Start Small, Grow Big?
Saving Multi-objective Function Evaluations

Tobias Glasmachers[1], Boris Naujoks[2], and Günter Rudolph[3]

[1]Ruhr-Universität Bochum, Germany
tobias.glasmachers@ini.rub.de
[2]Cologne University of Applied Sciences, Germany
boris.naujoks@fh-koeln.de
[3]Technische Universität Dortmund, Germany
guenter.rudolph@tu-dortmund.de

Abstract. The influence of non-constant population sizes in evolutionary multi-objective optimization algorithms is investigated. In contrast to evolutionary single-objective optimization algorithms an increasing population size is considered beneficial when approaching the Pareto-front. Firstly, different deterministic schedules are tested, featuring different parameters like the initial population size. Secondly, a simple adaptation method is proposed. Considering all results, an increasing population size during an evolutionary multi-objective optimization algorithm run saves fitness function evaluations compared to a fixed population size. In particular, the results obtained with the adaptive method are most promising.

1 Introduction

The size of the population is an external parameter in evolutionary algorithms (EA, [6,7]). Chosen once, it is expected to stay constant for the whole optimization run. However, the right choice of the population size has an enormous effect on the outcome of the EA run. Results vary from very good to very poor only with respect to a proper setting. Choosing a population too small may prevent the localization of optimal solutions, whereas choosing a population too large wastes considerable resources, in particular if fitness function evaluations are computationally expensive. Thus, a dynamic population size might help in saving many function evaluations without any loss in solution quality.

A typical single-objective EA run can be split into two phases. During the first phase the aim of the algorithm is to identify the basin of the globally best solution. To this end, a larger population size seems to be adequate. Having identified this basin or at least a good candidate for it, the goal of the algorithm shifts to identifying the best solution within this basin. Here, a smaller population size is sufficient. This possibly scales down to the $(1 + 1)$ selection scheme.

The situation changes completely if more than one objective is considered. The optimization run of an evolutionary multi-objective optimization algorithm

(EMOA, [3,4]) can be split into two phases as well. However, the population sizes are expected to be adapted best in a different way. Often, only a rather limited population size is sufficient to approach a Pareto-front. Once the front is reached, it needs to be explored and covered well, and, consequently, larger populations are expected to perform better at this task.

Another difference is the role of the population in single and multi-objective EA. In a single-objective EA, the population allows to sample the fitness landscape sufficiently well, and to maintain a set of diverse solutions. The performance of the population is measured by the fitness of the best individual. In a multi-objective EA, the population represents the algorithm's current approximation to the Pareto-front. All non-dominated individuals contribute to it's performance. Usually, there is a pre-defined upper limit on the allowable size of the result set, which is often considered the canonical value of the population size parameter.

As a consequence, the simple rule for adapting the population size of an EMOA could be: *start small, grow big.* Big refers here to the limit of the result set size, while we'd like to start with a much smaller size. In this study, we aim at proving that increasing the population size like this saves fitness function evaluations compared to a fixed population size.

Efforts for realizing a dynamic population size in evolutionary algorithms have been driven mainly by the desires to eliminate an external parameter and to improve performance. Only few attempts have been made to extend automatic population size control to *multi-objective* optimization. An early example (1977) of a dynamic population size in an EMOA was given by Peschel and Riedel [12], where the new population was formed by the non-dominated individuals from the union of parents and offspring. The PR1 algorithm [13] as well as the SEMO algorithm [10] work similarly. Tan et al. [14] as well as Lu and Yen [11] impose a cellular structure on the Pareto-front. This approach required prior knowledge of the front, which is only practical for synthetic benchmark problems. In [8] the population size develops as a function of a pre-defined time-dependent schedule (deterministic component) and the number of non-dominated individuals (adaptive component). It is maybe closest in spirit to the present study.

This study starts summarizing multi-criteria optimization basics before the schedules are defined in section 3. Section 4 presents experiments and results before we conclude our findings and provide a short outlook.

