A Fast and Elitist Multi-Objective **Genetic** Algorithm: **NSGA**-II

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan
Kanpur **Genetic** Algorithms Laboratory (KanGAL)
Indian Institute of Technology Kanpur
Kanpur, PIN 208 016, India
deb@iitk.ac.in

KanGAL Report No. 200001

Abstract

Multi-objective evolutionary algorithms which use non-dominated sorting and sharing have been mainly criticized for their (i) $O(MN^3)$ computational complexity (where M is the number of objectives and N is the population size), (ii) non-elitism approach, and (iii) the need for specifying a sharing parameter. In this paper, we suggest a non-dominated sorting based multi-objective evolutionary algorithm (we called it the Non-dominated Sorting GA-II or **NSGA**-II) which alleviates all the above three di culties. Speci cally, a fast non-dominated sorting approach with $O(MN^2)$ computational complexity is presented. Second, a selection operator is presented which creates a mating pool by combining the parent and child populations and selecting the best (with respect to tness and spread) N solutions. Simulation results on a number of di cult test problems show that the proposed **NSGA**-II, in most problems, is able to nd much better spread of solutions and better convergence near the true Pareto-optimal front compared to PAES and SPEAltwo other elitist multi-objective EAs which pay special attention towards creating a diverse Pareto-optimal front. Moreover, we modify the de nition of dominance in order to solve constrained multi-objective problems e ciently. Simulation results of the constrained **NSGA**-II on a number of test problems, including a ve-objective, seven-constraint non-linear problem, are compared with another constrained multi-objective optimizer and much better performance of **NSGA**-II is observed. Because of **NSGA**-II's low computational requirements, elitist approach, parameter-less niching approach, and simple constraint-handling strategy, **NSGA**-II should nd increasing applications in the coming years.

Keywords

Multi-objective optimization, **Genetic** algorithms, Multi-criterion decision making, Elitism, Constraint handling, Pareto-optimal solutions.

I. Introduction

The presence of multiple objectives in a problem, in principle, gives rise to a set of optimal solutions (largely known as Pareto-optimal solutions), instead of a single optimal solution. In the absence of any further information, one of these Pareto-optimal solutions cannot be said to be better than the other. This demands an user to nd as many Pareto-optimal solutions as possible. Classical optimization methods (including the multi-criterion decision-making (MCDM) methods) suggest converting the multi-objective optimization problem to a single-objective optimization problem by emphasizing one particular Pareto-optimal solution at a time. When such a method is to be used for nding multiple solutions, it has to be applied many times, hopefully nding a di erent solution at each simulation run.

Over the past decade, a number of multi-objective evolutionary algorithms (MOEAs) have been suggested [18], [6], [11], [24]. The primary reason for this is their ability to nd multiple Pareto-optimal solutions in one single simulation

run. Since EAs work with a population of solutions, a simple EA can be extended to maintain a diverse set of solutions. With an emphasis for moving towards the true Pareto-optimal region, an EA can be used to nd multiple Pareto-optimal solutions in one single simulation run.

The Non-dominated Sorting **Genetic** Algorithm (**NSGA**) proposed in Srinivas and Deb [18] was one of the rst such evolutionary algorithms. Over the years, the main criticism of the **NSGA** approach have been as follows:

High computational complexity of non-dominated sorting: The currently-used non-dominated sorting algorithm has a computational complexity of O(MN ^{3}) (where M is the number of objectives and N is the population size). This makes **NSGA** a computationally expensive algorithm for large population sizes. This large complexity arises because of the complexity involved in the non-dominated sorting procedure in every generation.

Lack of elitism: Recent results ([23], [16]) show clearly that elitism can speed up the performance of the GA signi cantly, also can help preventing the loss of good solutions once they are found.

Need for specifying the sharing parameter $_{share}$: Traditional mechanisms of insuring diversity in a populati get a wide variety of equivalent solutions have relied mostly on the concept of sharing. The main problem with sharing is that it requires the speci cation of a sharing parameter ($_{share}$). Though there has been some work on dynamic sizing of the sharing parameter [8], a parameter-less diversity preservation mechanism is desirable.

In this paper, we address all of these issues and propose an improved version of **NSGA**, which we call **NSGA**-II. From the simulation results on a number of di cult test problems, we nd that **NSGA**-II outperforms two other contemporary

---

multi-objective EAs|Pareto-archived evolution strategy (PAES), [12] and strength Pareto EA (SPEA) [22]|in terms of nding a diverse set of solutions and in converging near the true Pareto-optimal set.

Page 2

Constrained multi-objective optimization is important from the point of view of practical problem solving, but not much attention has been paid so far in this respect among the EA researchers. In this paper, we suggest a simple constraint handling strategy with **NSGA**-II, that suits well for any evolutionary algorithm. On four problems chosen from the literature, **NSGA**-II has been compared with another recently suggested constrained handling strategy. These results encourage the application of **NSGA**-II to more complex and real-world multi-objective optimization problems.

In the remainder of the paper, we brie y mention a number of existing elitist multi-objective EAs in section II. Thereafter, in section III we describe the proposed **NSGA**-II algorithm in details. Section IV presents simulation results of **NSGA**-II and compares them with two other elitist multi-objective EAs (PAES and SPEA). In section V, we highlight the issue of parameter interactions, a matter which is important in evolutionary computation research. Next section extends **NSGA**-II for handling constraints and compares the results with another recently-proposed constraint handling method. Finally, we outline the conclusions of this paper.

II. Elitist Multi-Objective Evolutionary Algorithms

During 1993-95, a number of di erent evolutionary algorithms were suggested to solve multi-objective optimization problems. Of them, Fonseca and Fleming's [6] MOGA, Srinivas and Deb's [18] **NSGA**, and Horn, Nafploitis, and Goldberg's [11] NPGA enjoyed more attention. These algorithms demonstrated the necessary additional operators for converting a simple EA to a multi-objective EA. Two common features on all three operators were the following: (i) assigning tness to population members based on non-dominated sorting and (ii) preserving diversity among solutions of the same non-dominated front. Although they have been shown to nd multiple non-dominated solutions on many test problems and a number of engineering design problems, researchers realized the need of introducing more useful operators (which have been found useful in single-objective EAs) so as to solve multi-objective optimization problems better. Particularly, the interest has been to introduce elitism to enhance the convergence properties of a multi-objective EA. In the study of Zitzler, Deb, and Thiele [23], it was clearly shown that elitism helps in achieving better convergence in MOEAs. Among the existing elitist MOEAs, Zitzler and Thiele's [24] strength Pareto EA (SPEA), Knowles and Corne's Pareto-archived evolution strategy (PAES) [12], and Rudolph's [16] elitist GA are well studied. We describe these approaches in brief. For details, readers are encouraged to refer to the original studies.

Zitzler and Thiele [24] suggested an elitist multi-criterion EA with the concept of non-domination in their strength

Pareto EA (SPEA). They suggested maintaining an external population at every generation storing all non-dominated solutions discovered so far beginning from the initial population. This external population participates in all **genetic** operations. At each generation, a combined population with the external and the current population is rst constructed. All non-dominated solutions in the combined population are assigned a tness based on the number of solutions they dominate and dominated solutions are assigned tness worse than the worst tness of any non-dominated solution. This assignment of tness makes sure that the search is directed towards the non-dominated solutions. A deterministic clustering technique is used to ensure diversity among non-dominated solutions. Although the implementation suggested in [24] is $O(MN^3)$, with proper book-keeping the complexity of SPEA can be reduced to $O(MN^2)$.

Knowles and Corne [12] suggested a simple MOEA using a single parent, single child evolutionary algorithm, similar to (1+1)-evolution strategy. Instead of using real parameters, authors have used binary strings and bit-wise mutations to create children. In their Pareto-archived ES (PAES) with one parent and one child, the child is compared with respect to the parent. If the child dominates the parent, the child is accepted as the next parent and the iteration continues. On the other hand, if the parent dominates the child, the child is discarded and a new mutated solution (a new child) is found. However, if the child and the parent do not dominate each other, the choice between the child and the parent is made by comparing them with an archive of best solutions found so far. The child is compared with the archive to check if it dominates any member of the archive. If yes, the child is accepted as the new parent and all the dominated solutions are eliminated from the archive. If the child does not dominate any member of the archive, both parent and child are checked for their nearness with the solutions of the archive. If the child resides in a least crowded region in the parameter space among the members of the archive, it is accepted as a parent and a copy of added to the archive. Crowding is maintained by deterministically dividing the entire search space in d^n subspaces, where d is the depth parameter and n is the number of decision variables and by updating the subspaces dynamically. Authors have calculated the worst case complexity of PAES for N evaluations as O(aMN), where a is the archive length. Since the archive size is usually chosen proportional to the population size N, the overall complexity of the algorithm is $O(MN^2)$.

Rudolph [16] suggested, but did not simulate, a simple elitist multi-objective EA based on a systematic comparison of individuals from parent and o spring populations. The non-dominated solutions of the o spring population are compared with that of parent solutions to form an overall non-dominated set of solutions, which becomes the parent population of the next iteration. If the size of this set is not greater than the desired population size, other individuals from the o spring population are included. With this strategy, he has been able to prove the convergence of this algorithm to the Pareto-optimal front. Although this is an important achievement in its own right, the algorithm lacks

motivation for the second task of maintaining diversity of Pareto-optimal solutions. An explicit diversity preserving mechanism must be added to make it more usable in practice. Since the determinism of the rst non-dominated front is $O(MN^2)$, the overall complexity of Rudolph's algorithm is also $O(MN^2)$.

In the following, we present the proposed non-dominated sorting GA approach which uses a fast non-dominated sorting procedure, an elitist-preserving approach, and a parameter-less niching operator.

III. Elitist Non-dominated Sorting **Genetic** Algorithm (**NSGA**-II)

The non-dominated sorting GA (**NSGA**) proposed by Srinivas and Deb in 1994 [18] has been subjected to a number of criticism, as mentioned earlier. In this section, we suggest **NSGA**-II, which alleviate all these di culties. We begin by presenting a number of di erent modules that form parts of **NSGA**-II.

