Sensitivity Analysis of Search-Space Dimensionality on Recent Multi-Objective Evolutionary Algorithms

Denny Verbeeck DENNY.VERBEECK@CS.KULEUVEN.BE
Dept. of Computer Science, Celestijnenlaan 200A, 3001 Heverlee, Belgium

Hendrik Blockeel HENDRIK.BLOCKEEL@CS.KULEUVEN.BE
Dept. of Computer Science, Celestijnenlaan 200A, 3001 Heverlee, Belgium

Abstract

Many methods for multi-objective optimisation exist, and there are multiple studies in which their performance is compared in terms of a wide range of evaluation metrics. Usually, these studies compare the end result of the optimisation process on given benchmarks; they do not evaluate how fast this end result is obtained, nor how properties of the benchmarks affect these results. In this paper, we investigate how the search space dimensionality of optimisation problems affects the behaviour of different methods, not only in terms of the end result but also in terms of how fast it is achieved. We compared two particle-swarm based optimisers, an elitist evolutionary algorithm and a scatter search algorithm. Our results show that while the PSO-based methods generally converge faster or equally fast compared to the others, they found a less diverse set of solutions.

1. Introduction

In many optimisation settings the task requires the optimisation of several objectives at once. Generally these objectives are conflicting, as a result no single optimal solution exists. Instead, an optimiser must find a set of mutually non-dominating solutions, called the Pareto-front. A solution dominates another one, if it scores strictly better for at least one objective, and equal or better on the other objectives, compared to the other solution.

Appearing in *Proceedings of BENELEARN 2014*. Copyright 2014 by the author(s)/owner(s).

Several multi-objective evolutionary algorithms (MOEAs) have been developed in recent years to deal with these so-called multi-objective optimisation problems. Some published work concerning an empirical comparison of these systems does exist (Bradstreet et al., 2007; Durillo et al., 2009; White & He, 2012). In general, this work executes a number of different algorithms on a number of different benchmark problems, and reports the final scores of several performance measures. However, this approach disregards how the values of the performance measures progress through the execution of the algorithm. For example, algorithm A may achieve a slightly sub-optimal solution very quickly, while algorithm B slowly creeps toward the global optimum. The end score reveals algorithm B having superior performance, however if we investigate the performance progress, we might prefer algorithm A since we find a decent solution more quickly.

Additionally it is useful to investigate an algorithm's behaviour on varying problem dimensionality. Different benchmark problems often have different dimensionalities, but this alone might not explain differences in performance of an algorithm, since the different benchmarks might have other characteristics which influence the algorithm. A more controlled experimental setup can provide more precise information.

In this work, we address both points. The work is motivated by a concrete application, namely compiler optimisation. Many compilers have a wide range of optimisation options, with possibly antagonistic effects (e.g., one optimisation destroys opportunities for another one). Depending on what is to be optimised (execution speed, memory space, energy requirements, compilation speed, ...), and what kind of program is being compiled, different combinations of options may be optimal, and finding them is nontrivial (Hoste & Eeckhout, 2008). Multi-objective optimisation methods can be used for this, but fitness evaluations are expensive (they require a full program compilation) and the input space is high-dimensional. In this context, a study of how search-space dimensionality affects the efficiency and solution quality is useful.

In this paper we revisit several MOEAs and investigate their performance progress during execution as well as their sensitivity to varying search-space dimensionality. We use the DTLZ (Deb et al., 2002b) and WFG (Huband et al., 2005) benchmark suites since their problems allow us to set the search-space dimensionality.

The remainder of this paper is organised as follows: in Section 2 we briefly introduce the MOEAs used in this comparison. Section 3 explains our experimental setup and Section 4 discusses the results of the experiments. In Section 5 we draw conclusions on these results.