2 Multi-Criteria Optimization

Considering only one objective in applied optimization is a simplification that does not mirror the complexity of the underlying application in most (or almost all) cases. Often multiple objectives $f_1, \ldots, f_n : X \to \mathbb{R}$ need to be considered. Here we focus on two objectives. The most common way to deal with multiple objectives appears to be aggregation, e.g., to a weighted sum $f(x) = \sum_i w_i f_i(x)$. In contrast, multi-objective or multi-criteria optimization (MCO) offers a different way to handle multiple objectives in a more unbiased, maybe more effective

way. For this approach, we have to consider the vector-valued objective function $f : X \to \mathbb{R}^n$, $f(x) = \big(f_1(x), \ldots, f_n(x)\big)$.

In MCO, an important concept is Pareto dominance, i.e., an objective value $y \in \mathbb{R}^n$ dominates another value $y' \in \mathbb{R}^n$ iff y is better in at least one dimension of the objective space and not worse in all the others. More formally and considering minimization, this reads

$$y \prec y' \quad \text{iff} \quad \forall i: \; y_i \leq y'_i \quad \wedge \quad \exists j: \; y_j < y'_j \; .$$

If an objective value y is not dominated by any other value in the image $A = f(X)$ of the objective function (or generated by the algorithm), it is said to be *non-dominated*, i.e., $\forall y' \in A : y' \not\prec y$. This concept allows for ranking of sets in the multi-dimensional objective space. MCO algorithms aim for the optimal set $A^* = \{y \in A \mid \nexists y' \in A : \; y' \prec y\}$. It has the property that every two points y and y' from A^* are mutually non-dominated, i.e. $y' \not\prec y \;\; \wedge \;\; y \not\prec y'$. This set is called the *Pareto-front*. The dominance relation is pulled back to the decision space X via the objective function by defining $x \prec x'$ iff $f(x) \prec f(x')$ for $x, x' \in X$. The resulting set $f^{-1}(A^*) \subset X$ of optimal solutions is called the *Pareto-set*.

In addition to the number of objectives, there is a structural change in the step from one to multiple objectives. The strict order of objective values in the single-criterion objective space turns into a partial order (induced by Pareto dominance) in the multi-criteria objective space. This structural change implies that besides Pareto dominance a secondary quality indicator is required for ranking and thus for rank-based selection in EA.

In recent years, the hypervolume [15,17] set indicator turned from a frequently used quality indicator to a well-established selection operator for EMOA. The hypervolume of a set Y is defined as the n-dimensional volume of the space spanned by the set and a reference point y_{ref} that needs to be defined by the user:

$$\Lambda \left(\bigcup_{y \in Y} \left\{ y' \in \mathbb{R}^n \,\middle|\, y \prec y' \prec y_{\text{ref}} \right\} \right)$$

with Λ being the n-dimensional Lebesgue measure of the given set. The hypervolume of a set $P \subset X$ of solutions (e.g., a population) is the hypervolume of the corresponding values $\{f(x) \,|\, x \in P\}$.

Maximization of the hypervolume covered by the population implicitly covers the traditional goals of convergence of the solution set to the optimal front as well as good solution spread. Most prominent instances of hypervolume based selection MCO algorithms are SMS-EMOA [2], Hyp-E [1], as well as MO-CMA-ES [9].

The $(\mu + 1)$ selection mechanism in SMS-EMOA provides an elegant way to enlarge and diminish the population size online. The population can be enlarged by skipping the selection step, and it can be reduced by skipping the offspring generation step. Therefore, this algorithm is used for the present study.[1]

[1] The software is available on request by email to the first author.

3 Schedules

A population size schedule is simply a rule defining which population size to use at which time. Such a rule may be a fixed function of the generation counter or an adaptive decision rule based on online indicators. We investigate both possibilities.

