U-NSGA-III: A Unified Evolutionary Algorithm for Single, Multiple, and Many-Objective Optimization

Haitham Seada and Kalyanmoy Deb
Computational Optimization and Innovation Laboratory (COIN)
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824, USA
Email: {seadahai,deb}@msu.edu

COIN Report Number 2014022

Abstract

Evolutionary algorithms (EAs) have been systematically developed to solve mono-objective, multi-objective and many-objective optimization problems, in this order, over the past few decades. Despite some efforts in unifying different types of mono-objective evolutionary and non-evolutionary algorithms, there does not exist many studies to unify all three types of optimization problems together. Such a unified optimization algorithm will allow a user to work with a single code or software enabling one-time implementation of a suitable solution representation, objective and constraint function coding and operator updates, if any, to achieve optimizations with different objective dimensions. In this study, for the first time, we propose a unified evolutionary optimization algorithm U-NSGA-III, based on the recently-proposed NSGA-III procedure, developed for solving many-objectives problems, for solving all three classes of problems specified above. Our proposed implementation is such that the U-NSGA-III algorithm degenerates to an equivalent and efficient population-based optimization procedure for each class, just from the description of the number of specified objectives of a problem. The algorithm works with usual EA parameters and no additional tunable parameters are needed. To demonstrate the working of U-NSGA-III, extensive simulations are performed on unconstrained and constrained test problems having single, two, multi and many-objectives, taken from the literature and on two engineering optimization design problems. The performance of U-NSGA-III is compared with a real-coded genetic algorithm for mono-objective problems, with well-known NSGA-II for two-objective problems, and with recently proposed NSGA-III for three to 15-objective problems. Results amply demonstrate the merit of our proposed unified approach, encourage its further application, and motivate similar studies for a richer understanding of the development of optimization algorithms.

1 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]. Initially researchers focused on solving optimization problems with no more than three objectives. 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, 24]. 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 front [23, 25, 26, 27]. Thus, in principle, current evolutionary multi-objective and many-objective optimization algorithms are different from each other.

Although certain evolutionary multi-objective optimization methodologies such as NSGA-II [6] does not scale up to solve many-objective optimization problems efficiently, they are found to work well in solving mono-objective optimization problems [28]. Based on NSGA-II framework, an omni-optimizer algorithm [29] 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 [23, 26, 27] 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 objective space becomes defunct. Thus, although these decomposed algorithms are efficient in handling many objectives, their working in mono- and bi-objective problems becomes questionable.

Besides the algorithmic deficiencies with existing mono and multi-objective methods to scale up to solve many-objective problems and many-objective methods to scale down suitably to solve mono- and multi-objective problems, there is another practical motivation for our study, which we discuss next. To solve an optimization problem either using a computer code or a commercial software, it must be first implemented (coded or expressed symbolically) within the code or software. Often, this implementation process involves linking the optimization code or software with a third-party evaluation software such as a finite element code or a computational fluid dynamics code or a network flow simulator, etc. To achieve a faster execution of the optimization algorithm, it is also recommended to customize the optimization procedure for the problem at hand [30, 31] by introducing new operators and/or modifying existing genetic operators with 'heuristics' of the problem. Often, 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. However, many multi- or many-objective optimization methods require the same problem to solved for individual objectives one at a time for obtaining ideal and nadir points prior to solving the multi- or many-objective version of the problem. Often, such several lower-dimensional runs are executed to verify or gain confidence in the obtained higher-dimensional efficient front [32]. In design exploration problems, objectives and constraints are interchanged to get a better idea of possible range of optimal solutions [33]. 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 reimplemented 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 evaluation softwares would be enough to solve different versions of the original problem, thereby providing flexibility to users in moving back and forth between different objective-dimensions of the problem and also in 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. But 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 lower-objective optimization algorithms.

Intuitively, such a unified approach would be generic to solving many-objective optimization problems. Thus, we base our algorithm using one of the recently proposed many-objective optimization algorithms – NSGA-III [26]. In the remainder of this paper, we provide a brief description of NSGA-III in Section 2 due to its algorithmic similarity with the proposed U-NSGA-III. Thereafter, we present our proposed unified approach U-NSGA-III in Section 3 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 4. Finally, conclusions are drawn in Section 5.

2 A Brief Introduction to NSGA-III

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. The hyper-plane is placed in a manner so that it intersects each objective axis at one. Das and Dennis's technique [34] is used to place $H = \binom{M+p-1}{p}$ reference points on the hyper-plane having $(p+1)$ points along each boundary. The population size N is chosen to be the smallest multiple of four greater than H, with the idea that for every reference point, one population member is expected to be found.

At a generation t, the following operations are performed. First, the whole population P_t is classified into different non-domination levels, in the same way it is done in NSGA-II, following the principle of non-dominated sorting. An offspring population Q_t is created from P_t using usual recombination and mutation operators. Since only one population member is expected to be found for each reference point, there is no need for any selection operation in NSGA-III, as any selection operator will allow a competition to be set among different reference points. A combined population $R_t = P_t \cup Q_t$ is then formed. Thereafter, points starting from the first non-dominated front is selected for P_{t+1} one at a time until all solutions from a complete front cannot be included. This procedure is also identical to that in NSGA-II. Let us denote the final front that could not be completely selected as F_L. In general, only a few solutions from F_L needs to be selected for 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 points have commensurate values. Thereafter, each member of P_{t+1} and F_L is *associated* with a specific reference point by using the shortest perpendicular distance ($d()$) of each population member with a reference line created by joining the origin with a supplied reference point. Then, a careful *niching* strategy is employed to choose those F_L members that are associated with the least represented reference points in P_{t+1}. The niching strategy puts an emphasis on selecting a population member for as many supplied reference points as possible. A population member associated with an under-represented or un-represented reference point 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 point close to the Pareto-optimal front, provided the genetic variation operators (recombination and mutation) are capable of producing respective solutions. The use of a well-spread reference points ensures a well-distributed set of trade-off points at the end.

The original NSGA-III study [26] 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 point set is adaptively updated on the fly based on the association status of each reference point over a number of generations. The algorithm is outlined in Algorithm 1.

Algorithm 1 Generation t of NSGA-III procedure

```
Input: $H$ structured reference points $Z^s$ or supplied aspiration points $Z^a$, 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:     Points 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 point: $[\pi(\mathbf{s}), d(\mathbf{s})] =$Associate($S_t, Z^r$)
        % $\pi(\mathbf{s})$: closest reference point, $d$: distance between s and $\pi(\mathbf{s})$
16:     Compute niche count of reference point $j \in Z^r$: $\rho_j = \sum_{\mathbf{s} \in S_t/F_l} ((\pi(\mathbf{s}) = j)\ ?\ 1\ :\ 0)$
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
```

2.1 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 down-scaling 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.

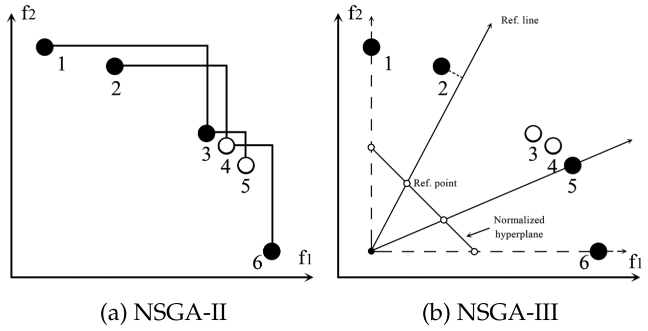


Figure 1: Working principles of NSGA-II and NSGA-III.

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.

3 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 points (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. 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 [34]), 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
```

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

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: $p_{best} = best(P)$
```

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

```
Input: $H$ structured reference points $Z^s$ or supplied aspiration points $Z^a$, 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)
```

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.

3.1 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 [34] strategy results in $\binom{1+p-1}{p}$ or one single reference point and is independent of the value of p. The resulting U-NSGA-III is a generational evolutionary algorithm (EA) that uses (i) a binary tournament selection, (ii) recombination and mutation operators, and (iii) an elite-preserving operator. Thus, our proposed U-NSGA-III is similar to other generational EAs, such as elite-preserving real-coded genetic algorithm [35] or the $(\mu/\rho + \lambda)$ evolution strategy [36], where $\mu = \lambda = N$ and $\rho = 2$.

3.2 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 points), additional $(N - H)$ points 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 points in NSGA-III should find a better distributed efficient points than adaptive discovery of points by NSGA-II.

3.3 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 desired optimal points 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 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.

4 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.