A. A Fast Non-dominated Sorting Approach

In order to sort a population of size N according to the level of non-domination, each solution must be compared with every other solution in the population to nd if it is dominated. This requires O(MN) comparisons for each solution, where M is the number of objectives. When this process is continued to nd the members of the rst non-dominated class for all population members, the total complexity is $O(MN^2)$. At this stage, all individuals in the rst non-dominated front are found. In order to nd individuals of the next front, the solutions of the rst front are temporarily discounted

and the above procedure is performed again. The procedure is repeated to nd subsequent fronts. As can be seen, the worst case (when there exists only one solution in each front) the complexity of this algorithm without any book-keeping is $O(MN^3)$. In the following, we describe a fast non-dominated sorting approach which will require at most $O(MN^2)$ computations.

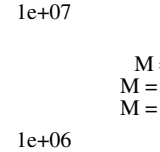
This approach is similar in principle to the above approach, except that a better book-keeping strategy is performed to make it a faster algorithm. In this approach, every solution from the population is checked with a partially lled population for domination. To start with, the rst solution from the population is kept in a set P_0. Thereafter, each solution p (the second solution onwards) is compared with all members of the set P_0 one by one. If the solution p dominates any member q of P_0, then solution q is removed from P_0. This way non-members of the non-dominated front get deleted from P_0. Otherwise, if solution p is dominated by any member of P_0, the solution p is ignored. If solution p is not dominated by any member of P_0, it is entered in P_0. This is how the set P_0 grows with non-dominated solutions. When all solutions of the population is checked, the remaining members of P_0 constitute the non-dominated set.

```
P0 = find-nondominated-front(P )
P0 = f1g                                          include rst member in P 0
for each p 2 P ^p 62 P 0                          take one solution at a time
      P0 = P 0 [fpg                               include p in P0 temporarily
      for each q 2 P0 ^q 6= p                       compare p with other members of P 0
            if p q, then P 0 = P 0nfqg              if p dominates a member of P0, delete it
            else if q p, then P0 = P 0nfpg          if p is dominated by other members of P0,
                                                  do not include p in P0
```

Here, we observe that the second population member is compared with only one solution of P_0, the third solution with at most two solutions of P_0, and so on. This requires a maximum of $O(N^2)$ domination checks. Since each domination check requires M function value comparisons, the maximum complexity of this approach to nd the rst non-dominated front is also $O(MN^2)$.

In order to validate this complexity estimate, we create di erent random populations of N solutions, each with a an objective vector of size M. Each member in the objective vector is chosen between zero and one at random. Thereafter, solutions are compared for domination according to the above algorithm and the total number of comparisons required to identify the rst non-dominated front is counted. This quantity is calculated for a number of di erent random populations and an average is calculated. Figure 1 shows the variation of this quantity with di erent population sizes and for M = 4, 10, and 20. It is clear that the number of comparisons (or computational complexity) increases with N. By tting a curve through the experimental quantities, we observe that the variation is polynomial and becomes quadratic with large M, as shown in Table I. The gure shows that slope of these tted straight lines on the log-log plot increases with M. The table shows these slopes as the exponent on N. As the number of objective functions increase, the exponent increases to approximately two.

To nd other fronts, the members of P_0 will be discounted from P and the above procedure is repeated, as outlined below.



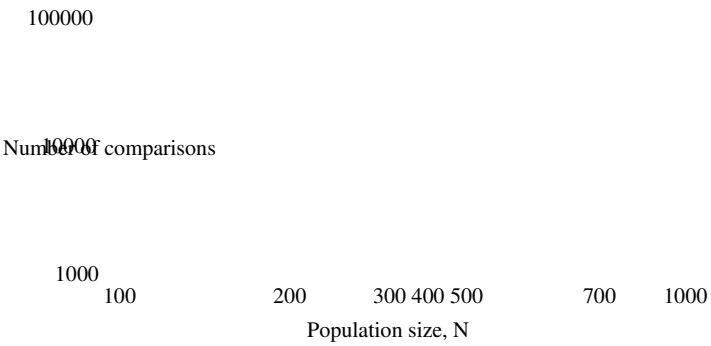


Fig. 1. Variation of computational complexity to nd the rst non-dominated front in a random population with population size, N. The lines are drawn by tting straight lines with the observed points in the log-log scale.

TABLE I

Computational complexity of finding the first non-dominated set for different M.

M	Variation
4	$25:354N_{1:1248}$
10	$8:145N_{1:8834}$
20	$9:772N_{2:0032}$

F = fast-non-dominated-sort(P) F is a set of non-dominated fronts
i = 1 i is the front counter and is initialized to one
until P 6= ;
Fi = find-nondominated-front(P) nd the non-dominated front
P = P n Fi remove non-dominated solutions from P
i = i + 1 increment the front counter

At the end of this operation, solutions of the rst non-dominated front are stored in F_1, solutions of the second non-dominated front are stored in F_2, and so on.

B. Diversity Preservation

We mentioned earlier that along with convergence to the Pareto-optimal set, it is also desired that an EA maintains a good spread of solutions in the obtained set of solutions. The original **NSGA** used the well-known sharing function approach, which has been found to maintain sustainable diversity in a population with appropriate setting of its associated parameters. The sharing function method involves a sharing parameter $_{share}$, which sets the extent of sharing desired in a problem. This parameter is related to the distance metric chosen to calculate the proximity measure between two population members. The parameter $_{share}$ denotes the largest value of that distance metric within which any two solutions share each other's tness. This parameter is usually set by the user, although there exist some guidelines [3]. There are two di culties with this sharing function approach:

1. The performance of the sharing function method in maintaining a spread of solutions largely depends on the chosen $_{share}$ value.
2. Since each solution must be compared with all other solutions in the population, the overall complexity of the sharing function approach is $O(N_2)$.

In the proposed **NSGA**-II, we replace the sharing function approach with a crowded comparison approach which eliminates both the above di culties to some extent. It will be clear in a while that the new approach does not require

any user-de ned parameter for maintaining diversity among population members. Also, the suggested approach has a better computational complexity. To describe this approach, we rst de ne a density estimation metric and then present the crowded comparison operator.

B.1 Density Estimation

To get an estimate of the density of solutions surrounding a particular solution in the population, we calculate the average distance of two points on either side of this point along each of the objectives. This quantity $i_{distance}$ serves as an estimate of the size of the largest cuboid enclosing the point i without including any other point in the population (we call this the crowding distance). In Figure 2, the crowding distance of the i-th solution in its front (marked with solid circles) is the average side-length of the cuboid (shown with a dashed box).

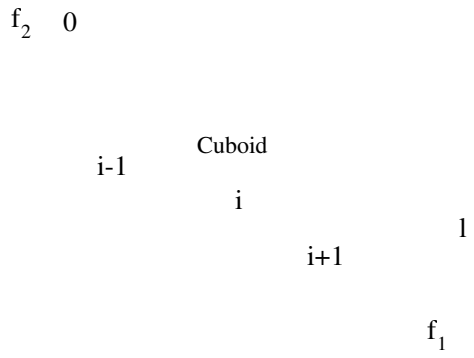


Fig. 2. The crowding distance calculation is shown.

The crowding distance computation requires sorting of the population according to each objective function value in their ascending order of magnitude. Thereafter, for each objective function, the boundary solutions (solutions with smallest and largest function values) are assigned an in nite distance value. All other intermediate solutions are assigned a distance value equal to the absolute di erence in the function values of two adjacent solutions. This calculation is continued with other objective functions. The overall crowding distance value is calculated as the sum of individual distance values corresponding to each objective.

The following algorithm clearly outlines the crowding distance computation procedure of all solutions in an non-dominated set I:

```
crowding-distance-assignment(I)
l = |I|                                  number of solutions in I
for each i, set I[i]distance = 0          initialize distance
for each objective m
   I = sort(I; m)                        sort using each objective value
   I[1]distance = I[l] distance = 1      so that boundary points are always selected
   for i = 2 to (l 1)                    for all other points
      I[i] distance = I[i] distance + (I[i + 1]:m I[i 1]:m)
```

Here $I_{[i]:m}$ refers to the m-th objective function value of the i-th individual in the set I. The complexity of this procedure is governed by the sorting algorithm. Since M independent sorting of at most N solutions (when all population members are in one front I) are involved, the above algorithm has O(MN log N) computational complexity.

After all population members in the set I are assigned a distance metric, we can compare two solutions for their extent of proximity with other solutions. A solution with a smaller value of this distance measure is, in some sense, more

crowded by other solutions. This is exactly what we compare in the proposed crowded comparison operator, described below.

B.2 Crowded Comparison Operator

The crowded comparison operator ($_n$) guides the selection process at the various stages of the algorithm towards a uniformly spread-out Pareto-optimal front. Let us assume that every individual i in the population has two attributes:

1. non-domination rank (i_{rank}), and
2. crowding distance ($i_{distance}$).

We now de ne a partial order $_n$ as :

$i \ _n \ j$ if ($i_{rank} < j_{rank}$) or (($i_{rank} = j_{rank}$) and ($i_{distance} > j_{distance}$))

That is, between two solutions with di ering non-domination ranks we prefer the solution with the lower (better) rank. Otherwise, if both solutions belong to the same front then we prefer the solution which is located in a lesser crowded region.

With these three new innovationsla fast non-dominated sorting procedure, a fast crowded distance estimation procedure, and a simple crowded comparison operator, we are now ready to describe the **NSGA**-II algorithm.

