can be freely downloaded from: http://jmetal.sourceforge.net.

[3]This implementation of AMGA2 can be freely downloaded from: http://www.iitk.ac.in/kangal/codes.shtml.

TABLE II
NUMBER AND PERCENTAGE OF PROBLEMS FOR WHICH AN ALGORITHM SHOWS THE BEST PERFORMANCE

Algorithms		ASMiGA 33 Test Problems		AMGA2 18 Test Problems		MOEA/D-DE 26 Test Problems		AbYSS 33 Test Problems		NSGA-II 33 Test Problems		SPEA2 33 Test Problems	
Function Evaluations		25000	50000	25000	50000	25000	50000	25000	50000	25000	50000	25000	50000
Performance Metrics	HV	23 (69.7%)	23 (69.7%)	3 (16.7%)	5 (27.8%)	2 (07.7%)	2 (07.7%)	4 (12.1%)	8 (24.2%)	2 (06.0%)	2 (06.0%)	1 (03.0%)	1 (03.0%)
	GD	15 (45.5%)	15 (45.5%)	1 (05.6%)	1 (05.6%)	6 (23.1%)	6 (23.1%)	4 (12.1%)	6 (18.2%)	4 (12.1%)	3 (09.1%)	4 (12.1%)	2 (06.0%)
	IGD	20 (60.6%)	16 (48.5%)	4 (22.2%)	3 (16.7%)	4 (15.4%)	2 (07.7%)	6 (18.2%)	10 (30.3%)	1 (03.0%)	2 (06.0%)	0 (00.0%)	0 (00.0%)
	GS	21 (63.6%)	19 (57.6%)	7 (38.9%)	6 (33.3%)	0 (00.0%)	0 (00.0%)	4 (12.1%)	8 (24.2%)	0 (00.0%)	0 (00.0%)	1 (03.0%)	0 (00.0%)

4) MOEA/D: population size $N = 100$, parameters for DE crossover $F = 0.5$, $CR = 1.0$, distribution index of polynomial mutation $\eta_m = 20$, probability of mutation $p_m = 1$/number of variables, neighbor size $T = 20$, maximal number of solutions replaced by each child solution $n_r = 2$, probability that parent solutions are selected from the neighborhood $\delta = 0.9$; weight vector parameter $H = 99$ for bi-objective, 13 for tri-objective, and 5 for penta-objective problems. Thus, 100, 105, and 126 weight vectors are generated, respectively. Then size of these sets of weight vectors are reduced to 100 using the method described in [18].
5) AMGA2: population size = 4 M, where M is the number of objectives; archive size = 100, parameters for DE-2 crossover $F = 0.5$, $CR = 0.1$, distribution index of polynomial mutation $\eta_m = 15$, minimum probability of mutation $p_{\min} = 1$/number of variables.
6) ASMiGA: minimum archive size $N_{\min} = 20$ for unconstrained and 25 for constrained problems, maximum archive size $N_{\max} = 100$, selection ratio $S_r = 0.25$ for SBX crossover and 0.15 for DE-3 crossover, parameters for DE-3 crossover $F = 0.5$, $CR = 0.1$, distribution index of SBX crossover $\eta_c = 20$, probability of SBX crossover $p_c = 1.0$, distribution index of polynomial mutation $\eta_m = 20$, probability of mutation $p_m = 1$/number of variables.

E. Results

We have considered median and interquartile range (IQR) as the statistical measures of central tendency and dispersion. For each problem and for each algorithm, we have computed the median and IQR of the 100 runs for each of the four metrics (HV, GD, IGD, and GS). The performance of the six algorithms in terms the median and IQR values of all four indices are summarized in Table II. Table II shows the number and percentage of test problems for which an algorithm exhibits the best performance. The cells, which exhibit the maximum percentage, are darkened in the table. The detailed values of the four indices with both 25 000 and 50 000 generations are provided in the supplementary materials as Tables S-I–S-VIII. The cells corresponding to the best median values are also darkened in these tables.

Table II reveals that ASMiGA outperforms other five algorithms in terms of all four metrics for both the cases with 25 000 and 50 000 function evaluations.

Table II shows that ASMiGA yields the best values of HV metric for 23 out of 33, i.e., 69.7% test problems for both 25 000 and 50 000 function evaluations. Similarly, ASMiGA shows the second best performance for eight (Schaffer, Fonseca, ZDT6, WFG1, Viennet4, DTLZ3, DTLZ5, and DTLZ6) and five (ZDT6, WFG4, Viennet4, DTLZ5, and DTLZ6) test problems respectively, with 25 000 and 50 000 function evaluations. So, out of 33 test cases ASMiGA shows either the best or the second best performance for 31 (23 + 8), i.e., 93.9% and 28 (23 + 5), i.e., 84.8% problems respectively, with 25 000 and 50 000 function evaluations. ASMiGA gives the best result for at least one test problem in each group for both with 25 000 and 50 000 function evaluations. Since, HV metric indicates both closeness to the PF and diversity of the solution set, we may conclude that ASMiGA is capable of providing well diverse as well as near PF solution set even with a lower number of function evaluations compared to the other algorithms. According to the values of HV, AMGA2 shows the second best performing algorithm. AMGA2 provides the best results for 3 and 5 out of 18 test problems respectively, with 25 000 and 50 000 function evaluations. SPEA2 shows the worst performance by providing the best result only for one test problem (Viennet4) in both cases with 25 000 and 50 000 function evaluations.

