On the Integrity of Performance Comparison for Evolutionary Multi-objective Optimisation Algorithms

Kevin Wilson and Shahin Rostami

Computational Intelligence Research Initiative (CIRI),
Bournemouth University, BH12 5BB,
srostami@bournemouth.ac.uk,
WWW home page: http://research.bournemouth.ac.uk/project/ciri

Abstract. This paper proposes the notion that the experimental results and performance analyses of newly developed algorithms in the field of multi-objective optimisation may not offer sufficient integrity for hypothesis testing. This is demonstrated through the multiple comparison of three implementations of the popular Non-dominated Sorting Genetic Algorithm II (NSGA-II) from well-regarded frameworks using the hypervolume indicator. The results show that of the thirty considered comparison cases, only four indicate that there was no significant difference between the performance of either implementation.

Keywords: Evolutionary Algorithms, Genetic Algorithms, Optimisation, Hypervolume indicator

1 Introduction

Evolutionary Multi-objective Optimisation (EMO) algorithms are well suited to solving complex real-world problems. One strength of EMO is the ability to produce a set of multiple trade-off solutions to a multi-objective problem within a single algorithm execution. This set is referred to as an approximation-set, and is considered to be an approximation of the theoretical true optimum (Pareto-optimal front) of a problem. When proposing a new EMO algorithm, it is often expected that the proposed algorithm is applied to a multi-objective test suite according to some experimental design. The approximation-sets generated from these experiments are then subjected to some performance metrics and compared to popular or state-of-the-art algorithms as a benchmark. The typical characteristics for assessing the quality of an approximation-set have been illustrated in Fig. 1 and have been listed in the following [20]:

- **Proximity**: the distance between the approximation-set and the Pareto-optimal front.
- **Diversity**: the uniformity and extent of the distribution of solutions within the approximation-set.

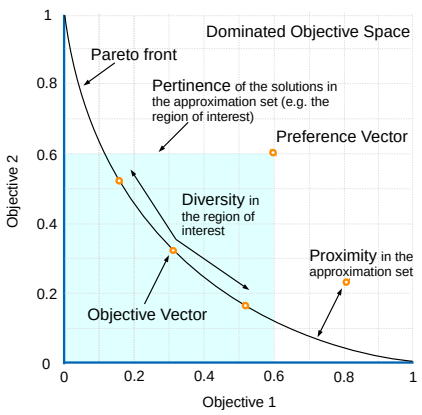

Fig. 1. Characteristics of an approximation-set in bi-objective space.

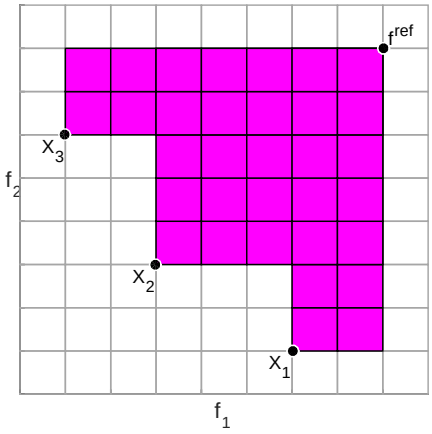

Fig. 2. An example of the hypervolume indicator in bi-objective space

– **Pertinence**: the relevance of the approximation-set in the presence of a Decision Maker (DM) preferences.

Many performance metrics exist for assessing approximation-sets against these desirable characteristics [32], and these have been employed in the comparison of EMO algorithms throughout the literature [17]. These pairwise or multiple comparisons resolve whether the proposed algorithm outperforms the benchmark algorithms in respect to desired the desired characteristics on specific test-cases. This approach allows an algorithm's comparative performance to be demonstrated, and allows the verification of hypotheses and contribution.

One popular algorithm which is often used as a benchmark in the EMO literature [14] is the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [4]. The NSGA-II, described in section 2.2, is a powerful EMO algorithm which employs elitism, non-dominated sorting, and diversity preservation in order to produce approximation sets with good proximity and diversity.

An implementation of the NSGA-II algorithm is required to generate results for comparison. The solution is to either implement NSGA-II based on details available in the related publication, or to acquire an existing implementation. Depending on the publication, it can be time-consuming or difficult to implement an EMO algorithm, e.g. due to unusual mathematical notation or the lack of low-level description of the operators. Where the authors have not released the implementation of their algorithm (which is often the case) [13], it is possible that implementations may be available as part of existing optimisation libraries.

