A Unified Evolutionary Optimization Procedure for Single, Multiple, and Many Objectives

Haitham Seada and Kalyanmoy Deb, IEEE Fellow
Department of Computer Science and Engineering
Michigan State University
Email: {seadahai,kdeb}@msu.edu
www.egr.msu.edu/~kdeb/COIN.shtml

Abstract—**Traditionally, evolutionary algorithms have been systematically developed to solve mono-objective, multi-objective and many-objective optimization problems, in this order. Despite some efforts in unifying different types of mono-objective evolutionary and non-evolutionary algorithms, researchers are not interested enough in unifying all three types of optimization problems together. Such a unified algorithm will allow users to work with a single software enabling one-time implementation of solution representation, operators, objectives and constraints formulations across several objective dimensions. For the first time, we propose a unified evolutionary optimization algorithm for solving all three classes of problems specified above, based on the recently-proposed elitist, guided non-dominated sorting procedure, developed for solving many-objectives problems. Using a new niching based selection procedure, our proposed unified algorithm automatically degenerates to an efficient equivalent population-based algorithm for each class. No extra parameters are needed. Extensive simulations are performed on unconstrained and constrained test problems having single, two, multi and many-objectives and on two engineering optimization design problems. Performance of the unified approach is compared to suitable population-based counterparts at each dimensional level. Results amply demonstrate the merit of our proposed unified approach and motivate similar studies for a richer understanding of the development of optimization algorithms.**

Index Terms—**Unified algorithms, mono-objective optimization, multi-objective optimization, many-objective optimization, NSGA-III.**

I. Introduction

DURING the past two decades, evolutionary multi-objective optimization (EMO) algorithms have demonstrated their usefulness in solving optimization problems having two and more objectives [1], [2], [3], [4]. With no more than three objectives in mind, most of the emphasis was put onto the ability of the algorithm to distribute population members over the entire efficient front [5], [6], [7], [8], [9], [10]. Recently, the term 'Many-Objective Optimization' was coined to refer to problems having more than three objectives [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23]. Because of the exponential increase in the number of non-dominated solutions with the increase in dimensions, most existing domination based EMO algorithms do not scale up to more than three objectives. To alleviate, new algorithms have been recently proposed mostly using an external guidance mechanism to help the algorithm distribute its population along higher-dimensional efficient fronts [22], [24], [25], [26].

Although certain evolutionary multi-objective optimization methodologies such as NSGA-II [6] do not scale up to solve many-objective optimization problems efficiently, they are found to work well in solving mono-objective optimization problems [27]. Based on NSGA-II framework, an omni-optimizer algorithm [28] was suggested to solve mono- and multi-objective optimization problems. This is because the domination operator used in NSGA-II's selection mechanism becomes an ordinal comparison operator, which is an essential operation for progressing towards the optimum solution for a mono-objective optimization problems. Thus, these multi-objective optimization methods can be considered as unified methods for solving mono- and multi-objective optimization problems, but omni-optimizer was certainly not suitable for solving many-objective problems.

On the other hand, existing many-objective optimization methods [22], [25], [26], [29], [30] are tested for three and more objective problems and have not been adequately evaluated for their performance in solving mono- and bi-objective optimization problems. One apparent difficulty of their scaling down to solve mono-objective problems is that the objective space becomes one-dimensional and the inherent guidance mechanism which ensures diversity of population members in the

All supplementary materials of this study are found at: http://www.msu.edu/~seadahai/supplementary/unsga3.htm referred to throughout the paper as Table *Sx* or Figure *Sx*.

objective space becomes defunct. Hence, currently there is no algorithm that can scale both ways - up and down - automatically (without additional setup or coding), while maintaining robust efficient performance across different dimensions.

In addition to the aforementioned difficulties, there is another practical motivation for our study, which we discuss next. To solve an optimization problem using an already existing software package, the problem must be first implemented (coded or expressed symbolically) within this software package. Often, this implementation process involves linking the optimization code or software with a third-party evaluation software such as a finite element code, a computational fluid dynamics code or a network flow simulator etc. In addition, it is recommended to introduce some algorithmic modifications to the optimization procedure itself according to the problem at hand [31], [32] by introducing new operators and/or modifying existing genetic operators with 'heuristics' of the problem. Also, instead of starting an optimization run from a random initial population, a heuristically biased initial population is created. Such algorithmic modifications and customized initializations involve careful analysis, efforts, and are certainly time-consuming. In practice, solving a higher dimensional problem usually involves solving lower-dimensional versions of the same problem. For example, pre-optimization knowledge of ideal and nadir points will aid the multi/many objective optimization process. Getting to know these two key pieces of information is only possible, through solving single objective optimization versions of the same problem, for each objective independently. Such several lower-dimensional runs are executed also to verify or gain confidence in the obtained higher-dimensional efficient front [33]. In design exploration problems, objectives and constraints are interchanged to get a better idea of the possible range of optimal solutions [34]. In such situations, if different optimization algorithms are needed for different objective-dimensions of the original optimization problem, the algorithmic modifications discussed above need to be re-implemented to every optimization algorithm used to solve every objective-dimensional version of the problem, thereby making the overall process slow, tedious, and also error-prone. If instead, one unified optimization algorithm capable of handling one to many-objective problems efficiently is available, a one-time algorithmic modification based on heuristics, one-time implementation of the problem description, and one-time linking with an evaluation software would be enough to solve different versions of the original problem, thereby providing flexibility, saving time, efforts, and most importantly making the process less error-prone.

In this paper, we make an effort to develop a single unified evolutionary optimization procedure that will solve mono-, multi- and many-objective optimization problems efficiently. Such an algorithm will not only allow a user to solve different types of problems, but also an understanding of the algorithmic features needed in such an efficient unified approach would be beneficial for EMO researchers. The successful development of a unified approach for handling one-to-many objectives will also provide a triumph of generic computing concept in optimization problem solving. The philosophy of computing through a computerized software is to implement an algorithm that is most generic capable of working with multiple and arbitrary number of input data. At the same time, when the software is applied to a lower-dimensional data or even to a single data, the software is expected to work as a specialized lower-dimensional or single-dimensional algorithm would perform. Unfortunately, optimization literature has traditionally followed an opposite philosophy. A lot of stress has been put in developing mono-objective optimization algorithms and often multi- or many-objective optimization problems are suitably converted to a mono-objective optimization problem so as to use mono-objective optimization algorithms. Our motivation in this paper is to explore the possibility of developing a unified optimization approach that naturally solves many-objective problems having four or more objectives and degenerates to solve one, two or three-objective problems, as efficiently as other competing optimization algorithms in each dimension.

Our proposed unified approach is based on one of the recently proposed many-objective optimization algorithms – NSGA-III [25]. The key behind the success of this unification in the new niching based selection operator, which adapts selection pressure automatically according to the dimensionality of the problem in hand. In the remainder of this paper, we provide a brief description of NSGA-III in Section II due to its algorithmic similarity with the proposed U-NSGA-III. Thereafter, we present our proposed unified approach U-NSGA-III in Section III and explain how the method degenerates to efficient mono- and multi-objective optimization algorithms. Simulation results on a variety of mono, multi- and many-objective problems, both constrained and unconstrained are presented using U-NSGA-III and compared with a real-parameter genetic algorithm, NSGA-II and NSGA-III in Section IV. Finally, conclusions are drawn in Section V.

II. A Brief Introduction to NSGA-III

From this point on, we will refer to number of objectives as M, population size as N, number of reference direction as H and number of divisions used to distribute reference directions on the front as p. Without loss of generality, all problems in this study are to be minimized. Maximization problems can be converted to minimization by negating its objective function. The proposed U-NSGA-III algorithm is based on the structure of NSGA-III; hence we first give a description of NSGA-III here.

NSGA-III starts with a random population of size N and a set of widely-distributed pre-specified M-dimensional reference points H on a unit hyper-plane having a normal vector of ones covering the entire R_+^M region. Das and Dennis's technique [35] is used to place $H = \binom{M+p-1}{p}$ reference points on the hyper-plane having $(p+1)$ directions along each boundary. A reference direction is a ray starting at the origin and passing through a supplied reference point. The population size N is chosen to be the smallest multiple of four greater than H, with the idea that for every reference direction, one population member is expected to be found.