From the HV values (Tables S-I and S-V) the following points are evident.

1) The output of ASMiGA with 50 000 function evaluations remain very close (almost unchanged) to the outputs obtained with 25 000 function evaluations. But, ASMiGA is comparatively benefited by higher function evaluations for complicated problems, e.g., problems in DTLZ family.
2) AbYSS and AMGA2 are mainly benefited by deeper exploration.
3) Some algorithms (mainly ASMiGA, AMGA2, and AbYSS) provide the same or almost the same output (mostly for ZDT family) with 50 000 function evaluations. From these observations, we can infer the following.
    a) With 50 000 function evaluations, for less complicated test problems all solutions by all methods lie on or very close to PF.
    b) ASMiGA has a much faster convergence rate compared to other five algorithms.

ASMiGA performs the best according to GD metric providing the best result for 15 out of 33 test problems, i.e., 45.5%

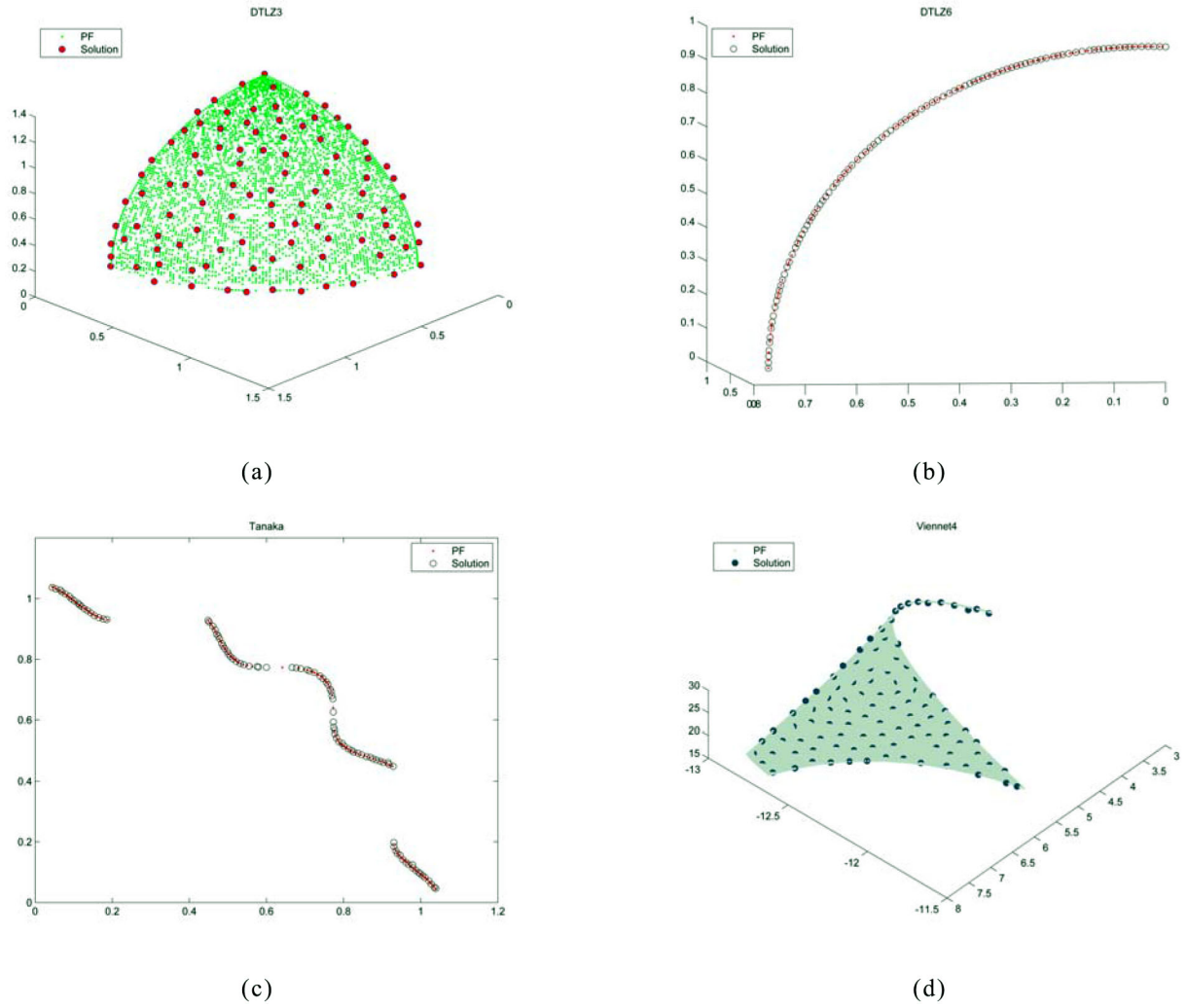

Fig. 2. Pareto fronts and obtained solution sets with 25 000 function evaluations of ASMiGA for (a) DTLZ3, (b) DTLZ6, (c) Tanaka, and (d) Viennet4.

test cases for both with 25 000 and 50 000 function evaluations, as shown in Table II (details in Tables S-II and S-VI). As measured by GD metric, both with 25 000 and 50 000 function evaluations, MOEA/D-DE performs the second best and AMGA2 exhibits the worst performance (Table II).

