A Fast and Effective Method for Pruning of Non-Dominated Solutions in Many-Objective Problems

Saku Kukkonen
Department of Information Technology
Lappeenranta University of Technology
P.O. Box 20, FIN-53851 Lappeenranta, Finland
saku.kukkonen@lut.fi

Kalyanmoy Deb
Kanpur Genetic Algorithms Laboratory (KanGAL)
Indian Institute of Technology Kanpur
Kanpur, PIN 208 016, India
deb@iitk.ac.in

KanGAL Report Number 2007004

Abstract

Diversity maintenance of solutions is an essential part in multi-objective optimization. Existing techniques are suboptimal either in the sense of obtained distribution or execution time. This paper proposes an effective and relatively fast method for pruning a set of non-dominated solutions. The proposed method is based on a crowding estimation technique using nearest neighbors of solutions in Euclidean sense, and a technique for finding these nearest neighbors quickly. The method is experimentally evaluated, and results indicate a good trade-off between the obtained distribution and execution time. Distribution is good also in many-objective problems, when number of objectives is more than two.

1 Introduction

Pruning a set of non-dominated solutions is a common and essential part of multi-objective evolutionary algorithms (MOEAs) such as the strength Pareto evolutionary algorithm (SPEA2) [1] and the elitist non-dominated sorting genetic algorithm (NSGA-II) [2]. An idea is to prune a non-dominated set to have a desired number of solutions in such a way that the remaining solutions have as good diversity as possible, meaning that the spread of extreme solutions is as high as possible, and the relative distance between solutions is as equal as possible. The best way to obtain a good distribution would be using some clustering algorithm. However, this is computationally expensive, since clustering algorithms usually take time $O(MN^2)$ to prune a set of size N with M objectives [3]. The complexity makes clustering techniques inapplicable to large population sizes, especially as pruning is usually done after each generation. The pruning technique of SPEA2 is based on finding the kth nearest neighbor of solutions, and has complexity of clustering.

In NSGA-II, the pruning of non-dominated solutions takes time $O(MN \log N)$ based on the *crowding distance*. The pruning method of NSGA-II provides good diversity in the case of two objectives, but when the number of objectives is more than two, the obtained diversity declines drastically [4]. The reason for the bad performance is the fact that the crowding distance fails to approximate the crowding of the solutions when the number of the objectives is more than two [5].

Pruning of non-dominated solutions can be seen as a multi-objective optimization problem in itself: the method should provide as good diversity as possible and be as fast as possible. It has been considered that these objectives are conflicting. However, this paper proposes a new algorithm with two different crowding estimation techniques for pruning non-dominated solutions in such a way that the obtained diversity is intended to be good also in the case of more than two objectives, and the consumed time is intended to be considerably less than in clustering.

The rest of the paper is organized as follows: The proposed pruning algorithm is described in Section 2 and experiments are reported in Section 3. Potential improvements of the algorithm are discussed in Section 4 and conclusions are drawn in Section 5.

2 Proposed Pruning Method

The basic idea of the proposed pruning method is to eliminate the most crowded members of a non-dominated set one by one, and update the crowding information of the remaining members after each removal. This idea is trivial but it contains two problems: how to efficiently determine the crowding of members and how to efficiently update the crowding information of remaining members after removal. Straightforward approaches for these computations are in time complexity class $O(MN^2)$, which makes them inapplicable to large population sizes. Therefore two approaches for crowding estimation based on the nearest neighbors of solution candidates are introduced, and then a technique for finding these nearest neighbors *quickly* is introduced.

2.1 Crowding Estimation Based on Nearest Neighbors

2.1.1 2 Nearest Neighbors

Probably the simplest crowding estimation technique is to measure the distance between a solution and its nearest neighbor solution, and use this distance to estimate crowding.

The solution having the smallest distance is considered as the most crowded. When the Euclidean (L_2) distance metric is used for distance calculations, there will always be two solutions having the same smallest distance to the nearest neighbor due to the symmetry property of the metric. Instead of selecting randomly one of the two solutions, the solutions can be ordered according to the distance to the second nearest neighbor. The solution having smaller distance to second nearest neighbor is more crowded. If the distance to the nearest and second nearest neighbors is marked with $L_2^{\mathrm{NN}_1}$ and $L_2^{\mathrm{NN}_2}$, respectively, then a distance vector $d_{2-\mathrm{NN}} = [L_2^{\mathrm{NN}_1}, L_2^{\mathrm{NN}_2}]$ is attached to each solution. In the case of real-coded variables and continuous objective functions, this provides a crowding measure, which usually establishes unambiguous ordering of solutions having the same smallest distance to the nearest neighbor.

The idea of using two nearest neighbors as primary and secondary sorting keys has been effectively used in the case of two objectives in [6]. However, when the number of objectives is more than two, pruning method described in [6] does not distinguish two nearest neighbors for crowding estimation and apparently fails to estimate crowding properly.

Distance to just two nearest neighbors can be used in crowding estimation even when number of objectives is more than two. This is because the distance to two nearest neighbors is already enough for finding most crowded solution unambiguously in most of cases.

2.1.2 M Nearest Neighbors

A bit more developed idea is to use the k nearest neighbors for crowding estimation in such a way that distances to the k nearest neighbors are *multiplied* together, and the solution having the smallest product is considered the most crowded. The product of distances, which is here called the *vicinity distance*, is a simple measure, but it already manages to contain information about vicinity and location. This is illustrated in Fig. 1 with two nearest neighbors in three different cases. In all the cases a solution (circle) locates on a line between two nearest neighbors (filled dots). In the first case (a)), the solution is closer to right neighbor than the left neighbor and vicinity distance will be $d_v = x\,(L - x) = xL - x^2$. When the solution moves between two nearest neighbors, its vicinity distance value will change along a paraboloid curve and has the maximum value $d_v = L^2/4$ when the solution is in the middle of two nearest neighbors as shown in b). If solution remains in the middle of two nearest neighbors but distance to these doubles as in case c) compared to b), then vicinity distance will increase to value $d_v = L^2$. Thus, value of vicinity distance will increase if distance to neighbors increases and it will decrease, if neighbors stay still but solution moves from the location, where vicinity distance has maximum value.

The idea is extended in such a way that the number of nearest neighbors for crowding estimation calculations is kept the same as the number of objectives, i.e., $k = M$. More formally vicinity distance is defined $d_{M-\mathrm{NN}} = \prod_{i=1}^{M} L_2^{\mathrm{NN}_i}$, where $L_2^{\mathrm{NN}_i}$ is distance to the ith nearest neighbor according to L_2 distance metric.

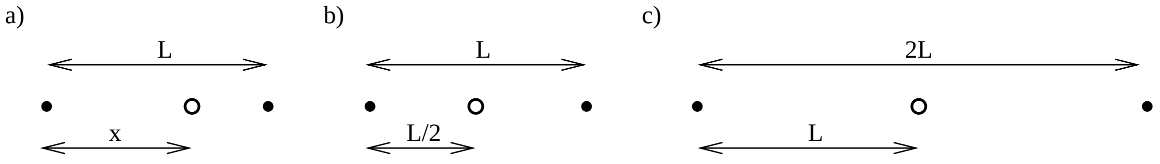

Figure 1: The effect of vicinity and location of neighbors to the vicinity distance value

2.2 Efficient k Nearest Neighbor Search

Finding the k nearest neighbors (k-NN) is a widely used operation in pattern recognition (PR) and image analysis. Especially, in vector quantization (VQ), which is used, e.g., in compression and classification, it is needed to find the nearest neighbor from a set of vectors for a given input vector. A naive approach for finding nearest neighbors is to calculate distance to all the vectors in given set and then find vectors, which have the shortest distance. Time complexity of this method is $O(kN)$, which is too high to be used in many applications. Therefore, there has been numerous research on this subject in the past three decades and more efficient algorithms have been developed.

Existing efficient methods are mainly based on two different approaches. First approach is to use some specialized data structure (such as k-d, R, or R* tree) for storing vectors, and search of nearest neighbor can be done fast because of the data structure. Another approach is to use some pre-calculated or easily evaluated information to reduce amount of vectors to which distance need to be calculated. This later approach is chosen here since it seems more flexible for a situation where set of vectors is changing (reducing in this case).