4.1 Mono-objective Problems

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

4.1.1 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 [37]. We employ an elite-preserving operator between parent and offspring populations in the proposed 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 below:

$$\begin{aligned} f_{\text{elp}}(\mathbf{x}) = &\sum_{i=1}^{n} i x_i^2, \\ &-10 \le x_i \le 10, i = 1, \ldots, n \end{aligned} \tag{1}$$

$$\begin{aligned} f_{\text{ros}}(\mathbf{x}) = &\sum_{i=1}^{n-1} [100({x_i}^2 - x_{i+1})^2 + (x_i - 1)^2], \\ &-10 \le x_i \le 10, i = 1, \ldots, n \end{aligned} \tag{2}$$

$$\begin{aligned} f_{\text{zak}}(\mathbf{x}) = &\sum_{i=1}^{n} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{n} i x_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{n} i x_i\right)^4, \\ &-1 \le x_i \le 1, i = 1, \ldots, n \end{aligned} \tag{3}$$

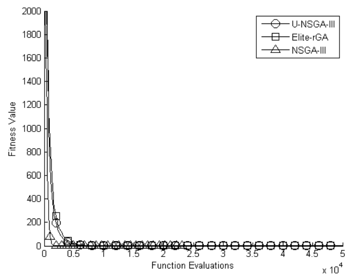
$$
\begin{aligned}
f_{\text{sch}}(\mathbf{x}) =& 418.9829n - \sum_{i=1}^{n}(x_i \sin\sqrt{|x_i|}), \\
& -500 \le x_i \le 500, i = 1, \ldots, n
\end{aligned} \tag{4}
$$

$$
\begin{aligned}
f_{\text{ack}}(\mathbf{x}) =& -20\exp\left[\frac{-1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} {x_i}^2}\right] \\
& -\exp\left[\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right] + 20 + e, \\
& -32.768 \le x_i \le 32.768, i = 1, \ldots, n
\end{aligned} \tag{5}
$$

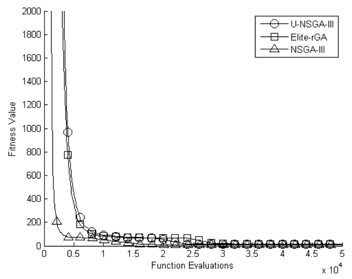
$$
\begin{aligned}
f_{\text{ras}}(\mathbf{x}) =& 10n + \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i)], \\
& -5.12 \le x_i \le 5.12, i = 1, \ldots, n
\end{aligned} \tag{6}
$$

For each problem, $n = 20$ is used and 11 simulations with the same set of parameters but from different initial populations are conducted. Although the first two problems are simple, f_{sch} is a difficult multi-modal problem. We use $N = 48$, 100, 100 and 300 for Ellipsoidal, Rosenbrock's, Zakharov's and Schwefel's problems, respectively, in order to handle the increasing complexity in these problems.

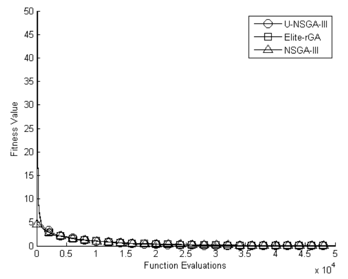
Figures 2a, 2b, 2c and 3 show the median function value (y-axis) for Ellipsoidal, Rosenbrock's, Zakharov's and Schwefel's problems over 11 runs for different function evaluations (x-axis). The three algorithms perform in a similar manner for unimodal and easy problems, while for the multi-modal problem U-NSGA-III performs the best.



(a) Ellipsoidal function



(b) Rosenbrock's function



(c) Zakharov's function

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

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. It was clear that NSGA-III is prone to getting 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 4 and 5 for Ackley's and Rastrigin's, respectively.

Table 1 summarizes the best, median and worst fitness values achieved by each of the three algorithms on unconstrained test problems.

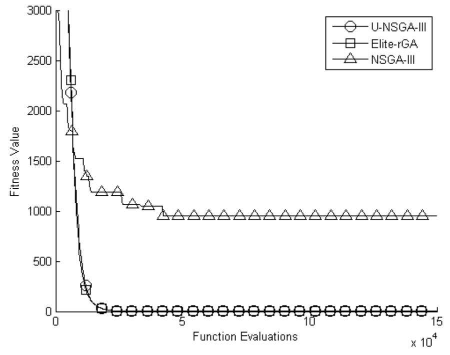
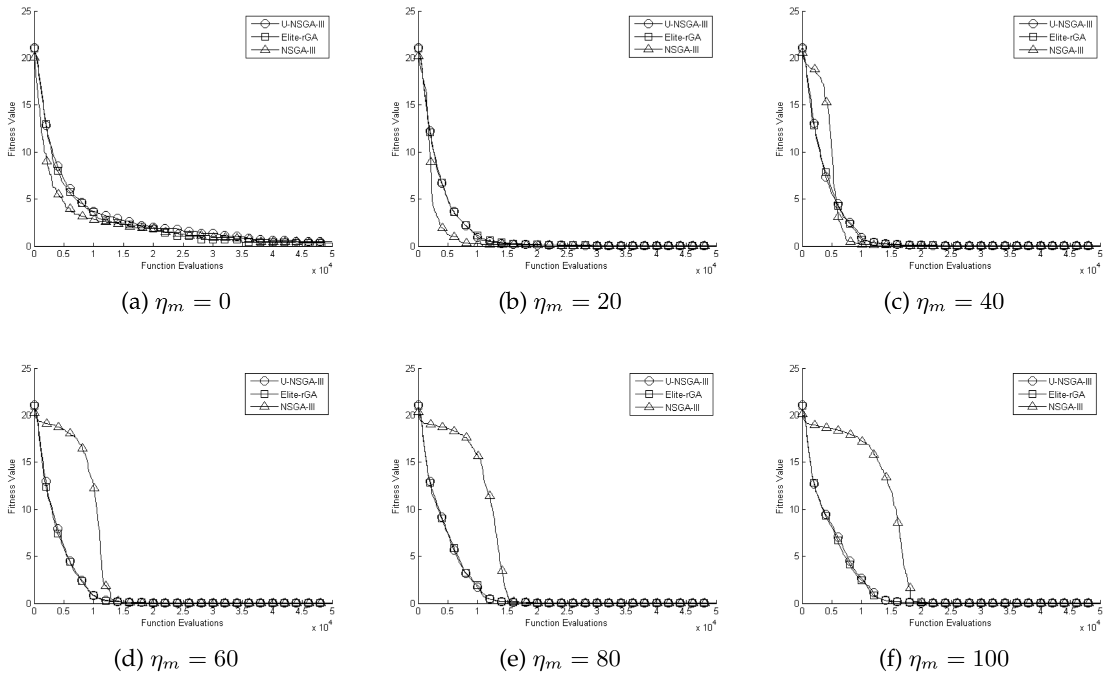


Figure 3: Generation-wise fitness for Schwefel function.



(a) $\eta_m = 0$

(b) $\eta_m = 20$

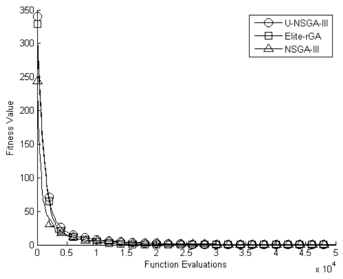
(c) $\eta_m = 40$

(d) $\eta_m = 60$

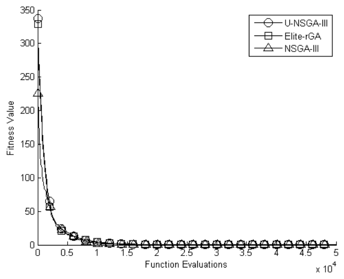
(e) $\eta_m = 80$

(f) $\eta_m = 100$

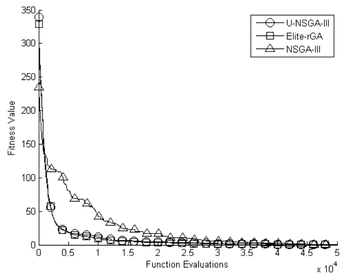
Figure 4: Effect of mutation distribution index η_m on Ackley's function.



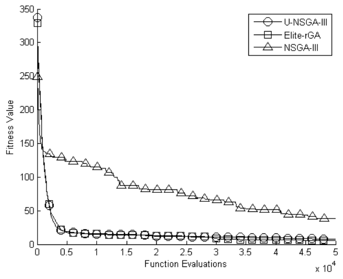
(a) $\eta_m = 0$



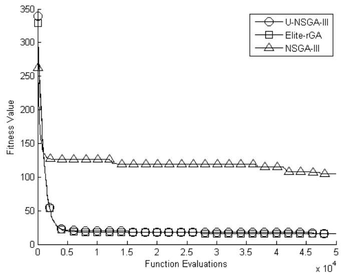
(b) $\eta_m = 20$



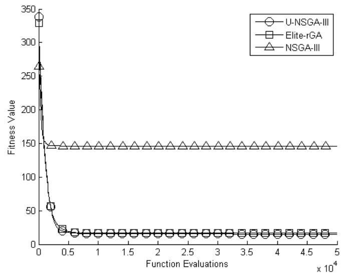
(c) $\eta_m = 40$



(d) $\eta_m = 60$



(e) $\eta_m = 80$



(f) $\eta_m = 100$

Figure 5: Effect of mutation distribution index η_m on Rastrigin's function.