According to the IGD metric, as depicted in Table II (details in Tables S-III and S-VII), the proposed algorithm ASMiGA yields the best performance for 20 and 16 out of 33 problems, i.e., 60.6% and 48.5% test cases, respectively, with 25 000 and 50 000 function evaluations. Table II reveals that not only ASMiGA but also AMGA2 and MOEA/D-DE perform poorer with higher number of function evaluations. On the contrary, AbYSS is the most profited algorithm by the same cause. According to IGD metric, ASMiGA always provides the best result for at least one problem in each of the test groups. The proposed algorithm provides excellent results for all problems of group four (having more than two objectives). For the case with 25 000 function evaluations, AMGA2 is the second best performing algorithm while; with 50 000 function evaluations, AbYSS shows the second best performance.

Table II (details in Tables S-IV and S-VIII) further reveals that according to GS, ASMiGA shows the best and second best results respectively, for 21 (63.6%) and eight test problems with 25 000 function evaluations; while it shows the best and the second best results respectively, for 19 (57.6%) and nine test problems with 50 000 function evaluations. ASMiGA yields the best result according to GS metric for at least one test problem in each test group. According to this metric, ASMiGA results in an excellent performance for group five (having more than two objectives, see Tables S-IV and S-VIII for details) showing the best result for all four problems for both with 25 000 and 50 000 function evaluations. Admirable performance of ASMiGA indicated by GS metric for more than bi-objective test problems (group five and six) demonstrates that the diversity preservation mechanism of the proposed archive maintenance strategy of ASMiGA is well suited for comparatively higher dimensional objective space. In terms of this metric, AMGA2 shows the second best performance for both with 25 000 and 50 000 function evaluations. With both 25 000 and 50 000 function evaluations, in terms of GS metric, MOEA/D-DE and NSGA-II exhibit the worst performance.

Among the six algorithms, AMGA2, MOEA/D-DE, NSGA-II, and SPEA2 yield the worst performance (in terms of number of problems, for which they provide the best results) at

least in one metric either with 25 000 or with 50 000 function evaluations (Table II). AbYSS yields quite poor performance for problems having more than two objectives (group five and six) according to the GS metric for both 25 000 and 50 000 function evaluations. For these two groups, AbYSS cannot produce the best result for any one of the test problems (details in Tables S-IV and S-VIII).

In Fig. 2, we depict the true PF and solutions obtained by ASMiGA with 25 000 function evaluations for two unconstrained problems DTLZ3 and DTLZ6, and for two constrained test problems Tanaka and Viennet4. We show the output of the first run of 100 random runs. The figures depict that for all these four problems ASMiGA can find well diverse solutions which are also close to the PF. They also cover the entire spectrum of the PF.

Though, ASMiGA performs the best in terms of for all four metrics for both 25 000 and 50 000 function evaluations, the overall performance of ASMiGA with 25 000 function evaluations is better (especially for IGD) than that with 50 000 function evaluations compared to other algorithms. Inspection of the detailed results reveals that for some test problems, the HV values are almost the same for ASMiGA with 25 000 and 50 000 function evaluations while this is not the case for some other algorithms. This suggests that ASMiGA has a faster convergence rate compared to the other algorithms and that makes it more suitable for real world problems, where each function evaluation demands high computational cost. But, for the classical test problems (Schaffer, Fonseca, and Kurswae), ASMiGA performs comparatively poor (with respect to its performance on other test groups). These three problems are relatively easier to solve as they are unconstrained, bi-objective, and have fewer number of variables compared to other test problems. On the other hand, ASMiGA performs very well for Viennet4 (3 objectives and 3 constraints) and Water (five objectives and seven constraints) test problems. Noticeably ASMiGA performs the best (always best according to all four metrics for both 25 000 and 50 000 function evaluations) for Water, which has five objectives and seven constraints. If we consider only the seven constrained test problems, according to all four performance measures, ASMiGA performs the best for 37 out of 56 (four measures, seven test problems, both with 25 000 and 50 000 function evaluations), i.e., 66.0%, test cases. In this limited experiment, we find that for constrained problems, ASMiGA outperforms other algorithms.

To check whether the improved performance of ASMiGA is statistically significant over the other algorithms, we have used the Wilcoxon signed-rank test [33] on each of the four performance indices. Table III shows that in most cases the null hypothesis is rejected for both cases with 25 000 and 50 000 function evaluations.

To summarize, the steady-state nature of ASMiGA allows it to be suitable for problems when parallelism in function evaluation is not possible, and higher convergence of the algorithm makes it suitable when maximum computational effort (number of evaluations) is given. ASMiGA is well suited for real-world problems when: 1) function evaluation is costly; 2) no parallelism is allowed; and 3) computational effort is fixed.