C. The Main Loop

Initially, a random parent population P_0 is created. The population is sorted based on the non-domination. Each solution is assigned a tness (or rank) equal to its non-domination level (1 is the best level, 2 is the next-best level and so on). Thus, minimization of tness is assumed. At rst, the usual binary tournament selection, recombination, and mutation operators are used to create a child population Q_0 of size N. Since elitism is introduced by comparing current population with previously-found best non-dominated solutions, the procedure is di erent after the initial generation. We rst describe a generation of the proposed algorithm:

$R_t = P_t [Q_t$	combine parent and children population
F = fast-nondominated-sort(R_t)	$F = (F_1; F_2; : : :)$, all non-dominated fronts of R_t
$P_{t+1} =$; and i = 1	
until $\|P_{t+1}\| + \|F_i\| \ N$	till the parent population is lled
crowding-distance-assignment(F_i)	calculate crowding distance in F_i
$P_{t+1} = P_{t+1} [F_i$	include i-th non-dominated front in the parent pop
i = i + 1	check the next front for inclusion
Sort(F_i ; $_n$)	sort in descending order using $_n$
$P_{t+1} = P_{t+1} [F_i [1 : (N - \|P_{t+1}\|)]$	choose the rst ($N - \|P_{t+1}\|$) elements of calF_i
Q_{t+1} = make-new-pop(P_{t+1})	use selection, crossover and mutation to create a new population Q_{t+1}
t = t + 1	increment the generation counter

The above step-by-step procedure shows that **NSGA**-II algorithm is simple and straightforward. First, a combined population $R_t = P_t [Q_t$ is formed. The population R_t will be of size 2N. Then, the population R_t is sorted according to non-domination. Since all previous and current population members are include in R_t , the elitism is ensured. Now, solutions belonging to the best non-dominated set F_1 are of best solutions in the combined population and must be emphasized more than any other solution in the combined population. If the size of F_1 is smaller then N, we de nitely choose all members of the set F_1 for the new population P_{t+1} . The remaining members of the population P_{t+1} is chosen from subsequent non-dominated fronts in the order of their ranking. Thus, solutions from the set F_2 are chosen next, followed by solutions from the set F_3 and so on. This procedure is continued till no more sets can be accommodated

followed by solutions from the set F_3, and so on. This procedure is continued till no more sets can be accommodated. Let us say that the set F_l is the last non-dominated set beyond which no other set can be accommodated. In general, the count of solutions in all sets from F_1 to F_l would be larger than the population size. To choose exactly N population members, we sort the solutions of the last front using the crowded comparison operator $_n$, in the descending order and choose the best solutions needed to fill all population slots. The **NSGA**-II procedure is also shown in Figure 3. The new

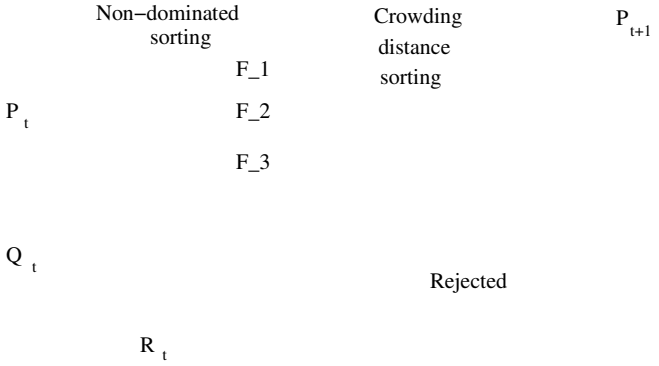


Fig. 3. A sketch of **NSGA**-II.

population P_{t+1} of size N is now used for selection, crossover and mutation to create a new population Q_{t+1} of size N.

It is important to note that we use a binary tournament selection operator but the selection criterion is now based on the crowded comparison operator $_n$. Since this operator requires both the rank and crowded distance of each solution in the population, we calculate these quantities while forming the population P_{t+1}, as shown in the above algorithm.

Let us now look at the complexity of one iteration of the entire algorithm. The basic operations and their worst case complexities are as follows:

1. Non-dominated sorting is $O(M(2N)^2)$,
2. Crowding distance assignment is $O(M(2N)\log(2N))$, and
3. Sorting on $_n$ is $O(2N \log(2N))$.

As can be seen, the overall complexity of the above algorithm is $O(MN^2)$, which is governed by the non-dominated sorting part of the algorithm.

The diversity among non-dominated solutions is introduced by using the crowding comparison procedure which is used in the tournament selection and during the population reduction phase. Since solutions compete with their crowding distance (a measure of density of solutions in the neighborhood), no extra niching parameter (such as $_{share}$ needed in the **NSGA**) is required here. Although the crowding distance is calculated in the objective function space, it can also be implemented in the parameter space, if so desired [2]. However, in all simulations performed in this study, we have used the objective function space niching.

IV. Simulation Results

In this section, we first describe the test problems used to compare the performance of **NSGA**-II with PAES and SPEA. For PAES and SPEA, we have identical parameter settings as suggested in the original studies. For **NSGA**-II, we have chosen a reasonable set of values and have not made any effort in finding the best parameter setting. We leave this task for a future study.

A. Test Problems

We first describe the test problems used to compare different multi-objective evolutionary algorithms. Test problems

are chosen from a number of signi cant past studies in this area. Veldhuizen [20] cited a number of test problems which many researchers have used in the past. Of them, we choose four problems, we call them SCH (from Scha er's study [17]), FON (from Fonseca and Fleming's study [8]), POL (from Poloni's study [14]), and KUR (from Kursawe's study [13]). In 1999, the rst author has suggested a systematic way of developing test problems for multi-objective optimization [2]. Zitzler, Deb, and Thiele [23] followed those guidelines and suggested six test problems. We choose ve of those six problems here and call them ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. All problems have two objective functions. None of these problems have any constraint. We describe these problems in Table II. The table also shows the number of variables, their bounds, the Pareto-optimal solutions, and the nature of the Pareto-optimal front for each problem.

All approaches are run for a maximum of 250 generations and with a population size 100. We use a crossover probability of $p_c = 0:9$ and a mutation probability of $p_m = 1=n$ or $1='$ (where n is the number of decision variables for real-coded GAs and ' is the string length for binary-coded GAs). For **NSGA**-II (real-coded), we use distribution indices [5] for crossover and mutation operators as $_c = 20$ and $_m = 20$, respectively. The population obtained at the end of 250 generations (the population after elitism mechanism is applied) is used to calculate a couple of performance metrics, which we discuss in the next subsection. For PAES, we use a depth value d equal to 4 and an archive size a of 100. We use all population members of the archive obtained at the end of 250 generations to calculate the performance metrics. For SPEA, we use a population of size 80 and an external population of size 20, so that overall population size becomes 100. We use the combination of these two populations at the nal generation to calculate the performance metrics used in this study. For PAES, SPEA, and **NSGA**-II (binary coded) we have used 30 bits to code each decision variable.

B. Performance Measures

Unlike in single-objective optimization, there are two goals in a multi-objective optimization|(i) convergence to the Pareto-optimal set, and (ii) maintenance of diversity in solutions of the Pareto-optimal set. Clearly, these two tasks cannot be measured with one performance metric adequately. A number of performance metrics have been suggested in the past [7], [22]. But, here, we de ne two performance metrics which are more direct in evaluating each of the above two goals in a solution set obtained by a multi-objective optimization algorithm.

The rst metric, , measures the extent of convergence to a known set of Pareto-optimal solutions. Since, multi-objective algorithms would be tested on problems having a known set of Pareto-optimal set, the calculation of this metric is possible. But, we realize that such a metric cannot be used for any arbitrary problem. First, we nd a set of H = 500 uniformly-spaced solutions from the true Pareto-optimal front in the objective space. For each solution obtained with an algorithm, we compute the minimum Euclidean distance of it from H chosen solutions on the Pareto-optimal front. The average of these distances is used as the rst metric (the convergence metric). Figure 4 shows the calculation procedure of this metric. The shaded region is the feasible search region and the solid curved lines specify the Pareto-

TABLE II

Test problems used in this study. All objective functions are to be minimized.

Page 8

Problem	n	Variable bounds	Objective functions	Op sol
SCH	1	$[-10^3; 10^3]$	$f_1(x) = x^2$ $f_2(x) = (x-2)^2$	$x \in$
FON	3	$[-4; 4]$	$f_1(x) = 1 - \exp\left(-\sum_{i=1}^{3}\left(x_i - \frac{1}{\sqrt{3}}\right)^2\right)$ $f_2(x) = 1 - \exp\left(-\sum_{i=1}^{3}\left(x_i + \frac{1}{\sqrt{3}}\right)^2\right)$	$x_1 = x$ 2 [1=
POL	2	[;]	$f_1(x) = 1 + (A_1 - B_1)^2 + (A_2 - B_2)^2$	

			$f_2(x) = (x_1 + 3)^2 + (x_2 + 1)^2$ $A_1 = 0.5 \sin 1 - 2 \cos 1 + \sin 2 - 1.5 \cos 2$ $A_2 = 1.5 \sin 1 - \cos 1 + 2 \sin 2 - 0.5 \cos 2$ $B_1 = 0.5 \sin x_1 - 2 \cos x_1 + \sin x_2 - 1.5 \cos x_2$ $B_2 = 1.5 \sin x_1 - \cos x_1 + 2 \sin x_2 - 0.5 \cos x_2$
KUR	3	$[-5, 5]$	$f_1(x) = \sum_{i=1}^{n-1} -10 \exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)$ $f_2(x) = \sum_{i=1}^{n} \lvert x_i\rvert^{0.8} + 5 \sin x_i^3$
ZDT1	30	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)\left[1 - \sqrt{x_1/g(x)}\right]$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$x_1 \in$ x_i $i = 2, \ldots,$
ZDT2	30	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)\left[1 - (x_1/g(x))^2\right]$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$x_1 \in$ x_i $i = 2, \ldots,$
ZDT3	30	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)\left[1 - \sqrt{x_1/g(x)} - (x_1/g(x))\sin(10\pi x_1)\right]$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$x_1 \in$ x_i $i = 2, \ldots,$
ZDT4	10	$x_1 \in [0, 1]$ $x_i \in [-5, 5]; f_2(x) = g(x)$ $i = 2, \ldots, n$	$f_1(x) = x_1$ $g(x) = 1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i^2 - 10\cos(4\pi x_i)\right)$	$x_1 \in$ x_i $i=2,\ldots,$
ZDT6	10	$[0, 1]$	$f_1(x) = 1 - \exp(-4x_1)\sin^6(4\pi x_1)$ $f_2(x) = g(x)\left[1 - (f_1(x)/g(x))^2\right]$ $g(x) = 1 + 9\left[\left(\sum_{i=2}^{n} x_i\right)/(n-1)\right]^{0.25}$	$x_1 \in$ x_i $i = 2,$

optimal solutions. Solutions with open circles are H chosen solutions on the Pareto-optimal front for the calculation of the convergence metric and solutions marked with dark circles are solutions obtained by an algorithm. It is clear that the smaller the value of this metric, the better is the convergence towards the Pareto-optimal front. When all obtained solutions lie exactly on H chosen solutions, this metric takes a value zero. In all simulations performed here, we present the average and variance of this metric calculated for solution sets obtained in multiple runs.

