Investigating the Normalization Procedure of NSGA-III

COIN Report 2018009

Julian Blank, Kalyanmoy Deb, and Proteek Chandan Roy

Michigan State University, East Lansing, MI, USA
{blankjul,kdeb,royprote}@msu.edu
http://www.coin-laboratory.com

Abstract. Most practical optimization problems are multi-objective in nature. Moreover, the objective values are, in general, differently scaled. In order to obtain uniformly distributed set of Pareto-optimal points, the objectives must be normalized so that any distance metric computation in the objective space is meaningful. Thus, normalization becomes a crucial component of an evolutionary multi-objective optimization (EMO) algorithm. In this paper, we investigate and discuss the normalization procedure for NSGA-III, a state-of-the-art multi- and many-objective evolutionary algorithm. First, we show the importance of normalization in higher-dimensional objective spaces. Second, we provide pseudo-codes which presents a clear description of normalization methods proposed in this study. Third, we compare the proposed normalization methods on a variety of test problems up to ten objectives. The results indicate the importance of normalization for the overall algorithm performance and show the effectiveness of the originally proposed NSGA-III's hyperplane concept in higher-dimensional objective spaces.

Keywords: Many-objective Optimization · NSGA-III · Normalization.

1 Introduction

The need to optimize several objectives at a time has been investigated for years, and various algorithms have been proposed [21]. The desired result is a non-dominated set of solutions close to the true Pareto-optimal front [13], instead of a single optimal solution. The non-dominated set of solutions gives us the possibility to make a suitable decision for choosing a single preferred solution following the algorithm's execution and provides useful information about optimal solutions with respect to different preferences. Also, the decision maker can compare the trade-offs between different solutions and therefore justify his/her choice.

However, the fact that the target space has more than one dimension brings new challenges which must be addressed in designing the optimization algorithm. To deal with multiple dimensions in the objective space, reference directions express the trade-off between solutions with respect to each objective. Usually, either the user provides them directly or they are sampled uniformly in the unit

space. If a uniformly distributed set of reference directions can be supplied and an EMO algorithm can find one or more Pareto-optimal solutions close to each reference direction, a widely distributed set will be achieved at the end.

Clearly, such a process will involve distance computations in the objective space, thereby necessitating a normalization procedure within the algorithm, which will consider the range of each objective on a same scale. In contrast to test problems where variables and objectives are already nicely scaled, in practical problems, the objective space range for each objective may differ by several magnitudes. Therefore, for any distance or trade-off calculation, normalization of objectives becomes an inevitable task.

In this paper, we investigate and discuss the normalization procedure of NSGA-III, a state-of-the-art evolutionary multi- and many-objective algorithm. In addition to the originally proposed normalization procedure of NSGA-III, we suggest a few other normalization methods. Our purpose in this paper is to: (i) show the importance of normalization in the objective space for high-dimensional multi-objective problems, (ii) compare different normalization procedures, and (iii) provide pseudo-codes for different normalization procedure.

In the remainder of this paper, we will first present a review of some past studies expanding upon or applying NSGA-III. Thereafter, in Section 3 we provide a brief description of the algorithm including the role of normalization. Then, different methodologies for normalization are discussed in depth, and a hands-on example is provided in Section 4. Afterwards, in Section 5, we present our results evaluated on a variety of test problems with up to ten objectives. Finally, conclusions of the study are presented in Section 6.

2 Related Studies

The need of optimizing more than one objective at a time brought attention of the multi-objective optimization research area. Also, normalization is often assumed implicitly and not discussed in detail, the importance to solve practical problems is indisputable.

The normalization procedure for MOEA/D [20] was investigated in [11]. The normalization was based on the PBI (penalty-based boundary intersection) measure by considering lower and upper bound estimations. The study showed that the normalization has both positive and negative effects on the performance on test problems. Interestingly, the normalization showed positive effects for test problems that do not need any normalization. Furthermore, three representative strategies for estimating the ideal point in MOEA/D were studied [17]. The ϵ value, which is subtracted from the minimum of each objective in the current population, is varied: small (pessimistic), large (optimistic), or decreasing over time (dynamic). The authors found out that the strategy has an effect on the exploration and exploitation of the algorithm and suggest to use the dynamic strategy for unknown problems. Also, the effect of local optimization to improve solutions contributing to the ideal point has been investigated [15]. The study showed that the local search helps to improve the diversity of the final non-dominated population for certain problems by converging close to the true ideal point in an early phase.

