Which performance indicator is meaningful?

Candidate Number 074936

ABSTRACT

The use of Evolutionary Algorithms (EA) have been popular in solving different types of multi-objective problems (MOP), particularly due to their advantages over classical and gradient-based algorithms. How effective EA are towards different problems, and how we measure their efficiency form the basis of this study. The measures used to quantify effectiveness of multi-objective Evolutionary Algorithms (MOEA) are known as Performance Indicators (PI). PIs are broadly divided into four categories: cardinality-based, convergence-based, distribution-based, and convergence- and distribution-based. How reliable these PIs are to justify the efficacy of MOEAs is a question in itself. This study addresses these questions by comparing and ranking two MOEAs - NSGA-II and SPEA2, by using three PIs, Inverse Generational Distance (IGD), Δ-metric, and Hypervolume indicator. The problems considered to perform this analyses are ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. The results presented show that IGD and hypervolume indicator are very helpful in determining the performance of MOEAs, however, Δ-metric has some limitations. This study concludes with possible causes to these limitations, and a better distribution-focussed PI that could be used towards MOPs.

1 INTRODUCTION

The use of Evolutionary Algorithms (EA) have been popular in solving multi-objective optimisation problems (MOP) due to their population-based approach that enables generation of several elements of the Pareto optimal set in a single run. These are particularly effective in solving complex MOPs with very large spaces, uncertainty, noise, disjoint Pareto curves, etc. Some examples of Multi-objective Evolutionary Algorithms (MOEA) are MOGA, NPGA, NSGA, NSGA-II, PAES, SPEA, SPEA2 and ϵ-MOEA[1, 2]. Selecting an efficient MOEA towards solving MOPs of complex nature is crucial to yield accurate optimal points, especially when there is a computational cost attached to it. Performance Indicators (PI) are metrics that are helpful in comparing and ranking multiple MOEAs to determine their efficiencies. There are several PIs that exist in the literature in ranking MOEA. These are broadly divided as per cardinality, convergence, distribution, and both convergence and distribution. Their description is provided in subsection 2.2. This project explores the correlation between the run time and ranking of two MOEAs using different PIs.

Section 2 describes relevant concepts in this study, including MOEAs, their significance, and their advantages over *classical* MOPs. It further expands on two EAs - NSGA-II and SPEA2, which is further used for performing analysis. Section 3 describes the experimental setup for this study, with problems used, MOEAs used, and performance metrics used. Section 4 lists results obtained from analysis for corresponding run times for each of the problems used. It further comments on results obtained and on which MOEA outperforms the other. Section 5 concludes the study with further work that could potentially be done to extend this study - one of which is to increase dimensionality to observe how the MOEAs perform in higher dimensions.

2 BACKGROUND

2.1 Multi-objective Evolutionary Algorithms

A multi-objective optimisation problem (MOP) has a number of objective functions which are to be minimised or maximised [3]. A MOP is defined in general form as:

$$\begin{aligned} \textbf{Minimize/Maximize } & f_m(\mathbf{x}), && m = 1, 2, \ldots, M; \\ \textbf{Subject to } & g_j(\mathbf{x}) \geq 0, && j = 1, 2, \ldots, J; \\ & h_k = 0, && k = 1, 2, \ldots, K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, && i = 1, 2, \ldots, n \end{aligned}$$

A solution $\mathbf{x}$ is a vector of n decision variables: $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$. The last set of constraints, $x_i^{(L)} \leq x_i \leq x_i^{(U)}$, are called variable bounds, restricting each decision variables x_i to take a value with a lower $x_i^{(L)}$ and an upper $x_i^{(U)}$ bound, constituting of the design space $\mathcal{D}$ [2]. "Classical" multi-objective algorithms, such as direct methods and gradient-based methods, are used in solving simple, convex optimisation problems.

Deb [2] lists the challenges that "classical" multi-objective algorithms are faced with -

(1) The convergence to an optimal solution depends on the chosen initial solution.
(2) Most algorithms tend to get stuck to a sub-optimal solution.
(3) An algorithm that is efficient in solving one optimisation problem might prove inefficient in solving a different problem.
(4) Algorithms are not efficient in handling problems having a discrete search space.
(5) Algorithms cannot be efficiently used on a parallel machine.