Even when all solutions converge to the Pareto-optimal front, the above convergence metric does not have a value zero. The metric will be zero only when each obtained solution lies exactly on each of the chosen solutions. Although this metric alone can provide some information about the spread in obtained solutions, we de ne an di erent metric to measure the spread in solutions obtained by an algorithm. The second metric, , measures the extent of spread achieved among the obtained solutions. Here, we are interested in getting a set of solutions which span the entire Pareto-optimal region. We calculate the Euclidean distance d_i between consecutive solutions in the obtained non-dominated set of solutions. We calculate the average d of these distances. Thereafter, from the obtained set of non-dominated solutions, we rst calculate the extreme solutions (in the objective space), by tting a curve parallel to that of the true

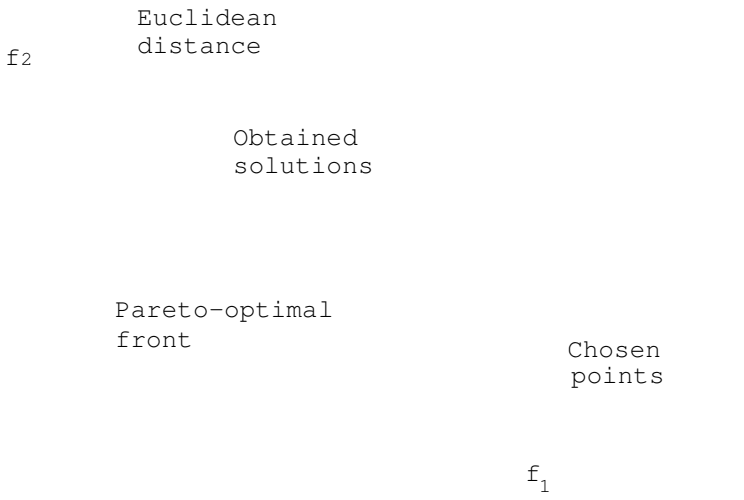


Fig. 4. Illustration of the distance metric .

Pareto-optimal front. Then, we use the following metric to calculate the non-uniformity in the distribution:

$$= \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}}.$$

Here, the parameters d_f and d_l are the Euclidean distances between the extreme solutions and the boundary solutions of the obtained non-dominated set, as depicted in Figure 5. illustrates all distances mentioned in the above equation.

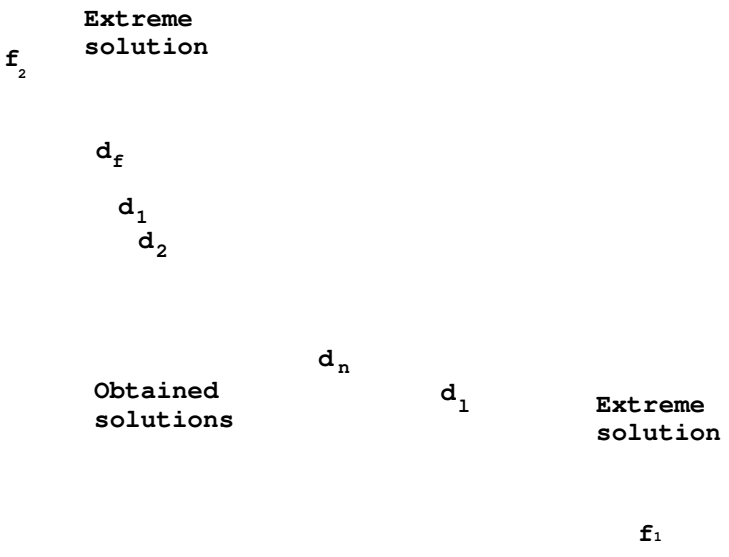


Fig. 5. Illustration of the diversity metric .

The parameter d is the average of all distances d_i , i = 1; 2; : : : ; (N 1), assuming that there are N solutions on the best non-dominated front. With N solutions, there are (N 1) consecutive distances. The denominator is the value of the numerator for the case when all N solutions lie on one solution. It is interesting to note that this is not the worst case spread of solutions possible. For a scenario with a large variance of the distances may have a numerator value greater than the denominator. Thus, the maximum value of the above metric can be greater than one. But, a good distribution would make all distances dconvergencei equal to d and would make $d_f = d_l = 0$ (with existence of extreme solutions in the non-dominated set). Thus, for the most widely and uniformly spread-out set of non-dominated solutions, the numerator of would be zero, making the metric to take a value zero. For any other distribution, the value of the metric would be greater than zero. For two distributions having identical values of d_f and d_l , the metric takes a

higher value with worse distributions of solutions within the extreme solutions. Note that the above diversity metric can be used on any non-dominated set of solutions, including one which is not the Pareto-optimal set.

C. Discussion of the Results

Table III shows the mean and variance of the convergence metric obtained using four algorithms **NSGA**-II (real-coded), **NSGA**-II (binary-coded), SPEA, and PAES.

TABLE III

Mean (shaded rows) and variance (unshaded rows) of the convergence metric .

Algorithm	SCH	FON	POL	KUR	ZDT1	ZDT2	ZDT3
NSGA-II	0.003391	0.001931	0.015553	0.028964	0.033482	0.072391	0.114500
Real-coded	0	0	0.000001	0.000018	0.004750	0.031689	0.007940	0.118460	0.013135
NSGA-II	0.002833	0.002571	0.017029	0.028951	0.000894	0.000824	0.043411
Binary-coded	0.000001	0	0.000003	0.000016	0	0	0.000042
SPEA	0.003403	0.125692	0.037812	0.045617	0.001799	0.001339	0.047517
	0	0.000038	0.000088	0.00005	0.000001	0	0.000047	6.57251
PAES	0.001313	0.151263	0.030864	0.057323	0.082085	0.126276	0.023872
	0.000003	0.000905	0.000431	0.011989	0.008679	0.036877	0.00001	0.5272

NSGA-II (real-coded or binary-coded) is able to converge better in all problems except in ZDT3 and ZDT6, where PAES found better convergence. In all cases with **NSGA**-II, the variance in 10 runs is also small, except in ZDT4 with **NSGA**-II (binary coded). The xed archive strategy of PAES allows better convergence to be achieved in two out of nine problems.

Table IV shows the mean and variance of the diversity metric obtained using all three algorithms.

TABLE IV

Mean (shaded rows) and variance (unshaded rows) of the diversity metric .

Algorithm	SCH	FON	POL	KUR	ZDT1	ZDT2	ZDT3
NSGA2R	0.477899	0.378065	0.452150	0.411477	0.390307	0.430776	0.738540
Real-coded	0.003471	0.000639	0.002868	0.000992	0.001876	0.004721	0.019706	0.064648	0.009923
NSGA-II	0.449265	0.395131	0.503721	0.442195	0.463292	0.435112	0.575606
Binary-coded	0.002062	0.001314	0.004656	0.001498	0.041622	0.024607	0.005078	0.009841	0.035042
SPEA	1.021110	0.792352	0.972783	0.852990	0.784525	0.755148	0.672938
	0.004372	0.005546	0.008475	0.002619	0.004440	0.004521	0.003587	0.014616	0.002713
PAES	1.063288	1.162528	1.020007	1.079838	1.229794	1.165942	0.789920
	0.002868	0.008945	0	0.013772	0.004839	0.007682	0.001653	0.101399	0.003916

NSGA-II (real or binary coded) performs the best in all nine test problems. The worst performance is observed with PAES. For illustration, we show one of the ten runs of PAES with an arbitrary run of **NSGA**-II (real-coded) on problem SCH in Figure 6.

On most problems, real-coded **NSGA**-II is able to nd a better spread of solutions than any other algorithm, including binary-coded **NSGA**-II.

In order to demonstrate the working of these algorithms, we also show typical simulation results of PAES, SPEA, and **NSGA**-II on the test problems KUR, ZDT2, ZDT4, and ZDT6. The problem KUR has three discontinuous regions in the Pareto-optimal front. Figure 7 shows all non-dominated solutions obtained after 250 generations with **NSGA**-II (real-coded). The Pareto-optimal region is also shown in the gure. This gure demonstrates the abilities of **NSGA**-II in converging to the true front and in nding diverse set of solutions in the front. Figure 8 shows the obtained non-dominated solutions with SPEA, which is the next best algorithm for this problem (refer to Tables III and IV). Although the convergence is adequate, the distribution in solutions is not as good as that with **NSGA**-II.

Next, we show the non-dominated solutions on the problem ZDT2 in Figures 9 and 10. This problem has a non-convex Pareto-optimal front. We show the performance of binary coded **NSGA**-II and SPEA on this function. Although the convergence is not a di culty here with both of these algorithms, both real-coded and binary-coded **NSGA**-II has better able to spread solutions in the entire Pareto-optimal region than SPEA (the next-best algorithm observed for this problem).

The problem ZDT4 has 21^9 or $7{:}94(10^{11})$ di erent local Pareto-optimal fronts in the search space, of which only one corresponds to the global Pareto-optimal front. The Euclidean distance in the decision space between solutions of two

---

Page 11

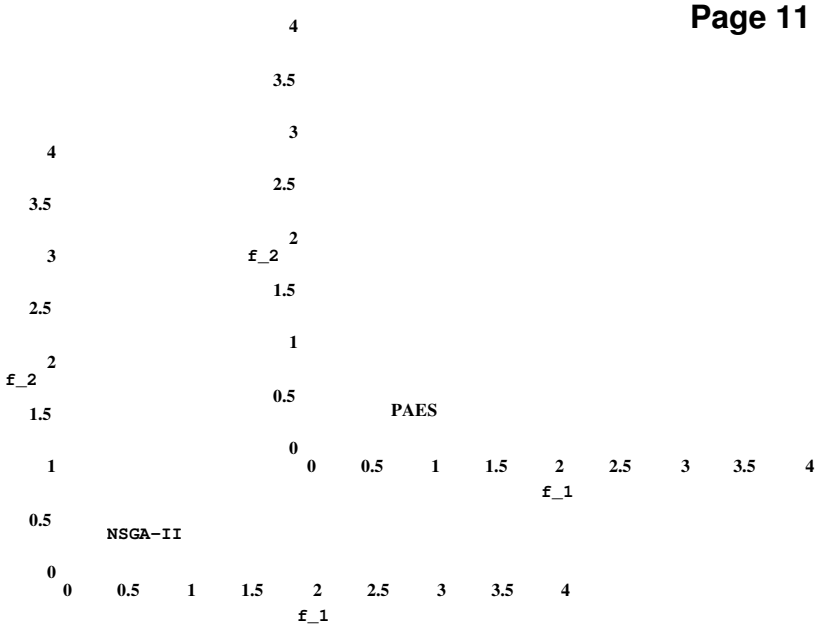


Fig. 6. **NSGA**-II nds better spread of solutions than PAES on SCH.