2. MOEAs

For this paper, we selected three recent MOEAs that have shown good performance in a previous comparative study (White & He, 2012), as well as NSGA-II (Deb et al., 2002a). The first two selected algorithms are OMOPSO (Sierra & Coello, 2005) and SMPSO (Nebro et al., 2009). Both are multi-objective extensions of particle swarm optimisation (PSO). PSO algorithms are inspired by the collective behaviour of social animals, like the collective movement of birds in a flock. The location of individuals in the search space is updated using a simple set of rules. Additionally solutions might be mutated using a polynomial mutation operator. The third algorithm is AbYSS (Nebro et al., 2008), which is a multi-objective algorithm based on scatter search. In scatter search, a small population or reference set is used from which new individuals are constructed systematically. NSGA-II is a well-known elitist genetic algorithm that uses fast non-dominated sorting and crowding distance assignment to select individuals which will generate the next offspring population. Although NSGA-II was developed over ten years ago, it is still widely used, and other authors have shown it is still competitive with more recent methods (White & He, 2012). Both NSGA-II and AbYSS use simulated binary crossover and polynomial mutation operators for dealing with real-valued parameter vectors.

3. Experimental Setup

We compare the algorithms using the DTLZ (Deb et al., 2002b) and WFG (Huband et al., 2005) benchmark suites. These benchmark suites consist of a number of artificial functions that are designed to make finding the Pareto-front difficult. Each algorithm is given a population of size 100, and is given a budget of 75000 function evaluations. Other algorithm parameters are kept at their defaults as suggested by their respective authors. All algorithms are executed 40 times on all problems. The search space for the optimization algorithms is exactly the parameter space of the test problems. We vary the number of parameters, i.e. the search space dimensionality, d of all problems in the set $d \in \{6, 10, 15, 20, 25, 30\}$.

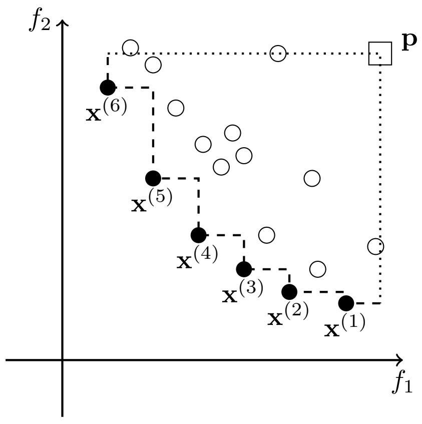


Figure 1: The dashed lines represent the Pareto front for two objective functions f_1 and f_2 in a minimisation setting. The points $\mathbf{x}^{(i)}$ denote Pareto-optimal solutions. The area bounded by the dashed and dotted lines represents the hypervolume metric for this Pareto front with respect to the reference point $\mathbf{p}$.

The DTLZ test problems were constructed by using a bottom-up approach. First the desired Pareto-front is expressed mathematically in the objective space. The rest of the feasible objective space should be located above this Pareto-front. This can be achieved by multiplying the Pareto-front expressions by a term greater than or equal to one. This term, as well as the variables for the Pareto-front expressions, can be constructed as a function of the input vector $\mathbf{x}$, resulting in an optimisation problem with the desired Pareto-front. As an example, Equation (1) shows the minimization problem DTLZ1 with two objectives and n parameters.

$$\begin{aligned}
\text{Minimize} \quad & f_1(\mathbf{x}) = \tfrac{1}{2}x_1(1+g(\mathbf{x})), \\
\text{Minimize} \quad & f_2(\mathbf{x}) = \tfrac{1}{2}(1-x_1)(1+g(\mathbf{x})), \\
\text{subject to} \quad & 0 \le x_i \le 1, \qquad \text{for } i = 1, \ldots, n \\
& g(\mathbf{x}) = 100\left[n + \textstyle\sum_{x_i \in \mathbf{x}} (x_i - 0.5)^2 - \right. \\
& \qquad \left. \cos(20\pi(x_i - 0.5))\right]
\end{aligned} \tag{1}$$