Table 1: Best, medium and worst function values for mono-objective unconstrained test problems.

Prob.	Func. Eval.	P	EliteRGA			NSGA-III			U-NSGA-III		
			Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
ELP	24000	48	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.01	**0.00**	**0.00**	**0.00**
ROS	50000	100	**0.04**	16.70	72.75	0.57	**8.97**	**70.90**	0.72	16.09	76.20
SCHL	150000	300	**0.00**	**0.00**	**0.00**	355.32	947.51	1421.26	**0.00**	**0.00**	355.32
ZAK	50000	100	0.02	0.04	**0.07**	0.03	0.08	0.21	**0.01**	**0.02**	**0.07**

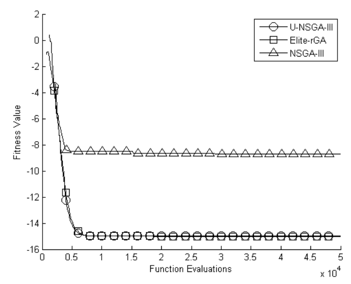
4.1.2 Constrained Problems

The superiority of U-NSGA-III is 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 [38]. Best, median and worst achieved fitness values by each algorithm in each problem are presented in Table 2. Figure 6 shows the median values of the three algorithms in each problem. In G07, all the three algorithms behave similarly, hence figures for G07 are removed for brevity. Obviously, NSGA-III in most cases is not able to converge to the global optimum. Again, U-NSGA-III and EliteRGA performs almost identically.

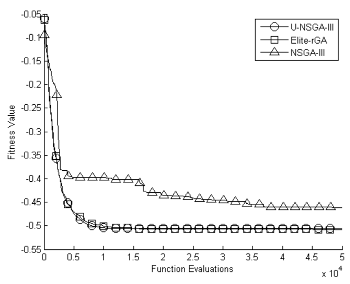
The performances of U-NSGA-III and EliteRGAare equivalent. As expected, NSGA-III with $N = 4$ performs well on unimodal problems, but fails miserably on multimodal problems.

4.2 Bi-objective Problems

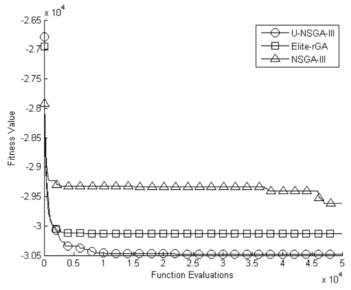
As mentioned before, the performance of NSGA-III was not tested on bi-objective optimization problems in the original study [26]. In this section, NSGA-II NSGA-III and U-NSGA-III are compared over an extensive set of unconstrained and constrained bi-objective problems. Two



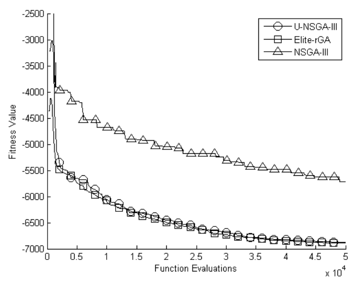
(a) G01



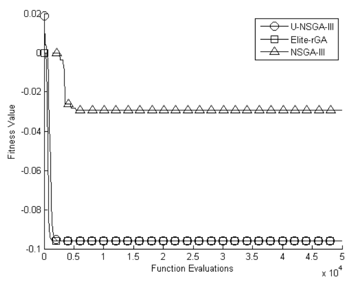
(b) G02



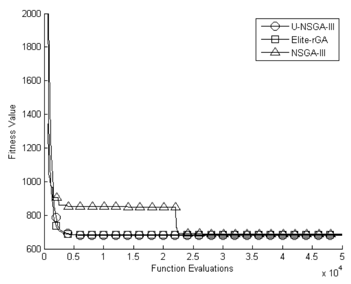
(c) G04



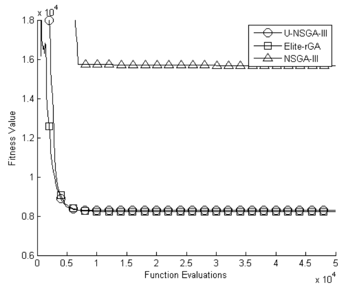
(d) G06



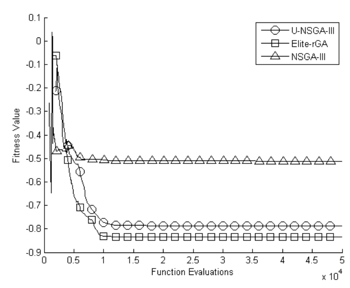
(e) G08



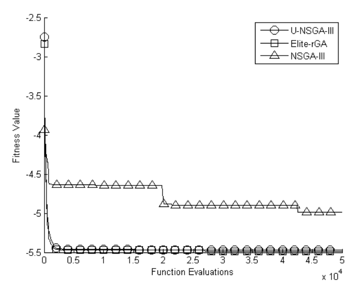
(f) G09



(g) G10



(h) G18



(i) G24

Figure 6: Performance of U-NSGA-III, elite EliteRGA, and NSGA-II on G-family constrained test problems.

Table 2: Best, medium and worst function values for mono-objective constrained test problems. 50,000 function evaluations and $P = 100$ are used.

Prob.	EliteRGA			NSGA-III			U-NSGA-III		
	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
G01	**-15.00**	**-15.00**	-12.43	-8.73	-8.73	-8.73	**-15.00**	**-15.00**	**-15.00**
G02	**-0.52**	**-0.51**	-0.48	-0.49	-0.46	-0.45	-0.50	-0.50	**-0.50**
G04	-30491.54	-30125.95	-29854.81	-30227.23	-29614.43	-28972.29	**-30659.34**	**-30483.61**	**-30309.43**
G06	**-6935.71**	**-6894.55**	**-6830.50**	-6325.48	-5716.84	-5353.46	-6918.78	-6880.34	-6816.80
G07	**24.77**	27.64	50.47	59.54	59.54	59.54	26.17	**26.17**	**26.17**
G08	**-0.10**	**-0.10**	**-0.10**	-0.03	-0.03	-0.03	**-0.10**	**-0.10**	**-0.10**
G09	**681.43**	683.74	687.50	689.02	689.02	689.02	681.57	**681.57**	**681.57**
G10	**7090.12**	**8243.44**	10947.69	15681.55	15681.55	15681.55	8285.18	8285.18	**8285.18**
G18	**-0.86**	**-0.83**	-0.49	-0.85	-0.51	-0.47	**-0.86**	-0.79	**-0.50**
G24	**-5.51**	**-5.48**	-5.30	-5.35	-4.99	-3.80	**-5.51**	-5.47	**-5.38**

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 – 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 11 different simulation runs are performed (except for ZDT6, where we use 10 variables, as used in the original study [39]).

4.2.1 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).

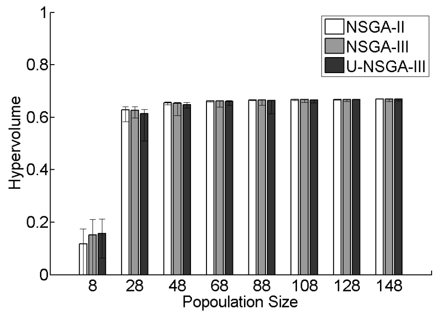
Figure 7 shows the best, median and worst hypervolume (HV) achieved by each of the three algorithms on ZDT1, ZDT2, ZDT4 and ZDT4 (a figure for ZDT6 is removed for the sake of brevity) for a reference point 1% larger in every component than the corresponding nadir point. Although the results are comparable 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 3 confirms that U-NSGA-III performs similar to NSGA-II on most ZDT problems.

The same observation can be noticed for constrained test problems in Figure 8. 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.

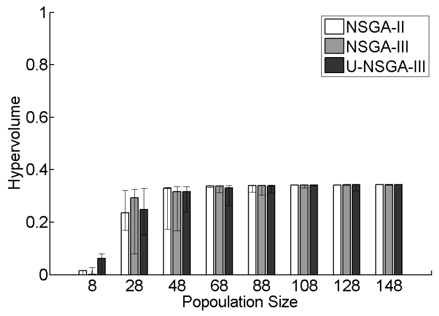
The same observation can be made in Figures 9 and 10 showing the results of welded beam and pressure vessel problems, respectively. The ideal and reference points used to compute HV

Table 3: Best, medium and worst HV for bi-objective unconstrained ZDT test problems.