In such cases, Evolutionary Algorithms prove to be effective in finding optimum solutions to different multi-objective optimisation problems, often simulating complex, real-world design problems. They are advantageous over classical algorithms in the following way [4]:

(1) EAs adopt population-based search, i.e., it generates more than one solution in an iteration. This provides parallel processing power, that helps in achieving a computationally quick overall search. It also finds multiple optimal solutions, thereby facilitating the solution of a multi-modal and multi-objective optimisation problem.
(2) EAs adopt a stochastic approach rather than a deterministic approach, yielding to desired outcomes by using biased probability distributions.
(3) EAs do not usually use gradient information in their search process. Their approach is more direct, enabling them to be used to a wide variety of optimisation problems.

Problem type	Number of objectives	Number of variables	Problem description
ZDT1	2	30	Continuous, unconstrained, convex Pareto front
ZDT2	2	30	Continuous, unconstrained, non-convex Pareto front
ZDT3	2	30	Continuous, unconstrained, disconnected Pareto front
ZDT4	2	30	Continuous, unconstrained, convex Pareto front
ZDT6	2	30	Non-uniform, unconstrained, non-convex Pareto front

Table 1: Description of problems considered [5]

EAs are based on the concept of biological evolution. A *population* of possible solutions to the problem; This population evolves over time and identifies better solutions [6]. The evolutionary algorithm searches for good solutions in the search space using this typical structure [7]:

(1) **Initialisation:** Randomly generate a population of samples from the search space.
(2) **Iteration:**
    (a) **Evaluation:** Compute the value of the objective function for each sample.
    (b) **Selection operator:** Use the values of objective function computed for the evaluated samples to select the samples to be used in the next step 2(c).
    (c) **Variation operators:** Apply variation operators to the selected samples to transform them into additional samples from the search space.
(3) **Termination:** If the termination criteria are met, halt the computation; if not, return to step 2(a).

The two MOEAs considered in this study are NSGA-II and SPEA2.

2.1.1 NSGA-II. The first version of Non-dominated Sorting Genetic Algorithm (NSGA) [8] considered the following operators:

- Assigning fitness to population members based on non-dominated sorting;
- Preserving diversity among solutions of the same non-dominated front.

NSGA-II [2], on the other hand, was more elitism-focussed, wherein the non-dominated members are assigned in the first iteration, however, from the following iterations, parent and offspring populations are combined, and then the iteration continues by incorporating selection, crossover, and mutation. The steps in the algorithm is defined by Deb *et al.* [2] as follows:

(1) Combine parent and offspring population.
(2) Sort all $\mathcal{F} = \{F_1, F_2, ...\}$ non-dominated fronts of R_t.
(3) Until the parent population is filled, calculate crowding distance and then update the non-dominated front.
(4) Use selection, mutation, and crossover to create a new population, Q_{t+1}.
(5) Repeat steps 1-4 until termination criteria is met.

The crowding distance from step (3) ensures diversity preservation. The crowding-distance computation requires sorting the population according to each objective function value in ascending order of magnitude. Thereafter, for each objective function, the boundary solutions (solutions with smallest and largest function values) are assigned an infinite distance value. All other intermediate solutions are assigned a distance value equal to the absolute normalized difference in the function values of two adjacent solutions. This calculation is continued with other objective functions. The overall crowding-distance value is calculated as the sum of individual distance values corresponding to each objective. Each objective function is normalized before calculating the crowding distance [2].

2.1.2 SPEA2. In the attempt to improving the original Strength Pareto Evolutionary Algorithm, Zitzler *et al.* [9] introduce SPEA2, where they claim that the EA is advantageous over its first version in the following way:

- An improved fitness assignment scheme is used, which takes for each individual into account how many individuals it dominates and it is dominated by.
- A nearest neighbor density estimation technique is incorporated which allows a more precise guidance of the search process.
- A new archive truncation methods guarantees the preservation of boundary solutions.

Zitzler *et al.* [9] define the structure of SPEA2 as follows:

(1) **Initialisation:** Generate an initial population P_0 and create the empty archive (external set) $\bar{P}_0 = \emptyset$. Set $t = 0$.
(2) **Fitness Assignment:** Calculate the fitness values P_t and $\bar{P}_t$.
(3) **Environmental selection:** Copy all nondominated individuals in P_t and $\bar{P}_t$ to P_{t+1}. If size of $\bar{P}_{t+1}$ exceeds $\bar{N}$ then reduce to a size of $\bar{N}$; If size of $\bar{P}_{t+1}$ is less than $\bar{N}$ then fill P_{t+1} with dominated individuals in P_t and $\bar{P}_t$.
(4) **Termination:** If $t \geq T$ or another stopping criteria, set $\mathbf{A}$ to the set of decision vectors represented by the nondominated individuals in $\bar{P}_{t+1}$. Stop.
(5) **Mating selection:** Perform binary tournament selection with replacement on $\bar{P}_{t+1}$ in order to fill the mating pool.
(6) **Variation:** Apply recombination and mutation operators to the mating pool and set $\bar{P}_{t+1}$ to the resulting population. Increment generation counter ($t = t + 1$) and go to step (2).