The WFG toolkit problems are defined by a vector of parameters $\mathbf{x}$. This vector is derived from a vector of working parameters $\mathbf{z}$, through a series of transition vectors. Each of these transition vectors adds complexity to the problem, such as multi-modality or non-separability. The optimisation method directly manipulates $\mathbf{z}$, through which $\mathbf{x}$ is indirectly manipulated. The parameters in $\mathbf{z}$ are labelled as either position-related or distance-related parameters. In our experiments we keep the number of position-related parameters constant at 4. We vary the number of distance-related parameters in the set $\{2, 6, 11, 16, 21, 26\}$, such that the total dimensionality is varied over the same set as for the DTLZ problems. Since the scope of this paper is limited to search-space dimensionality, we keep the number of objectives for all test problems at two.

Rather than bombarding the reader with dozens of performance metrics, we choose to base our experiments on the hypervolume metric (Zitzler & Thiele, 1999) and the spread metric (Deb et al., 2002a). The hypervolume metric measures the size of the objective space that is enclosed between the Pareto front found by an algorithm, and a reference point. Since the optimal Pareto fronts are known for DTLZ and WFG, this indicates how closely an algorithm is able to approximate the optimal Pareto front. Figure 1 illustrates the concept of hypervolume.

Spread is a measure of how well spread out solutions on the Pareto front are. A small spread measure indicates solutions are distributed evenly across the Pareto front. As larger gaps occur on the front, the spread metric increases.

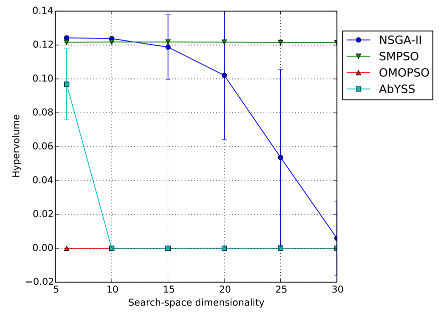
The solutions of each execution are stored in an archive of size 200, where solutions are sorted based on Pareto dominance and crowding distance.

4. Experimental Results

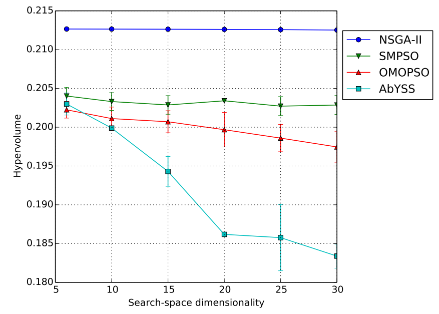
The graphs in this section all include error bars. These represent $\pm$ one standard deviation from the mean measured over 40 executions. Some test problem plots showed largely equivalent characteristics to others. These are not included separately, but are mentioned between parentheses in the title of the equivalent plots.

4.1. Sensitivity Analysis of Search-Space Dimensionality

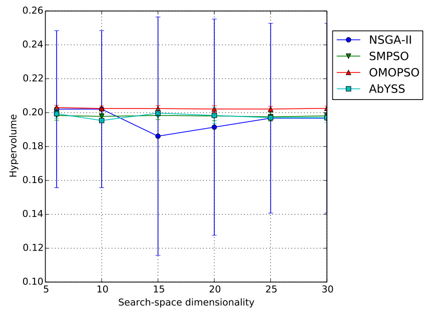
In the first part of our experimental setup, we look at the achieved hypervolume at the end of execution, in function of varying search-space dimensionality. Figure 2 shows plots for the DTLZ test problems. Intuitively, we would expect to see at least a small amount



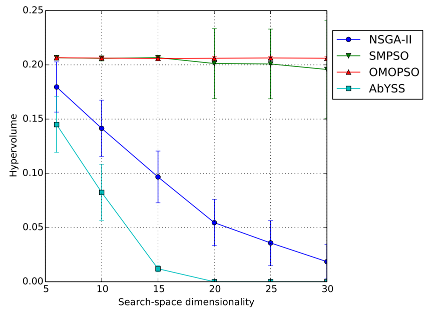
(a) DTLZ1 (DTLZ3)