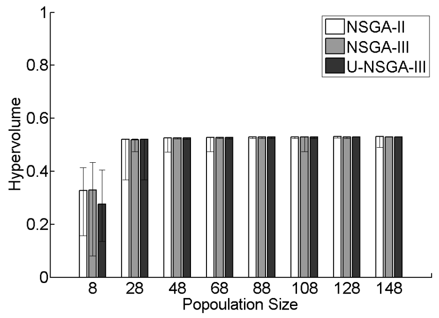
Prob.	G	P	NSGA-II			NSGA-III			U-NSGA-III		
			Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
ZDT1	100	8	0.17528	0.11695	—	0.21093	0.15223	—	**0.21168**	**0.15694**	**0.06435**
	100	48	**0.65818**	**0.65448**	**0.64795**	0.65731	0.65277	0.60657	0.65761	0.64815	0.63547
	100	88	0.66584	**0.66512**	**0.66359**	0.66588	0.66498	0.64587	**0.66589**	0.66462	0.61320
	100	128	0.66814	0.66799	0.66666	0.66850	0.66821	0.65966	**0.66867**	**0.66841**	**0.66728**
ZDT2	200	8	0.01487	0.01487	—	0.02715	0.00108	—	**0.07888**	**0.06267**	—
	200	48	0.33327	**0.32913**	0.17263	**0.33635**	0.31585	0.16824	0.33617	0.31567	**0.23919**
	200	88	0.34007	0.33936	**0.31527**	**0.34103**	**0.34030**	0.30458	0.34097	0.33788	0.31358
	200	128	0.34205	0.34186	**0.34163**	**0.34275**	0.34249	0.33959	0.34275	**0.34274**	0.32056
ZDT3	200	8	0.41452	0.32774	**0.15776**	**0.43267**	**0.32888**	0.08132	0.40587	0.27519	0.13611
	200	48	**0.52708**	**0.52606**	0.47280	0.52677	0.52602	**0.52185**	0.52647	0.52590	0.52123
	200	88	**0.52967**	**0.52942**	**0.52452**	0.52943	0.52919	0.52371	0.52946	0.52926	0.52361
	200	128	**0.53033**	**0.53025**	0.52506	0.53005	0.52997	0.52452	0.53019	0.52989	**0.52968**
ZDT4	500	48	0.21030	0.21002	—	0.33286	0.00433	—	**0.66260**	**0.47591**	—
	500	88	0.50025	0.34585	—	**0.66754**	**0.66752**	**0.34872**	0.66754	0.50273	—
	500	128	0.66841	0.50303	0.11423	**0.66935**	**0.66935**	0.21818	0.66935	0.66935	**0.50441**
	500	168	0.66948	0.66917	0.21796	**0.67029**	**0.67029**	**0.67029**	0.67029	0.67029	0.50523
ZDT6	350	8	0.27614	0.25236	**0.22579**	**0.28004**	**0.25278**	0.20354	0.26763	0.24376	0.21467
	350	48	**0.40227**	**0.39683**	**0.39117**	0.39864	0.39409	0.38856	0.39783	0.39019	0.38352
	350	88	**0.40959**	**0.40784**	**0.40547**	0.40690	0.40328	0.39861	0.40699	0.40428	0.39896
	350	128	**0.41167**	**0.41061**	**0.40959**	0.41163	0.40918	0.40777	0.41042	0.40912	0.40655



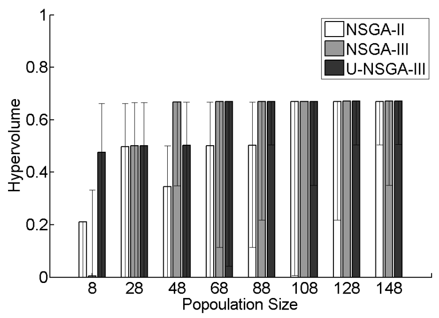
(a) ZDT1



(b) ZDT2



(c) ZDT3



(d) ZDT4

Figure 7: Performance of NSGA-II, NSGA-III, and U-NSGA-III with $N = H$ on unconstrained bi-objective test problems.

Table 4: Threshold HV values used in bi-objective ZDT problems for $N \geq H$ simulations.

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

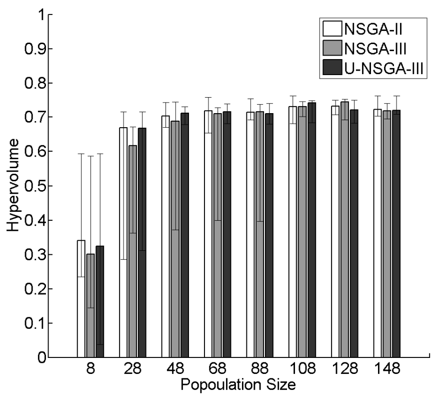
values for constrained problems are shown in Table 7.

4.2.2 N Greater Than H

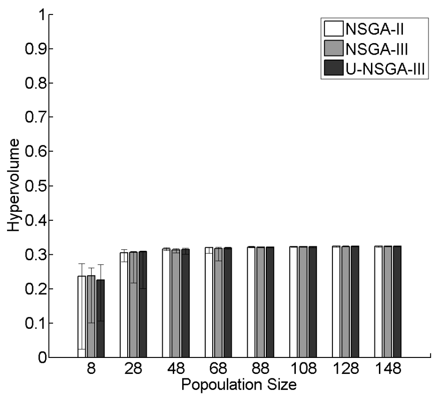
In the second set of experiments for handling bi-objective problems, we evaluate the usefulness of using $N \geq H$. In this experiment, two straight lines, one representing the performance of NSGA-II with $N = 16$ and another representing the performance of NSGA-III with $N = H = 16$ are shown for convenience. 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 4 presents the HV values used in our bi-objective simulations.

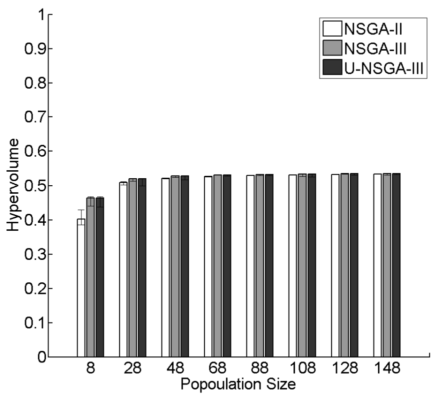
For U-NSGA-III, we have used different population sizes $N \geq H$. Average number of func-



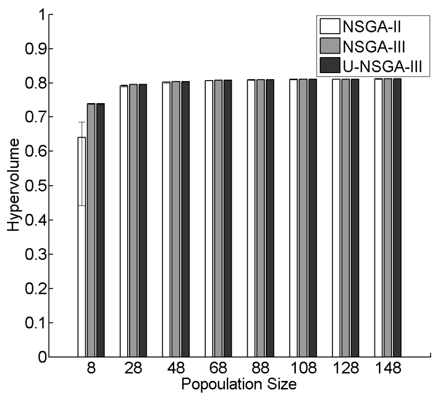
(a) OSY



(b) TNK



(c) SRN



(d) BNH

Figure 8: Constrained bi-objective problems ($N = H$).

tion evaluations over 11 runs needed to achieve the threshold HV value (Table 4) are plotted in Figure 11 for unconstrained ZDT test problems. For runs having a population size larger than number of reference points, the points that are closest to each H reference line in the objective space are used for the HV calculation, that is, for all U-NSGA-III simulations, only 16 individuals one closest to each specified reference lines 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 and ZDT3, the use of a larger population size is not found to be beneficial, whereas for ZDT2 and ZDT4 (more difficult problems), a larger population brings in the necessary diversity needed to solve the respective problem adequately. NSGA-III and NSGA-II are also applied with $N = H$ for all problems and the average number of solution evaluations in 11 runs needed to achieve an identical HV value is marked in respective figures. 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 all 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 points. These results are interesting and demonstrate the usefulness of a larger population size than the number of reference points for the proposed U-NSGA-III algorithm.