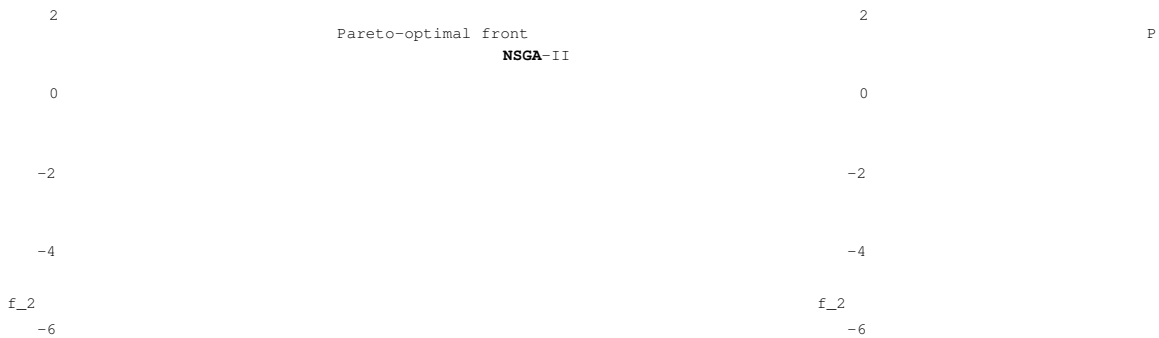


Fig. 7. Non-dominated solutions with **NSGA**-II (real-coded) on KUR.

Fig. 8. Non-dominated solutions with

consecutive local Pareto-optimal sets is 0:25. Figure 11 shows that both real-coded **NSGA**-II and PAES get stuck at di erent local Pareto-optimal sets, but the convergence and ability to nd a diverse set of solutions are de nitely better with **NSGA**-II. Binary-coded GAs have di culties in converging near the global Pareto-optimal front, a matter which is also been observed in previous single-objective studies [4]. On a similar 10-variable Rastrigin's function (the function g(x) here), that study clearly showed that a population of size of about at least 500 is needed for single-objective binary-coded GAs (with tournament selection, single-point crossover and bit-wise mutation) to nd the global optimum solution in more than 50% of the simulation runs. Since we have used a population of size 100, it is not expected that a multi-objective GA would nd the the global Pareto-optimal solution. Since SPEA performs poorly on this problem (Tables III and IV), we do not show SPEA results on this gure.

Finally, Figure 12 shows that PAES nds a better converged set of non-dominated solutions in ZDT6 compared to any other algorithm. However, the distribution in solutions is better with real-coded **NSGA**-II.

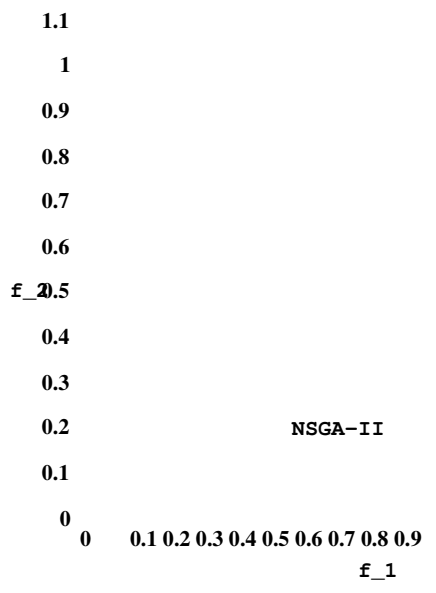


Fig. 9. Non-dominated solutions with **NSGA**-II (binary-coded) on ZDT2.

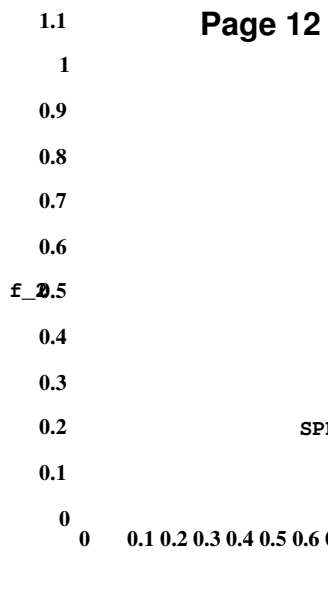


Fig. 10. Non-dominated solutions with

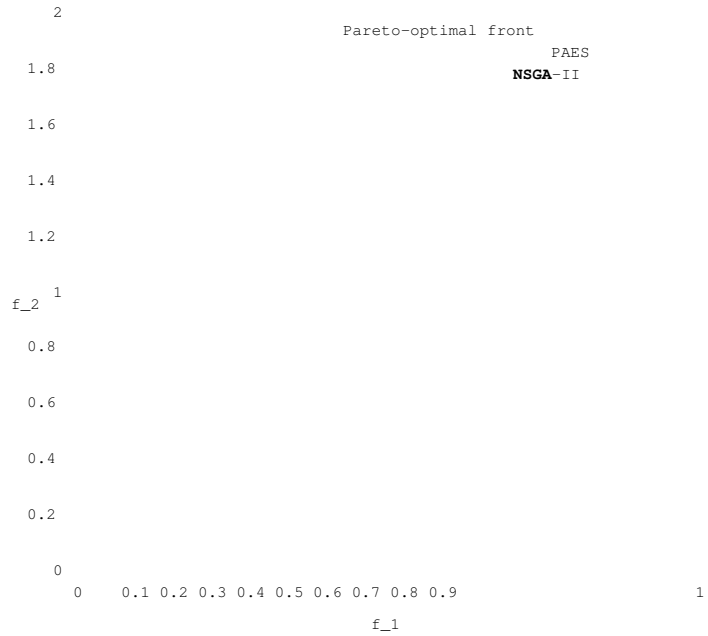


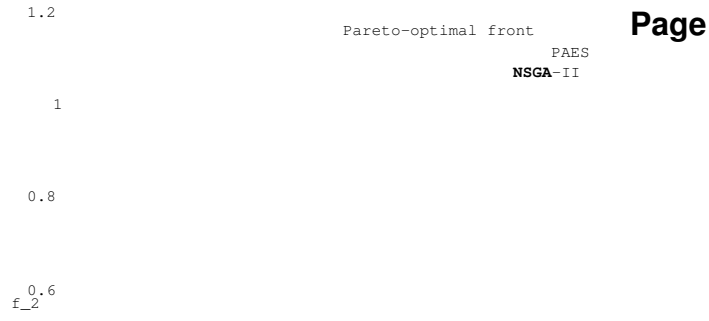
Fig. 11. **NSGA**-II nds better convergence and spread of solutions than PAES on ZDT4.

D. Di erent Parameter Settings

In this study, we do not make any serious attempt to nd the best parameter setting for **NSGA**-II. But in this section, we perform additional experiments to show the e ect of a couple of di erent parameter settings on the performance of **NSGA**-II.

First, we keep the all other parameters same as before, but increase the number of maximum generations to 500 (instead of 250 used before). Table V shows the convergence and diversity metrics for problems POL, KUR, ZDT3, ZDT4, and ZDT6. Now, we achieve a convergence very close to the true Pareto-optimal front and with a much better distribution. The table shows that in all these di cult problems, the real-coded **NSGA**-II has converged very close to the true optimal front, except in ZDT6, which probably requires a di erent parameter setting with **NSGA**-II. Particularly, the results on ZDT3 and ZDT4 improve with generation number.

The problem ZDT4 has a number of local Pareto-optimal fronts, each corresponding to particular value of g(x). To jump from one local optimum to the next best local optimum, a large change in the decision vector is needed. Unless, mutation or crossover operators are capable of creating solutions in the basin of another better attractor, the improvement in the convergence towards the true Pareto-optimal front is not possible. We use **NSGA**-II (real-coded)



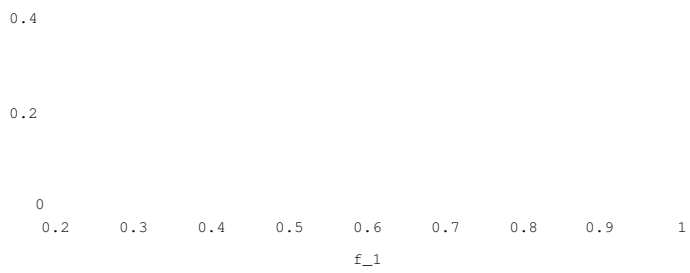


Fig. 12. Real-coded **NSGA**-II nds better spread of solutions than PAES on ZDT6, but PAES has a better convergence.

TABLE V

Mean and variance of the convergence and diversity metrics up to 500 generations.

	POL	KUR	ZDT3	ZDT4	ZDT6
Convergence metric,					
Mean	0.015882	0.026544	0.018510	0.090692	0.276609
Variance	0.000001	0.000017	0.000227	0.053460	0.015843
Diversity metric,					
Mean	0.467022	0.418889	0.688218	0.440022	0.655896
Variance	0.002186	0.000530	0.000610	0.026729	0.003302

with a smaller distribution index $_m = 10$ for mutation, which has an e ect of creating solutions wuth more spread than before. Rest of the parameter settings are identical as before. The convergence metric and diversity measure on problem ZDT4 at the end of 250 generations are as follows:

$$= 0{:}029544 \quad _2 = 0{:}002145$$
$$= 0{:}498409 \quad _2 = 0{:}003852$$

These results are much better than PAES and SPEA (as shown in Table III). To demonstrate the convergence and spread of solutions, we plot the non-dominated solutions of one of the runs after 250 generations in Figure 13. The gure shows that **NSGA**-II is able to nd solutions on the true Pareto-optimal front with g(x) = 1:0.