Any variable with a bar on top pertains to the archive (external set). N is the population size, $\bar{N}$ is the archive size, T is the maximum number of generations (evaluation size), which is used as a stopping/termination criteria, and $\mathbf{A}$ is the nondominated set, which is transferred to the archive (external set).

2.2 Performance Indicators

Performance Indicators for MOEAs are broadly categorised into four aspects [10]:

(1) **Cardinality:** Cardinality in this context is the number of non-dominated points that are generated by the MOEA.
(2) **Convergence:** This aspect quantifies how close a set of non-dominated points is from the Pareto front.
(3) **Distribution and spread:** This aspect is divided into two sub-groups: First one is how well-distributed the non-dominated points are on the Pareto front; Second one is the extent of Pareto front approximation and whether the Pareto front contains extreme points. This aspect would account for how diverse the non-dominated points are.
(4) **Convergence and distribution:** This aspect considers both the convergence and distribution of the set of non-dominated points.

There are several performance indicators described in the literature by various authors and their application towards MOPs. These are extensively described by Audet *et al.* [10].

3 METHODOLOGY AND EXPERIMENTAL DESIGN

3.1 Problems used

There are a range of problems used in this study, extracted from the *jMetalPy* [11] Python module, namely ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 to perform analysis. It was attempted to run simulations on DTLZ problems, however, as it was very time-consuming to run a single simulation of a run time as low as 10,000 evaluations and would not have yielded to a visual inspection of set of solutions due to an increase in dimensionality, DTLZ problems are regarded as out-of-scope for this study. The ZDT problem types are, however, diverse in nature, and their characteristics are tabulated below.

3.2 MOEAs used

The two MOEAs used for analysis is Non-dominated Sorting Genetic Algorithm (NSGA-II) and Strength Pareto Evolutionary Algorithm 2 (SPEA2). Their characteristics were described in subsubsection 2.1.1 subsubsection 2.1.2, respectively. The NSGA-II and SPEA2 methods are pre-defined in the *jMetalPy* Python module, and were utilised with the following arguments to perform analysis [11].

- Population Size: 100
- Offspring population size: 100
- Mutation: Polynomial mutation
- Crossover: SBX
- Probability for mutation and crossover: $\frac{1.0}{Number of variables}$
- Distribution indexes for mutation and crossover: 20.0
- Termination criteria: $\{4000, 8000, 12000, 16000\}$ unless stated otherwise

The run times / termination criteria for algorithms were chosen as per what run times demonstrate worst results and to run times that achieve convergence to true Pareto optimality.

3.3 Performance Indicators

The three performance indicators used for analysis are IGD (convergence- focussed), Δ-metric (diversity- and distribution-focussed), and hypervolume indicator (convergence- and distribution- focussed).

3.3.1 IGD. Pymoo [5] contains a pre-defined function to calculate IGD with respect to the true Pareto front, and outputs the measure when compared to each of the algorithm-generated set of solutions in the form of their respective Pareto fronts.
This metric aims to indicate that lower the magnitude, closer it is to the True Pareto, and therefore, the algorithm was able to find an optimum set of solutions. Therefore, if $IGD = 0.00$, the set of solutions have achieve true Pareto optimality.

3.3.2 Δ-metric. The Δ-metric is not defined in any of the MOP libraries present in Python, so this was defined as[12, 13]:

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

where, d_l and d_f are distances between two extreme points to their corresponding points from the set of solutions generated by MOEAs, with N number of solutions, and $\bar{d}$ is the mean distance of all the distance between the set of solutions.
This metric aims to indicate that lower the magnitude, more diverse the solutions. The most diverse possible magnitude for Δ-metric is 0.00.