It should be noted that the author of NSGA-II has made the source code available on their website. However, as described below, many researchers still make use of alternative implementations provided by commonly used libraries, and in some cases, even their own implementation of the algorithm. [13]

Given this situation, the implementation used may offer significantly different performance when compared to the authors published results. This would impact the integrity of performance comparisons when proposing a new algorithm.

To verify this, the following testable hypotheses can be defined:

Hypothesis 1 *The approximation-sets produced by multiple different implementations of NSGA-II on the same test case will not always achieve significantly similar performance.*

Hypothesis 2 *The final approximation-sets produced by multiple different implementations of NSGA-II may achieve significantly similar performance, but the performance of the populations maintained throughout the optimisation process will vary, i.e. the rate of convergence.*

The rest of the paper is organised as follows. Section 2.1 introduces the concept of Evolutionary Multi-objective Optimisation. Section 2.2 describes the NSGA-II algorithm and discusses its importance in the field. Section 3 details the experimental design and how the algorithms were configured. Section 3.1 lists the three implementations of the NSGA-II algorithm that were used in this experiment, and describes their distinguishing features, with details of how they have been used other experimental studies. Section 4.1 describes the hypervolume indicator, the key metric used to measure the performance of the various implementations of the NSGA-II algorithm in this study. The results of the experiment are summarised in section 4 and conclusions are drawn in section 5.

2 Background: Evolutionary Multi-objective Optimisation and NSGA-II

2.1 Evolutionary Multi-Objective Optimisation

A typical evolutionary algorithm will start with population of randomly generated candidate solutions. The object is to allow them to reproduce and generate another population of candidate solutions, which are closer to the theoretical optimum set of solutions, and to do this repeatedly, until the solution set is close enough to the optimum to be useful. As part of each iteration, the following steps are undertaken:

1. **Evaluation:** Each solution is evaluated and assigned a fitness level. This would involve calculating some function of its objective function value, with regard to any constraints that may exist. The optimisation is normally trying to minimise (or in some cases maximise) this value. In the simplest case, the fitness can just be made equal to the objective function value. This is often the case with single-objective optimisation, but with multi-objective problems a number of approaches to both fitness evaluation and selection have been developed to try and deal with the problem of multiple, possibly conflicting objectives.

2. **Selection:** A selection operator is then applied to the population. This has the job of making copies of good solutions and eliminating bad solutions from the population, whilst keeping the population size the same. Note that this step cannot create any new solutions - that is done by the remaining two steps.
   There are several approaches to selection when dealing with multi-objective problems, including Selection by switching objectives, and Selection by Aggregation. The one used by NSGA-II is called Pareto-based Selection.
   In this approach, the solutions are sorted into a number of sets. All non-dominated solutions are labelled as set one, and removed from consideration. The remaining non-dominated solutions are then labelled as set two, and removed from consideration, and so on. The fitness of a solution is determined by which set it belongs to.
3. **Crossover:** The crossover operator has the job of taking two solutions, and combining portions of them together to make two new solutions. As both portions come from solutions that have a certain level of fitness (otherwise they would not have survived the selection process) the chances are good that it will result in a solutions that also has a good level of fitness. There are a number of widely used crossover operators, including Linear [29], and Simulated Binary [15].
4. **Mutation:** The mutation operator introduces an element of random variation by changing some portion of a solution to create a new one. This ensures that the solution space is explored and as wide a range of optimum solutions as possible are identified. There are a number of widely used mutation operators, including Random [18] and Polynomial [5].

2.2 Non-dominated Sorting Genetic Algorithm II

The Non-dominated Sorting Genetic Algorithm II was first proposed by Deb et al in 2000 [4]. It generates an offspring population from the parent population, and then sorts them into a set of Pareto fronts. Solutions with high dominance are included in the next generation, whether they come from the parent or offspring population, thus making it an elitist algorithm. The algorithm also attempts to maximise diversity by employing a crowding distance operator which sorts the solutions according to their distance from their neighbours, to ensure as wide a spread as possible. NSGA-II has proved to be influential - Scopus reports a citation count of 12945 for Deb's paper, at the time of writing, and it has been described by Ishibuchi [13] as "the most frequently used evolutionary multiobjective algorithm in the literature since its proposal". He goes on to say that "New EMO algorithms were almost always compared with NSGA-II for performance evaluation in 2000-2010.". The execution life-cycle of NSGA-II is shown in algorithm 1.