V. Rotated Problems

It has been discussed in an earlier study [2] that interactions among decision variables can introduce another level of di culty to any multi-objective optimization algorithm including evolutionary algorithms. In this section, we create one such problem and investigate the working of previously three multi-objective evolutionary algorithms on such an epistatic problems.

$$\begin{aligned}
&\text{Minimize } f_1(\mathbf{y}) = y_1;\\
&\text{Minimize } f_2(\mathbf{y}) = g(\mathbf{y}) \exp(y_1 = g(\mathbf{y}));\\
&\text{where} \quad g(\mathbf{y}) = 1 + 10(n-1) + \sum_{i=2}^{n} \left(y_i^2 - 10\cos(4\pi y_i) \right);\\
&\text{and} \quad \mathbf{y} = \mathbf{R}\mathbf{x};\\
&\qquad -0{:}3 \le x_i \le 0{:}3; \text{ for } i = 1; 2; : : : ; n:
\end{aligned}$$

An EA works with the decision variable vector x, but the above objective functions are de ned in terms of the variable vector y, which is calculated by transforming the decision variable vector x by a xed rotation matrix $\mathbf{R}$. This way,

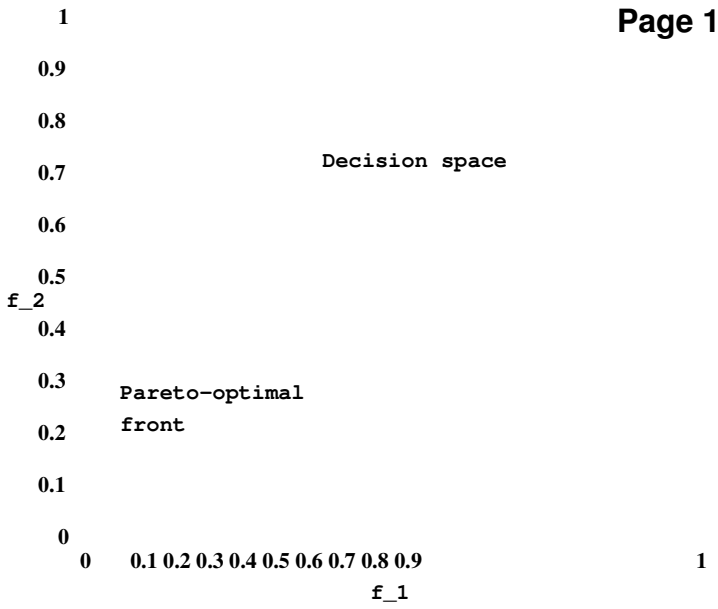


Fig. 13. Obtained non-dominated solutions with **NSGA**-II on problem ZDT4.

the objective functions are functions of a linear combination of decision variables. In order to maintain a spread of solutions over the Pareto-optimal region or even converge to any particular solution requires an EA to update all decision variables in a particular way. With a generic search operator, this becomes a di cult task to an EA. However, here, we are interested in evaluating the overall behavior of three elitist multi-objective EAs.

We use a population size of 100 and run each algorithm till 500 generations. For simulated binary crossover we use $_c = 10$ and mutation we use $_m = 50$. After rotation of x, the values of y may lie in a wide range. To restrict the Pareto-optimal solutions to lie within bounds, we discourage solutions with $\dot{j}f_1\dot{j} > 0{:}3$ by adding a xed large penalty to both objectives. Figure 14 shows the obtained solutions at the end of 500 generations using NSGA-II, PAES, and SPEA. It is observed that NSGA-II converged to the true front, but PAES and SPEA could not come close to the true front. The correlated parameter updates needed to progress towards the Pareto-optimal front makes this kind of problem di cult to solve. The elitism procedure along with the real-coded crossover and mutation operators used in

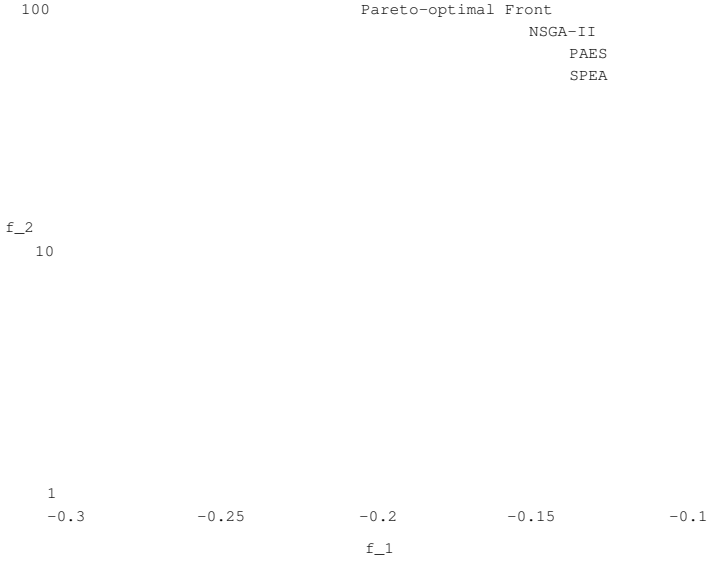


Fig. 14. Obtained non-dominated solutions with NSGA-II, PAES, and SPEA on the rotated problem.

NSGA-II are able to converge to the Pareto-optimal front (with g(y) = 1 resulting $f_2 = \exp(f_1)$). This example problem demonstrates that one of the known di culties (the linkage problem [9], [10]). of single-objective optimization algorithm can also cause di culties in a multi-objective problem. However, more systematic studies are needed to amply

address the linkage issue in multi-objective optimization.

VI. Constraint Handling

Last year, the rst author and his students implemented a penalty-parameter-less constraint handling approach for single-objective optimization. Those studies [1], [5] have shown how a tournament selection based algorithm can be used to handle constraints in a population approach much better than a number of other existing constraint handling approaches. A similar approach can be introduced with the above NSGA-II for solving constrained multi-objective optimization problems as well.

A. Proposed Constraint Handling Approach (Constrained NSGA-II)

This constraint handling method uses the binary tournament selection, where two solutions are picked from the population and the better solution is chosen. In the presence of constraints, each solution can be either feasible or infeasible. Thus, there may be at most three situations: (i) both solutions are feasible, (ii) one is feasible and other is not, and (iii) both are infeasible. For single objective optimization, we used a simple rule for each case:

Case (i) Choose the solution with better objective function value.

Case (ii) Choose the feasible solution.

Case (iii) Choose the solution with smaller overall constraint violation.

Since in no case constraints and objective function values are compared with each other, there is no need of having any penalty parameter, a matter which makes the proposed constrained handling approach useful and attractive.

In the context of multi-objective optimization, the latter two cases can be used as they are, and the rst case can be resolved by using the crowded comparison operator as before. To maintain the modularity in the procedures of NSGA-II, we simply modify the de nition of domination between two solutions i and j:

De nition 1 A solution i is said to constrained-dominate a solution j, if any of the following conditions is true:

1. Solution i is feasible and solution j is not.
2. Solutions i and j are both infeasible, but solution i has a smaller overall constraint violation.
3. Solutions i and j are feasible and solution i dominates solution j.

The e ect of using this constrained-domination principle is that any feasible solution has a better non-domination rank than any infeasible solution. All feasible solutions are ranked according to their non-domination level based on the objective function values. But, among two infeasible solutions, the solution with a smaller constraint violation has a better rank. Moreover, this modi cation in the non-domination principle does not change the computational complexity of NSGA-II. The rest of the NSGA-II procedure as described can be used as usual.

B. Ray-Kang-Chye's Constraint Handling Approach

T. Ray, T. Kang, and S. K. Chye [15] suggested a more elaborate constraint handling technique, where constraint violations of all constraints are not simply summed together, instead a non-domination check of constraint violations is also made. We give an outline of this procedure here.

Three di erent non-dominated rankings of the population is rst performed. The rst ranking is performed using M objective function values and the resulting ranking is stored in a N-dimensional vector R_{obj} . The second ranking R_{con} is performed using only the constraint violation values of all (J of them) constraints and no objective function

R_{con} is performed using only the constraint violation values of all (J of them) constraints and no objective function information is used. Thus, constraint violation of each constraint is used a criterion and a non-domination classi cation of the population is performed with the constraint violation values. Notice that for a feasible solution all constraint violations are zero. Thus, all feasible solutions have a rank 1 in R_{con}. The third ranking is performed using a combined objective function and constraint violation values (a total of (M +J) values). This produces the ranking R_{com}. Although objective function values and constraint violations are used together, one nice aspect of this algorithm is that there is no need of any penalty parameter. In the domination check, criteria are individually compared, thereby eliminating the need of any penalty parameter. Once these rankings are over, all feasible solutions having the best rank in R_{com} are chosen for the new population. If more population slots are available, they are created from the remaining solutions in a systematic manner. By giving importance to the ranking in R_{obj} in the selection operator and by giving importance to the ranking in R_{con} in the crossover operator, authors have laid out a systematic multi-objective GA, which also includes a niche preserving operator. For details, readers may refer to the original study [15]. Although authors did not compare their algorithm with any other method, they showed the working of this constraint handling method on a number of engineering design problems. However, since non-dominated sorting of three di erent sets of criteria are required and the algorithm introduces many di erent operators, it remains to be investigated how it performs on more complex problems, particularly from the point of view of computational burden associated with the method.

In the following section, we choose a set of four problems and compare the simple constrained NSGA-II with Ray-Kang-Chye's method.

C. Simulation Results

Page 16

For unconstrained multi-objective optimization, the rst author suggested a systematic procedure of developing test problems [2]. The procedure allows a simple way to introduce each aspect of di culties that a multi-objective optimizer can expect in a real-world problem. So far, no such systematic study exists to suggest test problem development for constrained multi-objective optimization. However, we choose problems (Table VI) which have been used in earlier studies. We only apply real-coded NSGA-II here.

TABLE VI

Constrained test problems used in this study. All objective functions are to be minimized.