3.3.3 Hypervolume Indicator. Pymoo [5] contains a pre-defined function to calculate this metric with respect to the true Pareto front. This metric does not necessarily look to minimise or maximise the magnitude to provide insights; Instead, these magnitudes calculated by the set of solutions by MOEAs is compared to the Hypervolume indicator of the true Pareto front. This is because the reference point, which is a factor to calculate the hypervolume indicator, which chosen by the user. In this study, the reference point is chosen to be the worst possible solution that could be possible, i.e., the Nadir point.

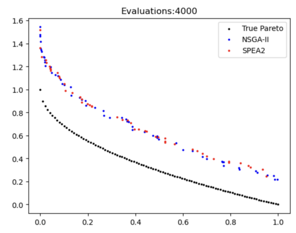
4 RESULTS

4.1 ZDT1

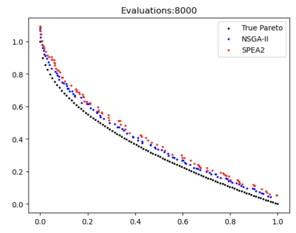
For the run times $\{4000, 8000, 12000, 16000\}$, it can be observed that whilst both MOEAs do not represent accurate set of solutions for the first one, NSGA-II outperforms SPEA2 for other run times. In addition to this, the IGD- and Δ-metric consistently reduces down to $\approx$ 0.00 when evaluations increase, whilst the Hypervolume indicator reaches the measure close to the true Pareto front. Other than these, there were no extreme insights and the metrics demonstrated magnitudes which were well expected.

4.2 ZDT2

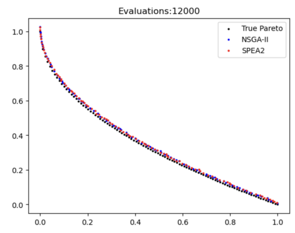
For the same run times used as for ZDT1, the first one starts off with the worst performance, where both MOEAs generated scattered randomly, thus the high Δ value. However, for the next two run times, 8000, 12000, the number of solutions generated by NSGA-II (31) are much less than SPEA2 (73); Despite of this, the Δ-metric is not that drastic for NSGA-II when compared to SPEA2 - this