(b) DTLZ2 (DTLZ5)

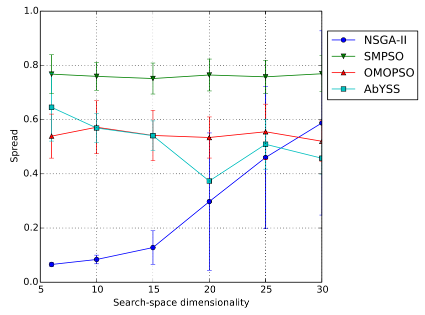


(c) DTLZ4 (DTLZ7)

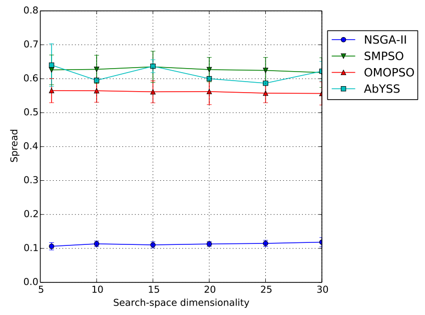


(d) DTLZ6

Figure 2: Hypervolume metric in function of search-space dimensionality for the DTLZ test problems.



(a) DTLZ1 (DTLZ3)



(b) DTLZ2 (DTLZ4–7)

Figure 3: Spread metric in function of search-space dimensionality for the DTLZ test problems.

of degrading performance as the dimensionality increases, but this is not always the case. In fact the particle swarm based optimisers (SMPSO and OMOPSO) seem to be largely unaffected. For the 1st, 3rd and 6th test problem, NSGA-II and AbYSS clearly shows sensitivity to increasing dimensionality. These test problems have multiple local Pareto-optimal fronts, a property that does not appear in the other DTLZ test problems. This result shows that NSGA-II and AbYSS are particularly sensitive to increasing search-space dimensionality when the objective space contains local Pareto-optimal fronts. Furthermore it is noteworthy that the PSO-based methods have generally small variance in their performance compared to the other algorithms. This indicates a more stable migration of the population towards the optimum.

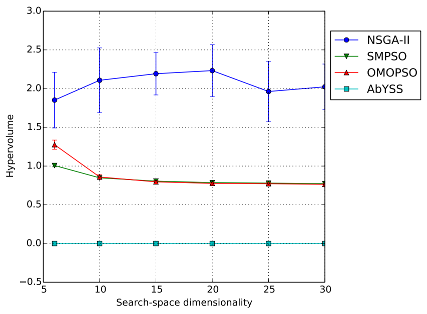
Figure 3 shows the impact of search-space dimensionality on the spread metric. Performance as measured by the spread metric seems largely unaffected by increasing dimensionality, with the exception of NSGA-II's performance on the 1st and 3rd test problems. Since these test problems contain multiple local Pareto fronts, it is possible not all solutions are on the same front, thus increasing the spread metric. However for all but the highest dimensionality setting, as well as for all other DTLZ test problems, NSGA-II outperforms its competitors in terms of the spread metric. Out of the compared MOEAs, NSGA-II is the only one with built-in preference towards a well-spread population by means of the crowding distance assignment, which explains these results.

Figure 4 shows plots for the hypervolume metric on the WFG test problems. For brevity, not all plots are reported here. In contrast to results obtained from the DTLZ test problems, here the PSO-based methods show the highest sensitivity to increasing search-space dimensionality, and are generally outperformed by NSGA-II and AbYSS. While some of the WFG test problems also incorporate multi-modality, the "hill size between the local fronts is 10 times higher in the DTLZ problems, making it easier for EAs to get stuck in a local optimum.