Moreover, NSGA-III [5,12], designed to solve problems with more than three objectives, was investigated since its publication in 2014, and some extensions and improved versions were proposed. For instance, a unified approach for mono-, multi- and many-objective problems, U-NSGA-III [14], introduces more selection pressure during the mating selection. Moreover, NSGA-III-OSD [3] decomposes the objective space into several subspaces by clustering the reference directions uniformly. Each subspace has its own population and PBI as decomposition method. Additionally, EliteNSGA-III [10] improves the diversity and accuracy of the resulting Pareto-front. An elite population archive is maintained to preserve previously generated elite solutions that would probably be eliminated by the reference survival selection procedure.

Moreover, NSGA-III has been applied to industry problems, for instance environmental dispatch problem [2], hydro thermal wind scheduling problem [19], and car engine design problem [9]. Also, NSGA-III has been implemented in different programming languages and popular optimization frameworks, such as jMetal [8], moeaframework [1], and PlatEMO [16].

3 NSGA-III

In the following, NSGA-III is explained and the role of normalization during the survival selection is illustrated. The basic framework remains similar to NSGA-II [6] with significant modifications to the mating and survival selection. In NSGA-III, parents to be used for recombination are selected randomly. The survival selection considers the M-dimensional objective space by using the reference direction concept. Reference directions Z represent trade-offs between solutions regarding their objective values. They are either provided a priori by the user or created uniformly, commonly executed using the Das and Dennis's technique [4].

An outline of the survival selection is shown in Algorithm 1. Considering an optimization problem with M objectives and an evolutionary algorithm with a population size of N, generation t begins with the current population $P^{(t)}$ known as the parent population, creates an offspring population $Q^{(t)}$ through recombination and mutation, and merges two populations together to create $R^{(t)} = P^{(t)} \cup Q^{(t)}$. The survival selection has to return $P^{(t+1)}$ – the next generation population of size N. The creation of $P^{(t+1)}$ is as follows. First, the individuals of the merged population $R^{(t)}$ are sorted by non-dominated rank which results in a list of fronts $(F_1, F_2, \ldots)$. To do this, the set of surviving solutions S is initialized as an empty set. Thereafter, it is iterated through the list of fronts and the current front F_i is appended to S, if the resulting number of individuals does not exceed N. The front where $|S \cup F_i| \geq N$ is the potential splitting front F_L. In case, $|S|+|F_L| = N$ no splitting is necessary and all surviving individuals are already determined. Otherwise, a niching method is employed to choose those F_L members that are associated with the least represented reference directions already associated by individuals in S. To assign individuals to the reference directions Z, S is normalized by using $\hat{z}^*$ as a lower and the nadir point estimation $\hat{z}^{nad}$ as an upper bound. Therefore, each already selected individual k in S is assigned to the closest reference direction π_k having a per-

Algorithm 1: NSGA-III Survival Selection

Input: Merged Population $R^{(t)}$, Number of surviving individuals N, Reference Directions Z, Ideal Point Estimation $\hat{z}^*$, Nadir Point Estimation $\hat{z}^{nad}$
Output: Surviving Individuals $P^{(t+1)}$

$(F_1, F_2, \ldots) \leftarrow$ `non_dominated_sort`$(R^{(t)})$
$S = \emptyset, \quad i = 1$
while $|S| + |F_i| < N$ **do** $S \leftarrow S \cup F_i; \quad i = i + 1$
$F_L \leftarrow F_i$
if $|S| + |F_L| = N$ **then** $S \leftarrow S \cup F_L$
else
  `/* Normalize objectives space and update boundary estimation */`
  $\bar{S}, \bar{F_L}, \hat{z}^*, \hat{z}^{nad} \leftarrow$ `normalize`$(S, F_L, \hat{z}^*, \hat{z}^{nad})$
  `/* niche count, assigned` Z_i`, perpendicular dist to` Z_i `*/`
  $\rho, \pi, d \leftarrow 0$
  for $k \leftarrow 1$ *to* $|S|$ **do**
    $\pi_k, d_k \leftarrow$ `associate`$(\bar{S}_k, Z); \quad \rho_{\pi_k} \leftarrow \rho_{\pi_k} + 1$
  end
  `// Remaining individuals from` F_L `to fill up` S
  $S \leftarrow S \cup$ `niching`$(\bar{F_L}, n - |S|, \rho, \pi, d)$
end
$P^{(t+1)} \leftarrow S$
return $P^{(t+1)}$