Generation-wise NSGA-III follows the same general outline of 2. Since only one population member is expected to be found for each reference direction, there is no need for selection in NSGA-III, as it will allow a competition to be set among different reference directions. The major difference between NSGA-II and NSGA-III is replacing $crowding-distance-based$ niching with $reference-directions-based$ niching. Let us denote the final front that could not be completely accommodated in the next generation as F_L. In general, only a few solutions from F_L needs to be selected for the next population P_{t+1} using a niche-preserving operator, which we describe next. First, each population member of P_{t+1} and F_L is *normalized* by using the current population spread so that all objective vectors and reference directions have commensurate values. Thereafter, each member of P_{t+1} and F_L is *associated* to the closest reference direction in terms of the shortest perpendicular distance ($d()$). Then, a careful *niching* strategy is employed to choose those F_L members that are associated with the least represented reference directions in P_{t+1}. A population member associated with an under-represented or un-represented reference direction is immediately preferred. With a continuous stress for emphasizing non-dominated individuals, the whole process is then expected to find one population member corresponding to each supplied reference direction close to the Pareto-optimal front.

The original NSGA-III study [25] have been demonstrated to work well from three to 15-objective DTLZ and other problems. A key aspect of NSGA-III is that it does not require any additional parameter. The method was also extended to handle constraints without introducing any new parameter. That study has also introduced a computationally fast approach by which the reference directions set is adaptively updated on the fly based on the association status of each reference direction over a number of generations. The algorithm is outlined in Algorithm 1.

Algorithm 1 Generation t of NSGA-III procedure

```
Input: H structured reference directions $Z^s$ or supplied
    aspiration directions $Z^a$, parent population $P_t$
Output: $P_{t+1}$
 1: $S_t = \emptyset$, $i = 1$
 2: $Q_t$ = Recombination+Mutation($P_t$)
 3: $R_t = P_t \cup Q_t$
 4: $(F_1, F_2, \ldots)$ = Non-dominated-sort($R_t$)
 5: repeat
 6:     $S_t = S_t \cup F_i$ and $i = i + 1$
 7: until $|S_t| \geq N$
 8: Last front to be included: $F_l = F_i$
 9: if $|S_t| = N$ then
10:     $P_{t+1} = S_t$, break
11: else
12:     $P_{t+1} = \cup_{j=1}^{l-1} F_j$
13:     Members to be chosen from $F_l$: $K = N - |P_{t+1}|$
14:     Normalize objectives and create reference set $Z^r$:
        Normalize($\mathbf{f}^n, S_t, Z^r, Z^s, Z^a$)
15:     Associate each member s of $S_t$ with a reference
        direction: $[\pi(\mathbf{s}), d(\mathbf{s})]$ =Associate($S_t, Z^r$)
        % $\pi(\mathbf{s})$: closest reference direction to s
        % $d$: distance between s and $\pi(\mathbf{s})$
16:     Compute niche count of reference direction $j \in Z^r$:
        $\rho_j = \sum_{\mathbf{s} \in S_t/F_l} ((\pi(\mathbf{s}) = j)\ ?\ 1\ :\ 0)$
17:     Choose $K$ members one at a time from $F_l$ to
        construct $P_{t+1}$: Niching($K, \rho_j, \pi, d, Z^r, F_l, P_{t+1}$)
18: end if
```

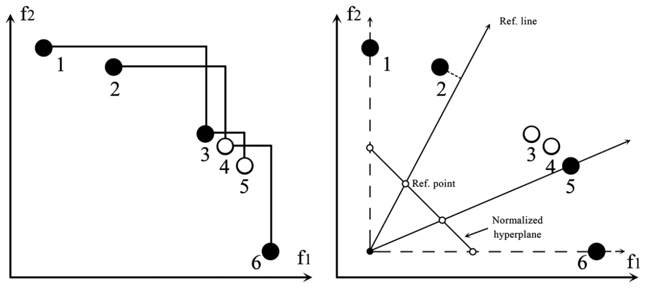
A. NSGA-III for Mono- and Multi-objective Problems

NSGA-III was primarily proposed to solve many-objective optimization problems having more than three objectives, although NSGA-III was demonstrated to work well on three-objective optimization problems. Authors of NSGA-III did not consider any bi-objective or mono-objective problems in the original study. Here, we discuss the potential of using NSGA-III in two-objective problems and then highlight its difficulties in downscaling to solve mono-objective optimization problems.

The differences in working principles of NSGA-II and NSGA-III on two-objective problems are outlined below:

1) NSGA-III does not use any explicit selection operator on P_t in the process of creating Q_t. On the other hand NSGA-II's selection operator uses non-dominated rank and a crowding distance value to choose a winner between two feasible individuals from P_t. It is worth noting however that, NSGA-III performs selection if and only if at least one of the two individuals being compared is infeasible. In that case NSGA-III prefers feasible over infeasible, and less violating over more violating individuals.
2) NSGA-III uses a set of reference directions to maintain diversity among solutions, while NSGA-II uses a more adaptive scheme through its crowding distance operator for the same purpose. As illustrated in Figure 1. Assuming the two algorithms are required to select two points out of six existing non-dominated points for the next generation, NSGA-II will prefer the black dots in Figure 1a over white dots, because they have the largest surrounding empty space. On the other hand, NSGA-III will prefer the four points that best represent the four

supplied reference directions, which are the black dots shown in Figure 1b.



(a) NSGA-II (b) NSGA-III

Fig. 1: Working principles of NSGA-II and NSGA-III.

If NSGA-III, having a population size almost identical to number of chosen reference directions is compared to NSGA-II having an identical population size as in NSGA-III, the former will introduce a milder selection pressure. This is because on average each population member in NSGA-III becomes associated with a different reference direction and becomes too important to be compared with another individual. The only selection pressure comes from their domination levels. However, the second point mentioned above may produce a significant difference in their performances. NSGA-III uses a pre-defined guidance mechanism to choose diverse solutions in the population, whereas NSGA-II uses no pre-defined guidance and emphasizes relatively diverse solutions on the fly. Thus, if the first aspect is taken care of somehow and more selection pressure is introduced, NSGA-III may become an equivalent or even a better algorithm than NSGA-II for solving bi-objective optimization problems.

Let us now discuss how NSGA-III would work on a mono-objective optimization problem. In mono-objective optimization, the domination concept degenerates to fitness superiority – a domination check between two solutions chooses the one having better objective value. At every generation, it is expected that one solution would occupy each non-dominated front in a mono-objective problem. Thus, it is expected to have N fronts in a population of size N. These characteristics of mono-objective problems affect the working of NSGA-III in the following manner:

1) First, in NSGA-III, there will be only one reference direction (the real line) to which all the individuals will be associated. Since the recommended population size is the smallest multiple of 4 greater than the number of reference directions, for all mono-objective optimization problems, NSGA-III will use a population of size four, which for all practical purposes is too small for NSGA-III's recombination operator to find useful offspring solutions. This is a major issue in developing a unified algorithm that will seamlessly work for many to mono-objective problems.
2) Moreover, since no explicit selection operator is used, the algorithm will pick a random solution for its recombination and mutation operators. The only selection effect comes from the elite-preserving operation for choosing P_{t+1} from a combination of P_t and Q_t. This is another major issue, which needs to be addressed while developing a unified approach.
3) Note also that the niching operation of NSGA-III becomes defunct for mono-objective problems, as there is no concept of perpendicular distance of a function value from the reference direction. Every function value falls on the real line, providing an identical perpendicular distance of zero to each population member.
4) NSGA-III's normalization also becomes a defunct operation for the same above reason.

It is now clear that a straightforward application of original NSGA-III to mono-objective optimization problems will result in an extremely small population size and a random selection process, neither of which is recommended for a successful evolutionary optimization algorithm. However, the niching and normalization operators of NSGA-III are essential for it to be successful in multi and many-objective optimization problem. Thus, a modification of NSGA-III is needed so that the resulting unified approach becomes efficient for mono to many-objective problems by making the niching and normalization operators automatically defunct for mono-objective problems and active for multi and many-objective problems.