Algorithm 1: NSGA-II

```
Generate random population P of size n
Apply selection and variation operators to produce offspring population Q of size n
repeat
    Combine P and Q to produce combined population R of size 2n
    Perform non-dominated sort of R to produce set of fronts F_1 to F_m
    x = 1
    while (enough room in P_new for all members of F_x) do
        Calculate crowding distance for each solution in F_x
        Copy all members of F_x to P_new
        x = x + 1
    end
    Sort F_x with crowded comparison operator in descending order
    Copy the best solutions from F_x to fill the remaining spaces in P_new
    Apply binary tournament selection (based on crowded comparison) to P_new
    Apply crossover operators to P_new
    Apply mutation operators to P_new
    Operators will produce an offspring population Q_new of size n
    P = P_new, Q = Q_new
until (stopping criteria satisfied);
Output P
```

3 Experimental Design

In order to test our hypotheses, three different implementations have been compared based on their hypervolume indicator performance on a set of synthetic test functions from the ZDT [30] test suite. These test functions, their configurations, and their salient features have been listed in Table 1.

Table 1. Salient features of the selected ZDT synthetic test problems and their respective parameter configurations

Problem	# Var	# Obj	Salient features	Reference Vector
ZDT1	30	2	convex front	11,11
ZDT2	30	2	concave front	11,11
ZDT3	30	2	disconnected front	11,11
ZDT4	10	2	convex front, many local optima	300,300
ZDT6	10	2	concave front, non-uniform distribution	11,11

Three implementations of NSGA-II have been configured to output their populations at each generation, allowing their performance to be measured throughout each execution. The parameters used to configure each of the considered implementations are taken from the author's original work [6] and listed in Table 2. Each execution has been configured to allow a maximum of 500 generations, with a sample size of 30 executions per test function.

The remainder of this section describes the NSGA-II implementations considered in this experiment, and introduces the hypervolume indicator.

3.1 Considered Implementations

Three widely used implementations of NSGA-II are compared, in this study. Each one is provided as part of a larger library of programs. Each library also

Table 2. Parameter configurations for all considered NSGA-II implementations

Crossover Type	Simulated Binary Crossover
Crossover Probability	0.9
Crossover Distribution Index	20
Mutation Type	Polynomial
Mutation Probability	1 / No of Decision Variables
Mutation Distribution Index	20
Selection Type	Binary Tournament
Population Size	100
Generation Count	500

contains implementations of other similar algorithms, and other features, such as test problems. The three libraries are introduced in the following three subsections.

jMetal jMetal is an object-oriented Java-based framework, developed by Durillo and Nebro [8]. It contains implementations of many multi-objective optimisation algorithms, as well as test problems, quality indicators and other tools useful to a researcher in the field. Development was started in 2006, with a major redesign and rewrite of the code in 2014. It is publicly available and licensed under the GNU Lesser General Public License. The current version is jMetal 5.1 and it is available on a public facing code repository [19].

The study in [25] uses the NSGA-II implementation offered by jMetal, as well as their implementation of SPEA-2 and IBEA, to test the hypothesis that their hyper-heuristic (HH) approach improves upon the results produced by the traditional algorithms. They compare the performance of the three algorithms unmodified and choosing the one that returns the best results, NSGA-II, they measure its performance again, having modified it to incorporate the HH feature.

Similar experimental scenarios which use implementations from jMetal can be found in [21], where they are used to test new types of Multi-objective Evolutionary Algorithm, and in [23] where they are used to aid in the design of software products.

Shark ML The Shark Machine Learning Library is an open-source C++ library which provides methods for optimisation and learning algorithms, neural networks and other machine learning techniques.

It is maintained by Igel, Heidrich-Meisner and Glasmachers and is publicly available and licensed under the GNU Lesser General Public License. The current version is 3.1.0 and it is available on a public facing code repository [12].

The study in [24] uses hybridised versions of the NSGA-II and MO-CMA-ES algorithms, as provided by the Shark Machine Learning Library, in conjunction with a partially observable Markov decision process (POMDP).

Other experimental scenarios which use the Shark ML framework can be found in [1] where they are used to develop a system for the topological optimisation of mechanical structures, and in [2] where they are used in parameter tuning for various types of algorithms, such as those used by search engines.