Problem	n	Variable bounds	Objective functions	Constraints
DEB	2	$x_1 \in [0:1 : 1:0]$ $x_2 \in [0 : 5]$	$f_1(x) = x_1$ $f_2(x) = (1 + x_2)=x_1$	$g_1(x) = x_2 + 9x_1$ $g_2(x) = x_2 + 9x_1$
SRN	2	$x_i \in [-20 : 20]$ $i=1;2$	$f_1(x) = (x_1 - 2)^2 + (x_2 - 1)^2 + 2$ $f_2(x) = 9x_1 - (x_2 - 1)^2$	$g_1(x) = x_2 + x_1$ $g_2(x) = x_1$
TNK	2	$x_i \in [0; \pi]$ $i=1;2$	$f_1(x) = x_1$ $f_2(x) = x_2$	$g_1(x) = 0:1\cos(16\arctan$ $g_2(x) = (x - 0:5)^2 +$
WATER	3	$0:01 \le x_1$ $0:01 \le x_2$ $0:01 \le x_3$	$0:45 \le f_1(x) = 106780:37(x_2 + x_3) + 61704:67$ $0:10 \le f_2(x) = 3000x_1$ $0:10 \le f_3(x) = (305700)2289x_2=(0:06 \cdot 2289)^{0:65}$ $f_4(x) = (250)2289\exp(-39:75x_2 +$	$g_1(x) = 0:00139=(x_1x_2) + 4:94x_3 - 0:08 \le 1$ $g_2(x) = 0:000306 - 0:0986 < 1$ $g_3(x) = 12:307=(\cdot - 4051:02 \le 50000$ $g_4(x) = 2:098=(x$

9:9x_3 + 2:74)
696:71 16000

$f_5(x) = 25(1:39=(x_1x_2) + 4940x_3$ 80) $g_5(x) = 2:138=(x_1x_2$
705:04 10000

$g_6(x) = 0:417(x_1$
136:54 2000

$g_7(x) = 0:164=(x$
54:48 550

In the rst problem, a part of the unconstrained Pareto-optimal region is not feasible. Thus, the resulting constrained Pareto-optimal region is a concatenation of the rst constraint boundary and some part of the unconstrained Pareto-optimal region. The second problem SRN was used in the original study of NSGA [18]. Here, the constrained Pareto-optimal set is a subset of the unconstrained Pareto-optimal set. The third problem TNK was suggested by Tanaka et al. [19] and has a discontinuous Pareto-optimal region, entirely falling on the rst constraint boundary. In the next subsection, we show the constrained Pareto-optimal region for each of the above problems. The fourth problem WATER is a ve-objective and seven-constraint problem, attempted to solve in [15]. With ve objectives, it is di cult to discuss the e ect of the constraints on the unconstrained Pareto-optimal region. In the next subsection, we show all 5 pair-wise plots of obtained non-dominated solutions.

D. Simulation Results

In all problems, we use a population size of 100, distribution indices for real-coded crossover and mutation operators of 20 and 100, respectively, and run NSGA-II (real-coded) and Ray, Kang, and Chye's constrained handling algorithm [15] for a maximum of 500 generations. We choose this rather large number of generations to investigate if spread in solutions can be maintained for a large number of generations. However, in each case, we obtain a reasonably good spread of solutions at early as at 200 generations. Crossover and mutation probabilities are the same as before.

Figure 15 shows the obtained set of 100 non-dominated solutions after 500 generations using NSGA-II. The gure shows that NSGA-II is able to uniformly maintain solutions in both Pareto-optimal region. It is important to note that in order to maintain a spread of solutions on the constraint boundary, the solutions must have to be modi ed in a particular manner dictated by the constraint function. This becomes a di cult task of any search operator. Figure 16 shows the obtained solutions using Ray-Kang-Chye's algorithm after 500 generations. It is clear that NSGA-II performs

better than Ray-Kang-Chye's algorithm in terms of converging to the true Pareto-optimal front and also in terms of maintaining a diverse population of non-dominated solutions.

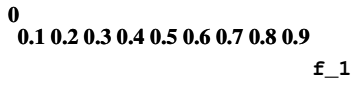


Fig. 15. Obtained non-dominated solutions with NSGA-II on the constrained problem DEB.

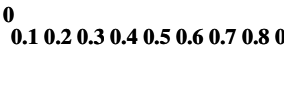


Fig. 16. Obtained non-dominated solutions wi algorithm on the constrained problem D

Next, we consider the test problem SRN. Figure 17 shows the non-dominated solutions after 500 generations using NSGA-II. The gure shows how NSGA-II can bring a random population on the Pareto-optimal front. For illustrating the feasible search space, we have created a number of random feasible solutions and plotted with 'dots' on the same gure. Ray-Kang-Chye's algorithm is also able to come close to the front on this test problem (Figure 18).

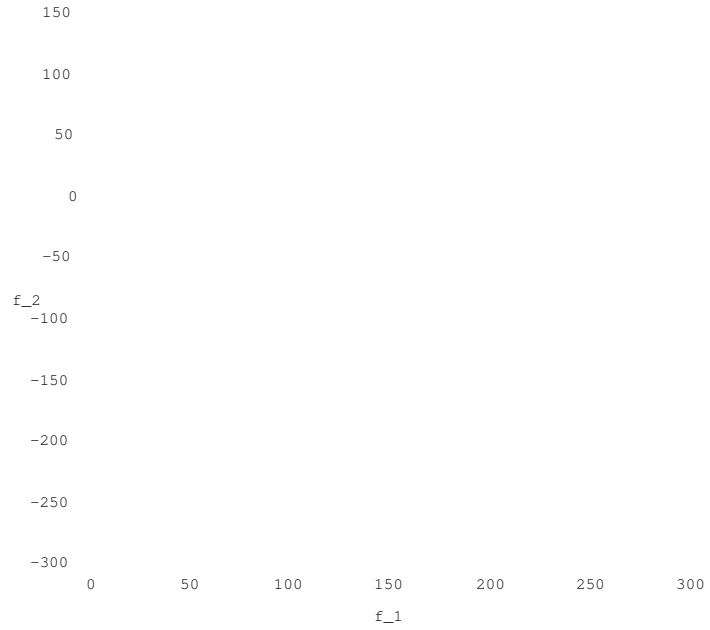


Fig. 17. Obtained non-dominated solutions with NSGA-II on the constrained problem SRN.

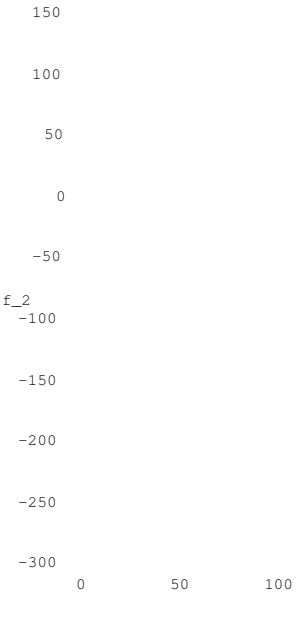


Fig. 18. Obtained non-dominated solutions wi algorithm on the constrained problem S

Figures 19 and 20 show the feasible objective space and the obtained non-dominated solutions with NSGA-II and Ray-Kang-Chye's algorithm. Here, the Pareto-optimal region is discontinuous and NSGA-II does not have any di culty in nding a wide spread of solutions over the true Pareto-optimal region. Although Ray-Kang-Chye's algorithm has found a number of solutions on the Pareto-optimal front, there exists many infeasible solutions even after 500 generations.

Ray, Kang, and Chye [15] have used the problem WATER in their study. They normalized the objective functions in the following manner:

$$f_1=8(10^4);\ f_2=1500;\ f_3=3(10^6);\ f_4=6(10^6);\ f_5=8000:$$

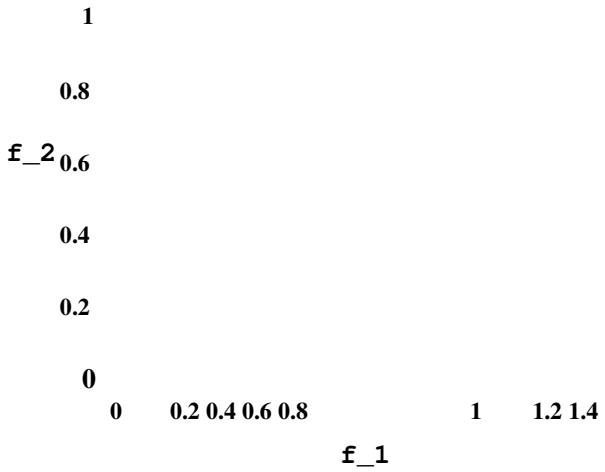


Fig. 19. Obtained non-dominated solutions with NSGA-II on the constrained problem TNK.

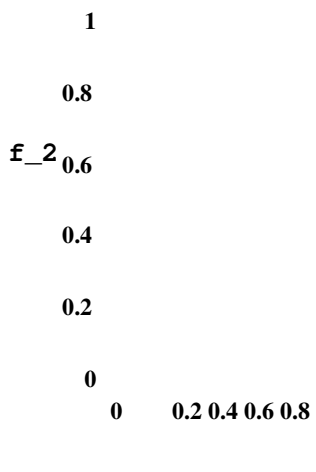


Fig. 20. Obtained non-dominated solutions wi algorithm on the constrained problem T

Since there are ve objective functions in the problem WATER, we observe the range of the normalized objective function values of the obtained non-dominated solutions. Table VII shows the comparison with Ray-Kang-Chye's algorithm. In

TABLE VII

Lower and upper bounds of the objective function values observed in the obtained non-dominated solutions.

Algorithm	f_1	f_2	f_3	f_4
NSGA-II	0.798 { 0.920	0.027 { 0.900	0.095 { 0.951	0.031 { 1.110	0.001 { 3.124
Ray-Kang-Chye	0.810 { 0.956	0.046 { 0.834	0.967 { 0.934	0.036 { 1.561	0.211 { 3.116

most objective functions, NSGA-II has found a better spread of solutions than Ray-Kang-Chye's approach. In order to show the pair-wise interactions among these ve normalized objective functions, we plot all $\binom{5}{2}$ in Figure 21 for both algorithms. NSGA-II results are shown in the upper diagonal portion of the gure and the Ray-Kang-Chye results are shown in the lower diagonal portion. The axes of any plot can be obtained by looking at the corresponding diagonal boxes and their ranges. For example, the plot at the rst row and third column has its vertical axis as f_1 and horizontal axis as f_3. Since this plot belongs in the upper side of the diagonal, this plot is obtained using NSGA-II. In order to compare this plot with a similar plot using Ray-Kang-Chye's approach, we look for the plot in the third row and rst column. For this gure, the vertical axis is plotted as f_3 and the horizontal axis is plotted as f_1. To get a better comparison between these two plots, we observe Ray-Kang-Chye's plot as it is, but turn the page 90 degrees in the clockwise direction for NSGA-II results. This would make the labelling and ranges of the axes same in both cases.

We observe that NSGA-II plots have better-formed patterns than in Ray-Kang-Chye's plots.

For example, gures f_1-f_3, f_1-f_4, and f_3-f_4 interactions are very clear from NSGA-II results. Although similar patterns exist in the results obtained using Ray-Kang-Chye's algorithm, the convergence to the true fronts is not adequate.