III. Proposed Unified Approach: U-NSGA-III

The above discussion suggests that the proposed U-NSGA-III method can retain the features of the original NSGA-III algorithm, as NSGA-III was shown to work well on three or more objectives. However, the difficulties in scaling down to two and mono-objective problems mentioned above require certain changes in NSGA-III algorithm, but we should be modifying NSGA-III in such a manner that the changes do not affect its working on three and more objective problems.

The difficulty for solving two-objective optimization problems seems to lie in the mild selection pressure that NSGA-III introduces to non-dominated solutions of a population, while the difficulties for solving mono-objective problems are small population size and the random selection process. One way to alleviate these difficulties is to use a population size N which is larger than the number of reference directions (H) and introduce a selection operator. Thus, unlike in NSGA-III, N and H will now be different parameters with a condition that $N \geq H$ and N is a multiple of four. Although this seems to introduce an additional parameter to our proposed U-NSGA-III, it doesn't. H is the desired number of optimal solutions expected at the end of a simulation run, and hence is not a parameter that needs to be tuned for U-NSGA-III to work well on different problems. Although in some cases it is difficult for a decision maker to determine his actual preferences, it is much easier for him to determine the number of desired alternatives among which he will choose from (H). Once H is determined, Das and Dennis's technique [35] can be used to generate a number of reference directions (as close to H as possible) covering the whole Pareto front i.e. representing the whole range of trade-offs among all objectives. of the whole front. We shall soon investigate the effect of this change for different problem sizes, but for mono-objective problems H is always one and N becomes simply the population size which is a generic parameter in all mono-objective evolutionary algorithms. For two-objective problems, H can be a handful of solutions (such as 10 or 20) for the decision-makers to consider, while the population size N can be much larger, such as 100 or 200. The population size consideration mainly comes from the complexity of the problem and an adequate sample size needed for the genetic operators (essentially the recombination operator) to work well. In this case, although N different Pareto-optimal solutions could be present in the final population, the H specific Pareto-optimal solutions, each closest to a different reference direction will be the outcome of the U-NSGA-III algorithm and will be presented to the decision-maker for choosing a single preferred solution. For three or more objective problems, since the number of specified reference directions (H) can already be quite high (due to the increase in H with M according to Das and Dennis's approach [35]), N can be made almost equal to H with the divisibility by four restriction.

Let us now discuss the algorithmic implications of introducing more population members than H for solving mono- and two-objective optimization problems. It is now expected that for each reference direction, there will be more than one population member associated. This then allows us to introduce a selection operator to have an adequate selection pressure for good population members. We add a niching-based tournament selection operator as follows. If the two solutions being compared come from two different associated reference directions, one of them is chosen at random, thereby introducing preservation of multiple niches in the population. Otherwise, the solution coming from a better non-dominated rank is chosen. In this case, if both solutions belong to the same niche (reference direction) and same non-dominated front, the one closer to the reference direction is chosen. Algorithm 2 presents the niched tournament selection procedure in a pseudo-code form, in which two feasible parent solutions (p_1 and p_2) are compared to choose a winner (p_s). If at least one of them is infeasible, the traditional NSGA-III selection is used. This operation can be repeated $N/2$ times systematically by using two consecutive population members of the parent population P_t to choose $N/2$ parents. The procedure can be repeated one more time by shuffling population P_t to obtain another set of $N/2$ parents. These two chosen parent sets can be combined to form the complete mating pool P'_t of size N in the NichingBasedSelection(P_t) procedure. The mating pool P'_t can then be used to create the offspring population Q_t by using usual recombination and mutation operators. Thus, the complete U-NSGA-III procedure can be achieved by simply replacing line 2 in Algorithm 1 with the two lines shown in Algorithm 4.

Algorithm 2 Niching based selection of U-NSGA-III.

```
Input: Two parents: $p_1$ and $p_2$
Output: Selected individual, $p_s$
 1: if $\pi(p_1) = \pi(p_2)$ then
 2:     if $p_1.rank < p_2.rank$ then
 3:         $p_s = p_1$
 4:     else
 5:         if $p_2.rank < p_1.rank$ then
 6:             $p_s = p_2$
 7:         else
 8:             if $d_\perp(p_1) < d_\perp(p_2)$ then
 9:                 $p_s = p_1$
10:             else
11:                 $p_s = p_2$
12:             end if
13:         end if
14:     end if
15: else
16:     $p_s = randomPick(p_1, p_2)$
17: end if
```

Algorithm 3 Degenerated U-NSGA-III algorithm for mono-objective problems.

```
Input: Mono-objective problem
Output: Best solution found, $p_{best}$
 1: $P = initialize()$
 2: while termination condition do
 3:     $Q = \phi$
 4:     while $|Q| < |P|$ do
 5:         $p_1 = tournamentSelect(P)$
 6:         $p_2 = tournamentSelect(P)$
 7:         $(c_1, c_2) = recombination(p_1, p_2)$
 8:         $c_1 = mutate(c_1)$
 9:         $c_2 = mutate(c_2)$
10:         $Q \cup \{c_1, c_2\}$
11:     end while
12:     $P = best(P \cup Q)$
13: end while
14: $p_{best} = best(P)$
```

For mono-objective problems, the flexibility of choosing an arbitrary population size alleviates one of the discussed difficulties. The niched selection operator degenerates to a usual binary tournament selection operator for which the solution having a better objective value becomes the winner. We now present the respective degenerative U-NSGA-III algorithm for solving mono-, multi- and many-objective optimization problems.

Algorithm 4 Generation t of U-NSGA-III procedure.

```
Input: H structured reference directions $Z^s$ or supplied aspiration directions $Z^a$, parent population $P_t$
Output: $P_{t+1}$
    ⋮  % Identical to Algorithm 1 (Line 1)
 2: $P'_t$ = NichingBasedSelection($P_t$)
 3: $Q_t$ = Recombination+Mutation($P'_t$)
    ⋮  % Identical to Algorithm 1 (Lines 3 to 18)
```

A. U-NSGA-III for Mono-objective Problems

Algorithm 3 presents a pseudo-code for the resulting U-NSGA-III algorithm when $M = 1$ is specified. It is interesting to note that Das and Dennis's [35] strategy results in $\binom{1+p-1}{p}$ or one single reference direction and is independent of the value of p. In this special case, all individuals will be attached to the same only-existing reference direction. Consequently our niching-based selection operator will choose one of two individuals based on their non-dominated ranking, which turns, in the presence of only one objective, into an ordinal comparison operator. Thus, the whole selection mechanism becomes an ordinal comparison operator between two randomly selected individuals, or what we simply call *binary tournament selection*. In addition, merging and non-dominated sorting - in this special case - will be equivalent to selecting the top N individuals after combining both parents and offspring populations. This is the classic elite-preserving mechanism used in Evolutionary Strategies (ES). Hence, the resulting U-NSGA-III in this special case is a generational evolutionary algorithm (EA) whose niching and normalization operators are defunct (as mentioned earlier), and uses (i) a binary tournament selection, (ii) recombination and mutation operators, and (iii) an elite-preserving operator. Thus, our proposed U-NSGA-III is said to *degenerate* into other generational EAs, such as elite-preserving real-coded genetic algorithm [36] or the $(\mu/\rho+\lambda)$ evolution strategy [37], where $\mu = \lambda = N$ and $\rho = 2$.

B. U-NSGA-III for Multi-objective Problems

For two or three-objective problems, in case N is chosen to be greater than H, U-NSGA-III is expected to have multiple population members for each reference direction. For multi-objective problems having two or three objectives, the non-dominated sorting will, in general, divide the population into multiple non-dominated fronts. The proposed niched tournament selection operator of U-NSGA-III then emphasizes (i) non-dominated solutions over dominated solutions and (ii) solutions closer to reference directions over other non-dominated but distant solutions from the reference directions. The rest of the U-NSGA-III algorithm works the same way as NSGA-II would work on multi-objective problems. However, although like in NSGA-II, all members of the final population of U-NSGA-III are also expected to be non-dominated to each other, the distribution of additional $(N-H)$ population members need not have a good diversity among them. Only H population members closest to each H reference directions are expected to be well distributed. But since the user is interested in getting H solutions at the end (implied in the beginning by specifying H reference directions), additional $(N-H)$ directions help in making the algorithm more efficient to arrive at precisely H target Pareto-optimal points.