The exact k-NN search technique used here is known as the *equal-average nearest neighbor search (ENNS) algorithm* [7, 8], and it uses the following theorem for the Euclidean (L_2) distance measure to reduce the amount of distance calculations:

$$\left|\frac{1}{M}\sum_{i=1}^{M} x_i - \frac{1}{M}\sum_{i=1}^{M} y_i\right| \le \frac{L_2(\vec{x},\vec{y})}{\sqrt{M}} \Leftrightarrow \left(\sum_{i=1}^{M} x_i - \sum_{i=1}^{M} y_i\right)^2 \le ML_2(\vec{x},\vec{y})^2 \ . \tag{1}$$

Thus, if the sums of elements of given vectors $\vec{x}$ and $\vec{y}$ are known, an upper bound for the Euclidean distance between them can be calculated. It is more convenient to use the squared Euclidean distance, since the actual distance values are not needed for finding neighbors, and calculating the square root is an expensive operation computationally. The geometrical interpretation of (1) is that vectors $\vec{x}$ and $\vec{y}$ are projected to a central axis of a hypercube (a projection vector starting from origin and going through point $(1, 1, \ldots, 1)$) and then the difference of projection values is at most equal to the Euclidean distance between vectors divided by the square root of the number of elements in the vectors. It has been proved that (1) holds also for any other projection vector $\vec{p}$ than the central axis of a hypercube, and (1) transforms into form [9]:

$$(p_x - p_y)^2 \le L_2(\vec{x},\vec{y})^2 \ , \quad \text{where} \quad p_x = \frac{\vec{p}\cdot\vec{x}}{|\vec{p}|} \quad \text{and} \quad p_y = \frac{\vec{p}\cdot\vec{y}}{|\vec{p}|} \ . \tag{2}$$

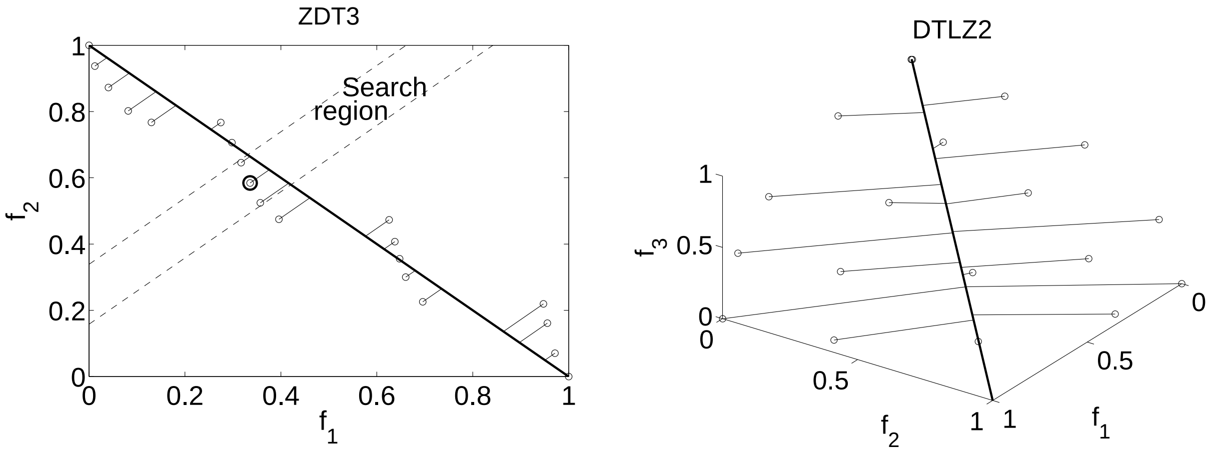

Figure 2: Selected projection axis in the case of two and three objectives when k-NN search based on (2) is used

It is useful to select the projection axis to represent the direction in which the vectors have the largest variance, and such axis can be obtained using, e.g., principal component analysis (PCA) [9]. However, data analysis is not necessarily needed in the case of a set of non-dominated solutions, since there already exists information on how the vectors/solutions are distributed. When all the objectives are to be minimized (or maximized), there exists such kind of monotonicity between objective values that when values of $M-1$ objectives increase, then the objective value of the remaining objective decreases. Thus, there is a negative correlation between objective values, and the projection axis can be chosen to go through points $(0, 0, \ldots, 0, 1)$ and $(1, 1, \ldots, 1, 0)$ if the objective values have been normalized to the value range $[0, 1]$. The projection axis chosen in this way in the case of two and three objectives has been illustrated in Fig. 2 for sets of non-dominated solution points. Projections of the points are also illustrated.

The ENNS technique can be used to find the nearest neighbor for a solution in such a way that first the projected values of the vectors are calculated, and these values are sorted. Then, for the solution vector, the distance to the solution having the nearest projected value is set as the best minimum distance found so far. The distances to other neighbor solutions according to the projected value are also calculated, and the minimum distance value is updated accordingly, as long as there exist neighbors for which (2) holds. This technique can be extended easily for finding the k nearest neighbors using k-th smallest distance instead of the minimum distance to reject distant solutions with (2) [9].

Besides ENNS, a technique called partial distortion search (PDS) algorithm [10] is used to speed up the execution further. It interrupts the distance calculation if a partial sum of elements exceeds a known minimum distance value. This technique adds one extra comparison but decreases the number of mathematical calculations speeding up the execution.

2.3 The Proposed Algorithm

The proposed pruning algorithm is based on the crowding estimation and k-NN search techniques presented above, and minimization of all objectives is assumed. For efficient maintenance of crowding information of remaining solutions after removal of the most crowded solution, a priority queue such as a heap [11, pp. 140–152] is used. The proposed algorithm is:

```
PRUNING OF NON-DOMINATED SET
input: a non-dominated set $\mathcal{F}$ (objectives are minimized),
       the size $n$ of a desired pruned set
output: elements of a heap $\mathcal{H}$

1  find minimum ($f_i^{\min}$) and maximum ($f_i^{\max}$) values of each objective $i$ of the members
   of $\mathcal{F}$ and normalize members according to the formula $f_i = (f_i - f_i^{\min})/(f_i^{\max} - f_i^{\min})$
2  for each member of $\mathcal{F}$, change $M$th objective $f_M$ to value $1 - f_M$ and calculate
   a sum of objective values, $m$
3  for each member of $\mathcal{F}$, find nearest neighbors (cf. Sect. 2.2) and calculate
   distance measure $d$ (cf. Sect. 2.1) for crowding estimation
4  for members of $\mathcal{F}$ having a minimum or maximum objective value, assign $d = \infty$
5  create an ascending heap $\mathcal{H}$ from the members of $\mathcal{F}$
6  while $|\mathcal{H}| > n$
7      remove an element (root node) with a minimum $d$ value from $\mathcal{H}$ and update $\mathcal{H}$
8      for the neighbors of the removed element
9          calculate a new $d$ value
10         replace old $d$ value in $\mathcal{H}$ with the new one and update $\mathcal{H}$
```

The first operation of the algorithm in line 1 is the normalization of objective values of the obtained non-dominated set. Different objectives might have very different value ranges, and using unnormalized values would distort the obtained distribution. The time complexity of the normalization is $O(MN)$. The normalization can lead to an infinite value if minimum and maximum values for some objective are same. Then this objective can be discarded from calculations. The second line of the algorithm re-maps the M-th objective value f_M to value $1 - f_M$ so that the original form (1) of ENNS can be used directly. The complexity of the operations in line 2 is $O(MN)$. Finding the k nearest neighbors for all the members of given set in line 3 is known as the *all-k-nearest-neighbor problem*, and it can be done in time $O(kN \log N)$ for fixed number of objectives M and fixed distance metric [12]. Line 4 takes time $O(M)$ and makes sure that members having extreme objective values are removed last. Creating a heap in line 5 takes time $O(N \log N)$. The while-loop in lines 6–10 is executed at most N times (on average $N/2$ times). Removing a minimum element from the heap and updating the heap to have correct structure in line 7 takes time $O(\log N)$. If the M-NN crowding measure is used, each member of a non-dominated set has on average M neighbors, whose crowding values are affected if the member is removed (these neighbors can be found easily if neighborhood information is also stored when the nearest neighbors have been searched in line 3). Therefore, the for-loop in lines 8–10 is executed M times on average. The calculation of a new crowding value in line 9 means finding the k nearest neighbors. This can be done in time $O(k \log N)$ for a static set of vectors with fixed dimension. Now, the set of vectors is changing (reducing) and the actual time complexity depends on how the vectors are distributed. Finally, replacing the crowding value to the heap, and updating the heap to have the correct structure in line 10 takes time $O(\log N)$.