VII. Conclusions

In this paper, we have proposed a computationally fast and elitist multi-objective evolutionary algorithm based on non-dominated sorting approach. On nine di erent di cult test problems borrowed from the literature, it has been found that the proposed NSGA-II has been able to maintain a better spread of solutions and convergence in the obtained non-dominated front compared to two other elitist multi-objective EAs|PAES and SPEA. However, on one problem PAES has been able to somewhat better converge close to the true Pareto-optimal front. PAES maintains diversity among solutions by controlling crowding of solutions in a deterministic and pre-speci ed number of equal-sized cells in the search space. In that problem, it is suspected that such a deterministic crowding coupled with the e ect of mutation-based approach has been bene cial in converging near the true front compared to the dynamic and parameter-less crowding

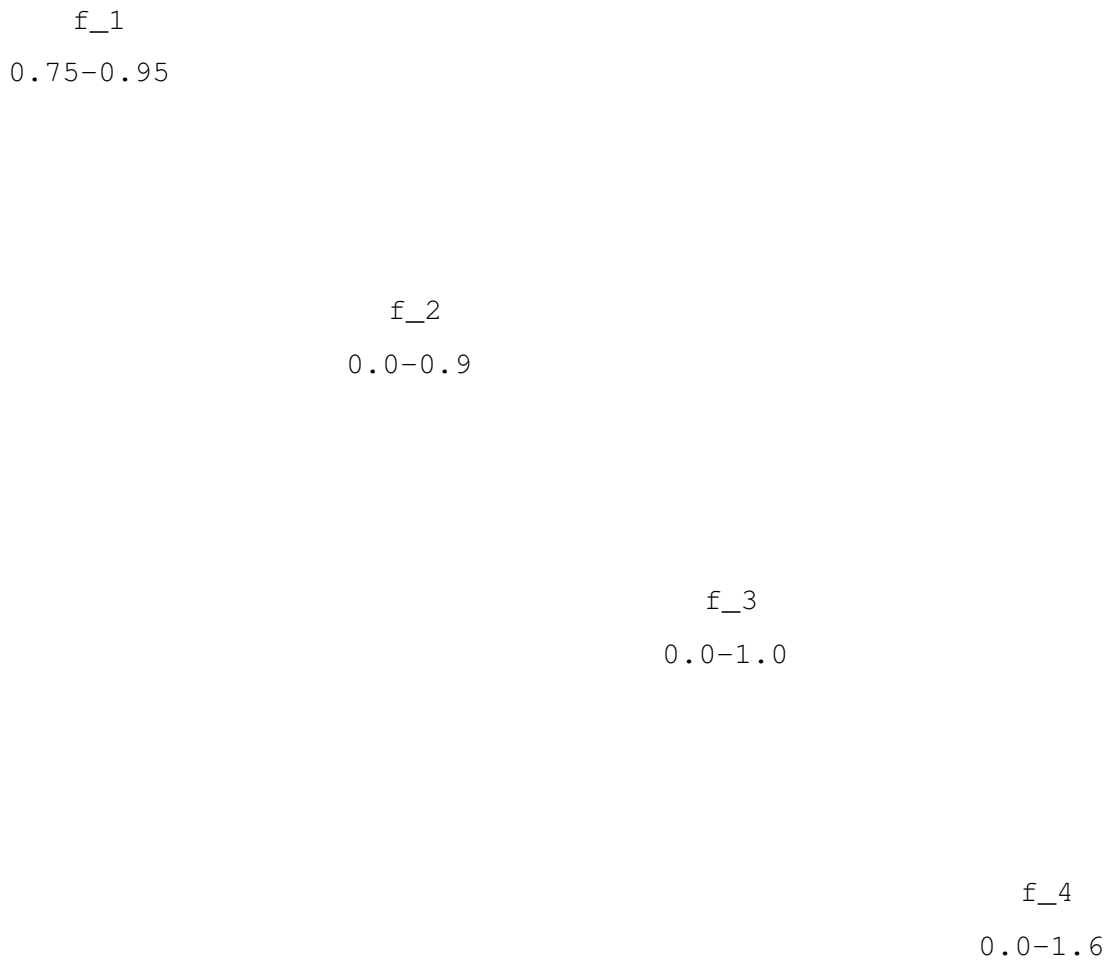


Fig. 21. Upper diagonal plots are for NSGA-II and lower diagonal plots are for Ray-Kang-Chye's algorithm. Compare (i; j) plot (Ray-Kang-Chye's algorithm with i < j) with (j; i) plot (NSGA-II). The label and ranges used for each axis are shown in the diagonal boxes.

approach used in NSGA-II and SPEA. But, the diversity preserving mechanism used in NSGA-II is undoubtedly the best among all three approaches.

On a problem having strong parameter interactions, NSGA-II has been able to come closer to the true front than other two approaches. But the important matter is that all three approaches faced di culties in solving this so-called highly

epistatic problem. Although this has been a matter of on-going research in single-objective EA studies, this paper shows that highly epistatic problems may also cause di culties to multi-objective EAs. More importantly, researchers in the eld must keep in mind of solving such problems while developing a new algorithm for multi-objective optimization.

We have also proposed a simple extension to the de nition of dominance for constrained multi-objective optimization. Although this new de nition can be used with any other multi-objective EAs, the real-coded NSGA-II with this de - nition has been shown to solve four di erent problems much better than another recently-proposed constraint handling approach.

With the properties of a fast non-dominated sorting procedure, an elitist strategy, a parameter-less approach, and a simple yet e cient constrained handling method, NSGA-II should nd increasing attention and applications in the near future.

Acknowledgements

Authors acknowledge the support provided by All India Council for Technical Education, India during the course of this study.

References

[1] K. Deb, \An e cient constraint handling method for genetic algorithms," Computer Methods in Applied Mechanics and Engineering, in press.

[2] K. Deb, \Multi-objective genetic algorithms: Problem di culties and construction of test Functions," Evolutionary Computation, Vol. 7, pp. 205{230, 1999.

[3] K. Deb and D. E. Goldberg, \An investigation of niche and species formation in genetic function optimization,", in Proceedings of the Third International Conference on Genetic Algorithms, J. D. Scha er, Ed. San Mateo, CA: Morgan Kau man, 1989, pp. 42{50.

[4] K. Deb, and S. Agrawal, \Understanding interactions among genetic algorithm parameters," in Foundations of Genetic Algorithms V, W. Banzhaf and C. Reeves, Eds. San Mateo, CA: Morgan Kau man, 1998, pp. 265{286.

[5] K. Deb, and R. B. Agrawal, \Simulated binary crossover for continuous search space," Complex Systems, Vol. 9, pp. 115{148, 1995.

[6] C. M. Fonseca, and P. J. Fleming, \Genetic algorithms for multi-objective optimization: Formulation, discussion and generalization." in Proceedings of the Fifth International Conference on Genetic Algorithms, S. Forrest, Ed. San Mateo, CA: Morgan Kau man, 1993, pp. 416{423.

[7] C. M. Fonseca, and P. J. Fleming, \On the performance assessment and comparison of stochastic multiobjective optimizers.", in Parallel Problem Solving from Nature IV, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds Be pp. 584{593.

[8] C. M. Fonseca, and P. J. Fleming, \ Multiobjective optimization and multiple constraint handling with evolutionary algorithms{Part II: Application example," IEEE Transactions on Systems, Man, and Cybernetics: Part A: Systems and Humans, pp. 38{47, 1998.

[9] D. E. Goldberg, B. Korb, and K. Deb, \Messy genetic algorithms: Motivation, analysis, and rst results," Complex Systems, Vol. 3, pp. 93{530, 1989.

[10] G. Harik, \Learning gene linkage to e ciently solve problems of bounded di culty using genetic algorithms", IlliGAL Report No. 97005, Urbana: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

[11] J. Horn, N. Nafploitis, and D. E. Goldberg, \A niched Pareto genetic algorithm for multi-objective optimization," in Michalewicz, Z., editor, Proceedings of the First IEEE Conference on Evolutionary Computation, Z. Michalewicz, Ed. Piscataway, New Jersey: IEEE Service Center, 1994, pp. 82{87.

[12] J. Knowles, and D. Corne, \The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation," in Proceedings of the 1999 Congress on Evolutionary Computation, Piscataway, New Jersey: IEEE Service Center, 1999, 98{105.

[13] F. Kursawe, \A variant of evolution strategies for vector optimization," in Parallel Problem Solving from Nature, I, H.-P. Schwefel and R. M anner, Eds Berlin: Germany: Springer, 1990, pp. 193{197.

[14] C. Poloni, \Hybrid GA for multi-objective aerodynamic shape optimization," in Genetic algorithms in engineering and computer science, G. Winter, J. Periaux, M. Galan, and P. Cuesta, Eds Chichester: Wiley, 1997, pp. 397{414.

[15] T. Ray, Kang, T., and S. Chye, \Multiobjective design optimization by an evolutionary algorithm", Engineering Optimization, in press.

[16] G. Rudolph, \Evolutionary search under partially ordered sets," Technical Report No. CI-67/99, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany.

[17] J. D. Scha er, \Multiple objective optimization with vector evaluated genetic algorithms," in Proceedings of the First International Conference on Genetic Algorithms, J. J. Grefensttete, Ed. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1987, pp. 93{100.

[18] N. Srinivas, and K. Deb, \Multi-Objective function optimization using non-dominated sorting genetic algorithms," Evolutionary Computation, Vol. 2, pp. 221{248, 1995.

[19] M. Tanaka et al., \GA-based decision support system for multi-criteria optimization,", in Proceedings of the International Conference on Systems, Man and Cybernetics-2, 1995, pp. 1556{1561.

[20] D. Van Veldhuizen, \Multiobjective evolutionary algorithms: Classi cations, analyses, and new innovations," PhD Dissertation and

Technical Report No. AFIT/DS/ENG/99-01, Dayton, Ohio: Air Force Institute of Technology, 1999.

[21] D. Van Veldhuizen, and G. Lamont, \Multiobjective evolutionary algorithm research: A history and analysis," Report Number TR-98-03. Wright-Patterson AFB, Ohio: Department of Electrical and Computer Engineering, Air Force Institute of Technology.

[22] Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: Methods and applications. Doctoral thesis ETH NO. 13398, Zurich: Swiss Federal Institute of Technology (ETH), Aachen, Germany: Shaker Verlag.

[23] E. Zitzler, K. Deb, and L. Thiele, \Comparison of multiobjective evolutionary algorithms: Empirical results." Evolutionary Computation, Vol. 8, pp. 173{195, 2000.

[24] E. Zitzler, and L. Thiele, \Multiobjective optimization using evolutionary algorithms|A comparative case study," in Parallel Problem Solving from Nature, V, A. E. Eiben, T. B ack, M. Schoenauer, and H.-P. Schwefel, Eds. Berlin, Germany: Springer, 1998, pp. 292{301.