However, when N/H is chosen to be one or close to one, U-NSGA-III algorithm may not have a solution for each reference direction in the early generations, but the selection pressure introduced by the niched tournament selection will emphasize finding and maintaining a single population member for each reference direction until they are all found. In either case of N/H being one or more than one, U-NSGA-III provides adaptively adequate selection pressure for it to be an efficient algorithm for handling the problem in hand. For three-objective problems, the uniform distribution of supplied reference directions in NSGA-III should find a better distributed efficient points than adaptive discovery of points by NSGA-II.

C. U-NSGA-III for Many-objective Problems

For many-objective optimization problems, most population members are expected to be non-dominated to each other. Hence, the niched tournament selection operator degenerates in choosing the closer of the two parent solutions with respect to their associated reference direction, when both parent solutions lie on the same niche. When N/H is much greater than one, this allows an additional filtering of choosing parent solutions closer to reference directions for their subsequent mating operation. This is, in general, a good operation to have particularly when there are multiple population members available around a specific reference direction, but due to requirement of a large value of H in many-objective problems, U-NSGA-III with N greater than H may end up with a large computational effort. However, if N/H is one or close to one (which is recommended for many-objective problems), the niched tournament selection, in most cases, becomes a defunct operator, and the algorithm degenerates to the original NSGA-III.

The above properties of U-NSGA-III suggests a possible way to construct a single unified optimization algorithm that automatically degenerates to efficient optimization algorithms for mono-, multi- and many-objective optimization problems simply by the specification of the number of objectives presented in the problem description. The number of reference directions H is dictated by the number of points desired along each objective axis. The population size parameter N is detached from H and the user is free to provide any value greater or equal to H. To make a systematic application pair-wise selection and pair-wise recombination operations, we still recommend to use N which is multiple of four. The proposed U-NSGA-III is capable of handling constraints the same way NSGA-II and NSGA-III are. The introduced niched selection operation requires $O(N)$ computations. As discussed in NSGA-III study, the rest of the computations is bounded by the maximum of $O(N^2 \log^{M-2} N)$ or $O(MN^2)$. Thus, $M > \log^{M-2} N$, the generation-wise complexity of the overall U-NSGA-III procedure is $O(MN^2)$, which is similar to those of NSGA-II and NSGA-III.

IV. Results

In the following sections, we present simulation results of U-NSGA-III applied to a wide variety of constrained and unconstrained mono-, multi- and many-objective problems. A couple of real-world engineering problems are also solved using the proposed U-NSGA-III algorithm. The stopping criterion is set according to the type of experiment being conducted. When the stopping criterion is a fixed maximum number of generations, this number is chosen according to the difficulty of the problem. In most situations, more-than-required number of generations is used to show that all algorithms were given enough time to reach their best performance. It is worth noting that all hypervolume values included in this study are calculated using JMetal optimization software package [38].

A. Mono-objective Problems

First, we present the results of U-NSGA-III on standard mono-objective optimization problems.

1) Unconstrained Problems: For the unconstrained case, we chose six mono-objective test problems as our test-bed, namely, Ellipsoidal, Rosenbrock's, Zakharov's, Schwefel's, Ackley's and Rastrigin's. The performance of U-NSGA-III is compared to a generational real-parameter genetic algorithm (EliteRGA) which was used to solve various problems in the past [39], as well as CMA-ES which is a state of the art algorithm for single objective optimization. We employ an elite-preserving operator between parent and offspring populations in the EliteRGA algorithm to make it equivalent to the degenerated form of U-NSGA-III for mono-objective problems (that is, the $(\mu/2 + \mu)$-ES form). Problem definitions are given in Equations *S1-S6* (supplementary material at www.msu.edu/~seadahai/supplementary/unsga3.htm):

For each problem, $n = 20$ is used and 31 simulations with the same set of parameters but from different initial populations are conducted. Although the first three problems are simple, f_{sch} and f_{ras} are difficult multi-modal problems. We use N = 48, 100, 100, 100 and 300 for Ellipsoidal, Rosenbrock's, Zakharov's, Rastrigin's and Schwefel's problems, respectively.

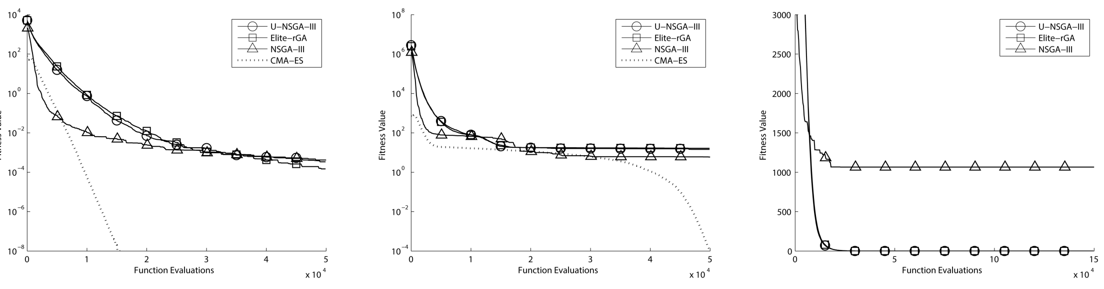
Figures 2a, 2b, *S1*, *S2* and 2c show the median function value (y-axis) for Ellipsoidal, Rosenbrock's, Rastrigin's, Zakharov's and Schwefel's problems over 31 runs for different function evaluations (x-axis). EliteRGA, NSGA-III and U-NSGA-III perform similarly for unimodal and easy problems. However the are totally outperformed by CMA-ES at this category of problems. On the other hand, for the multi-modal problems U-NSGA-III and EliteRGA are the best, since both NSGA-III and CMA-ES are very prone to be trapped in local optima, as shown in Figures 2c and *S1*.

In order to confirm the superiority of U-NSGA-III over NSGA-III in multi-modal problems, we did the same comparison again on two additional multi-modal test problems, which are Ackley's and Rastrigin's. Again, it was clear that NSGA-III is prone to get stuck into local optimum because of its prohibitive restriction on the population size. Even in the cases where NSGA-III was able to converge to the global optimum, we found it to be very sensitive to the polynomial mutation index (η_m). Only, a small (close to zero) η_m may enable NSGA-III to escape from local optimum to converge to the global optimum. This observation is clearly visible in Figures *S3* and *S4* for Ackley's and Rastrigin's, respectively.

Table *S3* summarizes the best, median and worst fitness values achieved by each of the four algorithms on unconstrained test problems.

2) Constrained Problems: The superiority of U-NSGA-III and its equivalence to EliteRGA are more evident when it comes to constrained problems. The three algorithms under investigation have been tested against ten constrained test problems from the G-family test suite [40]. Best, median and worst achieved fitness values by each algorithm in each problem are presented in Table *S4*. Obviously, according to Figures *S5* and *S6*, NSGA-III in most cases is not able to converge to the global optimum as expected. Again, U-NSGA-III and EliteRGA perform almost identically.