The complexity class of the whole algorithm is hard to estimate since, when the k-NN method presented earlier in this section is used in lines 3 and 9, the actual time complexity depends on number of objectives and how members of the non-dominated set are distributed on the selected projection axis. As the pruning method is used extensively, the most interesting thing is to know the expected complexity in practice. This, as well as the performance of the method according to the obtained diversity, is evaluated experimentally in the following.

3 Experiments

The proposed pruning method with the two introduced crowding estimation techniques was implemented in the generalized differential evolution 3 (GDE3) [13], which was then used to solve test problems. Also, pruning based on the crowding distance as in NSGA-II was implemented in GDE3. GDE3 is an extension of differential evolution (DE) [14] for constrained multi-objective optimization. Roughly speaking, the evolutionary part of the algorithm is DE and the multi-objective part is from NSGA-II [2]. This combination has been shown to give benefit over NSGA-II with rotated problems [15]. Furthermore, GDE3 has other improvements over NSGA-II, and an interested reader is advised to see references [5, 13].

NSGA-II and SPEA2 were used for comparison[1], and the diversity of the obtained results was measured using spacing, maximum spread, and hypervolume [16, pp. 327–333]. The spacing (S) measures the standard deviation of distances from each vector to the nearest vector in the obtained non-dominated set. A smaller value for S is better, and for an ideal distribution $S = 0$. The maximum spread (D) measures the length of the diagonal of a minimal hyperbox, which encloses the obtained non-dominated set, and a larger value tells about a larger spread between extreme solutions. The hypervolume (HV) calculates the volume of the objective space between the solutions and a reference (nadir) point, and a larger value is better.

[1]Code for NSGA-II was obtained from the web site www.iitk.ac.in/kangal and for SPEA2 from the web site www.tik.ee.ethz.ch/pisa.

3.1 Bi-Objective Problems

Bi-objective test problems, ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 [16, pp. 356–360] were solved using population size 100 and 1000 generations. A set of solutions for these problems are shown in Fig. 3, where sets of solutions have been sifted by 0.05 units along both objectives to alleviate observation. As it can be observed, the proposed pruning method and SPEA2 provide better diversity compared to diversity obtained using crowding distance although differences are small.

Tests were repeated 100 times and mean & standard deviation values are reported in Table 1. According to the spacing value, the proposed pruning method provides the best diversity, and the M-NN crowding estimation technique is slightly better. Differences in the maximal spread and hypervolume values are minimal compared to the differences in the spacing values. The proposed pruning method takes about two times longer to execute than the pruning method based on the crowding distance. However, the time needed for pruning is less than 20% from total CPU time needed[2].

3.2 Tri-Objective Problems

The proposed pruning method was also tested on a set of tri-objective test problems, DTLZ1, DTLZ2, DTLZ4, DTLZ5, and DTLZ7 [4], using population size 300 and 1000 generations. Results after one run are shown in Figs. 4–8, and numerical results for the problems from 100 repetition runs are shown in Table 2. The improvement over the pruning method based on the crowding distance is clearly visible, and the proposed pruning method provides similar or better (Fig. 8) results compared to SPEA2. Also the spacing metric in Table 2 indicates the same, this time the 2-NN crowding estimation technique being slightly better in most cases compared to M-NN. As in the case of bi-objective problems, the maximal spread and hypervolume values have only small differences. The proposed pruning method is now at worst about eight times slower than the pruning method based on the crowding distance. The pruning time is intelligibly less for the 2-NN than for the M-NN crowding estimation technique, and it is also less for DTLZ5 and DTLZ7, which do not have continuous and spread Pareto-fronts.

3.3 Measured Pruning Time Complexity

The time complexity of the proposed pruning method was verified experimentally in the case of two and three objectives using ZDT1 and DTLZ1. The measured pruning times for various population sizes are shown in Fig. 9.

In the bi-objective case (ZDT1), the proposed pruning method takes about twice the time of the pruning method based on the crowding distance, and complexity classes of

[2]The total CPU time for the proposed pruning method is smaller compared to GDE3 with the crowding distance because of overall speedups in the program code. On the other hand, GDE3 uses a naive $O(MN^2)$ non-dominated sorting implementation instead of faster $O(N \log^{M-1} N)$ implementation [3].

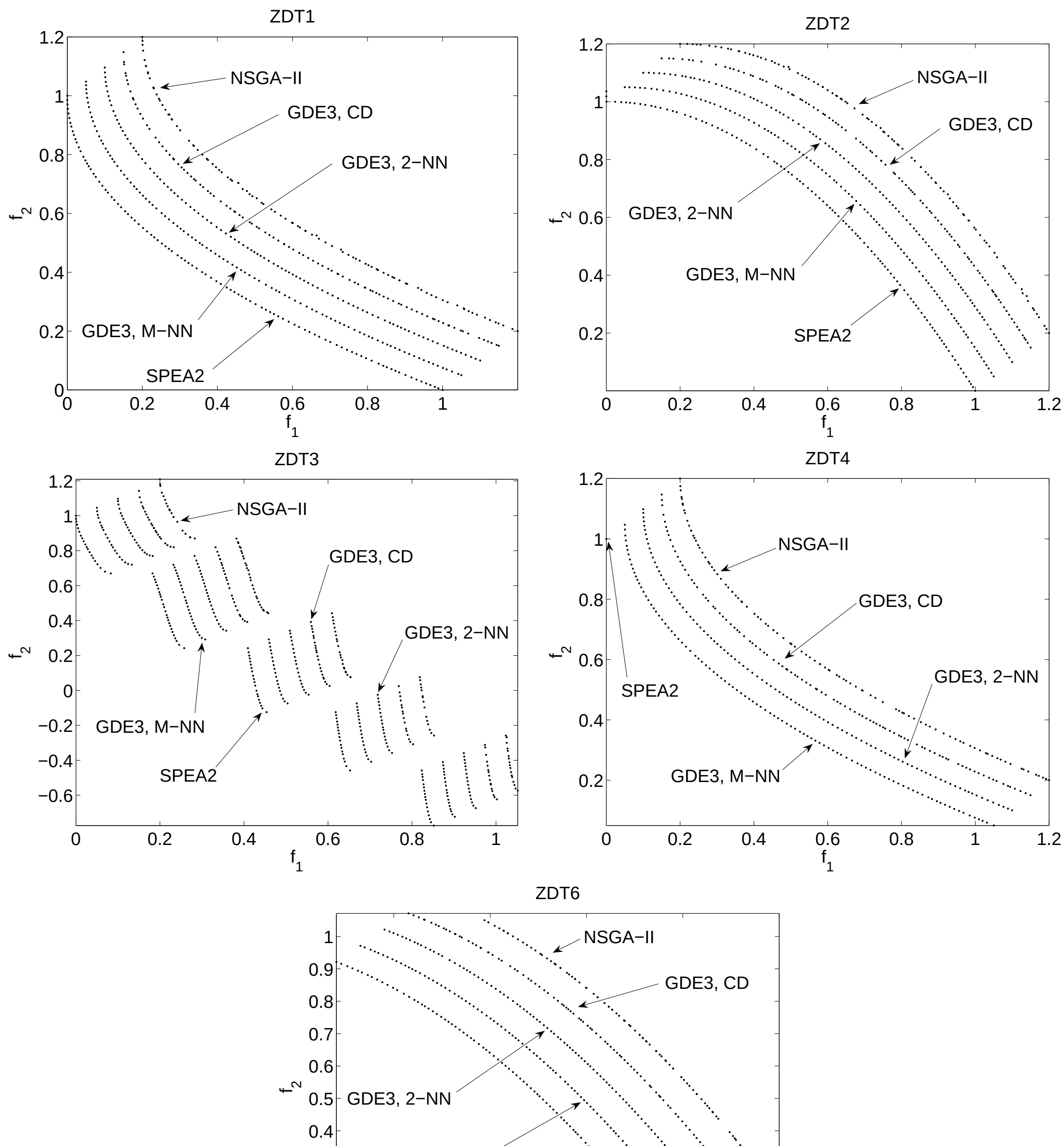

Figure 3: The results for the ZDT problems using NSGA-II and SPEA2, and GDE3 with the pruning methods based on crowding distance (CD) and described crowding estimation techniques (2-NN & *M*-NN)