3.1 Fixed Schedules

We start with the definition of a family of fixed, parametric schedules. The (increasing) population size is a function of time, measured by the number of fitness evaluations (FE), and normalized in relation to a budget of FE_{budget} fitness evaluations. In an application, this budget may be set to the affordable number of fitness evaluations. We think of it as a conservative estimate. We want to note clearly that the budget is a highly problem specific parameter. It is hard to guess a sound value without prior experimentation. Therefore, the requirement of providing an optimal budget parameter may not be realistic in practice. Here, this proceeding allows us to define comparable schedules for very different problems. As a practical solution, we also propose an adaptive scheduling strategy below.

The final population size S_{full} is the desired size of the Pareto-front approximation. In this study, it is fixed to $S_{\text{full}} = 100$. Besides these constants, each schedule is defined by four parameters $\alpha, \beta, \gamma, \delta \in [0, 1]$ as follows. The initial population size is set to $S_{\text{start}} = \lfloor \alpha \cdot S_{\text{full}} \rfloor$. The time FE_{full} by which the growing population size reaches S_{full} is represented as a fraction of the budget: $FE_{\text{full}} = \lfloor \beta \cdot FE_{\text{budget}} \rfloor$. In between we interpolate linearly. Thus, these parameters define linearly growing schedules with a cut-off at S_{full}. This class of schedules is further enriched by an intermediate point $(FE_{\text{inter}}, S_{\text{inter}})$, defined by $FE_{\text{inter}} = \lfloor \gamma \cdot FE_{\text{full}} \rfloor$ and $S_{\text{inter}} = \lfloor (1 - \delta) \cdot S_{\text{start}} + \delta \cdot S_{\text{full}} \rfloor$. This construction is illustrated in figure 1.

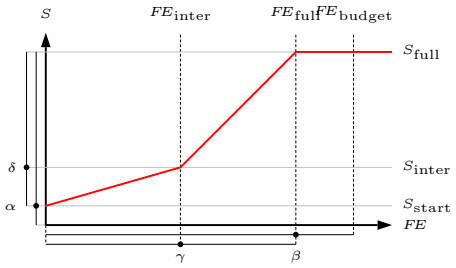

Fig. 1. Illustration of the fixed schedules and their parameters as functions of the population size S over the number of fitness evaluations FE. Refer to the text for details.

3.2 Adaptive Schedules

An alternative to a fixed schedule as a function of FE is an adaptive rule. We have found the following extremely simple rule to be effective: maintain a fading record of success probabilities, and increase the population size by a factor as

soon as the success rate drops below a threshold. This rule is inspired by Rechenberg's famous 1/5 rule for the adaptation of the step size (mutation strength) in evolution strategies. Here, a very similar rule is used successfully in a completely different context.

The algorithm works as follows. The low-pass filtered success rate is initialized to a value of $R \leftarrow 1/2$. In each generation it is updated according to $R \leftarrow (1-\eta) \cdot R + \eta \cdot \mathbf{1}_{\text{suc}}$ where the success indicator $\mathbf{1}_{\text{suc}}$ is one if the offspring generated in the current iteration survives the selection phase and zero otherwise. A learning rate of $\eta = 0.01$ results in sufficiently stable behavior. Once the success rate indicator R drops below the threshold of 1/5, the population size is increased:

$$S \leftarrow \min\left\{\lceil c \cdot S \rceil, S_{\text{final}}\right\}, \qquad R \leftarrow 1/4$$

At the same time, the success rate R is reset away from 1/5 in order to avoid multiple population size increases due to random effects. We set the increase factor to $c = 3/2$.

The intuition behind this scheme is that initially the success rate is high, since all individuals are selected with the same probability. As the population approaches the front, successes become harder to sample as the success probability slowly approaches zero. At this stage, progress can be made only by spreading out the population over the front, which requires an increase of the population size.

This adaptive strategy has a number of parameters such as the initial value of the success rate estimate, the success rate threshold of 1/5, the learning rate of 0.01 and the increase factor of 3/2. We did not tune these parameters. We have run a few trials with other parameter settings and we did not find the algorithm to be very sensitive to the exact values. However, the threshold should be kept around 1/5 for the procedure to work well.