Figure 12 shows the median hypervolume metric value comparison and Table 5 presents best, median, and worst hypervolume metric values of the three algorithms on constrained test

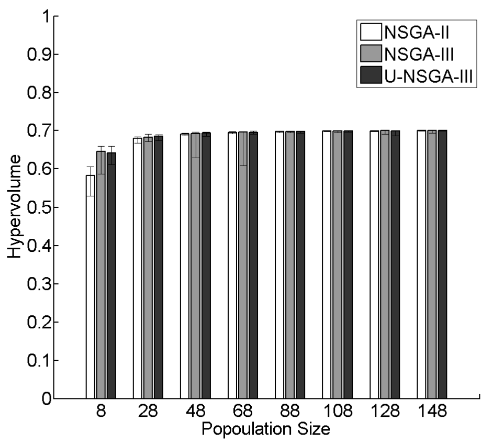


Figure 9: Welded beam design problem ($N = H$).

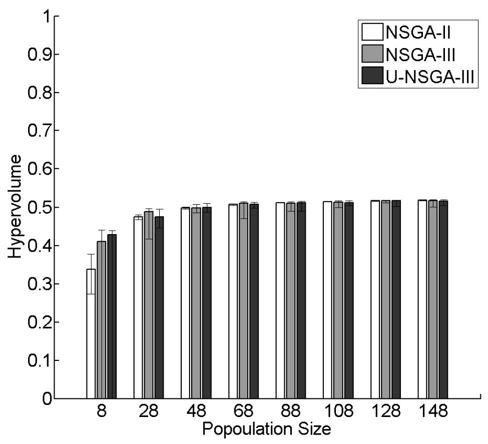


Figure 10: Pressure vessel design problem ($N = H$).

problems. Except in very small population sizes, the difference in the performance of all three algorithms is not that significant.

4.3 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 [27].

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 sampling based technique proposed elsewhere [40]. 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 [41]: $V_1(M) = (1/M!)(0.5^M)$. Therefore, the theoretically maximum hypervolume for a reference points at $\mathbf{z} = (1+\epsilon)(0.5, 0.5, \ldots, 0.5)^T$ is

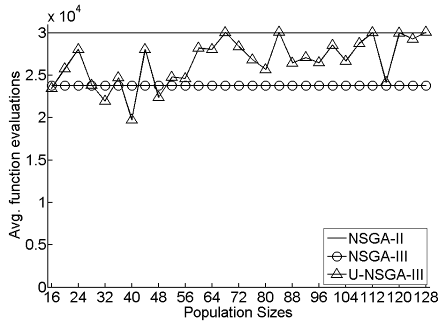
$$\mathrm{HV}_T = (0.5(1+\epsilon))^M - \frac{1}{M!}0.5^M. \tag{7}$$

The normalized hypervolume for a set of non-dominated points P is then defined from the calculated hypervolume HV(P), as follows:

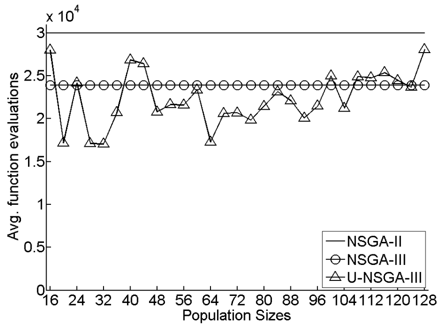
$$\mathrm{HV}_{\mathrm{norm}} = \frac{\mathrm{HV}(P)}{\mathrm{HV}_T}. \tag{8}$$

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 [42]:

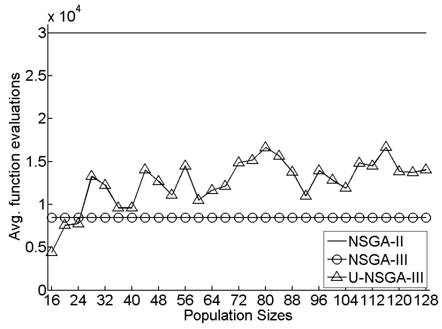
$$V_2(M) = \begin{cases} \frac{\pi^{M/2}}{2^M (M/2)!}, & \text{if } M \text{ is even,} \\ \frac{(\pi/2)^{(M-1)/2}}{M(M-2)(M-4)\cdots 1}, & \text{if } M \text{ is odd.} \end{cases}$$



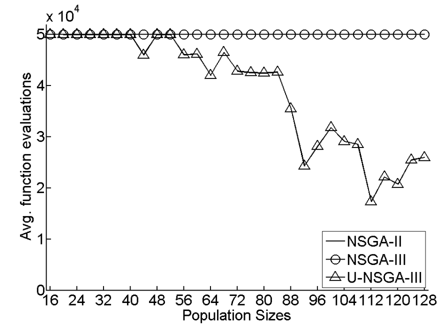
(a) ZDT1



(b) ZDT2



(c) ZDT3



(d) ZDT4

Figure 11: Performance of NSGA-II, NSGA-III, and U-NSGA-III with $N \geq H$ on unconstrained bi-objective ZDT test problems.

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

$$\mathrm{HV}_T = (1+\epsilon)^M - V_2(M), \tag{9}$$

and the normalized hypervolume can be computed by using Equation 8. Table 6 presents HV_T values for a few M values for both DTLZ1 and DTLZ2 for $\epsilon = 0.01$.

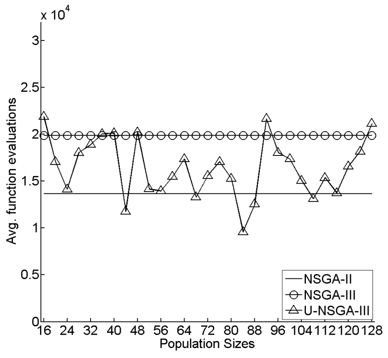
For solving three-objective problems, we have used $N = 92$ for all three algorithms and $H = 91$ for U-NSGA-III and NSGA-III. Each figure represents the best of 11 distinct runs. Figures 13a, 13b and 13c show the final population of the three algorithms for DTLZ1. Figures 13d, 13e and 13f show the same for DTLZ2.

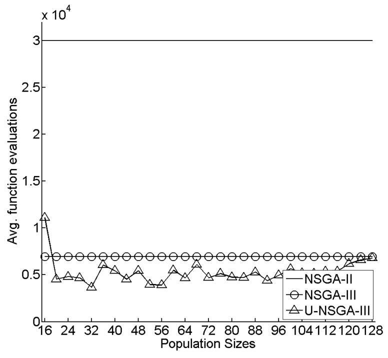
Results of the scaled versions of the two problems are shown in Figures 14a, 14b, 14c, 14d, 14e and 14f, respectively. Finally, Figure 15 presents the final population of the three algorithms for problems C3-DTLZ1 and C3-DTLZ4.

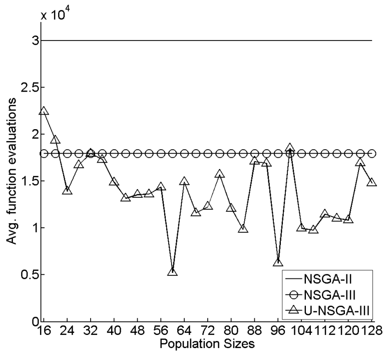
It can be seen form the figures that while NSGA-II fails to maintain adequate distribution of individuals, both NSGA-III and U-NSGA-III successfully achieve a uniformly distributed set of individuals covering the entire efficient front in each case. Only minor differences can be seen between the plots of NSGA-III and U-NSGA-III. Best, median and worst HV values in Tables 8 and 9 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 $\mathrm{HV}_{\mathrm{norm}}$ metric value.

4.4 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 almost equal


(a) OSY


(b) TNK


(c) SRN

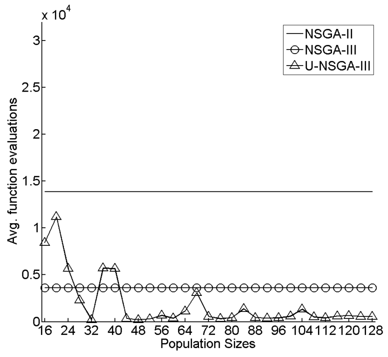
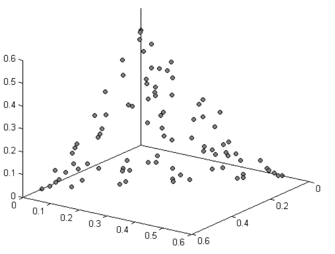
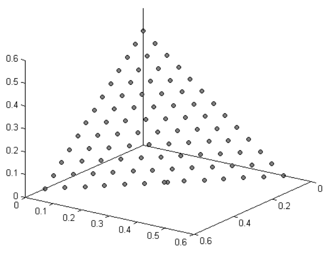
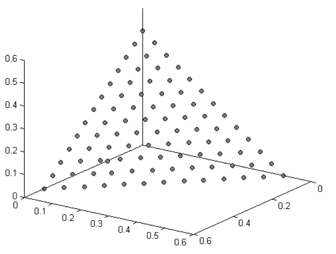
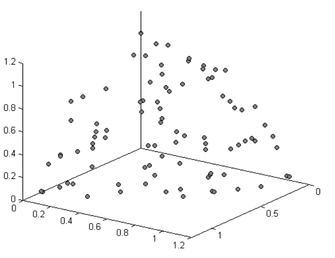

(d) BNH

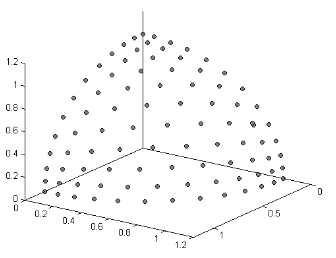
Figure 12: Performance of NSGA-II, NSGA-III, and U-NSGA-III with $N \geq H$ on constrained bi-objective problems.