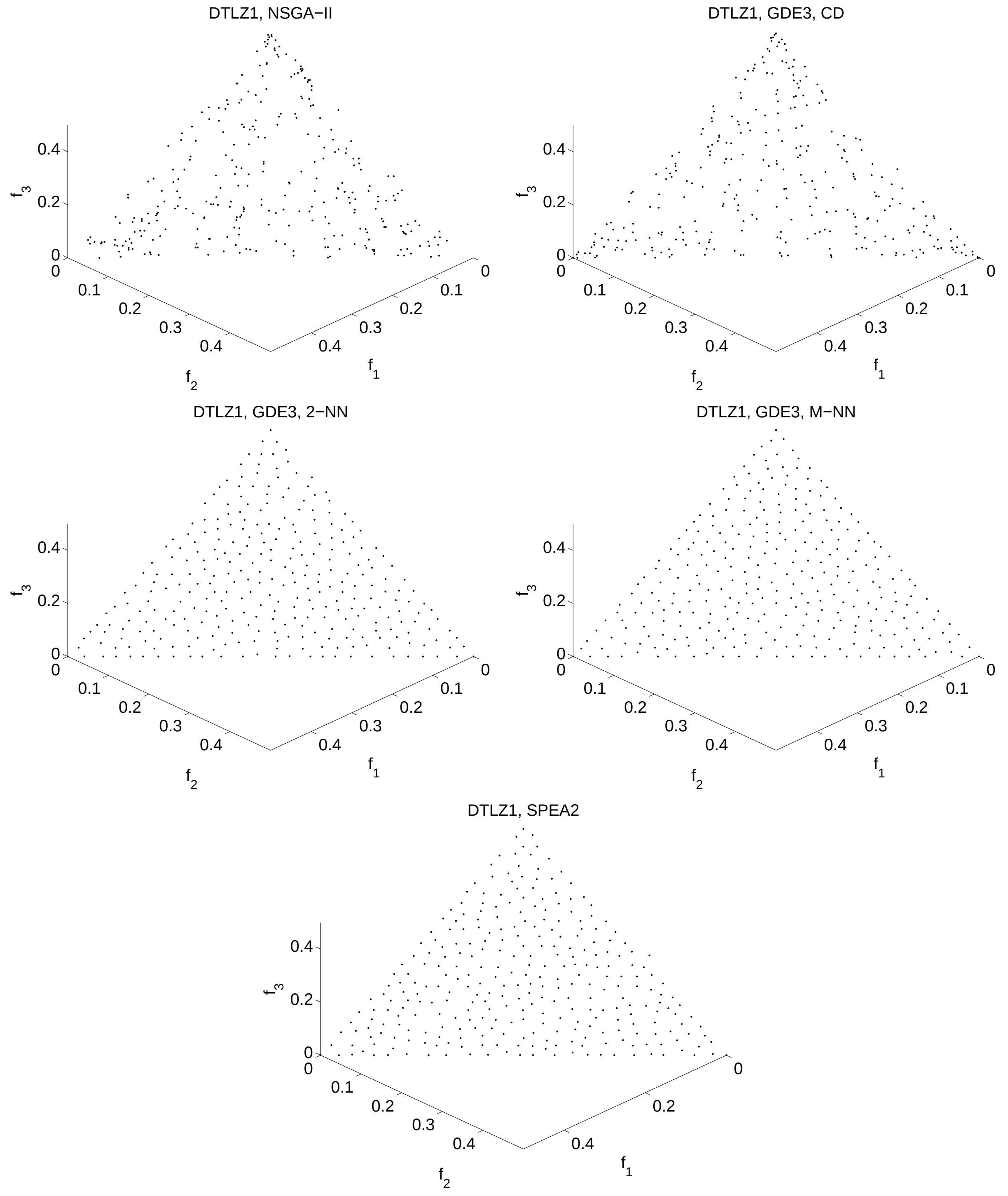

Figure 4: The results for the DTLZ1 problems using NSGA-II and SPEA2, and GDE3 with the pruning methods based on crowding distance (CD) and described crowding estimation techniques (2-NN & M-NN)

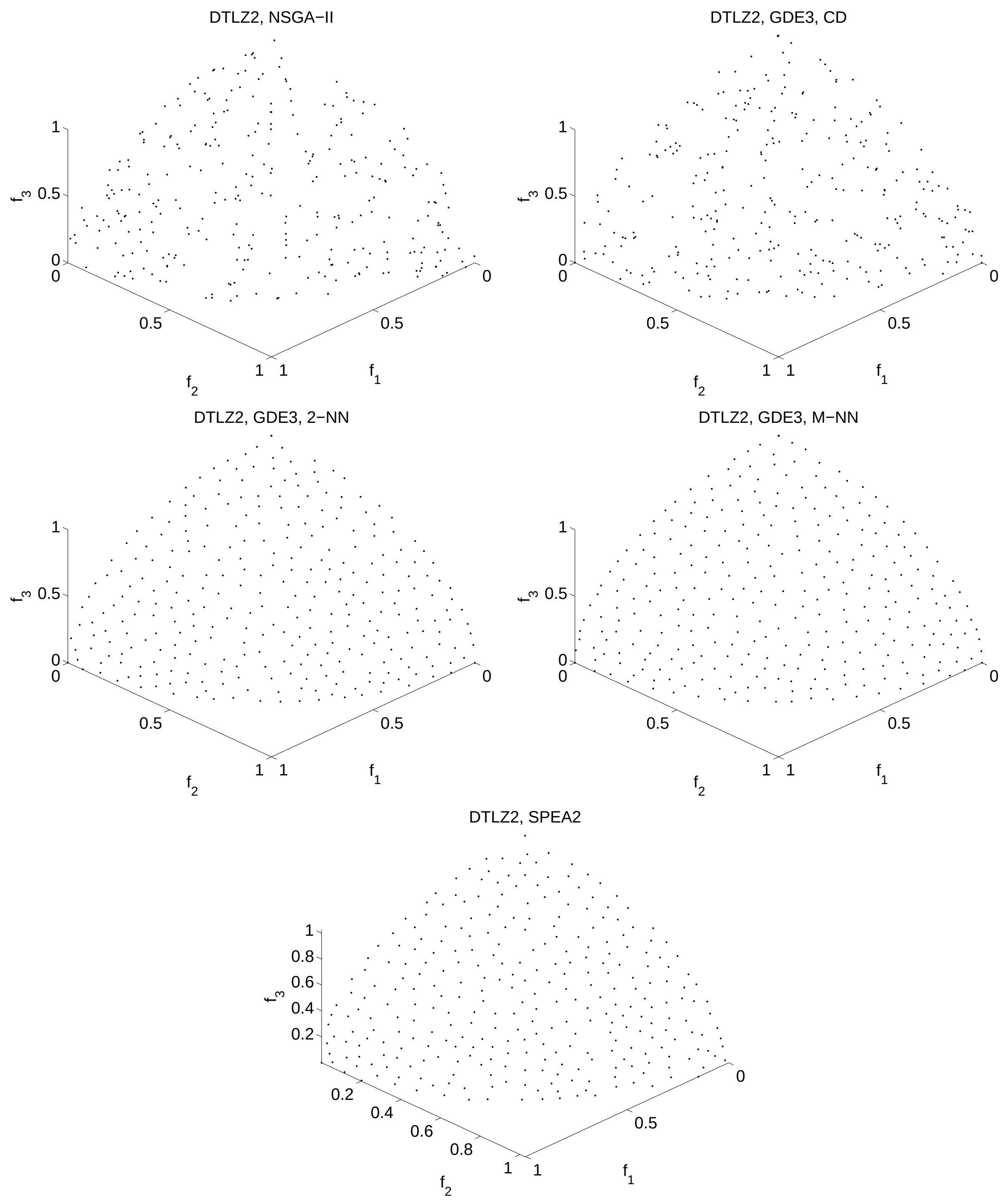

Figure 5: The results for the DTLZ2 problems using NSGA-II and SPEA2, and GDE3 with the pruning methods based on crowding distance (CD) and described crowding estimation techniques (2-NN & M-NN)