pendicular distance of d_k. The niche count ρ is kept track of and incremented by one for each assignment. Finally, the `niching` method selects from $\bar{F_L}$ the remaining $N - |S|$ individuals using ρ, π, d. 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.

4 Normalization Procedure

In this section, we investigate different normalization procedures for NSGA-III. The normalization relies on lower and upper boundaries in the objective space that correspond to the estimated ideal point $\hat{z}^*$ and the estimated nadir point $\hat{z}^{nad}$. Therefore, it is sufficient to provide the estimation for both points in order to normalize. The normalized value $\bar{a_i}$ in the i-th objective is then calculated by

$$\bar{a_i} = \frac{a_i - \hat{z}_i^*}{\hat{z}_i^{nad} - \hat{z}_i^*}. \tag{1}$$

The open question is how to find estimation the boundary points $\hat{z}^*$ and $\hat{z}^{nad}$, so that non-dominated solutions are properly emphasized. The ideal point estimation $\hat{z}^*$ is rather simple and the calculation is based on the smallest value in each objective we have observed since the start of the optimization run:

$$\hat{z}_j^* = \min(\hat{z}_j^* \cup R_j), \tag{2}$$

where R_j denotes the j-th objective of the merged population. Please note that the ideal point should not be calculated from R at each generation, but being updated. The survival selection of NSGA-III does not guarantee each individual contributing to an ideal point to survive in higher dimensions. For this reason, an update is necessary for a correct estimation of the ideal point.

The nadir point estimation is more tricky and is one of the main cruxes of this study. Let us first discuss the requirements and goals for estimating the nadir point in the context of many-objective evolutionary algorithms.

(i) Estimated ideal point must dominate estimated nadir point: Since it is normalized between the ideal and nadir point estimation, we need to make sure that $\forall\, i \in [1, .., M] : \hat{z}_i^{nad} > \hat{z}_i^*$. In practice, the formulation should be more strict where $\forall\, i \in [1, .., M] : \hat{z}_i^{nad} - \hat{z}_i^* > \epsilon_{nad}$ with ϵ_{nad} being our assumption about the minimum range of the Pareto-front for all objectives. A minimum difference ϵ_{nad} prevents having floating point issues and loosing the diversity during the survival selection.

(ii) Estimated nadir point must converge to the true nadir point with generations: Finally, when the population converges to the true optimum, the estimated nadir point should converge to the true nadir point. When both ideal and nadir points are estimated close to their true values, the EMO algorithm gets stabilized and works efficiently to find a well-distributed set of near Pareto-optimal points.

(iii) Estimated nadir point must gradually change from one generation to the next: This requirement is especially important in an evolutionary context, because the normalization is applied before assigning to reference directions and directly influences the survival method. An abrupt change of the normalization process will make previous generation's non-dominated solutions meaningless, thereby creating a restart situation.

In the following, we will suggest a number of possible normalization procedures. Towards this goal, we shall revise the hyperplane concept in Section 4.3 which was proposed in the original publication and present corner cases that must be handled on an implementation level.

4.1 Maximum of Non-dominated Front (MNDF)

Straightforwardly, we can concatenate the maximum of each objective of the non-dominated front of each generation and construct the nadir point. Assuming the algorithm converges eventually to the entire Pareto-optimal front, the estimated nadir point will be equal to the true nadir point. Since the method is based on the current non-dominated front in each generation, some special degenerate cases must be addressed. If the population has only one non-dominated solution, this solution might also be equal to the ideal point. This will cause a division by zero problem to Equation 1. In this case, we propose to consider the next non-dominated front for the estimation of the nadir point. This process can continue

until the difference between estimated nadir and ideal points becomes larger than a pre-specified threshold ϵ_{nad}.