(a) NSGA-II on DTLZ1


(b) NSGA-III on DTLZ1


(c) U-NSGA-III on DTLZ1


(d) NSGA-II on DTLZ2


(e) NSGA-III on DTLZ2

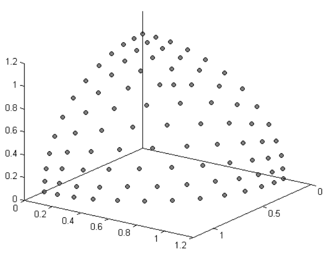

(f) U-NSGA-III on DTLZ2

Figure 13: Performance of NSGA-II, NSGA-III, and U-NSGA-III on unconstrained three-objective DTLZ problems.

Table 5: Best, medium and worst HV for bi-objective constrained test problems.

Prob.	G	P	NSGA-II			NSGA-III			U-NSGA-III		
			Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
OSY	100	8	0.59353	**0.34022**	**0.23561**	0.58761	0.30127	0.14430	**0.59369**	0.32459	0.03816
	100	48	0.74305	0.70292	0.67099	**0.74454**	0.68894	0.37136	0.73068	**0.71170**	**0.67890**
	100	88	**0.75465**	0.71471	**0.69245**	0.73756	**0.71627**	0.39624	0.74093	0.71095	0.68061
	100	128	0.75040	0.73164	**0.70700**	**0.75258**	**0.74515**	0.69238	0.75005	0.72177	0.68327
TNK	100	8	**0.27318**	0.23599	0.02417	0.26158	**0.23784**	0.10074	0.27074	0.22511	**0.10636**
	100	48	**0.31943**	**0.31556**	**0.31108**	0.31752	0.31330	0.30425	0.31858	0.31451	0.30048
	100	88	**0.32362**	0.32112	**0.31970**	0.32261	0.32067	0.31928	0.32279	**0.32141**	0.31935
	100	128	**0.32561**	0.32379	0.32108	0.32523	0.32323	0.32195	0.32513	**0.32396**	**0.32288**
BNH	500	8	0.68602	0.64094	0.44124	**0.73954**	**0.73914**	**0.73795**	0.73925	0.73875	0.73596
	500	48	0.80374	0.80243	0.79959	**0.80384**	0.80377	0.80373	0.80381	**0.80379**	**0.80375**
	500	88	0.80873	0.80833	0.80745	0.80892	0.80890	0.80887	**0.80893**	**0.80892**	**0.80888**
	500	128	0.81077	0.81048	0.81017	**0.81080**	0.81078	0.81077	0.81080	**0.81079**	**0.81077**
SRN	500	8	0.42888	0.40200	0.38589	0.46761	**0.46388**	**0.44001**	**0.46814**	0.46351	0.43743
	500	48	0.52318	0.52160	0.52018	0.52809	**0.52786**	**0.52359**	**0.52824**	0.52781	0.51651
	500	88	0.53022	0.52992	0.52948	0.53294	0.53281	**0.52997**	**0.53294**	**0.53286**	0.52982
	500	128	0.53293	0.53272	0.53207	0.53473	**0.53467**	**0.53318**	**0.53476**	0.53463	0.53043
WELDED	500	8	0.60589	0.58294	0.52943	0.65928	**0.64570**	0.58671	**0.65942**	0.64181	**0.61228**
	500	48	0.69353	0.69080	**0.68717**	**0.69677**	0.69284	0.62947	0.69623	**0.69423**	0.68537
	500	88	0.69867	0.69802	0.69457	**0.69955**	**0.69802**	**0.69535**	0.69950	0.69791	0.69319
	500	128	0.70017	0.69954	**0.69844**	0.70106	**0.70010**	0.69061	**0.70109**	0.69969	0.68764
PRESSURE	500	8	0.37783	0.33731	**0.27305**	**0.44090**	0.40992	0.00043	0.43856	**0.42860**	—
	500	48	0.50041	0.49835	**0.49533**	0.50801	0.49753	0.48582	**0.51024**	**0.49992**	0.48742
	500	88	0.51320	**0.51217**	**0.51129**	0.51438	0.51082	0.49018	**0.51592**	0.51102	0.48963
	500	128	0.51787	0.51714	**0.51643**	**0.51914**	0.51665	0.51191	0.51858	**0.51730**	0.50228

to H) in terms of hyper-volume values which are computed using the sampling based strategy proposed elsewhere [40] due to the computational complexities involved in the HV calculation 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 identical values of N and H as those used in NSGA-III study.

Tables 8 and 9 present performance of three algorithms on unscaled and scaled unconstrained DTLZ1 and DTLZ2 problems, and constrained C3-DTLZ1 and C3-DTLZ4 problems, respectively. It is amply clear that NSGA-II fails in terms of both convergence and maintaining diversity, meaning 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. The almost-identical PCP plots can be observed for the 10-objective versions of DTLZ problems in Figures 16, 17, and 18, respectively.

All these results demonstrate that the introduction of the niched tournament selection in

Table 6: 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

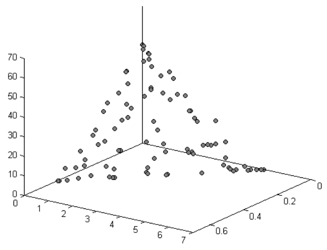
Table 7: Ideal and reference points used for constrained bi-objective problems.

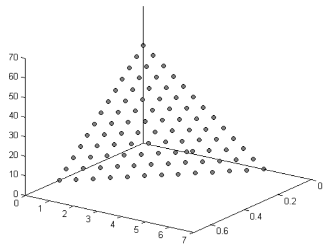
Problem	Ideal Point	Reference Point
OSY	$(-275, 5)$	$(-40.4, 77.77)$
TNK	$(0.029, 0.029)$	$(1.0605, 1.0605)$
BNH	$(0, 3.667)$	$(138.407, 50.5)$
SRN	$(5, -215)$	$(227.25, 0)$
Welded	$(0, 0)$	$(40.4, 0.00505)$
Pressure	$(42, -62761000)$	$(330270, -8080)$

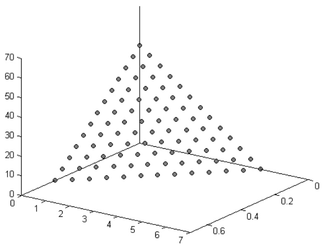
Table 8: Best, medium and worst HV_{norm} on multi/many-objective unconstrained DTLZ problems.

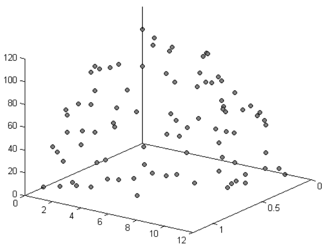
Prob.	Obj.	G	P	NSGA-II			NSGA-III			U-NSGA-III		
				Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
DTLZ1	3	400	92	.9198	.9136	.8597	**.9487**	**.9465**	**.9388**	.9462	.9464	.934
	5	600	212	—	—	—	**.9767**	**.9762**	**.9757**	.9766	.9760	.9751
	8	750	156	—	—	—	**.9953**	**.9953**	**.9953**	**.9953**	**.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

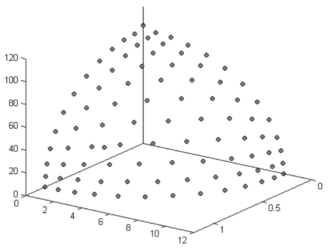
the original NSGA-III algorithm and the flexibility of using a different population size from the number of reference points do not change its performance in U-NSGA-III on many-objective


(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

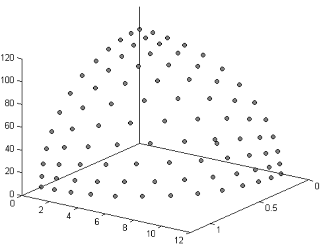

(f) U-NSGA-III on scaled DTLZ2

Figure 14: Performance of NSGA-II, NSGA-III, and U-NSGA-III on scaled unconstrained three-objective DTLZ problems.