TABLE III
WILCOXON SIGNED-RANK TEST (1-TAILED) FOR PAIR WISE COMPARISON WITH ASMiGA AT LEVEL OF SIGNIFICANCE $\alpha = 0.05$

Metrics	Function Evaluations	AMGA2	MOEA/ D-DE	AbYSS	NSGA-II	SPEA2
HV	25000	R	R	R	R	R
	50000	R	R	R	R	R
GD	25000	R	R	R	R	R
	50000	R	A	A	R	R
IGD	25000	A	R	R	R	R
	50000	A	R	R	R	R
GS	25000	A	R	R	R	R
	50000	A	R	R	R	R

R: H_0 Rejected, A: H_0 Accepted

F. Discussion

ASMiGA uses two additional parameters and proper choice of those parameters is important for a better performance of ASMiGA. An important parameter of ASMiGA is the choice of selection ratio (S_r). Based on some experiments, we found the value of S_r. For this, we tested ASMiGA for $S_r = 0.0$, 0.3, 0.6, 0.9, and 1.0 for all the test problems of WFG group and found that the best result is obtained with 0.30. Then we tested for $S_r = 0.35$, 0.25. We got better result with 0.25. We further tested with 0.20, 0.15, and 0.10. We plotted these results and found that $S_r = 0.15$ gives the best result. During this exercise, all test problems in WFG group were executed for 25 000 function evaluations for 100 times. We considered the median of the *HV* as the indicator. Undeniably, the procedure is *ad-hoc* and S_r can be fine-tuned further. Problem specific values of S_r may also lead superior performance of the algorithm. Note that, we have chosen the value of S_r using only WFG test problems but applied the same value for all test problems.

The parameter, $N_{\min}$, is also an additional parameter for our algorithm. A high value of $N_{\min}$ usually will make the algorithm start with too many dominated solutions, which is not good. At the same time, we want to start with a diverse population. A reasonable choice would be to take $N_{\min}$ between 20% and 30% of $N_{\max}$ so that when the algorithm generates $N_{\max}$ solutions in the archive, it will have a good nondominated set.

Dynamic archive size is one of the most important features of ASMiGA. To show how dynamic sized archive helps to enhance the performance of the proposed algorithm, we have simulated 100 runs of DTLZ3 and DTLZ6 test problems. For DTLZ3, after each 1000 function evaluations, we have calculated the *HV* corresponding to the nondominated solutions of the current archive. We then compute the average values of *HV* over the 100 runs and plot it in Fig. 3(a) with respect to the number of function evaluations. Similarly, the mean of the ratio of the current archive size to the maximum archive size (RCASMAS) is also plotted in the same figure. In Fig. 3(b), we make a similar plot for DTLZ6 but in this case *HV* is computed after every 100 function evaluations just to maintain the same level of clarity in Fig. 3(b) as that of Fig. 3(a). Fig. 3 depicts that when $N_{\min} = 100$, i.e., the archive size is fixed, the convergence rate is slower. When the *HV* saturates the archive size also gets (almost) fixed at $N_{\max}$.

The choice between SBX and DE-3 crossover is an important issue. SBX is not self-adaptable while DE is

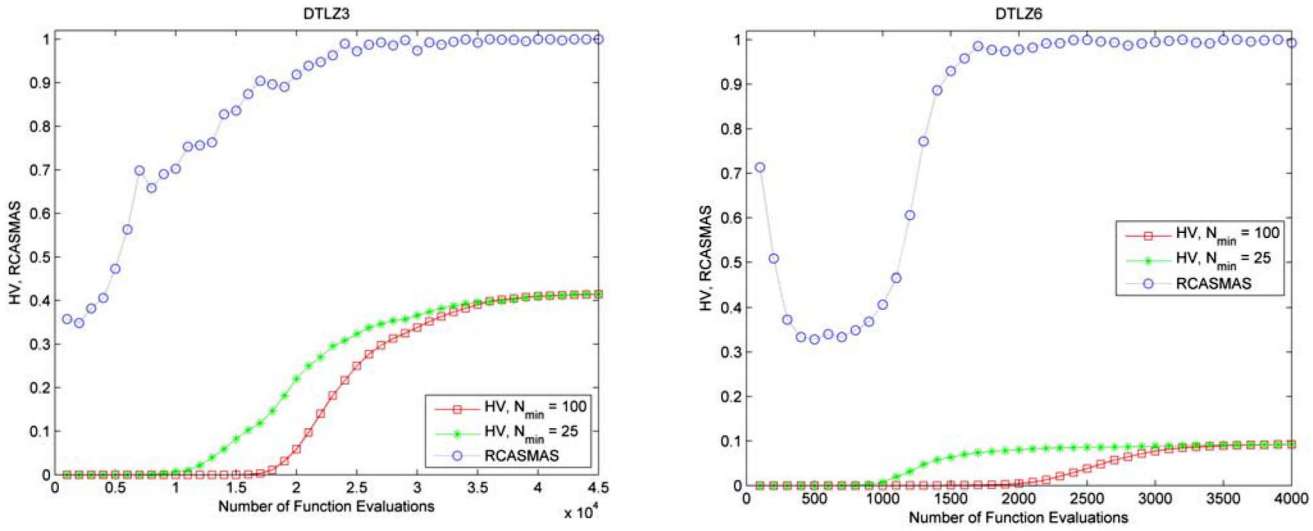

Fig. 3. Hypervolume and ratio of the current archive size to the maximum archive size (RCASMAS) averaged over 100 runs versus number of function evaluations for (a) DTLZ3 and (b) DTLZ6.