We conducted a Wilcoxon significance test over the median of the final result of our 31 runs in each experiment. Tables *S1* and *S2* show the $p - values$ of U-NSGA-III vs. NSGA-III, U-NSGA-III vs. EliteRGA and U-NSGA-III vs CMA-ES (the last comparison is conducted only for unconstrained test problems, because the original CMA-ES and its available implementation support only boxing constraints). Obviously, there is a statistically significant difference between U-NSGA-III and NSGA-III especially in multimodal problems (largest $p = 9.2668 \times 10^{-5}$). On the other hand, when comparing U-NSGA-III and EliteRGA these large $p - values$ reported in the majority of the problems, means that we have to accept the null hypothesis. The null hypothesis simply means that the two distributions – from which the two sets of results are taken – have the same median. In other words, the two algorithms perform similarly without any statistically significance difference between them. It is also evident, that there is a statistically significant difference between U-NSGA-III and CMA-ES (largest $p = 5.7257 \times 10^{-8}$). However, according to the corresponding figures, this difference does not mean that one of them is better than the other in all problems. Actually, U-NSGA-III is found to be much better



(a) Ellipsoidal function (b) Rosenbrock's function (c) Schwefel's function

Fig. 2: Objective function value with respect to the number of function evaluations.

than CMA-ES in multimodal problems (see Rastrigin's and Schwefel's), while CMA-ES significantly outperforms U-NSGA-III in simpler problems (see Ellipsoidal, Rosenbrock's and Zakharov's). Similar conclusions can be reached for constrained test problems.

B. Bi-objective Problems

As mentioned before, the performance of NSGA-III was not tested on bi-objective optimization problems in the original study [25]. In this section, NSGA-II NSGA-III and U-NSGA-III are compared over an extensive set of unconstrained and constrained bi-objective problems. Two types of experiments are shown here. These two experiments are supposed to help us draw conclusions about the niching mechanisms used in the three algorithms, and the effect of the niching-based selection operator of U-NSGA-III. We used ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 as our unconstrained testbeds while, OSY, TNK, BNH and SRN are used as constrained test problems. Two additional real-world engineering problems – welded beam and pressure vessel design [41] – are also included. For all runs, we use SBX $p_c = 0.9$, $\eta_c = 30$ and polynomial mutation $p_m = 1/n$ and $\eta_m = 20$. In all ZDT problems, we use 30 variables and 31 different simulation runs are performed (except for ZDT6, where we use 10 variables, as used in the original study [42]).

1) N Equal to H: The first experiment compares the performance of the three algorithms when population size (N) is equal to the number of reference directions (H). By comparing the results of NSGA-II and NSGA-III only (ignoring U-NSGA-III for the time being), we can draw conclusions about the effectiveness of both NSGA-II and NSGA-III on bi-objective problems. Also, we can draw conclusions about the effect of the new niching-based selection operator introduced in U-NSGA-III by comparing the results of NSGA-III and U-NSGA-III only (ignoring NSGA-II).

Figures 3 and *S7* show the best, median and worst hypervolume (HV) achieved by each of the three algorithms on ZDT1, ZDT2, ZDT4 and ZDT6 (the figure of ZDT6 is removed for the sake of brevity). HV is calculated using an HV reference point 1% larger in every component than the corresponding *nadir* point.

Although the results are comparable in most cases, NSGA-II's performance deteriorates significantly in ZDT4 compared to NSGA-III and U-NSGA-III especially at smaller population sizes. Table *S5* shows that the three algorithms are close to each other. However, from Table *S8*, we can see than in ZDT3 and ZDT6, there is a statistically significant difference in favor of NSGA-II, while in ZDT4, U-NSGA-III significantly outperforms NSGA-II.

The same observation can be noticed for constrained test problems in Figure *S8*.

The more difficult the problem is (BNH and SRN), the bigger the difference in favor of NSGA-III and U-NSGA-III. It is also clear that NSGA-II is the most negatively affected by very small population sizes (namely, $N = 8$), while both NSGA-III and U-NSGA-III are more robust with respect to small population sizes (see Table *S6*).

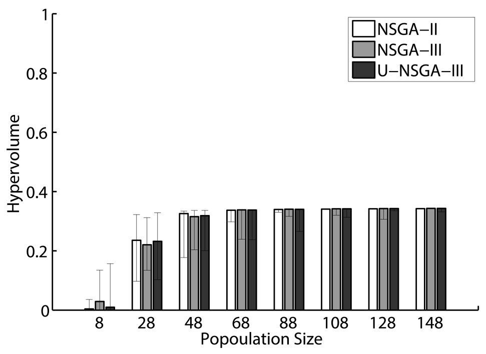
The same observation can be made in Figures *S9* and *S10* showing the results of welded beam and pressure vessel problems, respectively. The ideal and reference points used to compute HV values for constrained problems are shown in Table III.

The statistical significance test results (Table *S9*) shows that there is a statistically significant difference in favor of U-NSGA-III over NSGA-II in BNH, SRN and Pressure Vessel. The opposite however is never valid.

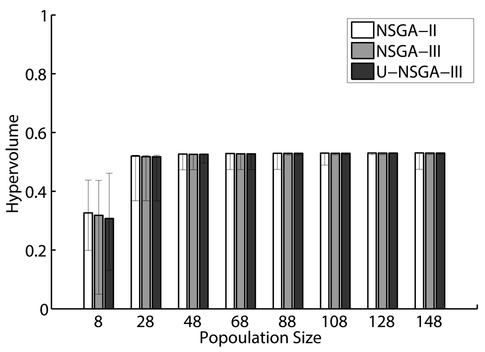
2) N Greater Than H: In the second set of experiments for handling bi-objective problems, we evaluate the usefulness of using $N \geq H$. In this experiment, two straight lines, one representing the performance of NSGA-II with $N = 16$ and another representing the performance of NSGA-III with $N = H = 16$ are shown for convenience. The jagged line represents the performance of U-NSGA-III with $N \geq H$ and $H = 16$.

The criterion of comparison used here is the number of function evaluations required to reach a pre-defined threshold hyper-volume (HV) value. Table I presents the HV values used in our bi-objective simulations.

For U-NSGA-III, we have used different population sizes $N \geq H$. Average numbers of function evaluations over 31 runs – needed to achieve the prespecified HV – are plotted in Figures 4 and *S11* for unconstrained ZDT



(a) ZDT2



(b) ZDT3

Fig. 3: Performance of NSGA-II, NSGA-III, and U-NSGA-III with $N = H$ on unconstrained bi-objective test problems.

TABLE I: Threshold HV values used in bi-objective ZDT problems for $N \geq H$ simulations.

Problem	Fixed HV	Problem	Fixed HV
ZDT1	0.640	ZDT4	0.530
ZDT2	0.316	ZDT6	0.390
ZDT3	0.512		

test problems. Constrained problems results are shown in Figure *S12*.

For runs having a population size larger than number of reference directions, out of all the individuals attached to a reference direction, only one contributes to the HV calculation (the closest). that is, for all U-NSGA-III simulations, only 16 individuals one closest to each specified reference direction are used to calculate the final HV value, no matter what population size is used. We did so to retain our ability to compare HV values for all simulations with different population sizes. For ZDT1, ZDT3 and TNK, the use of a larger population size is not found to be beneficial, whereas for ZDT2, ZDT4, BNH and SRN (more difficult problems), a larger population brings in the necessary diversity needed to solve the respective problem adequately. It is clear that in all problems, there exists certain population sizes, in general, higher than H that make U-NSGA-III to perform better than NSGA-III and NSGA-II. For relatively difficult problems, the difference is quite obvious. In most cases, however, the performance of NSGA-III is better than NSGA-II, due to the use of an external guidance for diversity through a uniformly distributed set of reference directions. These results are interesting and demonstrate the usefulness of a larger population size than the number of reference directions for the proposed U-NSGA-III algorithm.

C. Three-objective Problems

To enable U-NSGA-III to work well on mono- and bi-objectives, there should not be any performance degradation to three and many-objective problems. In this section, we present results on three-objective unconstrained DTLZ1, DTLZ2, scaled DTLZ1 and scaled DTLZ2 problems. We also include the two constrained test problems C3-DTLZ1 and C3-DTLZ4 proposed in [26]. In this section U-NSGA-III is compared to both NSGA-II and NSGA-III.