Other examples can be found in [9] where they were used to test a new type of hypervolume indicator, and in [22] where they were used to implement a way of updating covariance matrices.

MOEA Framework The MOEA Framework is a free and Open Source Java-based framework, developed by Hadka and first released in 2011 [10]. Like jMetal, it also contains implementations of a large group of multi-objective optimization algorithms, standard test problems and performance indicators.

The framework is also licensed under the GNU Lesser General Public License. The current version is 2.11 which was released Aug 16 2016 and it is available on a public facing code repository [10].

MOEA incorporates some algorithm implementations from the jMetal library, although it provides its own implementation of NSGA-II.

The study described in [7] makes use of the MOEA framework in its experimental design. The authors wish to determine the effectiveness of using a multi-objective evolutionary algorithm approach to solve the problem of how best to manage a robot. Similar experimental scenarios which use implementations from the MOEA framework can be found in [26] and [3].

4 Numerical Results

4.1 Hypervolume Indicator

The HV indicator (or s-metric) is a performance metric for indicating the quality of a non-dominated approximation set, introduced by [31] where it is described as the "*size of the space covered or size of dominated space*". It can be defined as [27]:

$$HV\left(f^{ref}, X\right) \quad = \quad \Lambda\left(\bigcup_{X_n \in X}\left[f_1(X_n), f_1^{ref}\right] \times \cdots \times \left[f_m(X_n), f_m^{ref}\right]\right) \quad (1)$$

where $HV\left(f^{ref}, X\right)$ resolves the size of the space covered by an approximation set X, $f^{ref} \in \mathbb{R}$ refers to a chosen reference point, and $\Lambda(.)$ refers to the Lebesgue measure [16]. This has been illustrated in Figure 2 in two-dimensional objective space (to allow for easy visualisation) with a population of three solutions.

The hypervolume (HV) indicator is appealing because it is compatible with any number of problem objectives and requires no prior knowledge of the true Pareto-optimal front. It is currently used in the field of multi-objective optimisation as both a proximity and diversity performance metric and also in the decision making process [11].

Unlike dominance-based criteria which require only two solutions for performing a comparison (which can be used on an ordinal scale), a reference vector is required to calculate the HV indicator value (i.e. it requires the objective to be measured on an interval scale). When used for pairwise or multiple comparison

of optimisation algorithms, this reference vector must be the same, otherwise the resulting HV indicator values are not comparable. The reference vectors used in these experiments are listed in Table 1.

Average hypervolume values over 30 runs for each test problem are shown in Fig. 3. The hypervolume indicator results for the worst, mean, and best execution at generation 500 and generation 100 have been presented in Table 3. The Wilcoxon signed-rank nonparametric test [28] has been employed to test for statistical significance, and the results of this are shown in Table 4. For each pair of implementations, the p value is shown, and a symbol is used to show relative performance. A '+' indicates that the first implementation outperformed the second, a '−' indicates that the second outperformed the first, and a '=' indicates that there was no significant difference between the first and second.

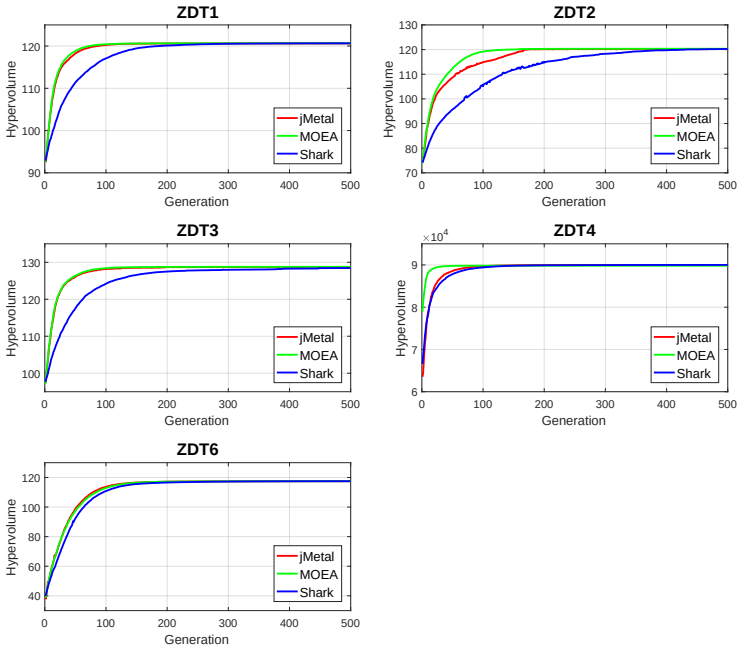

Fig. 3. Mean hypervolume indicator results for the ZDT test problems, 500 generations, for NSGA-II implementations taken from jMetal, MOEA Framework, and Shark ML.