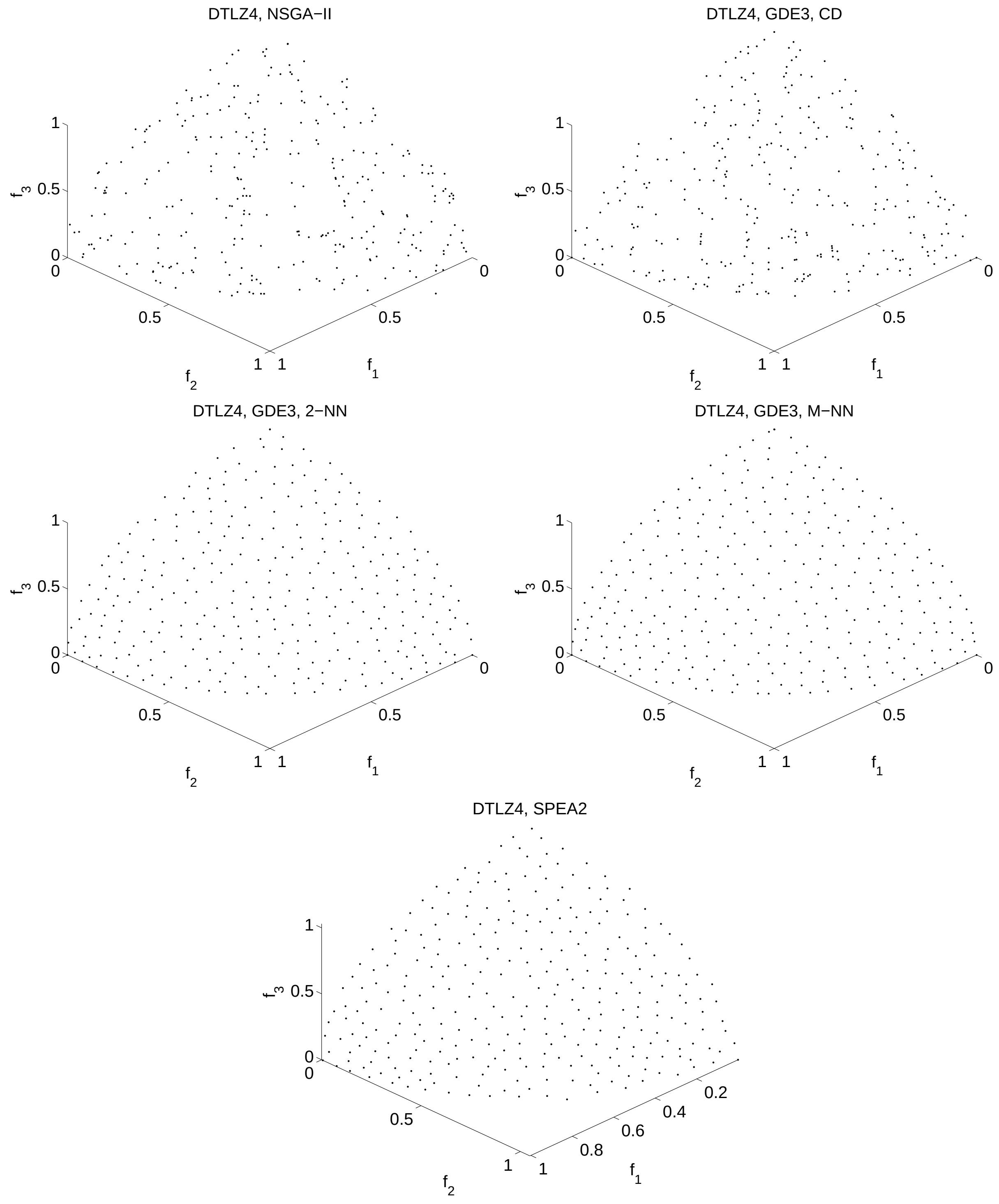

Figure 6: The results for the DTLZ4 problems using NSGA-II and SPEA2, and GDE3 with the pruning methods based on crowding distance (CD) and described crowding estimation techniques (2-NN & M-NN)

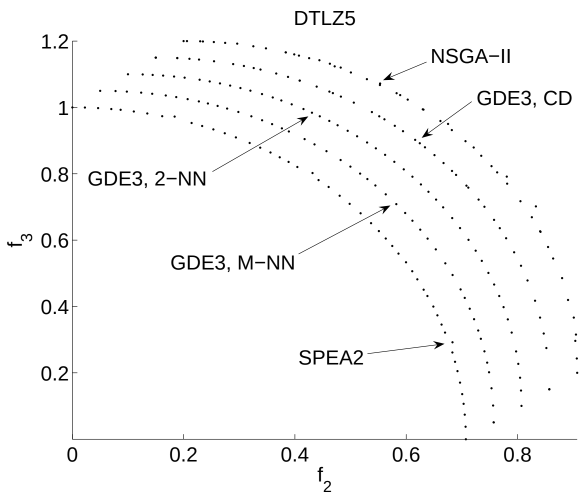

Figure 7: The results for the DTLZ5 problems using NSGA-II and SPEA2, and GDE3 with the pruning methods based on crowding distance (CD) and described crowding estimation techniques (2-NN & M-NN)

the methods are same. Intelligibly, there is no notable difference between the observed pruning times with the 2-NN and M-NN crowding estimation techniques.

In the tri-objective case (DTLZ1), the execution time of the proposed pruning method with the M-NN crowding estimation technique is at worst ten times more than the time with the pruning method based on the crowding distance, and the pruning time with the 2-NN crowding estimation technique is less than with the M-NN crowding estimation technique. It seems that the proposed pruning method scales well with the population size.

Estimation of the actual complexity class would require more problems with different Pareto-fronts and a larger number of objectives. However, the measured pruning times appear almost log-linear and considerably smaller than that of the pruning method in SPEA2 [5].

4 Potential Speedups of The Pruning Method

There exists several potential ways to speed up the new pruning method further. Exact nearest neighbor (NN) search itself could be enhanced. Currently, one projection axis is used in the ENNS method. Because (2) must hold for any projection axis, it would be possible to select several projection axes to different directions and project data to them. This would increase number of pre-calculations but presumably decrease number of search steps in finding NNs in ENNS. Different projection axes could be selected based on monotonicity relation between non-dominated solutions (cf. Sect. 2.2). Besides of projection values, also variances of vectors could be calculated to have a “tighter” inequality instead of (1) [17]. Also other improvements of ENNS exists [18, 19].

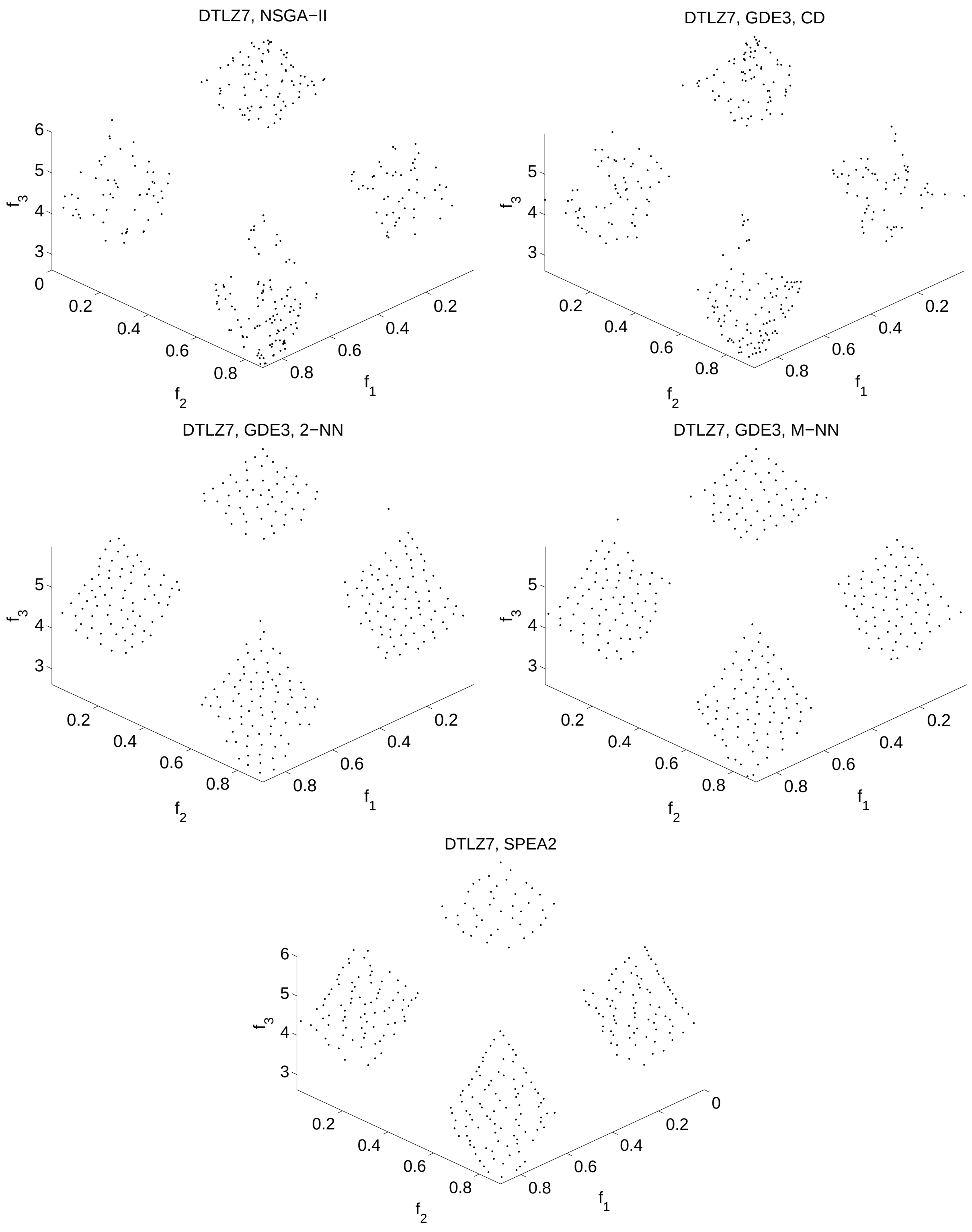

Figure 8: The results for the DTLZ7 problems using NSGA-II and SPEA2, and GDE3 with the pruning methods based on crowding distance (CD) and described crowding estimation techniques (2-NN & M-NN)