The performance metric used for these problems is also the hypervolume metric, but due to the increased computational efforts in extending the hypervolume computation to many-objective problems, we use the fast technique proposed elsewhere [43]. It is understood that DTLZ problems have mathematically defined description of their efficient fronts, thereby making it possible for us to compute the theoretical hypervolume if infinite points are put on the true efficient front. For DTLZ1 problem, the efficient front is a M-dimensional linear hyperplane making equal angle with all objective axis and intersecting each axis at 0.5. Thus, the efficient front is a M-simplex in M-dimensional space and the volume under the front is given as follows [44]: $V_1(M) = (1/M!)(0.5^M)$. Therefore, the theoretically maximum hypervolume for a reference points at $\mathbf{z} = (1+\epsilon)(0.5, 0.5, \ldots, 0.5)^T$ is

$$\mathrm{HV}_T = (0.5(1+\epsilon))^M - \frac{1}{M!}0.5^M. \tag{1}$$

The normalized hypervolume for a set of non-dominated individuals P is then defined from the calculated hypervolume $\mathrm{HV}(P)$, as follows:

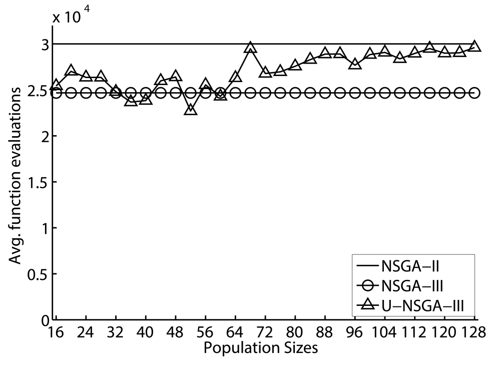
$$\mathrm{HV}_{\mathrm{norm}} = \frac{\mathrm{HV}(P)}{\mathrm{HV}_T}. \tag{2}$$

For DTLZ2 problem, the efficient front is a part of the M-dimensional hypersphere of radius one and the volume under the efficient front is given as follows [45]:

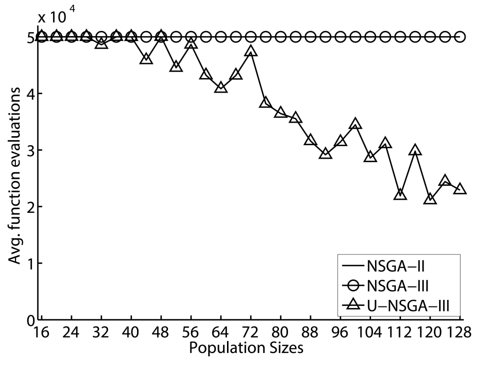
$$V_2(M) = \begin{cases} \frac{\pi^{M/2}}{2^M (M/2)!}, & \text{if } M \text{ is even,} \\ \frac{(\pi/2)^{(M-1)/2}}{M(M-2)(M-4)\cdots 1}, & \text{if } M \text{ is odd.} \end{cases}$$

For a reference point $\mathbf{z} = (1+\epsilon)(1, 1, \ldots, 1)^T$, the theoretical hypervolume is given as follows:

$$\mathrm{HV}_T = (1+\epsilon)^M - V_2(M), \tag{3}$$



(a) ZDT1



(b) ZDT4

Fig. 4: Performance of NSGA-II, NSGA-III, and U-NSGA-III with $N \geq H$ on unconstrained bi-objective ZDT test problems.

and the normalized hypervolume can be computed by using Equation 2. Table II presents HV$_T$ values for a few M values for both DTLZ1 and DTLZ2 for $\epsilon = 0.01$.

TABLE II: Theoretically maximum HV value (HV$_T$) for DTLZ1 and DTLZ2 problems for a few dimensions.

Prob-lem	Objective dimension, M				
	3	5	8	10	15
DTLZ1	0.107954	0.032584	0.004230	0.001079	0.000035
DTLZ2	0.506702	0.886517	1.067002	1.102132	1.160957

TABLE III: Ideal and reference points used for constrained bi-objective problems.

Problem	Ideal Point	Reference Point
OSY	$(-275, 5)$	$(-40.4, 77.77)$
TNK	$(0.029, 0.029)$	$(1.0605, 1.0605)$
BNH	$(0, 3.667)$	$(138.407, 50.5)$
SRN	$(5, -215)$	$(227.25, 0)$
Welded	$(0, 0)$	$(40.4, 0.00505)$
Pressure	$(42, -62761000)$	$(330270, -8080)$

For solving three-objective problems, we have used $N = 92$ for all three algorithms and $H = 91$ for U-NSGA-III and NSGA-III. Each figure represents the best of 11 distinct runs. Figure *S15* shows how the three algorithms perform on DTLZ1 and DTLZ2.

Results of the scaled versions of the two problems are shown in Figures 5a, 5b, 5c, 5d, 5e and 5f, respectively.

Finally, Figure *S16* presents the final population of the three algorithms for problems C3-DTLZ1 and C3-DTLZ4.

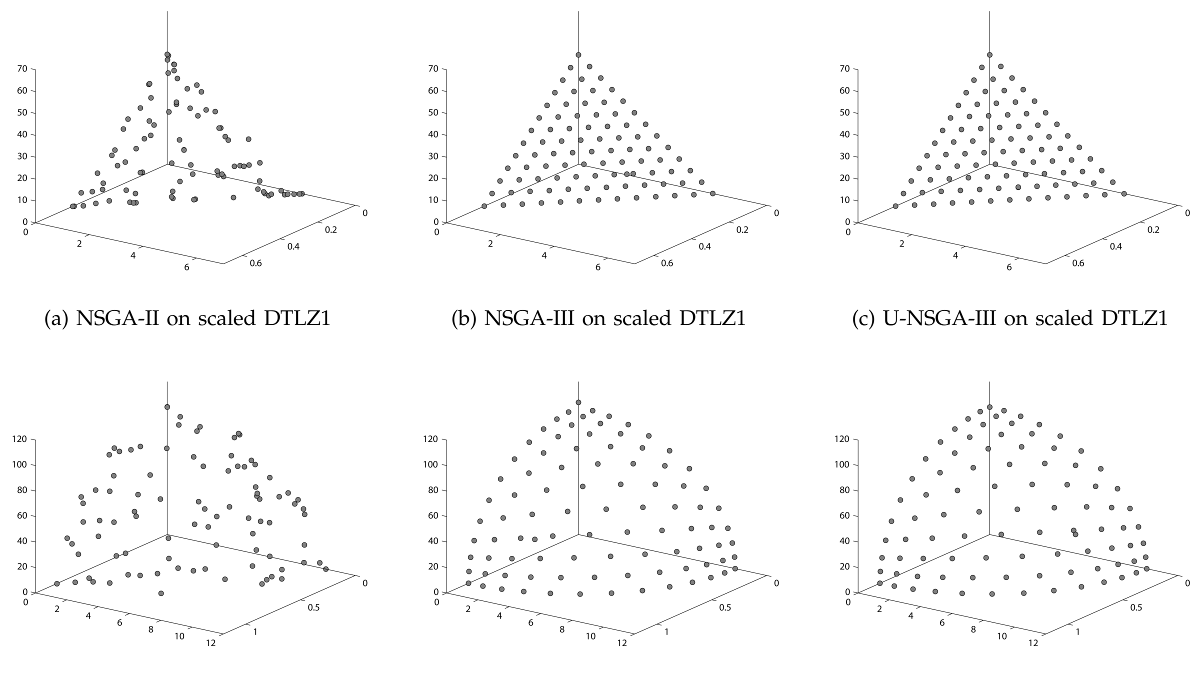
It can be seen from the figures that while NSGA-II fails to maintain adequate distribution of individuals, both NSGA-III and U-NSGA-III successfully achieve a uniformly distributed set of individuals covering the entire efficient front in each case. Only minor differences can be seen between the plots of NSGA-III and U-NSGA-III. Best, median and worst HV values in Tables IV and *S7* show the equivalence in performance between NSGA-III and U-NSGA-III, as anticipated. For scaled problems, we first unscale the objective values using the scaling factor used in the optimization process and then compute HV$_{\text{norm}}$ metric value.

D. Many-objective Problems

Finally, we consider five, eight, and 10-objective versions of the same 6 problems used in the previous subsection. We compare our proposed U-NSGA-III with NSGA-III (with N equal to H) in using hyper-volume values, which are computed using the sampling based strategy proposed elsewhere [43] due to the computational complexities involved in HV calculations in higher dimensions. The differences between U-NSGA-III and NSGA-III were analyzed to be negligible for many-objective optimization problems from an algorithmic point of view. Here, we investigate how both these methods perform empirically on a series of test problems. In these problems, we use the same N and H values used in the original NSGA-III study.