self-adaptable [5]. The effect of DE significantly depends on its parameters. Here, we have used $CR = 0.1$ and hence most of the genes of any off-spring are from the primary parent making it more exploitive. But, when the number of variables is not large (e.g., less than four) DE crossover may become enough explorative even with $CR = 0.1$. The off-spring will have only 0%, 45%, and 60% genes from its primary parent respectively when number of variables (n) is one, two, and three. This off-spring will then be mutated with 100%, 50%, and 33.33% probability of mutation for $n = 1$, 2, and 3 respectively. So, when the number of variables is small, DE crossover in conjunction with the mutation can become quite explorative. If the number of objectives, and/or the number of variables, and/or the number of constraints are high, then searching is likely to be complicated, because the first two conditions increase the search space, and the third condition may introduce discontinuities or holes in the search space. Therefore, we should concentrate on exploitation, and hence use of DE crossover with a low value of CR may be more effective. Yet, it is quite difficult to conclude anything precisely about the performance of an algorithm based on the parameters of DE. However, our limited experiments indeed reveal that for relatively less complex problems (with 2–3 objective functions or variables or constraints) SBX works better than DE while for larger more complex problems DE works better than SBX.

Except AbYSS, all other algorithms mentioned here use polynomial mutation with mutation probability $p_m = 1$/number of variables, and the number of variables for Schaffer is only one. So, for Schaffer $p_m = 1$, which suggests that all algorithms will be highly explorative for Schaffer and the off-spring will have no gene from its parents. Moreover, AbYSS uses a much more guided local search. This may be the reason behind the superior performance of AbYSS for Schaffer.

It is worth noting that only AbYSS, NSGA-II, and SPEA2 show zero value of *HV* metric for DTLZ3 and DTLZ6 test problems with 25 000 function evaluations and these three algorithms use SBX crossover while the other three algorithms use DE crossover. Genetic variation operator is an important factor in GA. The use of SBX crossover may be a contributor to the poorer performance of AbYSS, NSGA-II, and SPEA2.

Despite the promising result of ASMiGA, it raises a few issues opening up several avenues for further work.

1) Constraints are handled using the constraint domination method described in [1]. The amount of constraint violation could be used in mating selection to make the search more guided.
2) It uses crowding distance in bi-objective problems and the distances of the first and second nearest neighbors for problems with more than two objectives. It would be better to use only one measure for diversity.
3) ASMiGA uses either DE-3 or SBX depending on the problem characteristics. However, it would be better if we could use a single crossover operator for all problems.
4) The algorithm requires two extra parameters—$N_{\min}$ and S_r. Although, we have used the same fixed values for a wide range of problems, a systematic study for their choices is needed.
5) The CH has been done by an extension of single objective CH strategy. CH techniques, especially designed for constrained MOP might perform better for constrained problems.

V. Conclusion

We have proposed a new ASMiGA. ASMiGA uses new environmental and mating selection strategies. The proposed environmental selection strategy reduces exploration in less probable search region and the mating selection enhances exploration in more probable search region by increasing the exploitation of more desirable solutions. A proper tradeoff between exploration and exploitation is achieved by using

a parameter, called selection ratio, S_r. The archive maintenance strategy tries to maintain only nondominated solutions in the archive unless they are not sufficient to fill the minimum required archive size. A new crossover DE-3 has also been proposed, which increases the exploitation of existing solutions. ASMiGA uses either SBX or DE-3 crossover depending on the nature of the problem, which is determined by the number of variables, number of objectives, and number of constraints. The steady-state nature of ASMiGA maximizes the number of generations for a fixed number of function evaluations. The micro genetic nature of the algorithm makes the search much more guided in the search space. ASMiGA uses polynomial mutation as its default mutation operator.

We have compared the performance of ASMiGA with AMGA2, MOEA/D-DE, AbYSS, NSGA-II, and SPEA2 using four popular performance measures, HV, GD, IGD, and GS on several test problems. The experimental results show that ASMiGA outperforms the other five algorithms by providing a solution set, which is closer to the PF and well diverse. ASMiGA has a much faster convergence rate than the other algorithms. The proposed algorithm works very well especially in comparatively higher dimensional objective space and for constrained problems. ASMiGA is well suited when each function evaluation is costly, parallelism is not allowed, and the maximum computational effort (number of function evaluations) is fixed, and these make the algorithm very useful for real world problems.

In our view, the enhancement in the performance is a combined effect of all new ingredients. For example, our new environmental selection strategy reduces the chances of exploring in the less desirable area of the objective space. Our archive truncation strategy prunes solutions efficiently maintaining the diversity in the archive. It keeps only nondominated solutions in the archive unless archive size falls below a minimum limit. The new mating selection helps to exploit the desirable solutions. The new crossover operation, DE-3 is quite effective for steady-state genetic algorithm. We think, the new environmental selection scheme is the most effective of all these.

In future, we intend to improve the proposed algorithm by addressing the issues mentioned in the previous section.