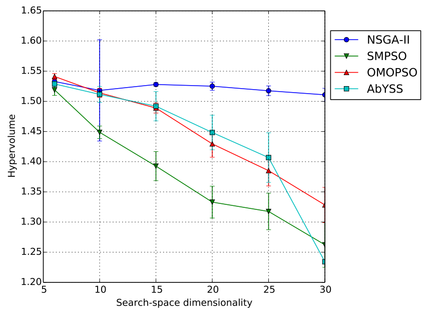
The contrast in performance between the two benchmark suites might also be explained by the structure of the WFG test problems themselves. As mentioned in Section 3, the parameters in the search space for WFG problems are labelled as either position-related or distance-related parameters. Even though in actuality, parameters can also be a mix of these two, the crossover operators of NSGA-II and AbYSS can exploit the distinction between position and distance parameters. Parent solutions on or near the Pareto-front would have roughly the same set of distance parameters. As a result, a crossover operation only significantly modifies the position parameters, generating offspring on different locations of the Pareto-front, facilitating a faster discovery of the Pareto-front. This theory can also explain the difference in spread metric between the compared methods. These results indicate that the lack of a crossover operator in PSO-based methods can in some cases result in a significant disadvantage compared to EAs that do incorporate one.

4.2. Performance Progress Analysis

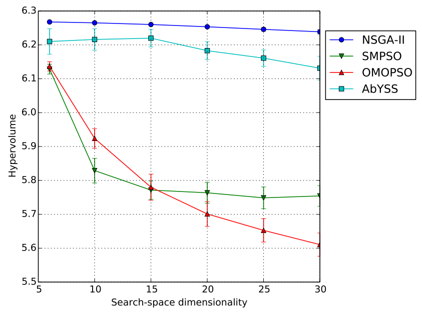
If we take a look at Figure 2(a), we might conclude that the performance of NSGA-II is comparable to SMPSO for DTLZ1 up to 15 input dimensions. However upon closer inspection of the run-time progress, we can see that this is not the case. Figure 5 shows the progress of the hypervolume dependent on the number of function evaluations for two dimensionalities of the DTLZ1 test problem. From these graphs we can see that increasing the dimensionality from 6 to 15, the amount of necessary function evaluations for NSGA-II to reach convergence is roughly tripled. This insight clearly gives preference to SMPSO even for the lower



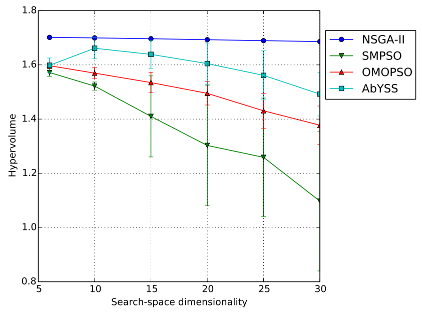
(a) WFG1



(b) WFG2

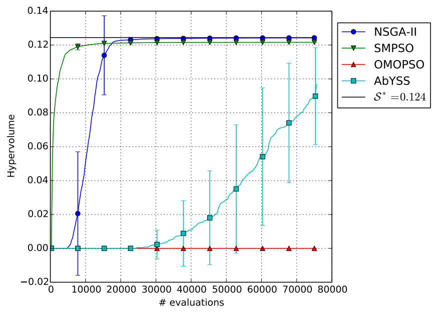


(c) WFG4 (WFG6)

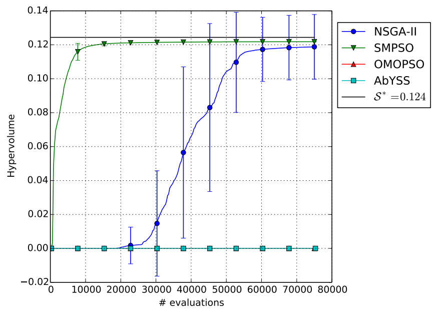


(d) WFG7 (WFG9)

Figure 4: Hypervolume metric in function of search-space dimensionality for the WFG test problems.