Tables IV and *S7* clearly show that NSGA-II fails in terms of both convergence and maintaining diversity, at the level of five or more objectives, to the extent that none of the individuals of the final population passed the reference point used to calculate HV. On the other hand, NSGA-III and U-NSGA-III produce very similar HV values in all problems. Almost-identical Parallel Coordinate Plots (PCP) can be observed for the 10-objective versions of DTLZ problems in Figures *S13*, 6 and *S14*. A PCP is a set of vertical bars, each represents one objective. Each zigzag horizontal line represents one individual. The point at which a zigzag line cuts a vertical bar represents the corresponding objective value of the corresponding individual. A PCP plot gives a rough picture of the diversity of set of solutions. The more intersections and the more coverage of the whole plot, the better.

Finally, Table *S10* shows that from a statistical point of view, U-NSGA-III and NSGA-III are equivalent on almost all multi/many-objective problems up to ten objectives. which means that the new niching based



(a) NSGA-II on scaled DTLZ1 (b) NSGA-III on scaled DTLZ1 (c) U-NSGA-III on scaled DTLZ1

(d) NSGA-II on scaled DTLZ2 (e) NSGA-III on scaled DTLZ2 (f) U-NSGA-III on scaled DTLZ2

Fig. 5: Performance of NSGA-II, NSGA-III, and U-NSGA-III on scaled unconstrained three-objective DTLZ problems.

TABLE IV: Best, medium and worst HV_{norm} on multi/many-objective unconstrained DTLZ problems.

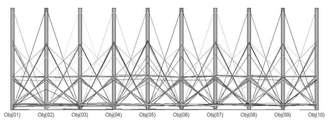
Problem	Obj.	G	P	NSGA-II			NSGA-III			U-NSGA-III		
				Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
DTLZ1	3	400	92	.9198	.9136	.8597	**.9487**	**.9465**	**.9388**	.9462	.9464	.934
	5	600	212	—	—	—	**.9767**	**.9762**	**.9757**	.9766	.9760	.9751
	8	750	156	—	—	—	**.9953**	**.9953**	**.9953**	**.9953**	**.9953**	**.9953**
	10	1000	276	—	—	—	**.9972**	**.9972**	**.9972**	**.9972**	**.9972**	**.9972**
DTLZ2	3	250	92	.8976	.8830	.8711	.9828	.9819	**.9812**	**.9838**	**.9824**	.9808
	5	350	212	.3763	.3143	.2072	**.8407**	.8396	.8371	.8404	**.8398**	**.8382**
	8	500	156	—	—	—	**.8532**	.8492	.8452	.8525	**.8497**	**.847**
	10	750	276	—	—	—	**.8769**	**.8760**	**.8743**	**.8769**	.8751	**.8743**
S-DTLZ1	3	400	92	.9204	.9111	.8993	**.9488**	**.9482**	**.9456**	.9485	.9472	.9445
	5	600	212	—	—	—	.9767	.9762	**.9675**	**.9768**	**.9764**	.9565
	8	750	156	—	—	—	**.9946**	**.9941**	**.9931**	.9943	**.9941**	.9920
	10	1000	276	—	—	—	**.9991**	**.9991**	**.9981**	**.9991**	**.9991**	**.9981**
S-DTLZ2	3	250	92	.8034	.7914	.7739	**.8756**	**.8741**	**.8715**	.8749	.8739	.8705
	5	350	212	.3761	.2895	.2376	**.8393**	.8349	**.8314**	.8384	**.8353**	.8285
	8	500	156	—	—	—	.8510	.8485	**.8463**	**.8512**	**.8490**	.8459
	10	750	276	—	—	—	.9218	.9173	**.9066**	**.9228**	**.9200**	.8935

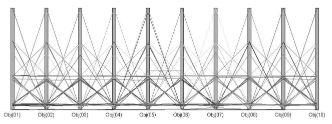
operator did not affect NSGA-III ability to tackle higher dimensional problems.

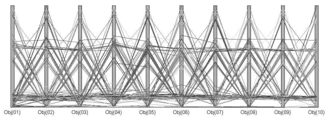
All these results demonstrate that the introduction of the niched tournament selection in the original NSGA-III algorithm and the flexibility of using a different population size from the number of reference directions do not change its performance in U-NSGA-III on many-objective optimization problems.

V. Conclusions

In this study, we have developed a unified evolutionary optimization algorithm U-NSGA-III which is a modification to a recently proposed evolutionary many-objective optimization method. U-NSGA-III has been carefully designed so as to solve mono-, multi-, and many-objective optimization problems. Simulation results on a number of single, two, three, five, eight and


(a) S-DTLZ1 using NSGA-III


(b) S-DTLZ1 using U-NSGA-III


(c) S-DTLZ2 using NSGA-III

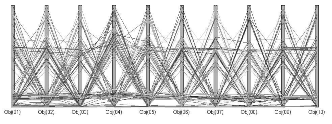

(d) S-DTLZ2 using U-NSGA-III

Fig. 6: Performance of NSGA-III and U-NSGA-III on scaled unconstrained many-objective DTLZ problems.

ten-objective constrained and unconstrained problems, have demonstrated the efficacy of the proposed unified approach. In each category having multiple problem instances, it has been found that the proposed U-NSGA-III performs in a similar manner and sometimes better than a respective specific EA – an elite-preserving RGA for mono-objective problems, NSGA-II for bi-objective problems, and NSGA-III for three and many-objective problems. The ability of one optimization algorithm to solve different types of problems equally efficiently and sometimes better, with the added flexibility brought in through a population size control remains a hallmark achievement of this study. In addition, several useful insights have been elaborated on, both algorithmically and empirically about the relative effectiveness of all the algorithms included in this study.

This type of unification has not been attempted before, except an omni-optimizer approach [28] which is not scalable for many-objective problems. In this regard, this study makes a key contribution in suggesting one single optimization algorithm that is able to degenerate into efficient mono-, multi- and many-objective optimization methods. dictated simply by the number of objectives in a given problem. Due to these reasons, the study is important from the efficient optimization software development point of view and its applicability to practical problems having separate one and many-objective versions. Such unification approaches also provide researchers the key insight about operator interactions needed to constitute scalable algorithms.

The unified optimization algorithm proposed here elevates the act of optimization as a computing-friendly approach. Computing algorithms are usually developed for handling a generic input having a large-dimensional attributes or parameters. However, the algorithm is also expected to work on a specific lower-dimensional or trivial input as a degenerate case of the generic case. For example, the Gauss-elimination computing algorithm was developed to solve a multi-variable linear system of equations $\mathbf{A}x = b$, but if the same algorithm is used to solve a single-variable linear equation, $ax = b$ (a degenerate case), the algorithm should find the solution $x = b/a$ without any hitch. In the same way, depending on the number of objectives, U-NSGA-III attempts to find multiple Pareto-optimal solutions if the objectives are greater than one, but when the number of objectives is only one, U-NSGA-III finds a single optimal solution as a degenerate case.

As an extension of this study, other efficient EMO methods e.g. MOEA/D can also be tried for the development of an equivalent unified approach. Also, U-NSGA-III can be modified to handle multi-modal problems for finding multiple optimal (and multiple Pareto-optimal [28]) solutions in a single simulation. The population approach and flexibility of EAs makes such approach possible and further such studies will demonstrate the usefulness of EAs in solving various optimization problems in a unified manner.

ACKNOWLEDGMENT

The authors gratefully acknowledge the Binational Fulbright Commision in Egypt and AMIDEAST Washington D.C. for funding the first author through his Ph.D. Fulbright grant.

REFERENCES

[1] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, vol. 16.

[2] C. A. C. Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2002, vol. 242.

[3] C.-K. Goh and K. C. Tan, "Evolutionary multi-objective optimization in uncertain environments," *Issues and Algorithms, Studies in Computational Intelligence*, vol. 186, 2009.