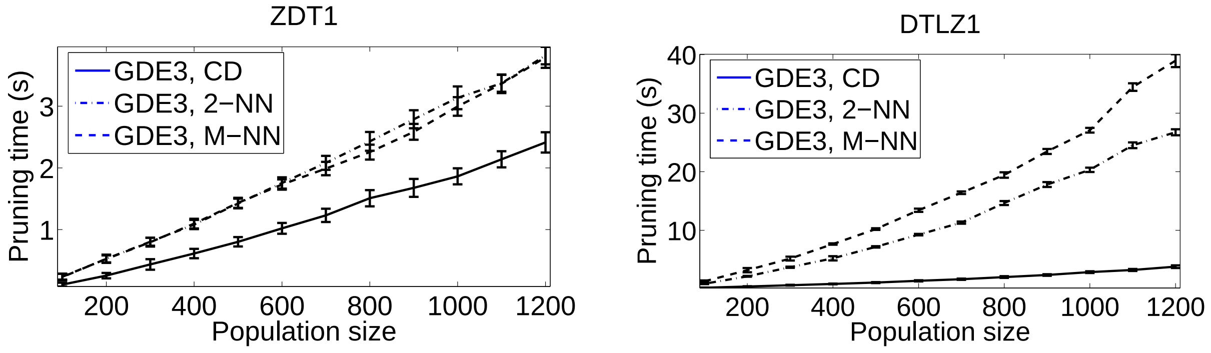

Figure 9: Average and standard error of pruning times measured from 100 runs of solving ZDT1 and DTLZ1 with the pruning methods based on the crowding distance (CD) and described crowding estimation techniques (2-NN & M-NN) implemented in GDE3

The new pruning method performs exact NN search during whole algorithm and the NN search appears to be the bottleneck of the method. An approximate NN technique could reduce calculations without notably declining obtained diversity. A clustering method similar to the new pruning method in this paper, has been recently proposed in [20]. These two methods were developed independently and major difference between them is that the method in [20] uses an approximate NN search. Based on [20], use of an approximate does not decline obtained clustering result considerable but gives notable speedup. Therefore, the algorithm in Sect. 2.3 could be modified to use an approximate NN search, e.g., in line 9, where the NN search is performed for crowding estimation.

5 Conclusions

A pruning method with two different crowding estimation techniques for pruning a set of non-dominated solutions has been proposed. The method is based on crowding estimation using nearest neighbors of solutions and a fast technique for finding the nearest neighbors.

According to the experimental results, the proposed pruning method provides a better diversity than the pruning method based on the crowding distance. Especially in the case of tri-objective problems, the obtained diversity is significantly better. The obtained distribution is observed to be similar to the distribution obtained with SPEA2. The execution time needed for the proposed pruning method is more than for the pruning method based on the crowding distance but significantly less than for the pruning method in SPEA2. Two different crowding estimation techniques provide similar diversity but simpler (the one, which uses two nearest neighbors) is also faster to execute.

Based on the results, the proposed method provides near optimal distribution, which does not need improvement. The execution time might be still reduced even thought it is currently reasonable. Evaluating the performance in the case of more than three objectives, remains as future work.

Acknowledgments

The first author gratefully acknowledges support from the East Finland Graduate School in Computer Science and Engineering (ECSE), the Academy of Finland, Technological Foundation of Finland, and Finnish Cultural Foundation. He also wishes to thank Drs. Fränti, Laumanns, and Kyrki for help and fruitful discussions.

Appendix

Control parameter values

GDE3

Crossover control parameter *CR* of 0.2 and mutation factor F of 0.2 were used for all the problems except ZDT4. For ZDT4, control parameter values $CR = 0.0$ and $F = 0.5$ were used.

NSGA-II

For all the problems probability of crossover was 0.9 and probability of mutation was $1/n$ (n is number of decision variables). SBX mutation (η_m) value was 20 and SBX recombination (η_c) value was 15.

SPEA2

Individual mutation probability was 1.0, individual recombination probability was 1.0, variable mutation probability was $1/n$ (n is number of decision variables), variable swap probability was 0.5, variable recombination probability was 0.5, SBX mutation (η_m) value was 20, and SBX recombination (η_c) value was 15 for all the problems except ZDT2. For ZDT2, variable recombination probability was 1.0.

Nadir points

The nadir point used with all the ZDT problems was {2.0, 2.0}.

The nadir point used with the DTLZ problems was {2.0, 2.0, 2.0} except the nadir point of {1.0, 1.0, 1.0} was used with DTLZ1 and the nadir point of {2.0, 2.0, 7.0} was used with DTLZ7.

References

[1] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Proceedings of the*

Third Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN 2001), pages 95–100, Athens, Greece, Sept 2002.

[2] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[3] Mikkel T. Jensen. Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms. *IEEE Transactions on Evolutionary Computation*, 7(5):503–515, Oct 2003.

[4] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. *Evolutionary Multiobjective Optimization*, chapter Scalable Test Problems for Evolutionary Multiobjective Optimization, pages 105–145. Springer-Verlag, London, 2005.

[5] Saku Kukkonen and Kalyanmoy Deb. Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)*, pages 3995–4002, Vancouver, BC, Canada, July 2006.

[6] Nasreddine Hallam, Peter Blanchfield, and Graham Kendall. Handling diversity in evolutionary multiobjective optimisation. In *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, pages 2233–2240, Edinburgh, Scotland, Sept 2005.

[7] L. Guan and M. Kamel. Equal-average hyperplane partioning method for vector quantization of image data. *Pattern Recognition Letters*, 13(10):693–699, Oct 1992.

[8] S.-W. Ra and J.-K. Kim. A fast mean-distance-ordered partial codebook search algorithm for image vector quantization. *IEEE Transactions on Circuits and Systems-II*, 40(9):576–579, Sept 1993.

[9] SeongJoon Baek and Koeng-Mo Sung. Fast K-nearest-neighbour search algorithm for nonparametric classification. *IEE Electronics Letters*, 36(21):1821–1822, Oct 2000.

[10] Chang-Da Bei and Robert M. Gray. An improvement of the minimum distortion encoding algorithm for vector quantization. *IEEE Transactions on Communications*, COM-33(10):1132–1132, Oct 1985.

[11] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. Prentice-Hall, 1990.

[12] Pravin M. Vaidya. An $O(n \log n)$ algorithm for the all-nearest-neigbors problem. *Discrete & Computational Geometry*, 4:101–115, Dec 1989.

[13] Saku Kukkonen and Jouni Lampinen. GDE3: The third evolution step of Generalized Differential Evolution. In *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, pages 443–450, Edinburgh, Scotland, Sept 2005.

[14] Kenneth V. Price, Rainer Storn, and Jouni Lampinen. *Differential Evolution: A Practical Approach to Global Optimization.* Springer-Verlag, Berlin, 2005.

[15] Antony Iorio and Xiaodong Li. Solving rotated multi-objective optimization problems using Differential Evolution. In *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence (AI 2004)*, pages 861–872, Cairns, Australia, Dec 2004.

[16] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons, Chichester, England, 2001.

[17] SeongJoon Baek, BumKi Jeon, and Koeng-Mo Sung. Fast encoding algorithm for vector quantization. *IEEE Signal Processing Letters*, 4(12):325–327, Dec 1997.