4.2 Maximum of Extreme Points (ME)

We can use the achievement scalarization function (ASF) [18] along the axes to find M extreme points. The ASF function is defined by:

$$ASF(f(x), w, \hat{z}^*) = \max_{i=1}^{M} \frac{f_i(x) - \hat{z}_i^*}{w_i}, \tag{3}$$

where the weight vector w of the k-th objective is $w_k = 1$ and $w_i = \epsilon_{asf}$, if $i \neq k$. For our experiments we set $\epsilon_{asf} = 10^{-6}$. The procedure to update the extreme points each generation is presented in Algorithm 2. We find the extreme points by combining the merged population with the current extreme points. This ensures that the extreme points get an update, instead of a straightforward replacement all the time. Then, the ASF function scalarizes multiple objectives into a single value. We simply choose the solution with the minimum ASF value having at least ϵ_{nad} different in the i-th objective. Finally, we set the extreme point $e^{(i)}$ to the objective vector of the index found.

Algorithm 2: Maximum of Extreme Points

Input: R, $\hat{z}^*$, Current Extreme Points e
Output: Updated Extreme Points e

$A \leftarrow R \cup e$
for $i \leftarrow 1$ *to* M **do**
  $w \leftarrow (\epsilon_{asf}^1, \ldots, \epsilon_{asf}^M)$
  $w_i \leftarrow 1$
  $k \leftarrow \texttt{argmin}(\texttt{ASF}(A, w, \hat{z}^*), \epsilon_{nad})$,
  $e^{(i)} \leftarrow A^{(k)}$
end

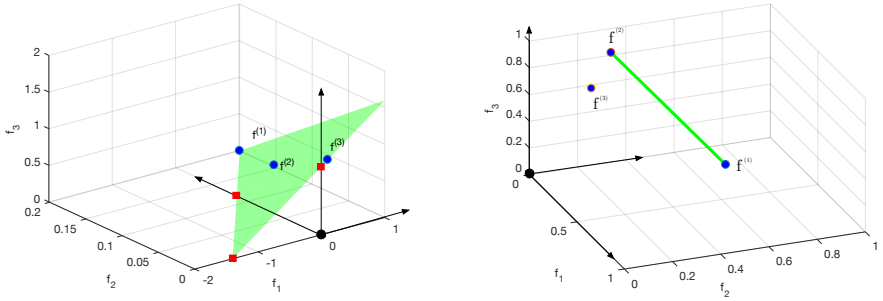
4.3 Revised Hyperplane through Extreme Points (HYP)

In the following, we revise the idea implemented in the original NSGA-III. We analyze the hyperplane concept on implementation level, where a number of exceptions must be handled to ensure the algorithm will not fail – a number of which we describe below.

Negative Intercepts The hyperplane is found in the translated space $e' \leftarrow e - \hat{z}^*$ and the intercepts I are the intersections with the coordinate axes. The intercepts with the axes in the translated space represent the estimation of the range of the Pareto-front. For this reason, the intercepts are required to be positive. However, the hyperplane through the extreme points might intersect the axis not in the positive orthant. Let us consider the following scenario, illustrated in Figure 1a, with solutions in the objective space $f^{(1)} = (1.0, 0.2, 0.0)$, $f^{(2)} = (0.4, 0.1, 0.4)$, and $f^{(3)} = (0.1, 0.0, 1.0)$. Each point is an extreme point with respect to the achievement scalarization function along an axis: $f^{(1)}$ along f_1, $f^{(2)}$ along f_2, and $f^{(3)}$ along f_3. The intercepts of the hyperplane will result in

$I = (-1.4, 0.1167, 0.933)$. Obviously, the intercept with the first axis is negative and cannot be used for normalization.

No unique hyperplane exists A unique M-dimensional hyperplane through the extreme points can only be found if all points are linearly independent from each other. On implementational level, a matrix with the extreme points as row vectors has to be inverted to obtain the axis intercepts. Linearly dependent rows form a singular matrix, where an exception will be thrown during the inversion. Moreover, the extreme point selection does not guarantee to select different points for different axes. For instance, let us consider three non-dominated points $f^{(1)} = (0.8, 0.5, 0.5)$, $f^{(2)} = (0.1, 0.3, 0.9)$, and $f^{(3)} = (0.4, 0.1, 0.9)$, as shown in Figure 1b. The extreme points are selected using the achievement scalarization function for each axis. Here, $f^{(1)}$ will be chosen for both f_1 and f_2 axes, and $f^{(2)}$ for f_3. Because $f^{(1)}$ is chosen for two axes, the matrix to be inverted is singular and a unique hyperplane does not exist. Additionally, numerical instability through floating point calculations depending on the library used for the inversion must be addressed as well.