The only remaining critical parameter is the initial population size S_{start}. Analog to the fixed schedules defined above, we express this parameter by means of $\alpha \in [0,1]$ as $S_{\text{start}} = \lfloor \alpha \cdot S_{\text{full}} \rfloor$.

4 Experimental Evaluation

The goal of our experimental evaluation is two-fold. First of all, we aim for an overview of whether and how many fitness evaluations can be saved with a non-uniform population size schedule. To this end, we test a large number of deterministic schedules against the baseline method, which is to run the EMOA with the full target population size. Second, and maybe more importantly, we investigate the performance of our adaptive population size control method. The primary performance comparison is with the uniform baseline. The systematic grid evaluation of non-adaptive schedules serves as a second baseline. It allows to judge the performance of the algorithm relative to the possible gain that could be expected from *any* population size adaptation algorithm.

We consider the benchmark problems ZDT1-4 and ZDT6 from [16], the two-objective versions of DTLZ1-4 from [5], as well as Schaffer's problem. We use 30 variables for ZDT1-3 and 10 variables for all other problems. The goal of optimization is to cover 99.9% of the hypervolume theoretically achievable with 100 individuals, relative to the reference point $(1.1, 1.1)$. Visual inspection reveals that this formalized goal corresponds to a reasonably accurate problem solution. For the ZDT and DTLZ problems, the achievable hypervolume can be obtained from the website `http://www.tik.ee.ethz.ch/~sop/download/supplementary/testproblems/` for the reference point $(11, 11)$ and converted easily. For Schaffer's problem we use the formulation $f_1(x) = |x_1 - \frac{1}{2}|$ and $f_2(x) = |x_1 + \frac{1}{2}|$, so that the optimal front fits inside the unit square. The achievable hypervolume with N points is $1.1^2 - 0.5 - 0.5/(N-1)$.

The first experiment compares SMS-EMOA with population size 100 to the same algorithm with increasing population size schedules as described in section 3. We have tested a four-dimensional grid of schedules given by the parameters $\alpha \in \{0.01, 0.05, 0.1, 0.2, 0.5\}$, $\beta \in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, $\gamma \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$, and $\delta \in \{0.0, 0.1, 0.2, 0.3, 0.5, 0.7\}$. The budget FE_{budget} was set to the number of *FE* required by the baseline, rounded up, see table 1. The algorithm was run 100 times for each of the 1 050 schedules. The median number of *FE* relative to the baseline is reported compactly in figure 2.

Table 1. Budgets for the definition of the fixed schedules. The budget values were determined by rounding up the median *FE* required by plain SMS-EMOA for reaching 99.9% of the optimal hypervolume.

Problem	Schaffer	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6	DTLZ1	DTLZ2	DTLZ3	DTLZ4
Budget	6 000	9 000	11 000	10 000	40 000	7 000	36 000	6 000	105 000	6 000

The results show two basic facts. First, the region where a good schedule is found varies from problem to problem. This clearly shows the need for an adaptive strategy. Second, problems exist where the baseline is hard to beat with any schedule. This means that increasing populations help in many cases, but not always, while it (nearly) never harms.

On the DTLZ4 benchmark all strategies with initial population size of less than 50 hit the maximum of 500 000 *FE in the median*. It turns out that this result is not due to an algorithmic flaw but must be attributed solely to numerical problems.[2]

[2] The term $\cos(\pi/2 \cdot x_i^{100})$ in the DTLZ4 problem applied to numbers $x_i < 0.83$ gives exactly one when evaluated with 64bit IEEE double precision numbers. This leads to a large fitness plateau and a spurious local optimum. Control experiments with (a) higher precision, (b) lower exponent, and (c) larger population confirm this finding. Therefore, we do not consider the DTLZ4 benchmark any longer.