[4] K. C. Tan, E. F. Khor, and T. H. Lee, *Multiobjective evolutionary algorithms and applications*. Springer Science & Business Media, 2006.

[5] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[6] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," *Lecture notes in computer science*, vol. 1917, pp. 849–858, 2000.

[7] E. Zitzler, M. Laumanns, L. Thiele, E. Zitzler, E. Zitzler, L. Thiele, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," 2001.

[8] D. W. Corne, J. D. Knowles, and M. J. Oates, "The pareto envelope-based selection algorithm for multiobjective optimization," in *Parallel Problem Solving from Nature PPSN VI*. Springer, 2000, pp. 839–848.

[9] C. A. C. C. Coello and G. T. Pulido, "A micro-genetic algorithm for multiobjective optimization," in *Evolutionary multi-criterion optimization*. Springer, 2001, pp. 126–140.

[10] E. F. Khor, K. C. Tan, and T. H. Lee, "Tabu-based exploratory evolutionary algorithm for effective multi-objective optimization," in *Evolutionary Multi-Criterion Optimization*. Springer, 2001, pp. 344–358.

[11] M. Garza-Fabre, G. T. Pulido, and C. A. C. Coello, "Ranking methods for many-objective optimization," in *MICAI 2009: Advances in Artificial Intelligence*. Springer, 2009, pp. 633–645.

[12] S. F. Adra and P. J. Fleming, "Diversity management in evolutionary many-objective optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 2, pp. 183–195, 2011.

[13] V. Khare, X. Yao, and K. Deb, "Performance scaling of multi-objective evolutionary algorithms," in *Evolutionary Multi-Criterion Optimization*. Springer, 2003, pp. 376–390.

[14] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review." in *IEEE congress on evolutionary computation*. Citeseer, 2008, pp. 2419–2426.

[15] D. Hadka and P. Reed, "Borg: An auto-adaptive many-objective evolutionary computing framework," *Evolutionary Computation*, vol. 21, no. 2, pp. 231–259, 2013.

[16] H. K. Singh, A. Isaacs, and T. Ray, "A pareto corner search evolutionary algorithm and dimensionality reduction in many-objective optimization problems," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 4, pp. 539–556, 2011.

[17] H. Aguirre and K. Tanaka, "Many-objective optimization by space partitioning and adaptive ϵ-ranking on mnk-landscapes," in *Evolutionary Multi-Criterion Optimization*. Springer, 2009, pp. 407–422.

[18] H. Sato, H. E. Aguirre, and K. Tanaka, "Pareto partial dominance moea and hybrid archiving strategy included cdas in many-objective optimization," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 2010, pp. 1–8.

[19] X. Zou, Y. Chen, M. Liu, and L. Kang, "A new evolutionary algorithm for solving many-objective optimization problems," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 5, pp. 1402–1412, 2008.

[20] U. K. Wickramasinghe and X. Li, "A distance metric for evolutionary many-objective optimization algorithms using user-preferences," in *AI 2009: Advances in Artificial Intelligence*. Springer, 2009, pp. 443–453.

[21] R. C. Purshouse and P. J. Fleming, "On the evolutionary optimization of many conflicting objectives," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 6, pp. 770–784, 2007.

[22] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 6, pp. 712–731, 2007.

[23] K. Sindhya, K. Miettinen, and K. Deb, "A hybrid framework for evolutionary multi-objective optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 17, no. 4, pp. 495–511, 2013.

[24] A. Britto and A. Pozo, "I-mopso: A suitable pso algorithm for many-objective optimization," in *Neural Networks (SBRN), 2012 Brazilian Symposium on*. IEEE, 2012, pp. 166–171.

[25] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints," *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 4, pp. 577–601, 2014.

[26] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: handling constraints and extending to an adaptive approach," *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 4, pp. 602–622, 2014.

[27] D. Sharma, K. Deb, and N. Kishore, "Towards generating diverse topologies of path tracing compliant mechanisms using a local search based multi-objective genetic algorithm procedure," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, 2008, pp. 2004–2011.

[28] K. Deb and S. Tiwari, "Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1062–1087, 2008.

[29] J. Bader and E. Zitzler, "Hype: An algorithm for fast hypervolume-based many-objective optimization. tik report 286," *Computer Engineering and Networks Laboratory (TIK), ETH Zurich*, 2008.

[30] R. Wang, R. C. Purshouse, and P. J. Fleming, "Preference-inspired coevolutionary algorithms for many-objective optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 17, no. 4, pp. 474–494, 2013.

[31] K. Deb, A. R. Reddy, and G. Singh, "Optimal scheduling of casting sequence using genetic algorithms," *Materials and Manufacturing Processes*, vol. 18, no. 3, pp. 409–432, 2003.

[32] K. Deb, P. Jain, N. K. Gupta, and H. K. Maji, "Multiobjective placement of electronic components using evolutionary algorithms," *Components and Packaging Technologies, IEEE Transactions on*, vol. 27, no. 3, pp. 480–492, 2004.

[33] K. Deb, K. Mitra, R. Dewri, and S. Majumdar, "Towards a better understanding of the epoxy-polymerization process using multi-objective evolutionary computation," *Chemical engineering science*, vol. 59, no. 20, pp. 4261–4277, 2004.

[34] D. K. Saxena and K. Deb, "Trading on infeasibility by exploiting constraints criticality through multi-objectivization: A system design perspective," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE, 2007, pp. 919–926.

[35] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 631–657, 1998.

[36] K. Deb, "Geneas: A robust optimal design technique for mechanical component design," in *Evolutionary algorithms in engineering applications*. Springer, 1997, pp. 497–514.

[37] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–a comprehensive introduction," *Natural computing*, vol. 1, no. 1, pp. 3–52, 2002.

[38] J. J. Durillo and A. J. Nebro, "jmetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760–771, 2011.

[39] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 3, pp. 1–15, 1994.

[40] J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. C. Coello, and K. Deb, "Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization," *Journal of Applied Mechanics*, vol. 41, p. 8, 2006.

[41] J. Knowles, D. Corne, and K. Deb, *Multiobjective problem solving from nature: from concepts to applications*. Springer Science & Business Media, 2007.

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

[43] L. While, L. Bradstreet, and L. Barone, "A fast way of calculating exact hypervolumes," *Evolutionary Computation, IEEE Transactions on*, vol. 16, no. 1, pp. 86–95, 2012.

[44] P. Stein, "A note on the volume of a simplex," *American Mathematical Monthly*, pp. 299–301, 1966.

[45] X. Wang, "Volumes of generalized unit balls," *Mathematics Magazine*, pp. 390–395, 2005.

Haitham Seada is a Ph.D student at Department of Computer Science and Engineering, Michigan State University, USA. He received his Bachelor's degree in Computer Science from the Faculty of Computers and Informatics, Zagazig University, Egypt in 2005. Received his master's degree from the same university in bioinformatics in 2009. worked as a teaching assistant and an assistant lecturer for 8 years. His research interests include evolutionary algorithms and their applications in optimization, and applications of optimization in computer science.

Kalyanmoy Deb is Koenig Endowed Chair Professor at Department of Electrical and Computer Engineering in Michigan State University, USA. Prof. Deb received his Bachelor's degree in Mechanical Engineering from Indian Institute of Technology Kharagpur in 1985. After a short job experience, he received his master's and doctral degrees from University of Alabama, USA in 1989 and 1991, respectively. Prof. Deb's research interests are in evolutionary algorithms and their application in optimization, modeling, and machine learning. He was awarded Infosys Prize, TWAS Prize in Engineering Sciences, CajAstur Mamdani Prize, Distinguished Alumni Award from IIT Kharagpur, Edgeworth-Pareto award, Bhatnagar Prize in Engineering Sciences, and Bessel Research award from Germany. He is fellow of IEEE, ASME, and three Indian science and engineering academies. He has published over 405 research papers with Google Scholar citation of 71,000 with h-index 88. He is in the editorial board on 20 major international journals. More information about his research contribution can be found from http://www.egr.msu.edu/ kdeb.