(a) Negative Intercepts in the objective space: $f^{(i)}$ are illustrated by blue dots and axes intercepts by red squares.

(b) No unique hyperplane exists: Any hyperplane through the line is possible .

Fig. 1: Two degenerate cases of HYP normalization method.

Pseudo-code of HYP For any implementation using the hyperplane idea, the above presented scenarios must be addressed. In these cases, we propose the algorithm to fall back to the worst point of the current non-dominated front or of the population. Algorithm 3 illustrates the procedure using the hyperplane idea and handling the degenerate cases. Note, that our procedure requires the worst point estimation $\hat{z}^w$. In contrast to the ideal point estimation $\hat{z}^*$, we define $\hat{z}^w$ having the largest value observed so far, for each objective. We use $\hat{z}^w$ as an upper bound of $\hat{z}^{nad}$. With the check, if any intercept is smaller than ϵ_{nad}, we ensure that the hyperplane has no negative intercepts and the resulting nadir point estimation is significantly larger than $\hat{z}^*$. Also, we make sure that the resulting nadir point estimation $I_k + \hat{z}^*_k$ is not larger than our upper bound $\hat{z}^w_k$. If one of these requirements is not met, we declare the hyperplane as not useful

for the normalization purpose and, therefore, the nadir point estimation is set as the maximum of each objective of current non-dominated population members.

Finally, we make sure the nadir point estimation satisfies the first requirement, being dominated by the estimated ideal point. If it is not, then we use the maximum of corresponding violating objective in the population.

Algorithm 3: Hyperplane through extreme points.

Input: Merged Population R, Non-dominated Fronts $(F_1, .., F_K)$, $\hat{z}^*$, Worst Point Estimation $\hat{z}^w$, Extreme Points e
Output: Nadir Point Estimation $\hat{z}^{nad}$

$e \leftarrow$ `update_extreme_points`$(R, \hat{z}^*, e)$
$b \leftarrow$ `FALSE`
try:
  $A' \leftarrow$ `find_hyperplane`$(e, \hat{z}^*)$
  $I \leftarrow$ `find_intercepts`(A')
  for $k \leftarrow 1$ *to* M **do**
    $\hat{z}_k^{nad} \leftarrow I_k + \hat{z}_k^*$
    if $I_k < \epsilon_{nad}$ ***or*** $\hat{z}_k^{nad} > \hat{z}_k^w$ **then**
      $b \leftarrow$ `TRUE`
      break
    end
  end
catch ***Error***: $b \leftarrow$ `TRUE`

`/* Fall back to the maximum in each objective of current front */`
if $b =$ *TRUE* **then**
  for $i \leftarrow 1$ *to* M **do** $\hat{z}_i^{nad} \leftarrow$ `max_in_objective`(F_1, i)
end

`/* Nadir point must be significantly larger in each objective */`
for $i \leftarrow 1$ *to* M **do**
  if $\hat{z}_i^{nad} - \hat{z}_i^* < \epsilon_{nad}$ **then** $\hat{z}_i^{nad} \leftarrow$ `max_in_objective`$((F_1 \cup ... \cup F_k), i)$
end

5 Results

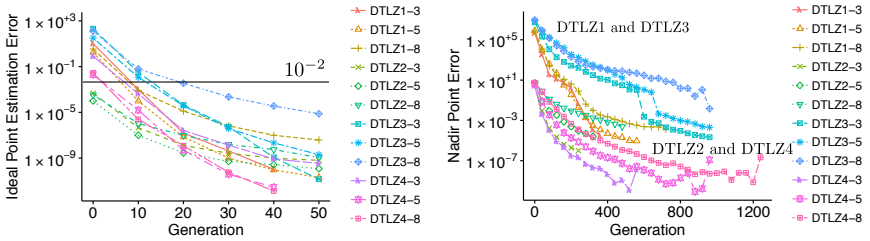
In this section, we present the simulation results of NSGA-III using the proposed normalization procedures[1]. First, we analyze the ideal and nadir point estimation error over generations. Second, we present the performance of the proposed normalization procedures on test problems. We use the scalable multi-objective optimization test problems suite, DTLZ [7], for our evaluation. Also, we investigate scaled versions of these problems, where objectives are multiplied with increasing factors. We conducted the experiment analogous to the original publication of NSGA-III. Therefore, we refer to [5] for details about the experimental setup and algorithm parameters, such as references lines, population size, number of generation and recombination operators. We run each algorithm 50 times on each test problem.