The MOEA Framework implementation outperforms the implementations from jMetal and Shark ML on 26 of the 30 results listed in Table 3, with the fastest overall rate of convergence. The greatest difference in performance can be seen early in the optimisation process, e.g. before 200 generations (Figure 3). This suggests that the number of generations used in an experiment will significantly impact the integrity of the results, i.e. if an experiment is using

Shark ML's implementation of NSGA-II as a benchmark, the comparison would not be the same as if they were to use jMetal's implementation of NSGA-II.

Hypothesis 1 is concerned with the final approximation sets produced by each of the implementations. Although the magnitude of the variance is smaller in the later generations, even in the final approximation set, 13 of the 15 pairwise comparisons rejected the null hypothesis. This shows a significant difference in the hypervolume quality of each implementation's final approximation set.

The results suggest that MOEA Framework's NSGA-II implementation offers the best performance, despite being configured with the same parameters and operators. It outperformed the other two implementations on all of the test problems except ZDT4, where it performed the worst.

The Shark implementation was second best overall, given that it performed best on ZDT4, and second best on three of the test problems (ZDT1, ZDT2 and ZDT6), with only its performance on ZDT3 being the worst out of the three.

The jMetal implementation was the worst performer on three of the test problems, and came second on ZDT3 (behind MOEA) and ZDT4 (behind Shark).

This provides experimental confirmation of hypothesis 1 - that the approximation sets produced by multiple different implementations of NSGA-II on the same test case will not always achieve significantly similar performance.

Table 3. Mean hypervolume results from 30 executions of three versions of NSGA-II on the ZDT test suite. The boldface values indicate superior performance

$gen = 100$	jMetal			MOEA			Shark		
Problem	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best
ZDT1	118.8605	120.2570	120.4908	**120.3037**	**120.4232**	**120.5114**	114.7115	117.0696	118.6003
ZDT2	109.9355	114.9227	119.7335	**110.4201**	**119.1809**	**119.8377**	99.3786	105.1081	113.5910
ZDT3	125.1561	128.1568	128.5589	**128.1541**	**128.4240**	**128.5620**	121.2054	124.1729	126.9816
ZDT4	89532.74	89712.37	89856.94	**89538.86**	**89815.22**	**89999.52**	88683.47	89403.65	89722.09
ZDT6	111.8097	**113.7393**	**114.7459**	**111.3142**	112.9072	114.3293	108.3006	110.8531	112.8922

$gen = 500$	jMetal			MOEA			Shark		
Problem	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best
ZDT1	120.5031	120.6304	120.6550	**120.6595**	**120.6601**	**120.6608**	120.6234	120.6510	120.6601
ZDT2	119.3920	120.1841	120.3202	**120.3260**	**120.3268**	**120.3275**	119.3120	120.2408	120.3249
ZDT3	128.2403	128.7063	128.7711	**128.7745**	**128.7748**	**128.7751**	125.5630	128.4439	128.7729
ZDT4	89983.72	89996.25	89999.61	89619.57	89818.45	**89999.66**	**89992.71**	**89998.54**	89999.53
ZDT6	116.7795	117.4678	117.4982	**117.4874**	**117.4963**	**117.5044**	117.4800	117.4900	117.5005

Hypothesis 2 is concerned with the rate of convergence of each implementation. Fig. 3 clearly illustrates that, at generation 100 of the optimisation process, for all considered test functions, the Shark ML implementation of NSGA-II offers the worst performance, with regards to the mean hypervolume indicator quality achieved by each population. Table 3 lists results indicating that Shark ML also achieved the lowest hypervolume indicator quality for the worst performing and best performing populations for each problem.