(a) DTLZ1, $d = 6$

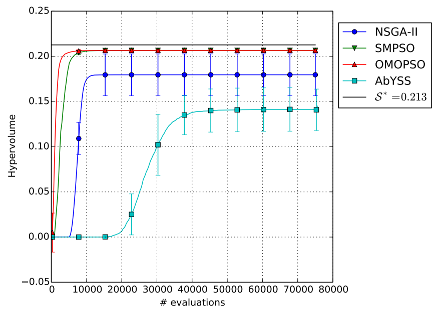


(b) DTLZ1, $d = 15$

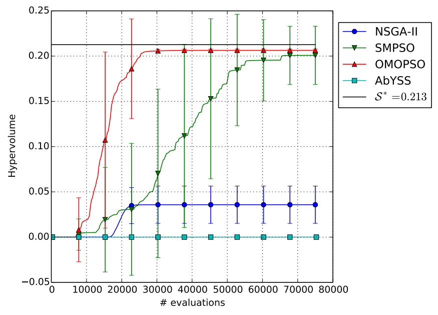
Figure 5: Progress of the Hypervolume metric for the DTLZ1 benchmark. The optimal hypervolume $\mathcal{S}^*$ is also shown.

dimensionality settings. Figure 6 shows the same effect on the DTLZ6 test problem, while the final hypervolumes of SMPSO and OMOPSO are comparable, SMPSO converges more slowly in higher dimensions than OMOPSO.

Another interesting observation is the fact that while some methods converge prematurely, it is possible they reach that state orders of magnitude faster than the method that is superior in the end. Figure 7(a) demonstrates this effect clearly for the 30-dimensional WFG1 problem. It takes NSGA-II about 30000 function evaluations to reach a hypervolume that the PSO-based methods reached almost immediately. However after that point, NSGA-II's performance surpasses that of its competitors. A more subtle example is given by the 30-dimensional WFG6 problem, shown in Figure 7(b). Here the PSO-based methods double the performance of NSGA-II, but only for the first 5000 function evaluations. These results show that in certain settings, e.g. where only a limited amount of function evaluations are permitted, the best choice of method might not always be clear. On the other WFG test problems, all algorithms converged to their maximum hypervol-



(a) DTLZ6, $d = 6$



(b) DTLZ6, $d = 25$

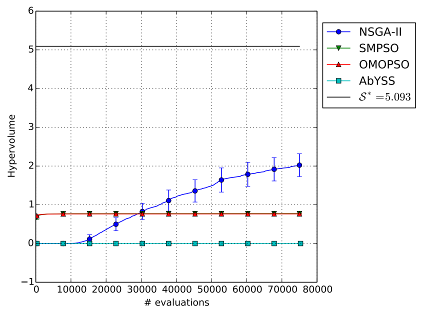
Figure 6: Progress of the Hypervolume metric for the DTLZ6 benchmark. The optimal hypervolume $\mathcal{S}^*$ is also shown.

ume at roughly the same speed, therefore these plots are not shown here.

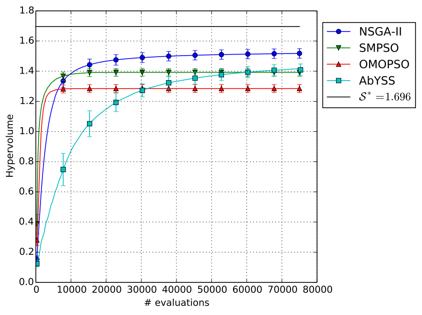
5. Conclusion

In this paper we investigated the impact of varying search space dimensionality on the performance of several multi-objective evolutionary algorithms. We showed that NSGA-II and AbYSS are particularly sensitive to increasing dimensionality for multi-modal problems. SMPSO and OMOPSO seem largely unaffected by multi-modality or increasing dimensionality, however the set of solutions generated by these methods are in almost all test cases less well spread out over the objective space compared to NSGA-II and AbYSS.