[18] Jeng-Shyang Pan, Zhe-Ming Lu, and Sheng-He Sun. An efficient encoding algorithm for vector quantization based on subvector technique. *IEEE Transactions on Image Processing*, 12(3):265–270, March 2003.

[19] Spiridon Florin Beldianu. A fast nearest neighbor search algorithm for image vector quantizatio. In *Proceedings of the International Symposium on Signals, Circuits & Systems (ISSCS 2005)*, pages 485–488, Iasi, Romania, July 2005.

[20] Pasi Fränti, Olli Virmajoki, and Ville Hautamäki. Fast agglomerative clustering using k-nearest neighbor grapht. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1875–1881, Nov 2006.

Table 1: Mean and standard deviation values of spacing (S), maximum spread (D), hypervolume (HV), and CPU times for ZDT test problems measured from 100 independent runs. GDE3 is implemented with the pruning methods based on the crowding distance (CD) and described crowding estimation techniques (2-NN & M-NN)

Problem	Method	S	D	HV	Total time (s)	Pruning time (s)
ZDT1	NSGA-II	$6.9175e-03 \pm 5.6118e-04$	$\mathbf{1.4146e+00} \pm 2.2506e-03$	$3.6604e+00 \pm 2.3269e-04$	$2.4076e+00 \pm 1.0456e-02$	$1.5600e-01 \pm 3.8006e-02$
	GDE3, CD	$6.4240e-03 \pm 5.5946e-04$	$1.4113e+00 \pm 1.4634e-03$	$3.6610e+00 \pm 1.2739e-03$	$1.4877e+00 \pm 1.3246e-02$	$\mathbf{1.0530e-01} \pm 3.1669e-02$
	GDE3, 2-NN	$2.8501e-03 \pm 2.8597e-04$	$1.4114e+00 \pm 1.5524e-03$	$3.6615e+00 \pm 1.7195e-03$	$\mathbf{1.3560e+00} \pm 6.5134e-03$	$2.3340e-01 \pm 3.6048e-02$
	GDE3, M-NN	$\mathbf{2.6305e-03} \pm 2.8921e-04$	$1.4115e+00 \pm 1.5376e-03$	$3.6616e+00 \pm 1.1364e-03$	$1.3570e+00 \pm 7.4536e-03$	$2.3260e-01 \pm 4.1013e-02$
	SPEA2	$3.2063e-03 \pm 2.9122e-04$	$1.4142e+00 \pm 7.2339e-05$	$\mathbf{3.6619e+00} \pm 3.6100e-05$	$1.1756e+01 \pm 7.0249e-02$	$7.6469e+00 \pm 1.9474e-01$
ZDT2	NSGA-II	$6.9795e-03 \pm 6.7784e-04$	$1.4145e+00 \pm 1.9127e-03$	$3.3274e+00 \pm 2.5553e-04$	$2.3769e+00 \pm 1.0120e-02$	$1.5490e-01 \pm 3.4772e-02$
	GDE3, CD	$6.3904e-03 \pm 5.4882e-04$	$1.4140e+00 \pm 5.4740e-04$	$3.3274e+00 \pm 2.5905e-03$	$1.4718e+00 \pm 6.8048e-03$	$\mathbf{1.0521e-01} \pm 3.1254e-02$
	GDE3, 2-NN	$2.9513e-03 \pm 2.6492e-04$	$1.4141e+00 \pm 3.2235e-04$	$\mathbf{3.3282e+00} \pm 1.4963e-03$	$1.3489e+00 \pm 6.7854e-03$	$2.2688e-01 \pm 3.9345e-02$
	GDE3, M-NN	$\mathbf{2.7640e-03} \pm 2.2573e-04$	$1.4140e+00 \pm 7.4998e-04$	$3.3281e+00 \pm 3.5560e-03$	$\mathbf{1.3466e+00} \pm 6.4609e-03$	$2.3094e-01 \pm 4.2527e-02$
	SPEA2	$3.4023e-03 \pm 3.6818e-04$	$\mathbf{1.4332e+00} \pm 8.6015e-03$	$3.3271e+00 \pm 2.1421e-03$	$1.2291e+01 \pm 9.7998e-02$	$8.1818e+00 \pm 2.2531e-01$
ZDT3	NSGA-II	$4.9342e-03 \pm 4.5888e-04$	$\mathbf{1.9676e+00} \pm 1.0790e-03$	$4.8146e+00 \pm 1.3933e-04$	$2.3584e+00 \pm 9.3980e-03$	$1.5770e-01 \pm 3.7196e-02$
	GDE3, CD	$4.3573e-03 \pm 4.1398e-04$	$1.9639e+00 \pm 1.7943e-03$	$4.8151e+00 \pm 1.0630e-04$	$1.3838e+00 \pm 5.4643e-03$	$\mathbf{8.5400e-02} \pm 2.9074e-02$
	GDE3, 2-NN	$1.7614e-03 \pm 1.9583e-04$	$1.9600e+00 \pm 3.6983e-02$	$4.8117e+00 \pm 3.6699e-02$	$1.2235e+00 \pm 5.9246e-03$	$1.6420e-01 \pm 4.5797e-02$
	GDE3, M-NN	$\mathbf{1.6415e-03} \pm 1.6563e-04$	$1.9639e+00 \pm 1.5796e-03$	$\mathbf{4.8154e+00} \pm 2.5151e-05$	$\mathbf{1.2194e+00} \pm 5.4717e-03$	$1.6510e-01 \pm 4.3797e-02$
	SPEA2	$3.0892e-03 \pm 3.4177e-04$	$1.9673e+00 \pm 1.0494e-04$	$2.8151e+00 \pm 6.1957e-05$	$1.1500e+01 \pm 7.2117e-02$	$7.3613e+00 \pm 2.2574e-01$
ZDT4	NSGA-II	$7.3229e-03 \pm 5.9883e-04$	$\mathbf{1.4144e+00} \pm 1.4268e-04$	$3.6604e+00 \pm 4.7253e-04$	$1.3384e+00 \pm 6.7749e-03$	$1.2450e-01 \pm 2.8333e-02$
	GDE3, CD	$6.0789e-03 \pm 5.5549e-04$	$1.4112e+00 \pm 1.8113e-03$	$3.6613e+00 \pm 1.6689e-04$	$\mathbf{9.5180e-01} \pm 5.7525e-03$	$\mathbf{8.6200e-02} \pm 2.6772e-02$
	GDE3, 2-NN	$2.3983e-03 \pm 3.2217e-04$	$1.4111e+00 \pm 1.6083e-03$	$\mathbf{3.6619e+00} \pm 7.9959e-05$	$9.7260e-01 \pm 7.4698e-03$	$1.5910e-01 \pm 4.3440e-02$
	GDE3, M-NN	$\mathbf{2.2912e-03} \pm 2.9246e-04$	$1.4123e+00 \pm 8.3372e-03$	$3.6589e+00 \pm 3.0332e-02$	$9.6950e-01 \pm 6.0927e-03$	$1.5970e-01 \pm 4.0935e-02$
	SPEA2	$2.8862e-03 \pm 2.9266e-04$	$1.4143e+00 \pm 1.6038e-04$	$3.6614e+00 \pm 5.5024e-04$	$1.0639e+01 \pm 9.3916e-02$	$6.3440e+00 \pm 2.8208e-01$
ZDT6	NSGA-II	$8.3199e-03 \pm 6.6005e-04$	$1.0463e+00 \pm 1.2942e-04$	$2.9204e+00 \pm 2.8917e-04$	$1.3882e+00 \pm 8.2118e-03$	$1.4790e-01 \pm 3.3584e-02$
	GDE3, CD	$6.6047e-03 \pm 6.5536e-04$	$1.1648e+00 \pm 2.5260e-02$	$3.0209e+00 \pm 1.3448e-01$	$\mathbf{1.2147e+00} \pm 5.7753e-03$	$\mathbf{9.6869e-02} \pm 3.4866e-02$
	GDE3, 2-NN	$2.7362e-03 \pm 3.0119e-04$	$1.1656e+00 \pm 2.0350e-02$	$3.0265e+00 \pm 1.0448e-01$	$1.2347e+00 \pm 5.7656e-03$	$2.3490e-01 \pm 3.5971e-02$
	GDE3, M-NN	$\mathbf{2.6386e-03} \pm 2.8531e-04$	$1.1632e+00 \pm 3.1709e-02$	$3.0114e+00 \pm 1.8180e-01$	$1.2309e+00 \pm 6.2109e-03$	$2.3240e-01 \pm 3.6985e-02$
	SPEA2	$3.1010e-03 \pm 3.0937e-04$	$\mathbf{1.1685e+00} \pm 3.6318e-05$	$\mathbf{3.0412e+00} \pm 1.5286e-04$	$1.1662e+01 \pm 5.7654e-02$	$7.5451e+00 \pm 2.0404e-01$