The relative performance of the three implementations at generation 100 is also different when compared with generation 500. The MOEA implementation

Table 4. Results showing pairwise comparison of the mean hypervolume values using the Wilcoxon signed-ranks non-parametric test

	Generation 100						Generation 500					
	jMetal/MOEA		MOEA/Shark		jMetal/Shark		jMetal/MOEA		MOEA/Shark		jMetal/Shark	
Problem	p-value		p-value		p-value		p-value		p-value		p-value	
ZDT1	6.8714e-02	=	1.7344e-06	+	1.7344e-06	+	1.7344e-06	-	1.7344e-06	+	3.6094e-03	-
ZDT2	1.2506e-04	-	1.7344e-06	+	2.6033e-06	+	1.7344e-06	-	1.7344e-06	+	4.0483e-01	=
ZDT3	1.5886e-01	=	1.7344e-06	+	1.7344e-06	+	1.7344e-06	-	1.7344e-06	+	1.7138e-01	=
ZDT4	8.9443e-04	-	1.9209e-06	+	4.2857e-06	+	1.9209e-06	+	1.9209e-06	-	1.1748e-02	-
ZDT6	4.5336e-04	+	3.1817e-06	+	1.9209e-06	+	1.6566e-02	-	1.4773e-04	+	4.9498e-02	-

is still the best performer, outperforming the other implementations on all of the test problems except ZDT6, where it comes second after jMetal. The jMetal implementation is second best on all of the other test problems, which leaves Shark as the worst performer on all 5 test problems.

Table 4 shows that there is a statistically significant difference in 13 of the 15 pairwise comparisons in the generation 100 results. The two comparisons which do support the null hypothesis are between jMetal and MOEA, on the test problems ZDT1 and ZDT3.

It is clear from the results listed in Table 4, and from the graphs shown in Fig. 3 that the rate of convergence of the three implementations is different, in some cases markedly so. Shark has a lower rate of convergence in all of the test problems, the difference being most marked in ZDT2, but almost as large in ZDT1 and ZDT3. jMetal and MOEA converge at a similar rate in ZDT1, ZDT3 and ZDT6, but in ZDT4 the convergence rate of jMetal and Shark seem to be similar, but both are slower than MOEA.

However, although Shark has the slowest rate of convergence in all of the test problems, it actually results in the the best final approximation set in the case of ZDT1, and the second best in three of the other test problems.

Any discussion of which implementation is best will involve a tradeoff between the following factors - the speed at which the algorithm can converge towards an approximation set which is close enough to the Pareto front to be useful, and the desired quality of the final approximation set.

5 Conclusions

In this study, three implementations of the NSGA-II algorithm were configured with the same parameters and operators, and applied to the ZDT test suite in a three-way comparison. The hypervolume indicator was used as a performance metric to identify the worst, mean, and best performance of populations generated by each implementation for each test function. The implementations should not have offered significantly different performance, however, the null hypothesis was rejected in 26 of the 30 comparisons.

MOEA Framework's implementation of NSGA-II outperformed those from jMetal and Shark ML - this shows that, when comparing a newly proposed

algorithm to two implementations of NSGA-II, it is possible for the algorithm to both outperform NSGA-II, and be outperformed by NSGA-II, simultaneously.

It seems increasingly important that researchers should publish source code for their algorithm, and specify which implementation (and version) of a benchmark algorithm is used in their comparisons. The difference in performance can emerge for many reasons, including the interpretation of the pseudo-code or algorithm listing, if the author does not publish their source code, and the features of a particular programming language e.g. the accuracy of floating point numbers.

Further work in this research direction would benefit the field of Evolutionary Computation, as there are many new algorithms proposed in the literature and the majority of them are compared against existing algorithms as a benchmark. It is also worth noting that differences in the implementation of test functions and performance metrics may also exist, which would also impact the integrity of performance comparison.