Additionally we tracked the performance during the execution of these algorithms rather than just considering the final performance measures. By investigating the performance progress we showed that only observing a final score can be deceiving. In most comparative studies, authors make sure to provide the methods under study with enough function evaluations to reach convergence. In some cases, what happens before this point is also of interest, and should be considered. We observed that in most cases, the PSO-based methods converged after less or roughly equal amounts of function evaluations compared to the evolutionary methods. However these solutions might be suboptimal to the ones discovered by NSGA-II or AbYSS, this was particularly true for the WFG test problems. We theorised the lack of a crossover operator and the lack of pressure from a crowding distance assignment might be to blame for the difference in performance. This theory can also explain why SMPSO and OMOPSO consistently perform worse in terms of their spread metric compared to NSGA-II and AbYSS. However additional research is warranted in order to validate this theory. In future work PSO-based methods could be extended with a mechanism to pressure particles in the swarm to be better spread out over the Pareto-front.



(a) WFG1, $d = 30$



(b) WFG6, $d = 30$

Figure 7: Progress of the Hypervolume metric for the two of the WFG test problems in high search-space dimensionality. The optimal hypervolume $\mathcal{S}^*$ is also shown.

Existing comparative studies generally do not perform this type of analysis. They focus on the performance only at the end of an algorithm's execution, and do not vary the search-space dimensionality of test problems.

Our results show that including these analyses can lead to additional and more detailed insights.

Acknowledgements

This research was funded by FWO Research project 3E091191: "Multi-objective compiler optimisation space exploration.

References

Bradstreet, L., Barone, L., While, L., Huband, S., & Hingston, P. (2007). Use of the wfg toolkit and pisa for comparison of moeas. *Computational Intelligence in Multicriteria Decision Making, IEEE Symposium on* (pp. 382–389).

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

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2002b). Scalable multi-objective optimization test problems. *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on* (pp. 825–830).

Durillo, J. J., García-Nieto, J., Nebro, A. J., Coello, C. A., Luna, F., & Alba, E. (2009). Multi-objective particle swarm optimizers: An experimental comparison. *Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization* (pp. 495–509). Berlin, Heidelberg: Springer-Verlag.

Hoste, K., & Eeckhout, L. (2008). Cole: compiler optimization level exploration. *Proceedings of the 6th annual IEEE/ACM international symposium on Code generation and optimization* (pp. 165–174). New York, NY, USA: ACM.

Huband, S., Barone, L., While, L., & Hingston, P. (2005). A scalable multi-objective test problem toolkit. In C. Coello Coello, A. Hernndez Aguirre and E. Zitzler (Eds.), *Evolutionary multi-criterion optimization*, vol. 3410 of *Lecture Notes in Computer Science*, 280–295. Springer Berlin Heidelberg.

Nebro, A., Durillo, J., Garcia-Nieto, J., Coello Coello, C., Luna, F., & Alba, E. (2009). Smpso: A new pso-based metaheuristic for multi-objective optimization. *Computational intelligence in miulti-criteria decision-making, 2009. mcdm '09. ieee symposium on* (pp. 66–73).

Nebro, A., Luna, F., Alba, E., Dorronsoro, B., Durillo, J., & Beham, A. (2008). Abyss: Adapting scatter search to multiobjective optimization. *Evolutionary Computation, IEEE Transactions on*, *12*, 439–457.

Sierra, M. R., & Coello, C. A. C. (2005). Improving pso-based multi-objective optimization using crowding, mutation and ϵ-dominance. *In EMO2005, pages 505519. LNCS 3410* (pp. 505–519). Springer-Verlag.

White, T., & He, S. (2012). An empirical comparison of several recent multi-objective evolutionary algorithms. In L. Iliadis, I. Maglogiannis and H. Papadopoulos (Eds.), *Artificial intelligence applications and innovations*, vol. 381 of *IFIP Advances in Information and Communication Technology*, 48–57. Springer Berlin Heidelberg.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *Evolutionary Computation, IEEE Transactions on*, *3*, 257–271.