ACKNOWLEDGMENT

The authors would like to thank J. J. Durillo, S. Tiwari, and Q. Zhang for their help. They are also thankful to the referees for their valuable suggestions. They also thank the jMetal team for providing such a nice framework.

REFERENCES

[1] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[2] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," Comput. Eng. Netw. Lab. (TIK), ETH, Zurich, Switzerland, Tech. Rep. 103, 2001.

[3] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of generalized differential evolution," in *Proc. IEEE CEC*, Edinburgh, Scotland, 2005, pp. 443–450.

[4] S. Tiwari, P. Koch, G. Fadel, and K. Deb, "AMGA: An archive-based micro genetic algorithm for multi-objective optimization," in *Proc. GECCO*, 2008, pp. 729–736.

[5] S. Tiwari, G. Fadel, and K. Deb, "AMGA2: Improving the performance of the archive-based micro-genetic algorithm for multi-objective optimization," *Eng. Optim.*, vol. 43, no. 4, pp. 371–401, 2011.

[6] J. J. Durillo, A. J. Nebro, F. Luna, and E. Alba, "On the effect of the steady-state selection scheme in multi-objective genetic algorithms," in *Proc. 5th Int. Conf. EMO*, Nantes, France, Apr. 2009, pp. 183–197.

[7] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications," Doctoral dissertation ETH 13398, Computer Engineering and Networks Laboratory, SWISS Federal Institute of Technology, Zurich, Switzerland, 1999.

[8] K. Deb and S. Tiwari, "Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1062–1087, 2006.

[9] H. Eskandari, C. D. Geiger, and G. B. Lamnot, "FastPGA: A dynamic population sizing approach for solving expensive multiobjective optimization problems," in *Proc. EMO*, Matsushima, Japan, pp. 141–155, 2007.

[10] D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto envelop-based selection algorithm for multiobjective optimization," in *Proc. PPSN*, Paris, France, 2000, pp. 839–848.

[11] M. Kim, T. Hiroyasu, M. Miki, and S. Watanabe, "SPEA2+: Improving the performance of the strength Pareto evolutionary algorithm 2," in *Proc. PPSN*, Birmingham, U.K., 2004, pp. 742–751.

[12] H. Lu and G. G. Yen, "Rank-density-based multiobjective genetic algorithm and benchmark test function study," *IEEE Trans. Evol. Comput.*, vol. 7, no. 4, pp. 325–343, Aug. 2003.

[13] G. G. Yen and H. Lu, "Dynamic multiobjective evolutionary algorithm: Adaptive cell-based rank and density estimation," *IEEE Trans. Evol. Comput.*, vol. 7, no. 3, pp. 253–274, Jun. 2003.

[14] K. Deb, M. Mohan, and S. Mishra, "Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions," *Evol. Comput. J.*, vol. 13, no. 4, pp. 501–525, 2005.

[15] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates, "PESA-II: Region based selection in multiobjective evolutionary optimization," in *Proc. 6th Int. Conf. PPSN-VI*, 2000, pp. 839–848.

[16] A. J. Nebro *et al*., "AbYSS: Adapting scatter search to multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 4, pp. 439–453, Aug. 2008.

[17] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[18] Q. Zhang, W. Liu, and H. Li, "The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances," in *Proc. 11th Conf. CEC*, Trondheim, Norway, 2009, pp. 203–208.

[19] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 284–302, Apr. 2009.

[20] C. A. C. Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Knowl. Inf. Syst.*, vol. 1, no. 3, pp. 269–308, 1999.

[21] C. A. C. Coello, "Evolutionary multi-objective optimization: A historical view of the field," *IEEE Comput. Intell. Mag.*, vol. 1, no. 1, pp. 28–36, Feb. 2006.

[22] A. Konak, D. W. Cott, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliab. Eng. Sys. Safety*, vol. 91, no. 9, pp. 992–1007, Sep. 2006.

[23] C. A. C. Coello, "List of references on evolutionary multiobjective optimization," Tech. Rep. CINVESTAV-IPN, 2009 [Online]. Available: http://delta.cs.cinvestav.mx/ccoello/EMOO

[24] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Opt.*, vol. 11, no. 4, pp. 341–359, 1997.

[25] W. L. Loh, "On Latin hypercube sampling," *Ann. Stat.*, vol. 33, no. 6, pp. 2058–2080, 2005.

[26] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2000.

[27] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multi-objective optimization," in *Evolutionary Multiobjective Optimization*, A. Abraham, L. Jain, and R. Goldberg, Eds. Berlin, Germany: Springer-Verlag, 2005, pp. 105–145.

[28] J. J. Durillo, A. J. Nebro, F. Luna, B. Dorronsoro, and E. Alba, "jMetal: A Java framework for developing multi-objective optimization metaheuristics," Dept. Lenguajes y Ciencias de la Computación, Univ. Málaga, Málaga, Spain, E.T.S.I. Informática, Campus de Teatinos, Tech. Rep. ITI-2006-10, 2006.

[29] K. Deb, *Multi Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: Wiley, 2001.

[30] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: A history and analysis," Dept. Elec. Comput. Eng., Graduate School Eng., Air Force Inst. Technol., Wright-Patterson AFB, OH, USA, Tech. Rep. TR-98-03, 1998.