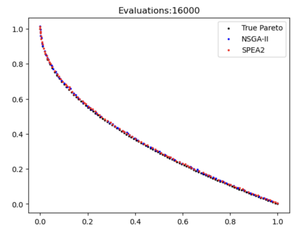
(a) ZDT1:4000



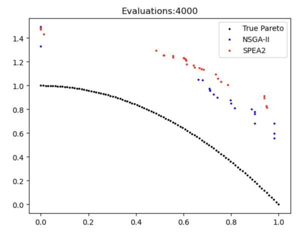
(b) ZDT1:8000



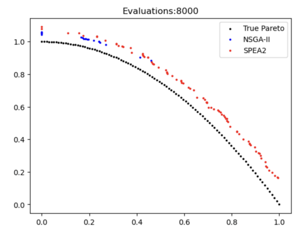
(c) ZDT1:12000



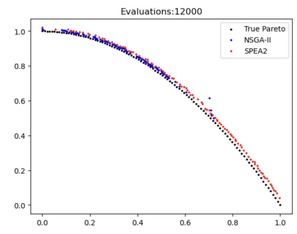
(d) ZDT1:16000



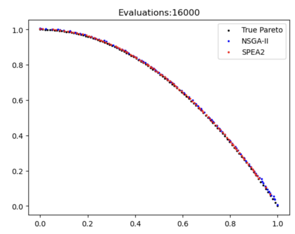
(e) ZDT2:4000



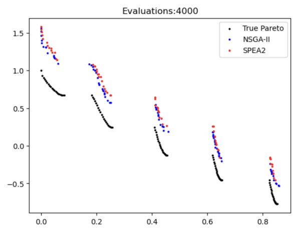
(f) ZDT2:8000



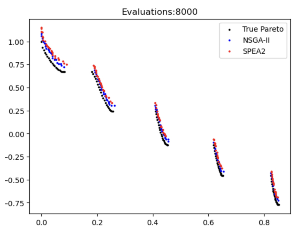
(g) ZDT2:12000



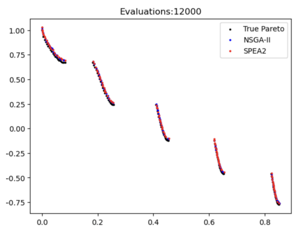
(h) ZDT2:16000



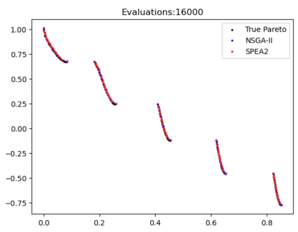
(i) ZDT3:4000



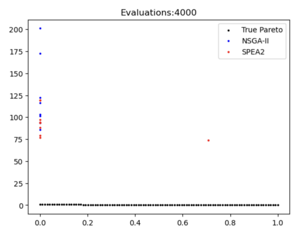
(j) ZDT3:8000



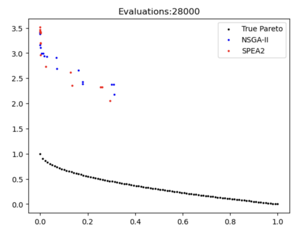
(k) ZDT3:12000



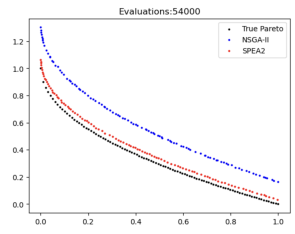
(l) ZDT3:16000



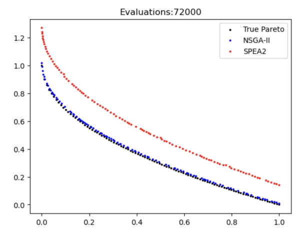
(m) ZDT4:4000



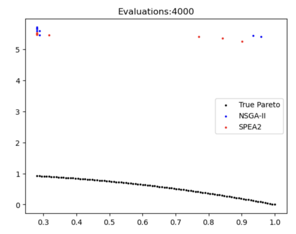
(n) ZDT4:28000



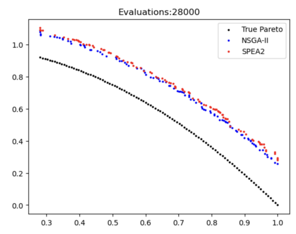
(o) ZDT4:54000



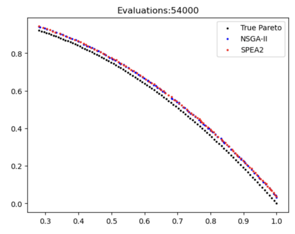
(p) ZDT4:72000



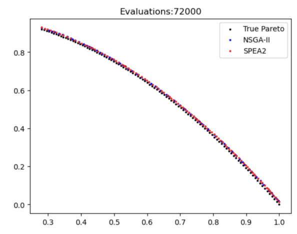
(q) ZDT6:4000



(r) ZDT6:28000



(s) ZDT6:54000



(t) ZDT6:72000

Figure 1: For the above simulations, the problem name and number of evaluations are given as graph titles. The blue points represent set of solutions for NSGA-II, and the red points represent set of solutions for SPEA2, while the black points represent the true Pareto front

Problem	Evaluations	IGD		Δ-metric		Hypervolume Indicator			Number of solutions	
		NSGA-II	SPEA2	NSGA-II	SPEA2	NSGA-II	SPEA2	True Pareto	NSGA-II	SPEA2
ZDT1	4000	0.171	0.1477	0.046	0.048	0.731	0.769	0.992	57	44
	8000	0.028	0.025	0.029	0.023	0.674	0.677	0.714	100	100
	12000	0.009	0.008	0.05	0.045	0.663	0.664	0.674	100	100
	16000	0.006	0.006	0.006	0.006	0.67	0.67	0.675	100	100
ZDT2	4000	0.394	0.449	0.108	0.098	0.418	0.345	0.938	23	19
	8000	0.045	0.049	0.027	0.023	0.326	0.322	0.392	31	73
	12000	0.088	0.014	0.02	0.022	0.125	0.125	0.137	54	100
	16000	0.006	0.006	0.029	0.022	0.324	0.325	0.331	100	100
ZDT3	4000	0.16	0.176	0.058	0.056	0.964	0.939	1.28	44	74
	8000	0.02	0.033	0.056	0.068	0.809	0.784	0.853	100	100
	12000	0.009	0.007	0.036	0.041	0.788	0.795	0.807	100	100
	16000	0.005	0.005	0.037	0.035	0.776	0.777	0.782	100	100
ZDT4	4000	96.05	62.62	0.24	0.354	7.545	1.042	1.461	6	8
	28000	1.956	1.937	0.047	0.04	1.756	1.369	3.869	12	17
	54000	0.0813	0.095	0.123	0.0227	0.51	0.527	0.619	100	100
	72000	0.01	0.135	0.039	0.023	0.918	0.745	0.93	100	100
ZDT6	4000	4.155	4.399	0.122	0.25	0.13	0.157	0.33	11	6
	28000	0.138	0.151	0.0209	0.0217	0.235	0.222	0.386	100	100
	54000	0.017	0.018	0.0208	0.0207	0.2613	0.2606	0.2816	100	100
	72000	0.006	0.007	0.021	0.021	0.263	0.263	0.27	100	100