[1] The source code is freely available at https://github.com/msu-coinlab/pymoo

We compute the following squared ideal point and nadir point estimation errors to track the progress of them through generations:

$$\hat{e}^* = \sum_{i=1}^{m} \left(\frac{\hat{z}_i^* - z_i^*}{z_i^{nad} - z_i^*} \right)^2 \qquad \hat{e}^{nad} = \sum_{i=1}^{m} \left(\frac{\hat{z}_i^{nad} - z_i^{nad}}{z_i^{nad} - z_i^*} \right)^2 \tag{4}$$

The squared ideal point estimation error $\hat{e}^*$ is calculated by summing up the normalized squared difference in each objective. The squared nadir point estimation error $\hat{e}^{nad}$ is also defined accordingly. Figure 2a shows the median $\hat{e}^*$ during the first 50 generations. Note that the error decreases below one percent after at most 20 generations for all considered test problems. This confirms the hypothesis, that the ideal point estimation is quick, less problematic and the assumption to use the smallest values for each objective reduces the estimation error effectively. Analogous, we illustrate $\hat{e}^{nad}$ in Figure 2b. Clearly, the overall estimation error is higher than the ideal point estimation error and the convergence is slower. Furthermore, we can cluster the error into two groups, where DTLZ1 and DTLZ3 start with a smaller estimation error compared to DTLZ2 and DTLZ4. This is caused by the multimodality introduced through the convergence function for DTLZ1 and DTLZ3.



(a) Median of $\hat{e}^*$ over generations. (b) Median of $\hat{e}^{nad}$ over generations.

Fig. 2: Variation of ideal and nadir point estimation errors using HYP.

Next, let us discuss the performance of the different normalization procedures. We use the inverse generational distance (IGD) as a performance metric for our study. For scaled problems (SDTLZ), we used the weighted Euclidean distance for the IGD computation, where the distance in objective k is divided by $z_k^{nad} - z_k^*$. Figure 3 shows the box plots of the IGD values on the DTLZ test problem suite for three, five, and ten objectives. In addition to the proposed methods, we evaluate a normalization procedure (TRUE) where the true boundary points $\hat{z}^* = z^*$ and $\hat{z}^{nad} = z^{nad}$ are used all along. The following observations are made:

- It can be concluded from the experiments, that MNDF is too naive and performed mostly worse than the other approaches. Due to the fact that the current non-dominated front may not be close to the true Pareto-optimal front, the search is easily biased to specific regions of the objective space.
- For scaled DTLZ problems, HYP has more outliers compared to the other problems. For SDTLZ2, MNDF shows surprisingly good results. Because the