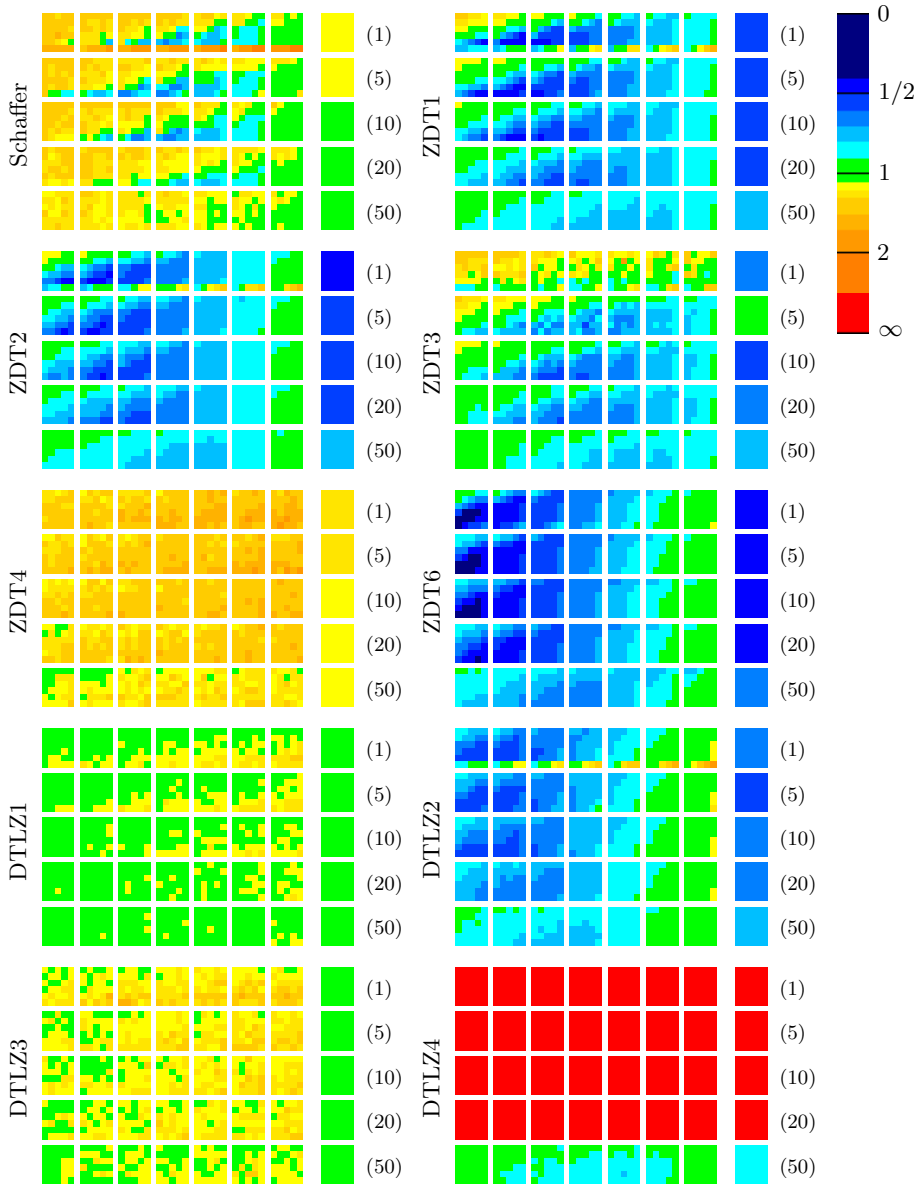

Fig. 2. Performance of fixed and adaptive schedules. Each 5×7 matrix of bitmaps encodes values of the parameters α (rows) and β (columns). In addition the initial population size is listed in brackets on the right. Within each bitmap, rows and columns encode parameters δ and γ, respectively, so that the position within the bitmap resembles the position of the intermediate point as indicated in figure 1. The column on the right of the matrix reports results for the adaptive schedule. Pixel colors indicate relative runtime, measures as number of *FE* divided by number of *FE* required by SMS-EMOA. Values smaller than 1 (blueish color) indicate an improvement over the baseline, values close to 1 (green) mark performance indifferent to the baseline, and values larger than 1 (yellow to red) indicate deterioration of the performance as compared to the baseline.