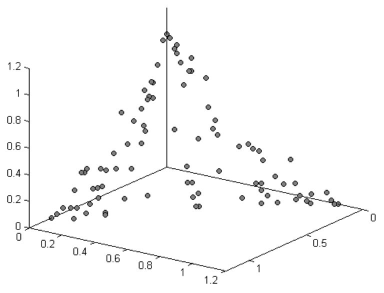
Table 9: Best, medium and worst HV on multi/many-objective constrained C3-DTLZ problems.

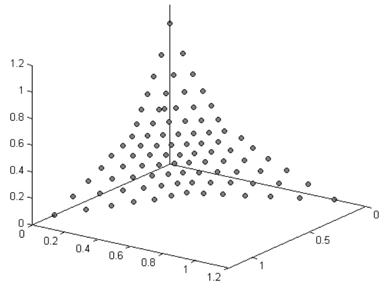
Prob.	M	G	P	NSGA-II			NSGA-III			U-NSGA-III		
				Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
C3-DTLZ1	3	750	92	.8786	.8679	.7705	**.9114**	**.9078**	**.9031**	9.107E-1	.9072	.8970
	5	1250	212	.8862	.0797	—	**.9942**	.9935	**.9933**	.9941	**.9938**	.9932
	8	2000	156	—	—	—	**1.083**	**1.082**	1.082E0	1.082	1.082	**1.082**
	10	3000	276	—	—	—	**1.105**	**1.105**	1.104	**1.105**	**1.105**	**1.105**
C3-DTLZ4	3	750	92	.7046	.6893	.6794	.7358	.7316	.7229	**.7362**	**.7326**	**.7308**
	5	1250	212	.7699	.7515	.6972	**.9454**	**.9450**	**.9444**	.9453	.9449	.9442
	8	2000	156	183	159	147	**275**	**275**	**275**	**275**	**275**	**2.75**
	10	3000	276	809	737	686	**1,130**	**1,130**	**1,130**	**1,130**	**1,130**	**1,130**

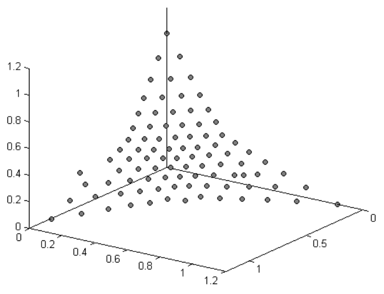
optimization problems.

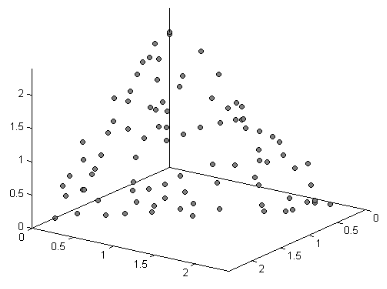
5 Conclusions

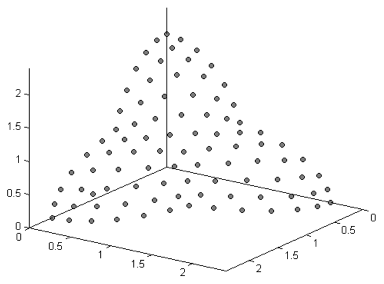
In this study, we have developed a unified evolutionary optimization algorithm U-NSGA-III which is a modification to a recently proposed evolutionary many-objective optimization method. U-NSGA-III has been carefully designed so as to solve mono-, multi-, and many-objective optimization problems. Simulation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


(a) NSGA-II on C3-DTLZ1


(b) NSGA-III on C3-DTLZ1


(c) U-NSGA-III on C3-DTLZ1


(d) NSGA-II on C3-DTLZ4


(e) NSGA-III on C3-DTLZ4

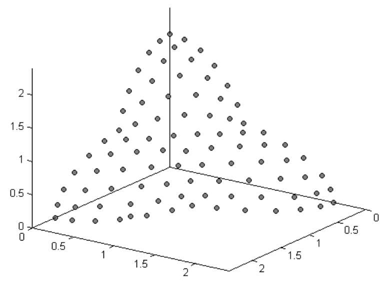
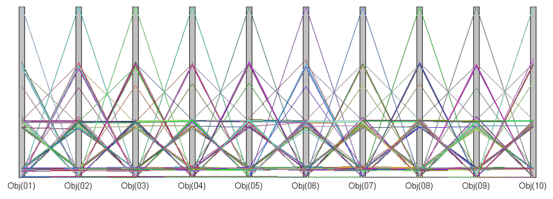

(f) U-NSGA-III on C3-DTLZ4

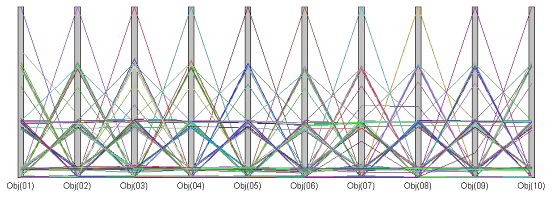
Figure 15: Performance of NSGA-II, NSGA-III, and U-NSGA-III on constrained three-objective C3-DTLZ problems.

cific 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, over a wide range of problems.

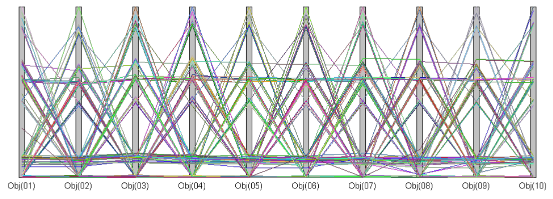
Despite efforts in unifying different evolutionary and non-evolutionary methods for optimizing a single objective, unification of dimension-specific optimization algorithms has not been attempted before, except an omni-optimizer approach [29] in which a unified single and bi-objective approach was suggested. However, the omni-optimizer algorithm 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.



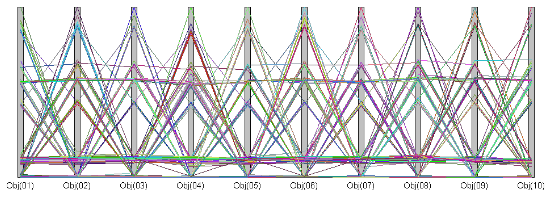
(a) DTLZ1 using NSGA-III



(b) DTLZ1 using U-NSGA-III



(c) DTLZ2 using NSGA-III



(d) DTLZ2 using U-NSGA-III

Figure 16: Performance of NSGA-III and U-NSGA-III on unconstrained many-objective DTLZ problems.

Due to these reasons, the study is important from the point of view of an efficient optimization software development and application of practical problems involving solution of separate one or many-objective versions of the same problem. Such unification approaches also provide researchers the key insight about operator interactions needed to constitute scalable algorithms.

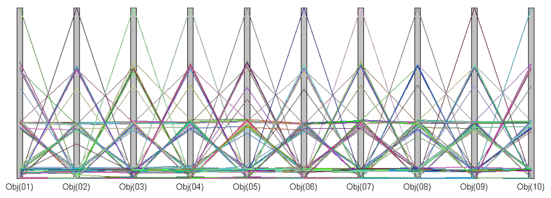
The unified optimization algorithm proposed here elevates the act of optimization as a computing-friendly approach. Computing algorithms are usually developed for handling a generic input having a large-dimensional attributes or parameters. However, the algorithm is also expected to work on a specific 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. For the first time, we have suggested a unified optimization algorithm that works in this computing-friendly principle. 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 supplied to be one, the U-NSGA-III algorithm finds a single optimal solution as a degenerate case.

As an extension of this study, MOEA/D and other efficient EMO methods can also be tried for the development of an equivalent unified approach. As a further extension, the unified approach can be modified to handle multi-modal problems for finding multiple optimal (and multiple Pareto-optimal [29]) solutions in a single simulation. The population approach and flexibility of EAs makes such a unified approach possible and further such studies will demonstrate the overall usefulness of EAs in solving various optimization problems in a unified manner.

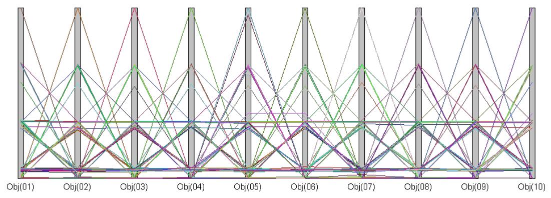
References

[1] K. Deb, *Multi-objective optimization using evolutionary algorithms*. Chichester, UK: Wiley, 2001.