Table 2: Mean and standard deviation values of spacing (S), maximum spread (D), hypervolume (HV), and CPU times for DTLZ test problems measured from 100 independent runs. GDE3 is implemented with the pruning methods based on the crowding distance (CD) and described crowding estimation techniques (2-NN & M-NN)

Problem	Method	S	D	HV	Total time (s)	Pruning time (s)
DTLZ1	NSGA-II	$2.7301e-02 \pm 3.6882e-03$	$8.0296e-01 \pm 4.8071e-02$	$9.6937e-01 \pm 2.0890e-03$	$\mathbf{9.3375e+00} \pm 6.6284e-02$	$6.3347e-01 \pm 1.2919e-01$
	GDE3, CD	$2.3927e-02 \pm 1.1410e-03$	$9.0238e-01 \pm 1.7048e-01$	$9.6748e-01 \pm 3.4917e-02$	$1.3648e+01 \pm 1.1168e-01$	$\mathbf{6.2460e-01} \pm 6.3444e-02$
	GDE3, 2-NN	$1.0591e-02 \pm 7.4208e-04$	$8.8532e-01 \pm 1.2251e-01$	$9.7272e-01 \pm 2.3476e-02$	$1.6237e+01 \pm 1.0560e-01$	$3.5605e+00 \pm 8.7404e-02$
	GDE3, M-NN	$1.1260e-02 \pm 6.8360e-04$	$\mathbf{9.1129e-01} \pm 1.9040e-01$	$9.6780e-01 \pm 3.6437e-02$	$1.7627e+01 \pm 1.6803e-01$	$5.1749e+00 \pm 1.0265e-01$
	SPEA2	$\mathbf{8.7750e-03} \pm 2.0557e-03$	$8.6981e-01 \pm 2.1762e-02$	$\mathbf{9.7622e-01} \pm 3.9209e-05$	$1.2211e+02 \pm 1.2821e-01$	$7.6808e+01 \pm 1.7632e-01$
DTLZ2	NSGA-II	$3.0825e-02 \pm 1.5370e-03$	$1.7372e+00 \pm 7.1500e-03$	$7.3607e+00 \pm 1.9157e-02$	$1.5280e+01 \pm 7.9015e-02$	$7.8230e-01 \pm 6.6816e-02$
	GDE3, CD	$2.8914e-02 \pm 1.3827e-03$	$1.7320e+00 \pm 7.7029e-06$	$7.4234e+00 \pm 2.9540e-03$	$\mathbf{1.3536e+01} \pm 3.3132e-02$	$\mathbf{6.4540e-01} \pm 7.4947e-02$
	GDE3, 2-NN	$1.5029e-02 \pm 8.0864e-04$	$1.7320e+00 \pm 4.9954e-06$	$7.4422e+00 \pm 6.2471e-04$	$1.6183e+01 \pm 9.2480e-02$	$3.7446e+00 \pm 7.2341e-02$
	GDE3, M-NN	$1.5958e-02 \pm 8.2783e-04$	$1.7320e+00 \pm 4.3583e-06$	$\mathbf{7.4433e+00} \pm 3.5137e-04$	$1.7789e+01 \pm 4.0661e-02$	$5.3663e+00 \pm 6.8955e-02$
	SPEA2	$\mathbf{1.2919e-02} \pm 6.9397e-04$	$\mathbf{1.7593e+00} \pm 1.9827e-02$	$7.4311e+00 \pm 2.1360e-03$	$1.3095e+02 \pm 7.7438e-02$	$8.7535e+01 \pm 1.5309e-01$
DTLZ4	NSGA-II	$3.1285e-02 \pm 1.5006e-03$	$1.7385e+00 \pm 1.0485e-02$	$7.3842e+00 \pm 1.5108e-02$	$1.5752e+01 \pm 8.6729e-02$	$7.7610e-01 \pm 6.4790e-02$
	GDE3, CD	$2.8450e-02 \pm 1.2669e-03$	$1.7321e+00 \pm 1.0347e-06$	$7.4252e+00 \pm 2.9245e-03$	$\mathbf{1.3813e+01} \pm 5.8934e-02$	$\mathbf{6.4060e-01} \pm 6.9614e-02$
	GDE3, 2-NN	$1.4207e-02 \pm 1.6561e-03$	$1.7289e+00 \pm 3.1785e-02$	$7.4330e+00 \pm 1.0168e-01$	$1.6918e+01 \pm 4.9857e+00$	$4.1224e+00 \pm 4.9245e+00$
	GDE3, M-NN	$1.5221e-02 \pm 1.1413e-03$	$1.7321e+00 \pm 1.6773e-07$	$\mathbf{7.4437e+00} \pm 2.2990e-04$	$1.8109e+01 \pm 1.1213e-01$	$5.1986e+00 \pm 1.7774e-01$
	SPEA2	$\mathbf{1.2543e-02} \pm 2.4636e-03$	$\mathbf{1.7465e+00} \pm 7.0831e-02$	$7.3918e+00 \pm 1.9807e-01$	$1.3105e+02 \pm 1.0151e+00$	$8.7409e+01 \pm 1.2951e+00$
DTLZ5	NSGA-II	$3.5634e-03 \pm 1.7765e-04$	$\mathbf{1.4143e+00} \pm 2.4335e-04$	$6.1078e+00 \pm 1.1647e-04$	$\mathbf{1.1695e+01} \pm 9.0123e-02$	$6.5980e-01 \pm 6.9949e-02$
	GDE3, CD	$3.4106e-03 \pm 1.7730e-04$	$1.4142e+00 \pm 2.7905e-05$	$6.1084e+00 \pm 1.0564e-04$	$1.2149e+01 \pm 8.9120e-02$	$\mathbf{5.7600e-01} \pm 6.6999e-02$
	GDE3, 2-NN	$1.7253e-03 \pm 1.0064e-04$	$1.4142e+00 \pm 1.7727e-05$	$\mathbf{6.1089e+00} \pm 6.4457e-05$	$1.2531e+01 \pm 7.2325e-02$	$1.1271e+00 \pm 8.1851e-02$
	GDE3, M-NN	$\mathbf{1.6092e-03} \pm 7.8633e-05$	$1.4142e+00 \pm 1.6933e-05$	$\mathbf{6.1089e+00} \pm 5.4523e-05$	$1.2863e+01 \pm 4.7087e-02$	$1.5448e+00 \pm 6.8748e-02$
	SPEA2	$2.3039e-03 \pm 1.1166e-04$	$\mathbf{1.4143e+00} \pm 4.4545e-04$	$6.1087e+00 \pm 5.9630e-05$	$1.2363e+02 \pm 8.3786e-02$	$7.7715e+01 \pm 1.2489e-01$
DTLZ7	NSGA-II	$2.3073e-02 \pm 1.6451e-03$	$3.5925e+00 \pm 4.8045e-02$	$1.3493e+01 \pm 2.4877e-02$	$1.4823e+01 \pm 1.4667e-01$	$7.7070e-01 \pm 1.0215e-01$
	GDE3, CD	$2.3658e-02 \pm 2.6283e-03$	$\mathbf{3.6179e+00} \pm 7.8436e-03$	$1.3545e+01 \pm 1.9480e-02$	$\mathbf{1.2066e+01} \pm 2.2497e-02$	$\mathbf{5.5920e-01} \pm 6.4834e-02$
	GDE3, 2-NN	$\mathbf{9.1687e-03} \pm 1.1766e-03$	$3.6153e+00 \pm 3.0744e-03$	$1.3586e+01 \pm 5.3207e-03$	$1.2809e+01 \pm 2.9450e-01$	$2.2352e+00 \pm 1.0391e-01$
	GDE3, M-NN	$9.7859e-03 \pm 8.8586e-04$	$3.6155e+00 \pm 2.6746e-03$	$\mathbf{1.3588e+01} \pm 3.7791e-03$	$1.3540e+01 \pm 3.3633e-02$	$3.0734e+00 \pm 6.1664e-02$
	SPEA2	$1.5140e-02 \pm 8.2394e-04$	$3.6067e+00 \pm 7.1954e-03$	$1.3577e+01 \pm 1.1132e-02$	$1.3342e+02 \pm 6.7649e-02$	$8.8697e+01 \pm 1.0649e-01$