In the second experiment, the population size online adaptation procedure is tested against plain SMS-EMOA and also ranked relative to the extensive grid of deterministic schedules. The experimental setup remains unchanged. The adaptive schedule has a single parameter α controlling the initial population size S_{start}. For a fair comparison, this parameter was varied in the same range as before.

The results for the adaptive schedules are reported graphically in the column right to the bitmap matrix in figure 2, as well as numerically in table 2. The performance of the algorithm is rather robust w.r.t. its only parameter, the initial population size. Our results suggest a default setting of $\alpha = 0.1$ (corresponding to $S_{\text{start}} = 10$ in our experiments).

It becomes clear that on average the adaptive schedule works about as well as a good fixed schedule. Importantly, this is achieved across different problems that require different types of schedules, without prior knowledge of the budget, and with a practically parameter free method.

Table 2. Performance (median number of *FE*, lower is better) of plain SMS-EMO and adaptive population size schedule, with the same initial population size of $S_{\text{start}} = 10$.

Problem	Schaffer	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6	DTLZ1	DTLZ2	DTLZ3
Baseline	5 482	8 867	10 271	9 533	39 422	6 012	35 052	5 015	102 185
Adaptive	5 848	4 851	5 544	5 712	46 591	2 701	34 926	3 058	97 880

An interesting question is how exactly the online adaptation evolves the population size. Figure 3 answers this question. There is a high correlation between the population size staying low for an extended period and a significant performance improvement over the baseline (compare to figure 2). This is not surprising since increasing the population size quickly basically means that plain SMS-EMOA takes over quickly. Such behavior is hard to avoid on multi-modal problems such as ZDT4, DTLZ1, and DTLZ3. Importantly, although in these cases online adaptation does not help, it also does not (seriously) impair performance.

In contrast, for problems ZDT1, ZDT2, ZDT3, ZDT6, and DTLZ2 adaptively increasing the population size results in considerable savings of *FE*, in the order of about 50%. All of these problems can be solved by approaching the front with a small population, resulting in increased selection pressure, and spreading the increasing population over the front as soon as the progress rate drops.

In summary, starting small and growing the population big over the course of a multi-objective optimization run can save a significant fraction of fitness evaluations, while it nearly never hurts. Our adaptive algorithm performs in most cases about as well as the (in general unknown) best deterministic schedule.

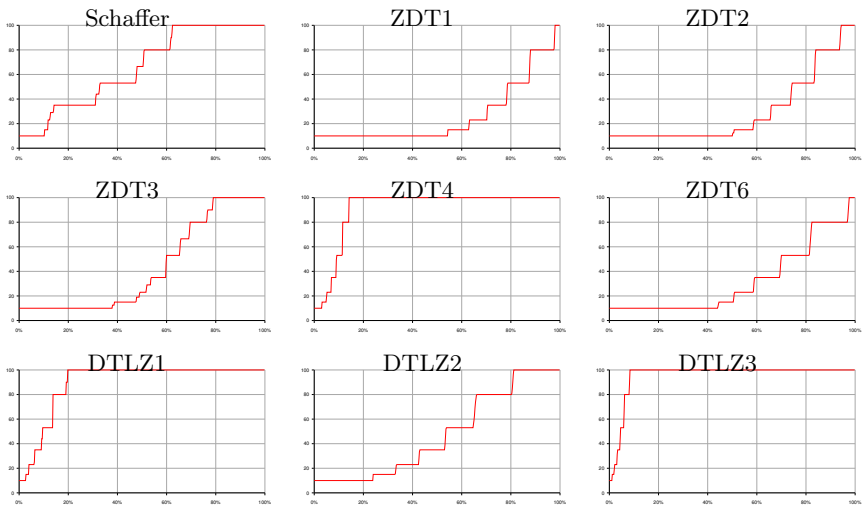

Fig. 3. Evolution of the adaptively controlled population size starting with 10 individuals. All successful runs were rescaled to the same length which is displayed on a percentage scale.