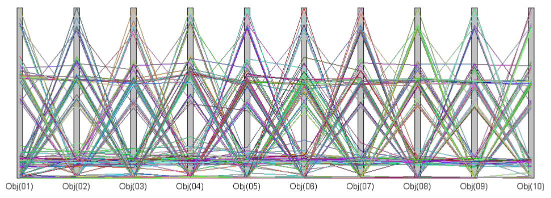
[2] C. A. C. Coello, D. A. VanVeldhuizen, and G. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Boston, MA: Kluwer, 2002.



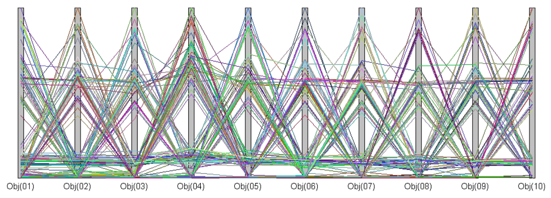
(a) S-DTLZ1 using NSGA-III



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



(c) S-DTLZ2 using NSGA-III



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

Figure 17: Performance of NSGA-III and U-NSGA-III on scaled unconstrained many-objective DTLZ problems.

[3] C. K. Goh and K. C. Tan, *Evolutionary Multi-objective Optimization in Uncertain Environments: Issues and Algorithms*. Springer-Verlag, 2009.

[4] K. C. Tan, E. F. Khor, and T. H. Lee, *Multiobjective Evolutionary Algorithms and Applications*. London, UK: Springer-Verlag, 2005.

[5] N. Srinivas and K. Deb, "Multi-objective function optimization using non-dominated sorting genetic algorithms," *Evolutionary Computation Journal*, vol. 2, no. 3, pp. 221–248, 1994.

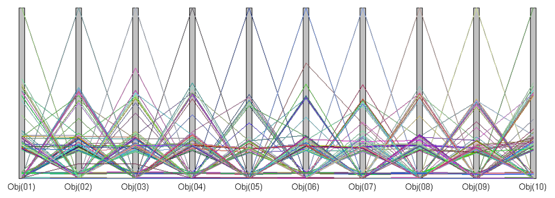
[6] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[7] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou, and T. Fogarty, Eds. Athens, Greece: International Center for Numerical Methods in Engineering (CIMNE), 2001, pp. 95–100.

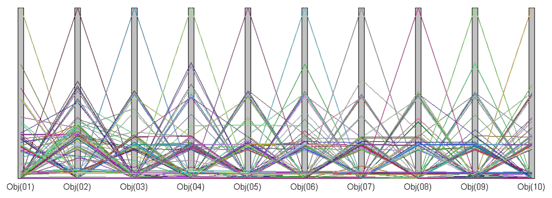
[8] D. W. Corne, J. D. Knowles, and M. Oates, "The Pareto envelope-based selection algorithm for multiobjective optimization," in *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature VI (PPSN-VI)*, 2000, pp. 839–848.

[9] C. A. C. Coello and G. Toscano, "A micro-genetic algorithm for multi-objective optimization," Laboratoria Nacional de Informatica AvRyerkerk, M., Averill, R., Deb, K., and Goodman, E. (2012). Optimization for Variable-Size Problems Using Genetic Algorithms. *Proceedings of the 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference.* (Indianapolis, USA). AIAA 2012-5569, Reston, VA: AIAA.anzada, Xalapa, Veracruz, Mexico, Tech. Rep. Lania-RI-2000-06, 2000.

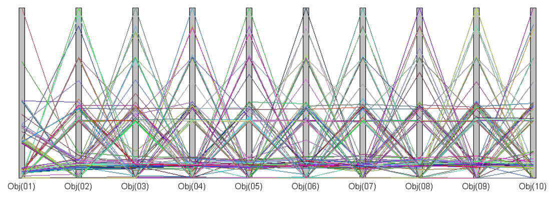
[10] E. F. Khor, K. C. Tan, and T. H. Lee, "Tabu-based exploratory evolutionary algorithm for effective multi-objective optimization," in *Proceedings of the First Evolutionary Multi-Objective Optimization (EMO-01)*, 2001, pp. 344–358.



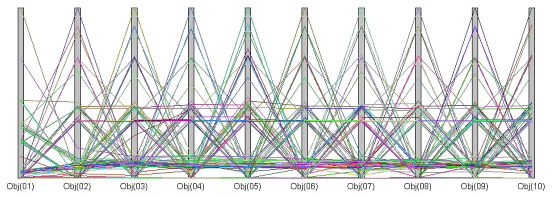
(a) C3-DTLZ1 using NSGA-III



(b) C3-DTLZ1 using U-NSGA-III



(c) C3-DTLZ4 using NSGA-III



(d) C3-DTLZ4 using U-NSGA-III

Figure 18: Performance of NSGA-III and U-NSGA-III on constrained many-objective DTLZ problems.

[11] M. Garza-Fabre, G. T. Pulido, and C. A. C. Coello, "Ranking methods for many-objective optimization," in *Proceedings of Mexican International Conference on Artificial Intelligence (MICAI-2009)*. Springer, 2009, pp. 633–645.

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

[13] V. Khare, X. Yao, and K. Deb, "Performance scaling of multi-objective evolutionary algorithms," in *Proceedings of the Second Evolutionary Multi-Criterion Optimization (EMO-03) Conference (LNCS 2632)*, 2003, pp. 376–390.

[14] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review," in *Proceedings of Congress on Evolutionary Computation (CEC-2008)*, 2008, pp. 2424–2431.

[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," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 4, pp. 539–556, 2011.

[17] H. Aguirre and K. Tanaka, "Many-objective optimization by space partitioning and adaptive epsilon-ranking on MNK-landscapes," in *Proceedings of 5th International Conference on Evolutionary Multi-Criterion Optimization (EMO 2009)*, 2009, pp. 407–422, lecture Notes in Computer Science 5467.

[18] H. Sato, H. E. Aguirre, and K. Tanaka, "Pareto partial dominance moea in many-objective optimization," in *Proceedings of Congress on Computational Intelligence (CEC-2010)*, 2010, pp. 1–8.

[19] X. Zou, Y. Chen, M. Liu, and L. Kang, "A new evolutionary algorithm for solving many-objective optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 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 *Proceedings of Artificial Intelligence conference (AI-2009), (LNAI 5866)*. Heidelberg, Germany: Springer-Verlag, 2009, pp. 443–453.

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

[22] M. Köppen and K. Yoshida, "Substitute distance assignments in NSGA-II for handling many-objective optimization problems," in *Proceedings of Fourth International Conference on Multi-Objective Optimization, EMO 2007 (LNCS 4403)*. Heidelberg, Germany: Springer-Verlag, 2007, pp. 727–741.

[23] 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.

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

[25] A. Britto and A. Pozo, "I-MOPSO: A suitable PSO algorithm for many-objective optimization," *2012 Brazilian Symposium on Neural Networks*, pp. 166–171, 2012.

[26] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, Part I: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

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

[28] D. Sharma, K. Deb, and N. N. Kishore, "Towards generating diverse topologies of path tracing compliant mechanisms using a local search based multi-objective genetic algorithm procedure," in *IEEE Congress on Evolutionary Computation'08*, 2008, pp. 2004–2011.

[29] K. Deb and S. Tiwari, "Omni-optimizer: A generic evolutionary algorithm for global optimization," *European Journal of Operations Research (EJOR)*, vol. 185, no. 3, pp. 1062–1087, 2008.

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

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

[32] 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.

[33] D. Saxena and K. Deb, "Trading on infeasibility by exploiting constraint's criticality through multi-objectivization: A system design perspective," in *Proceedings of the Congress on Evolutionary Computation (CEC-2007)*, in press.

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

[35] K. Deb and M. Goyal, "A robust optimization procedure for mechanical component design based on genetic adaptive search," *Transactions of the ASME: Journal of Mechanical Design*, vol. 120, no. 2, pp. 162–164, 1998.

[36] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies: A comprehensive introduction," *Natural computing*, vol. 1, pp. 3–52, 2003.

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

[38] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. C. Coello, and K. Deb, "Problem definitions and evaluation criteria for the cec-2006," Singapore: Nanyang Technological University, Tech. Rep., 2006, special Session on Constrained Real-Parameter Optimization, Technical Report.

[39] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation Journal*, vol. 8, no. 2, pp. 125–148, 2000.

[40] L. Bradstreet, L. While, and L. Barone, "A fast incremental hypervolume algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 714–723, 2008.

[41] P. Stein, "A note on the volume of a simplex," *The American Mathematical Monthly*, vol. 73, no. 3, pp. 299–301, 1966.

[42] X. Wang, "Volumes of generalized unit balls," *Mathematics Magazine*, vol. 78, no. 5, 2005.