[31] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang, "Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion," in *Proc. IEEE CEC*, Vancouver, BC, Canada, 2006, pp. 3234–3241.

[32] S. Kukkonen and K. Deb, "Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems," KanGAL 200607, IIT, Kanpur, India, Tech. Rep. 7, 2006.

[33] J. Derrac, S. Garcia, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011.

[34] O. Isao, S. Hiroshi, and K. Shigenobu, "Areal-coded genetic algorithm for function optimization using the unimodal normal distribution crossover," *J. Jap. Soc. Artif. Intell.*, vol. 14, no. 6, pp. 1146–1155, 1999.

[35] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst.*, vol. 9, no. 2, pp. 115–148, 1995.

[36] K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter optimization," *Evol. Comput. J.*, vol. 10, no. 4, pp. 371–395, 2002.

[37] K. Miettinen, *Nonlinear Multiobjective Optimization*. Norwell, MA, USA: Kluwer, 1999.

[38] J. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algor.*, Hillsdale, NJ, USA, 1987, pp. 93–100.

[39] C. Fonseca and P. Flemming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms—Part II: Application example," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, no. 1, pp. 38–47, Jan. 1998.

[40] F. Kursawe, "A variant of evolution strategies for vector optimization," in *Proc. PPSN*, H. Schwefel and R. Männer, Eds. Berlin, Germany: Springer-Verlag, 1990, pp. 193–197.

[41] S. Huband, L. Barone, R. L. While, and P. Hingston, "A scalable multi-objective test problem toolkit," in *Proc. EMO*, Guanajuato, Mexico, 2005, pp. 280–295.

[42] A. Osyczka and S. Kundo, "A new method to solve generalized multicriteria optimization problems using a simple genetic algorithm," *Struct. Optim.*, vol. 10, no. 2, pp. 94–99, 1995.

[43] M. Tanaka, H. Watanabe, Y. Furukawa, and T. Tanino, "GA-based decision support system for multicriteria optimization," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Vancouver, BC, Canada, 1995, pp. 1556–1561.

[44] N. Srinivas and K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1995.

[45] R. Viennet, C. Fontiex, and I. Marc, "Multicriteria optimization using a genetic algorithm for determining a Pareto set," *J. Syst. Sci.*, vol. 27, no. 2, pp. 255–260, 1996.

[46] T. Ray, K. Tai, and K. Seow, "An evolutionary algorithm for multiobjective optimization," *Eng. Optim.*, vol. 33, no. 3, pp. 399–424, 2001.

[47] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.

[48] K. Nag and T. Pal, "A new archive-based steady-state genetic algorithm," in *Proc. IEEE WCCI*, Brisbane, QLD, Australia, Jun. 2012, pp. 853–859.

[49] E. Mezura-Montes and C. A. Coello Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm Evol. Comput.*, vol. 1, no. 4, pp. 173–194, 2011.

[50] T. Ray, T. Kang, and S. K. Chye, "An evolutionary algorithm for constrained optimization," in *Proc. GECCO*, San Francisco, CA, USA, 2000, pp. 771–777.

[51] A. Isaacs, T. Ray, and W. Smith, "Blessings of maintaining infeasible solutions for constrained multi-objective optimization problems," in *Proc. CEC*, Hong Kong, China, 2008, pp. 2785–2792.

[52] T. Ray, H. K. Singh, A. Isaacs, and W. Smith, "Infeasibility driven evolutionary algorithm for constrained optimization," in *Constraint Handling in Evolutionary Optimization*, E. Mezura-Montes, Ed. Berlin, Germany: Springer-Verlag, 2009, pp. 145–165.

[53] K. Harada, J. Sakuma, I. Ono, and S. Kobayashi, "Constraint-handling method for multi-objective function optimization: Pareto descent repair operator," in *Proc. 4th Int. Conf. EMO*, Matshushima, Japan, 2007, pp. 156–170.

[54] N. Young, "Blended ranking to cross infeasible regions in constrained multiobjective problems," in *Proc. CIMCA-IAWTIC*, Vienna, Austria, 2005, pp. 191–196.

[55] H. K. Singh, T. Ray, and W. Smith, "C-PSA: Constrained Pareto simulated annealing for constrained multi-objective optimization," *Inf. Sci.*, vol. 180, no. 13, pp. 2499–2513, 2010.

[56] B. Qu and P. Suganthan, "Constrained multi-objective optimization algorithm with ensemble of constraint-handling methods," *Eng. Optim.*, vol. 43, no. 4, pp. 403–416, 2011.

[57] K. Deb, A. Sinha, and S. Kukkonen, "Multi-objective test problems, linkages, and evolutionary methodologies," in *Proc. GECCO*, Washington, DC, USA, 2006, pp. 1141–1148.

[58] A. W. Iorio and X. Li, "Solving rotated multi-objective optimization problems using differential evolution," in *Proc. AJCAI*, Cairns, QLD, Australia, 2004, pp. 861–872.

[59] J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 50–66, Feb. 2005.

[60] K. C. Giannakoglou, "Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence," *Int. Rev. J. Progr. Aerosp. Sci.*, vol. 38, no. 1, pp. 43–76, 2002.