Table 2: Results from running simulations for different number of evaluations (run time)

potentially could be a discrepancy on the application of this metric towards this type of a problem. For this problem, SPEA2 seems to outperform NSGA-II.

4.3 ZDT3

For the same run times used as for ZDT1, for the first run, it seems that the number of solutions generated by NSGA-II is markedly less than SPEA2. Despite of this, the Δ-metric is almost identical for both MOEAs. Given the well-spread nature of this problem, the diversity of solutions for NSGA-II is questionable, considering the number of solutions generated. This confirms that Δ-metric may not seem that meaningful to interpret results, and one has to consider other metrics such as visual inspection and number of solutions generated. On the other hand, the two metrics, IGD and Hypervolume, seem to be helpful in interpreting the effectiveness of MOEAs towards this problem - the first run presenting worst results, and consistently improves with higher run time.

4.4 ZDT4

The run times considered are $\{4000, 28000, 54000, 72000\}$ for the ZDT4 problem. The first run generates very few solutions, not remotely close to the true Pareto front. For the second run, the solutions are scattered, yet far from the true Pareto front - and therefore, IGD is significantly high. Surprisingly, the Δ-metric is 0.04, which theoretically means that the solutions generated are good. However, considering figure 1(n), such solutions cannot be said to be well-spread and diverse. Again, it could be said that Δ-metric is not meaningful in this problem. On hypervolume indicator, NSGA-II is closer to true Pareto front, and as observed from visual representation, NSGA-II significantly outperforms SPEA2. Additionally, the SPEA2 seems to be presenting the same pareto front between 54000 to 72000, reaching its optimum convergence in finding solutions, irrespective of any run time beyond 54000.

4.5 ZDT6

The run times considered are the same as ZDT4. As expected, both MOEAs perform poorly for the first run. For the other three runs, the results consistently improve, and these are apparent with all three PIs, IGD, Δ-metric, and hypervolume indicator. For this problem, NSGA-II starts off with a better performance (as observed in Figure 1(r)), but for higher run times, both MOEAs perform equivalently well.

5 DISCUSSION AND FUTURE WORK

It is apparent that IGD and Hypervolume indicator requires the true Pareto front to be known, as this is used as a reference to compare efficiency of MOEAs. The diversity metric, Δ-metric, is the only exception, where diversity of solutions could be assessed without the need for having any information on the true Pareto front. However, it comes with limitations. Yan *et al.* [13] rightly point this discrepancy of using Δ-metric towards assessing diversity of solutions, which was also recognised when performing analysis. Δ-metric did not seem effective in problems ZDT2, ZDT3, and ZDT4, where the magnitude represented promising result, however, this was not the case when visual representations were considered. One of the reasons of this inaccuracy could be due to the fact that only extreme points and the distances within them are considered (as per the formula), with no reference to the true Pareto front, the solutions between extreme points might

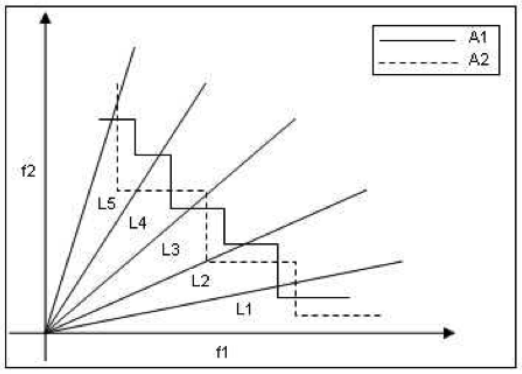


Figure 2: Use of attainment surfaces: *A1* and *A2* are two Pareto fronts. Multiple surfaces (beyond two-dimensions)/ lines (for two-dimensions) are drawn from the origin $(0.00, ..., 0.00)_n$ and distances of the two fronts are measured [14]

be well-spread and uniform, however, they would not represent diverse solutions in context to the true Pareto front. Cai *et al.* [12] present a diversity metric that they claim is more efficient than the Δ-metric. They employ a *coverage vector*, where different angles are drawn from the origin $(0.00, ..., 0.00)_n$, where n is the number of dimensions, which is then incorporated to the DIR-metric. The DIR-metric accounts for both uniformity and coverage of solutions. A similar PI that considers both distribution and convergence is the use of attainment surfaces, where lines from different angles are drawn from the origin, and the distances between multiple Pareto fronts are compared in ranking different MOEAs [14]. Other convergence- and divergence-based PIs might also be effective in this case.