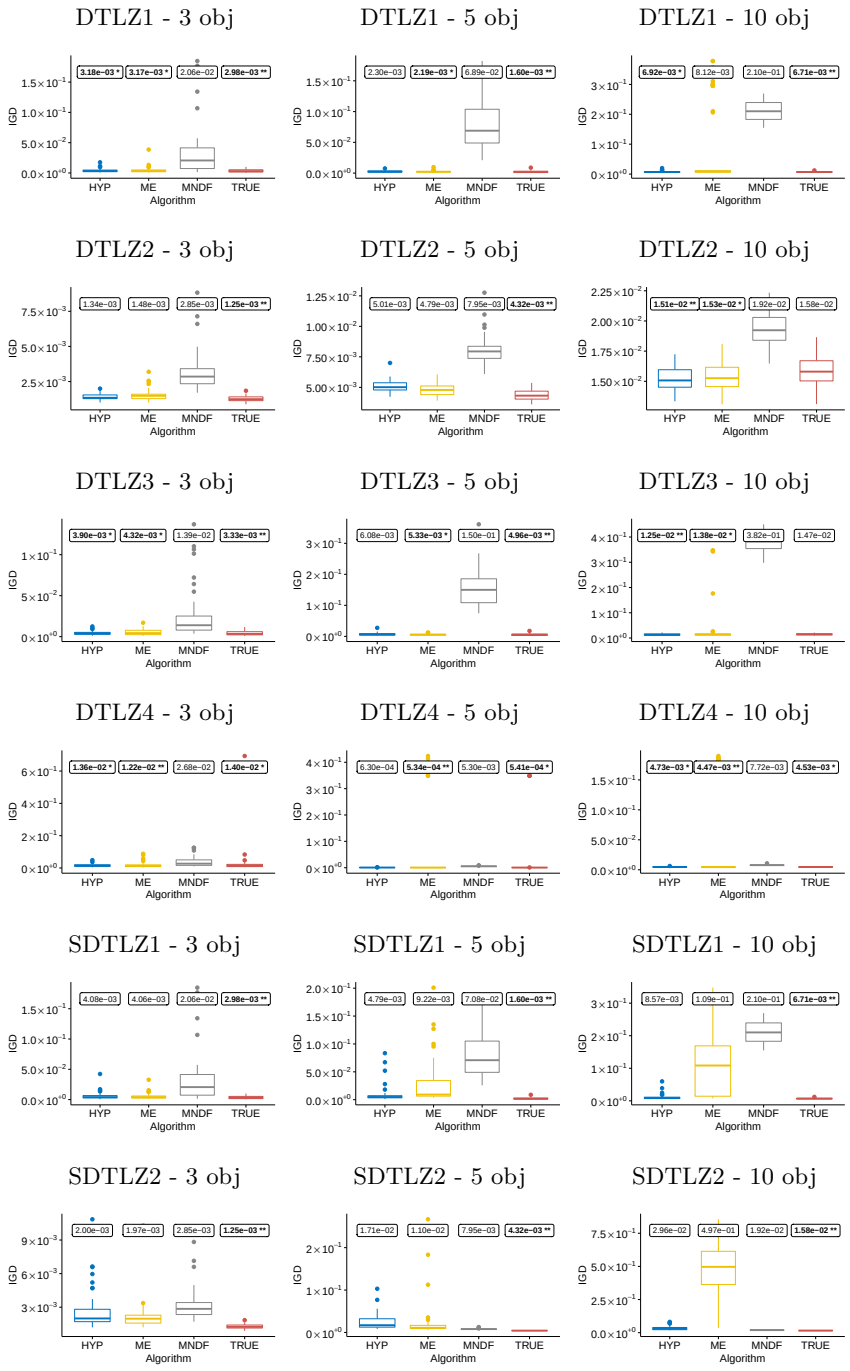


Fig. 3: Box plots showing IGD values for DTLZ (normalized IGD for SDTLZ) problems. The median values are presented for each algorithm annotated by ** if best and * if not significantly worse than best (according to Wilcoxon signed-rank test with $p = 0.05$)

convergence function is rather simple, the non-dominated front seems to be a good representative of the true Pareto-optimal front.
- By comparing the median IGD values, we can observe that TRUE performs 17 out of 18 times the best. However, in practice the boundary points of the Pareto-front are unknown and this information can not be utilized.
- ME and MNDF showed more outliers and significantly higher median performances compared to HYP. Moreover, seven times HYP did not perform significantly worse than TRUE. We recommend using HYP whenever the true boundary points of the Pareto front are unknown.

6 Conclusions

In this paper, different normalization procedures for NSGA-III for solving many-objective optimization problems have been investigated. It has been shown that normalization is a crucial component for multi-objective algorithms and necessary to solve problems where objectives are scaled differently. The original proposed normalization method has been analyzed and degenerate cases, such as negative intercepts and no unique hyperplanes, have been discussed. The proposed methods have been applied to test problems up to ten objectives, where the ideal as well as the nadir point estimation errors over generations has been analyzed. The results confirm that the ideal point estimation is less problematic and gets settled quickly, whereas the nadir point estimation is tricky and requires a large number of generations to get settled. Moreover, the overall performance of NSGA-III with different normalization procedures has been evaluated. Although the original proposed hyperplane concept HYP must handle degenerate cases carefully (see Algorithm 3), it shows the best performance besides TRUE. The hyperplane concept is not only applicable for NSGA-III and can now be tested with other multi- and many-objective algorithms where normalization is not addressed properly or naively implemented. Moreover, the effect of normalization on different shapes of the Pareto-optimal front must be studied next.