[61] Z. S. Davies *et al*., "Efficient improvement of silage additives by using genetic algorithms," *Appl. Environ. Microbiol.*, vol. 66, no. 4, pp. 1435–1443, Apr. 2000.

[62] J. R. G. Evans, M. J. Edirisinghe, and P. V. C. J. Eames, "Combinatorial searches of inorganic materials using the inkjet printer: Science philosophy and technology," *J. Eur. Ceramic Soc.*, vol. 21, no. 13, pp. 2291–2299, 2001.

[63] D. Weuster-Botz and C. Wandrey, "Medium optimization by genetic algorithm for continuous production of formate dehydrogenase," *Process Biochem.*, vol. 30, no. 6, pp. 563–571, 1995.

[64] S. Vaidyanathan, D. I. Broadhurst, D. B. Kell, and R. Goodacre, "Explanatory optimization of protein mass spectrometry via genetic search," *Anal. Chem.*, vol. 75, no. 23, pp. 6679–6686, 2003.

[65] S. Vaidyanathan, D. Kell, and R. Goodacre, "Selective detection of proteins in mixtures using electrospray ionization mass spectrometry: Influence of instrumental settings and implications for proteomics," *Anal. Chem.*, vol. 76, no. 17, pp. 5024–5032, 2004.

[66] A. Zhou *et al*., "Multiobjective evolutionary algorithms: A survey of the state-of-the-art," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, Mar. 2011.

[67] B. Y. Qu and P. N. Suganthan, "Multi-objective evolutionary algorithms based on the summation of normalized objectives and diversified selection," *Inf. Sci.*, vol. 180, no. 17, pp. 3170–3181, 2010.

[68] B. Y. Qu and P. N. Suganthan, "Multi-objective differential evolution based on the summation of normalized objectives and improved selection method," in *Proc. IEEE SDE*, Paris, France, Apr. 2011, pp. 1–8.

[69] C. Aporntewan and P. Chongstitvatana, "A hardware implementation of the compact genetic algorithm," in *Proc. CEC*, Seoul, Korea, 2001, pp. 624–629.

[70] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 287–297, Nov. 1999.

[71] Z. H. Zhan *et al*., "Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 445–463, Apr. 2013.

[72] H. Xia, J. Zhuang, and D. Yu, "Combining crowding estimation in objective and decision space with multiple selection and search strategies for multi-objective evolutionary optimization," *IEEE Trans. Cybern.*, vol. 44, no. 3, pp. 378–393, May 2013.

[73] L. Ke, Q. Zhang, and R. Battiti, "MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and ant colony," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1845–1859, Dec. 2013.

Kaustuv Nag received the B.Tech. degree in computer science and engineering from the West Bengal University of Technology, Kolkata, India, and the M.Tech. degree in computer science and engineering from the National Institute of Technology, Durgapur, India, in 2010 and 2012, respectively, and is currently pursuing the Ph.D. degree from Jadavpur University, Kolkata.

He was a Visiting Researcher at Indian Statistical Institute, Kolkata. His current research interests include genetic algorithm, genetic programming, and artificial neural networks.

Mr. Nag is a recipient of INSPIRE Fellowship.

Tandra Pal (M'02) received the B.Sc. degree (Hons.) in physics, the B. Tech. degree in computer science from Kolkata University, Kolkata, India, the M.E. degree in computer science, and the Ph.D. degree in engineering from Jadavpur University, Kolkata, India, in 1986, 1991, 1993, and 2010, respectively.

She joined the National Institute of Technology, Durgapur, India, in 1994, and is currently an Associate Professor with the Computer Science and Engineering Department. Her current research interest includes fuzzy sets theory, fuzzy systems including fuzzy control, fuzzy decision-making, artificial neural networks, and genetic algorithms.

Nikhil R. Pal (M'91–SM'00–F'05) is a Professor with the Electronics and Communication Sciences Unit of the Indian Statistical Institute. His current research interest includes bioinformatics, brain science, fuzzy logic, pattern analysis, neural networks, and evolutionary computation.

Dr. Pal was the Editor-in-Chief of the IEEE TRANSACTIONS ON FUZZY SYSTEMS from January 2005 to December 2010. He has served/been serving on the editorial/advisory board/steering committee of several journals, including the *International Journal of Approximate Reasoning, Applied Soft Computing, International Journal of Knowledge-Based Intelligent Engineering Systems, International Journal of Neural Systems, Fuzzy Sets and Systems, International Journal of Intelligent Computing in Medical Sciences and Image Processing, Fuzzy Information and Engineering: An International Journal,* IEEE TRANSACTIONS ON FUZZY SYSTEMS, and the IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS—CYB. He has given several plenary/keynote speeches in different premier international conferences in the area of computational intelligence. He has served as the General Chair, Program Chair, and Co-Program Chair of several conferences. He was a Distinguished Lecturer of the IEEE Computational Intelligence Society (CIS) and was a member of the Administrative Committee of the IEEE CIS. He is currently the Vice President for Publications of the IEEE CIS. He is a fellow of the National Academy of Sciences, India, a fellow of the Indian National Academy of Engineering, a fellow of the Indian National Science Academy, and a fellow of the International Fuzzy Systems Association.