Whilst IGD and hypervolume indicator seem meaningful in presenting results across ZDT problems considered, it would be interesting to analyse whether this accuracy stays true for higher dimensional problems, such as DTLZ. This could potentially be a future work to develop on this study further.

Additionally, it is observed that for ZDT2 with run times 8000 and 12000, the solutions for NSGA-II are not that diverse and the number of solutions is much less than expected, whilst SPEA2 performs as expected. Although this has just occurred for 2 of 20 simulations performed, which might show that such situations are rare, this is unusual behaviour for NSGA-II as the algorithm explicitly accounts for diversity preservation (see section 2.1.1). By running further simulations for run times with an increment of 1000 from the range (4000, 12000), it might be more apparent to see whether this diversity preservation issues is demonstrated for all those simulations.

REFERENCES

[1] Carlos Coello, David Veldhuizen, and Gary Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems Second Edition.* 01 2007.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[3] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms.* Wiley-Interscience series in systems and optimization. Wiley, 2001.

[4] Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowinski, editors. *Multiobjective Optimization, Interactive and Evolutionary Approaches [outcome of Dagstuhl seminars]*, volume 5252 of *Lecture Notes in Computer Science*. Springer, 2008.

[5] Julian Blank. Pymoo: Multi-objective optimization in python, 2022. Accessed on July 12, 2023.

[6] A.R. Leach. 4.05 - ligand-based approaches: Core molecular modeling. In John B. Taylor and David J. Triggle, editors, *Comprehensive Medicinal Chemistry II*, pages 87–118. Elsevier, Oxford, 2007.

[7] L. Altenberg. Evolutionary computation. In Richard M. Kliman, editor, *Encyclopedia of Evolutionary Biology*, pages 40–47. Academic Press, Oxford, 2016.

[8] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[9] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. 2001.

[10] Charles Audet, Jean Bigeon, Dominique Cartier, Sébastien Le Digabel, and Ludovic Salomon. Performance indicators in multiobjective optimization. *Eur. J. Oper. Res.*, 292(2):397–422, 2021.

[11] Antonio Benítez-Hidalgo. jmetalpy, 2019. Accessed on July 11, 2023.

[12] Xinye Cai, Haoran Sun, and Zhun Fan. A diversity indicator based on reference vectors for many-objective optimization. *Information Sciences*, 430-431:467–486, 2018.

[13] Jingyu Yan, Chongguo Li, Zhi Wang, Lei Deng, and Demin Sun. Diversity metrics in multi-objective optimization: Review and perspective. In *2007 IEEE International Conference on Integration Technology*, pages 553–557, 2007.

[14] Lam T. Bui, Hussein A. Abbass, and Daryl Essam. Fitness inheritance for noisy evolutionary multi-objective optimization. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, GECCO '05, page 779–785, New York, NY, USA, 2005. Association for Computing Machinery.