References

1. Aulig, N., Olhofer, M.: Neuro-evolutionary topology optimization of structures by utilizing local state features. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, GECCO '14, pp. 967–974. ACM (2014)
2. Branke, J., Elomari, J.A.: Meta-optimization for parameter tuning with a flexible computing budget. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, GECCO '12, pp. 1245–1252. ACM (2012)
3. Cocaña-Fernández, A., Sánchez, L., Ranilla, J.: Improving the eco-efficiency of high performance computing clusters using EECluster. Energies **9**(3), 197 (2016)
4. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In: Parallel Problem Solving from Nature - PPSN VI, 6th International Conference, Paris, France, September 18-20, 2000, Proceedings, pp. 849–858 (2000)
5. Deb, K., Goyal, M.: A combined genetic adaptive search (geneas) for engineering design. Computer Science and informatics **26**, 30–45 (1996)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
7. Desjardins, B., Falcon, R., Abielmona, R., Petriu, E.: A multi-objective optimization approach to reliable robot-assisted sensor relocation. In: 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 956–964 (2015)
8. Durillo, J.J., Nebro, A.J.: jMetal: A java framework for multi-objective optimization. Advances in Engineering Software **42**(10), 760–771 (2011)
9. Friedrich, T., Bringmann, K., Voß, T., Igel, C.: The logarithmic hypervolume indicator. In: Proceedings of the 11th Workshop Proceedings on Foundations of Genetic Algorithms, FOGA '11, pp. 81–92. ACM (2011)
10. Hadka, D.: Moea - a free and open source java framework for multiobjective optimization (2015). URL https://github.com/MOEAFramework/MOEAFramework
11. Helbig, M., Engelbrecht, A.P.: Performance measures for dynamic multi-objective optimisation algorithms. Information Sciences **250**, 61–81 (2013)

12. Igel, C., Heidrich-Meisner, V., Glasmachers, T.: The shark machine learning library (2013). URL https://github.com/Shark-ML/Shark
13. Ishibuchi, H., Imada, R., Setoguchi, Y., Nojima, Y.: Performance comparison of NSGA-II and NSGA-III on various many-objective test problems. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 3045–3052 (2016)
14. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: A short review. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 2419–2426 (2008)
15. Kumar, K.D.A.: Real-coded genetic algorithms with simulated binary crossover: studies on multimodal and multiobjective problems. Complex Systems **9**, 431–454 (1995)
16. Lebesgue, H.: Intégrale, longueur, aire. Annali di matematica pura ed applicata **7**(1), 231–359 (1902)
17. Li, M., Yang, S., Liu, X.: A performance comparison indicator for pareto front approximations in many-objective optimization. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO '15, pp. 703–710. ACM (2015)
18. Michalewicz, Z., Hartley, S.J.: Genetic algorithms+ data structures= evolution programs. Mathematical Intelligencer **18**(3), 71 (1996)
19. Nebro, A.: jmetal: a framework for multi-objective optimization with metaheuristics (2014). URL https://github.com/jMetal/jMetal
20. Purshouse, R.C.: On the evolutionary optimisation of many objectives. University of Sheffield Sheffield, UK (2003)
21. Rostami, S., Neri, F.: Covariance matrix adaptation pareto archived evolution strategy with hypervolume-sorted adaptive grid algorithm. Integrated Computer-Aided Engineering **23**(4), 313 (2016)
22. Rostami, S., Shenfield, A.: A multi-tier adaptive grid algorithm for the evolutionary multi-objective optimisation of complex problems. Soft Computing pp. 1–17 (2016)
23. dos Santos Neto, P.d.A., Britto, R., Rabêlo, R.d.A.L., Cruz, J.J.d.A., Lira, W.A.L.: A hybrid approach to suggest software product line portfolios. Applied Soft Computing **49**, 1243–1255 (2016)
24. Soh, H., Demiris, Y.: Evolving policies for multi-reward partially observable markov decision processes (MR-POMDPs). In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11, pp. 713–720. ACM (2011)
25. Strickler, A., Prado Lima, J.A., Vergilio, S.R., Pozo, A.T.R.: Deriving products for variability test of feature models with a hyper-heuristic approach. Applied Soft Computing **49**, 1232–1242 (2016)
26. Svensson, M.K.: Using evolutionary multiobjective optimization algorithms to evolve lacing patterns for bicycle wheels. Master's thesis, NTNU-Trondheim (2015)
27. Voß, T., Hansen, N., Igel, C.: Improved step size adaptation for the MO-CMA-ES. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, GECCO '10, pp. 487–494. ACM (2010)
28. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin **1**(6), 80–83 (1945)
29. Wright, A.H., et al.: Genetic algorithms for real parameter optimization. Foundations of genetic algorithms **1**, 205–218 (1991)
30. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary computation **8**(2), 173–195 (2000)
31. Zitzler, E., Thiele, L.: An evolutionary algorithm for multiobjective optimization: The strength pareto approach. Citeseer, Swiss Federal Institute of Technology (1998)

32. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Fonseca, V.G.d.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation **7**(2), 117–132 (2003)