References

1. Moeaframework. `http://moeaframework.org`, accessed: 2018-09-26
2. Bhesdadiya, R.H., Trivedi, I.N., Jangir, P., Jangir, N., Kumar, A.: An nsga-iii algorithm for solving multi-objective economic/environmental dispatch problem. Cogent Engineering **3**(1), 1269383 (2016)
3. Bi, X., Wang, C.: An improved nsga-iii algorithm based on objective space decomposition for many-objective optimization. Soft Computing **21**(15), 4269–4296 (Aug 2017)
4. Das, I., Dennis, J.E.: Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. SIAM Journal on Optimization **8**(3), 631–657 (1998)
5. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. IEEE Transactions on Evolutionary Computation **18**(4), 577–601 (Aug 2014)

6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. Trans. Evol. Comp **6**(2), 182–197 (Apr 2002)
7. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization, pp. 105–145. Springer London, London (2005)
8. Durillo, J., Nebro, A., Alba, E.: The jmetal framework for multi-objective optimization: Design and architecture. In: CEC 2010. pp. 4138–4325. Barcelona, Spain (July 2010)
9. Gaur, A., Talukder, A.K.M.K., Deb, K., Tiwari, S., Xu, S., Jones, D.: Finding near-optimum and diverse solutions for a large-scale engineering design problem. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1–8 (Nov 2017)
10. Ibrahim, A., Rahnamayan, S., Martin, M.V., Deb, K.: Elitensga-iii: An improved evolutionary many-objective optimization algorithm. In: 2016 IEEE Congress on Evolutionary Computation (CEC). pp. 973–982 (July 2016)
11. Ishibuchi, H., Doi, K., Nojima, Y.: On the effect of normalization in moea/d for multi-objective and many-objective optimization. Complex & Intelligent Systems **3**(4), 279–294 (Dec 2017)
12. Jain, H., Deb, K.: An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach. IEEE Transactions on Evolutionary Computation **18**(4), 602–622 (Aug 2014)
13. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston (1999)
14. Seada, H., Deb, K.: A unified evolutionary optimization procedure for single, multiple, and many objectives. IEEE Transactions on Evolutionary Computation **20**(3), 358–369 (June 2016)
15. Singh, H.K., Yao, X.: Improvement of reference points for decomposition based multi-objective evolutionary algorithms. In: Shi, Y., Tan, K.C., Zhang, M., Tang, K., Li, X., Zhang, Q., Tan, Y., Middendorf, M., Jin, Y. (eds.) Simulated Evolution and Learning. pp. 284–296. Springer International Publishing, Cham (2017)
16. Tian, Y., Cheng, R., Zhang, X., Jin, Y.: Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum]. IEEE Computational Intelligence Magazine **12**(4), 73–87 (Nov 2017)
17. Wang, R., Xiong, J., Ishibuchi, H., Wu, G., Zhang, T.: On the effect of reference point in moea/d for multi-objective optimization. Applied Soft Computing **58**, 25 – 34 (2017)
18. Wierzbicki, A.P.: The use of reference objectives in multiobjective optimization. In: Fandel, G., Gal, T. (eds.) Multiple Criteria Decision Making Theory and Applications, pp. 468–486. Berlin: Springer-Verlag (1980)
19. Yuan, X., Tian, H., Yuan, Y., Huang, Y., Ikram, R.M.: An extended nsga-iii for solution multi-objective hydro-thermal-wind scheduling considering wind power cost. Energy Conversion and Management **96**, 568 – 578 (2015)
20. Zhang, Q., Li, H.: Moea/d: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation **11**(6), 712–731 (Dec 2007)
21. Zhou, A., Qu, B.Y., Li, H., Zhao, S.Z., Suganthan, P.N., Zhang, Q.: Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm and Evolutionary Computation **1**(1), 32 – 49 (2011)