APPENDIX

Code

```
############################################################
This code pertains for both 2D and 3D problems. As 3D
problems took a very long time, these were not used towards
the report. They were experimented upon, nevertheless.
############################################################

# Pymoo for visualising True Pareto
from pymoo.problems import get_problem
from pymoo.util.ref_dirs import get_reference_directions

# jMetalPy for importing MOEAs
from jMetalPy.jmetal.algorithm.multiobjective.nsgaii import NSGAII
from jMetalPy.jmetal.algorithm.multiobjective.spea2 import SPEA2

# jMetalPy operators and utilities libraries for performing MOEA analysis
from jMetalPy.jmetal.operator import SBXCrossover, PolynomialMutation
from jMetalPy.jmetal.problem.multiobjective.zdt import ZDT1, ZDT2, ZDT3, ZDT4, ZDT6
from jMetalPy.jmetal.problem.multiobjective.dtlz import DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5, DTLZ6, DTLZ7
from jMetalPy.jmetal.util.termination_criterion import StoppingByEvaluations
from jMetalPy.jmetal.util.solution import get_non_dominated_solutions

# NumPy and Matplotlib for plotting
import matplotlib.pyplot as plt
import numpy as np

# SciPy and Pymoo for Performance Indicators
from scipy.spatial.distance import cdist
from pymoo.indicators.igd import IGD
from pymoo.indicators.hv import HV

# Importing a specific problem for analysis -- Replace 'zdt4' and ZDT4 to your choice
p = get_problem('zdt4')
#ref_dirs = get_reference_directions('das-dennis', n_dim=3, n_partitions=13)  ## Only used for 3D problems
pareto = p.pareto_front()
problem = ZDT4(number_of_variables=30)

# MOEAs to use towards the problem
max_evaluations = 72e3    # Run time
algorithm1 = NSGAII(
    problem=problem,
    population_size=100,
    offspring_population_size=100,
    mutation=PolynomialMutation(probability= 1.0/30.0, distribution_index=20),
    crossover=SBXCrossover(probability=1.0, distribution_index=20),
    termination_criterion=StoppingByEvaluations(max_evaluations)
)
algorithm2 = SPEA2(
    problem=problem,
    population_size=100,
    offspring_population_size=100,
    mutation=PolynomialMutation(probability= 1.0/30.0, distribution_index=20),
    crossover=SBXCrossover(probability=1.0, distribution_index=20),
    termination_criterion=StoppingByEvaluations(max_evaluations)
)
```

```
algorithm1.run()
algorithm2.run()
solutions1 = algorithm1.get_result()
solutions2 = algorithm2.get_result()

# Retrieve Pareto fronts for NSGA-II and SPEA2
front1 = get_non_dominated_solutions(solutions1)
front2 = get_non_dominated_solutions(solutions2)
nsgaii_front = Plot.get_points(front1)
spea2_front = Plot.get_points(front2)

# Print IGD measures for both MOEA
ind = IGD(pareto)
nsgaii_front = np.array(nsgaii_front)
spea2_front = np.array(spea2_front)
print("IGD [NSGA-II]:", ind(nsgaii_front))
print("IGD [SPEA2]:", ind(spea2_front))

# Print delta-metric for both MOEA
def delta_metric(algo_front, true_front):
    n = len(algo_front)
    distances = cdist(algo_front, true_front, metric='euclidean')
    closest_distances = np.min(distances, axis=1)
    df = np.min(closest_distances)
    dl = np.max(closest_distances)
    d_avg = np.mean(closest_distances)
    delta = (df + dl + np.sum(closest_distances - d_avg)) / (df + dl + (n - 1) * d_avg)
    return delta

nsgaii_front = np.array(nsgaii_front)
spea2_front = np.array(spea2_front)
delta1 = delta_metric(nsgaii_front, pareto)
delta2 = delta_metric(spea2_front, pareto)
print('Delta [NSGA-II]:', delta1)
print('Delta [SPEA2]:', delta2)

# Print Hypervolume metric for both MOEA
nsgaii_front = np.array(nsgaii_front)
spea2_front = np.array(spea2_front)

if delta1 > delta2:
    ref_point = np.max(spea2_front, axis=0)
else:
    ref_point = np.max(nsgaii_front, axis=0)

ind = HV(ref_point=ref_point)

print('HV [NSGA-II]:', ind(nsgaii_front))
print('HV [SPEA2]:', ind(spea2_front))
print('True_HV', ind(np.array(pareto)))

# Plot MOEA solutions with reference to true Pareto
if len(nsgaii_front[0]) == 3:
    fig = plt.figure(figsize = (10, 7))
    ax = plt.axes(projection ="3d")
    ax.scatter3D(*zip(*pareto), c='k', s=5, label='True Pareto')
    ax.scatter3D(*zip(*nsgaii_front), c='b', s=5, label='NSGA-II')
    ax.scatter3D(*zip(*spea2_front), c='r', s=5, label='SPEA2')
elif len(nsgaii_front[0]) == 2:
    plt.scatter(*zip(*pareto), c='k', s=3, label='True Pareto')
    plt.scatter(*zip(*nsgaii_front), c='b', s=3, label='NSGA-II')
    plt.scatter(*zip(*spea2_front), c='r', s=3, label='SPEA2')

plt.title(f'Evaluations:%d' %max_evaluations)
plt.legend()
plt.show()
```