5 Conclusion and Outlook

We have proposed an online adaptation scheme for the population size of an EMOA. This algorithm was compared with the usual proceeding of fixing the population size to the desired cardinality of the result set, as well to a large number of systematically chosen deterministic increasing population size schedules. The proposed adaptive algorithm compares favorably. It saves up to about 50% of the fitness evaluations of the standard algorithm in case uni-modal problems, whereas it shows nearly unchanged behavior on multi-modal benchmarks. The comparison of the performance of the adaptive schedule to the large set of deterministic schedules reveals that significantly better results cannot be expected with *any* population size schedule. Thus the strategy to *start small and grow big* turns out to be successful in MCO.

A few open questions remain for future research. We did not present an adaptation rule for shrinking of the population size, although a similar success-based adaptive rule is straightforward to design. However, at least on standard benchmarks shrinking is not very useful. Another open question is how the adaptation rule can be adapted to work with an evolution strategy without interfering with the (often success-based) step size adaptation mechanism. An extension of this study to more than two objectives is work in progress.

References

1. Bader, J., Zitzler, E.: HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. Evolutionary Computation 19(1), 45–76 (2011)
2. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research 181(3), 1653–1669 (2007)
3. Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Springer, New York, 2nd edn. (2007)
4. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley, Chichester, UK (2001)
5. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Multi-Objective Optimization Test Problems. In: Congress on Evolutionary Computation (CEC 2002). pp. 825–830. IEEE Press, Piscataway (NJ) (2002)
6. DeJong, K.A.: Evolutionary Computation: A Unified Approach. MIT Press, Cambridge, MA (2006)
7. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Natural Computing Series, Springer, Berlin (2003)
8. Eskandari, H., Geiger, C.D., Lamont, G.B.: FastPGA: A dynamic population sizing approach for solving expensive multiobjective optimization problems. In: Obayashi, S., et al. (eds.) Proceedings of 4th International Conference on Evolutionary Multicriterion Optimization (EMO 2007). pp. 141–155. Springer, Berlin (2007)
9. Igel, C., Hansen, N., Roth, S.: Covariance Matrix Adaptation for Multi-objective Optimization. Evolutionary Computation 15(1), 1–28 (2007)
10. Laumanns, M.: Analysis and Applications of Evolutionary Multiobjective Optimization Algorithms. Ph.D. thesis, Swiss Federal Institute of Technology (ETH), Zurich, Zurich, Switzerland (2003)
11. Lu, H., Yen, G.G.: Dynamic population size in multiobjective evolutionary algorithms. In: Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC2002). pp. 1648–1653. IEEE Press, Piscataway (NJ) (2002)
12. Peschel, M., Riedel, C.: Use of vector optimization in multiobjective decision making. In: Bell, D.E., Keeney, R.L., Raiffa, H. (eds.) Conflicting Objectives in Decisions, pp. 97–121. Wiley, Chichester (1977)
13. Rudolph, G., Agapie, A.: Convergence properties of some multi-objective evolutionary algorithms. In: Zalzala, A., et al. (eds.) Proceedings of the 2000 Congress on Evolutionary Computation (CEC 2000), Vol. 2. pp. 1010–1016. IEEE Press, Piscataway (NJ) (2000)
14. Tan, K.C., Lee, T.H., Khor, E.F.: Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization. IEEE Transactions on Evolutionary Computation 5(6), 565–588 (2001)
15. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Ph.D. thesis, Swiss Federal Institute of Technology (ETH), Zurich, Zurich, Switzerland (November 1999)
16. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2), 173–195 (2000)
17. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In: Eiben, A.E. (ed.) Parallel Problem Solving from Nature (PPSN V). pp. 292–301. Springer, Berlin (1998)