New Ways to Calibrate Evolutionary Algorithms

Gusz Eiben and Martijn C. Schut

Department of Computer Science, Faculty of Science, VU University, Amsterdam,
The Netherlands.
ae.eiben@few.vu.nl
mc.schut@few.vu.nl

Abstract

The issue of setting the values of various parameters of an evolutionary algorithm (EA) is crucial for good performance. One way to do it is by controlling EA parameters on-the-fly, which can be done in various ways and for various parameters. We briefly review these options in general and present the findings of a literature search and some statistics about the most popular options. Thereafter, we provide three case studies indicating a high potential for uncommon variants. In particular, we recommend focusing on parameters regulating selection and population size, rather than those concerning crossover and mutation. On the technical side, the case study on adjusting tournament size shows by example that global parameters can also be self-adapted, and that heuristic adaptation and pure self-adaptation can be successfully combined into a hybrid of the two.

Key words: Parameter Control, Self-adaptive, Selection, Population Size

1 Introduction

In the early years of evolutionary computing the common opinion was that evolutionary algorithm (EA) parameters are robust. The general claim was that the performance of an EA does not depend heavily on the right parameter values for the given problem instance at hand. Over the last two decades the EC community realised that setting the values of EA parameters is crucial for good performance. One way to calibrate EA parameters is by controlling them on-the-fly, which can be done in various ways and for various parameters [13,16,18]. The purpose of this chapter is to present a general description of this field, identify the main stream of research, and argue for alternative approaches that do not fall in the main stream. This argumentation is based on three case studies published earlier [7,15,17].

The rest of the chapter is organised as follows. Section 2 starts off with giving a short recap of the most common classification of parameter control techniques. Then we continue in Sect. 3 with an overview of related work, including some statistics on what types of parameter control are most common in the literature. Section 4 presents three case studies that substantiate our argument regarding the choice of the parameter(s) to be controlled. Section 5 concludes the chapter.

2 Classification of Parameter Control Techniques

In classifying parameter control techniques of an evolutionary algorithm, many aspects can be taken into account [1,13,16,18,53]. In this chapter we consider the three most important ones:

1. *What* is changed (e.g., representation, evaluation function, operators, selection process, mutation rate, population size, and so on)?
2. *How* the change is made (i.e., deterministic heuristic, feedback-based heuristic, or self-adaptive)?
3. *The evidence* upon which the change is carried out (e.g., monitoring performance of operators, diversity of the population, and so on)?

Each of these is discussed in the following.

2.1 *What* is Changed?

To classify parameter control techniques from the perspective of what component or parameter is changed, it is necessary to agree on a list of all major components of an evolutionary algorithm, which is a difficult task in itself. For that purpose, let us assume the following components of an EA:

- Representation of individuals
- Evaluation function
- Variation operators and their probabilities
- Selection operator (parent selection or mating selection)
- Replacement operator (survival selection or environmental selection)
- Population (size, topology, etc.)

Note that each component can be parameterised, and that the number of parameters is not clearly defined. For example, an offspring $\overline{v}$ produced by an arithmetical crossover of k parents $\overline{x}_1, \ldots, \overline{x}_k$ can be defined by the following formula:

$$\overline{v} = a_1\overline{x}_1 + \ldots + a_k\overline{x}_k,$$

where $a_1, \ldots, a_k$, and k can be considered as parameters of this crossover. Parameters for a population can include the number and sizes of subpopulations, migration rates, and so on for a general case, when more than one population is involved. Despite the somewhat arbitrary character of this list of components and of the list of parameters of each component, the "what-aspect" is one of the main classification features, since this allows us to locate where a specific mechanism has its effect.

2.2 *How* are Changes Made?

Methods for changing the value of a parameter (i.e., the "how-aspect") can be classified into: **parameter tuning** and **parameter control**. By parameter tuning we mean the commonly practised approach that amounts to finding good values for the parameters *before* the run of the algorithm and then running the algorithm using these values, which remain fixed during the run. Parameter control forms an alternative,

as it amounts to starting a run with initial parameter values that are changed *during* the run.

We can further classify parameter control into one of the three following categories: deterministic, adaptive and self-adaptive. This terminology leads to the taxonomy illustrated in Fig. 1.

Deterministic parameter control This takes place when the value of a strategy parameter is altered by some deterministic rule. This rule fires at fixed moments, predetermined by the user (which explains the name "deterministic") and causes a predefined change without using any feedback from the search. Usually, a time-varying schedule is used, i.e., the rule is used when a set number of generations have elapsed since the last time the rule was activated.

Adaptive parameter control This takes place when there is some form of feedback from the search that serves as inputs to a mechanism used to determine the direction or magnitude of the change to the strategy parameter. The assignment of the value of the strategy parameter may involve credit assignment, based on the quality of solutions discovered by different operators/parameters, so that the updating mechanism can distinguish between the merits of competing strategies. Although the subsequent action of the EA may determine whether or not the new value persists or propagates throughout the population, the important point to note is that the updating mechanism used to control parameter values is externally supplied, rather than being part of the "standard" evolutionary cycle.

Self-adaptive parameter control The idea of the evolution of evolution can be used to implement the self-adaptation of parameters [6]. Here the parameters to be adapted are encoded into the chromosomes and undergo mutation and recombination. The better values of these encoded parameters lead to better individuals, which in turn are more likely to survive and produce offspring and hence propagate these better parameter values. This is an important distinction between adaptive and self-adaptive schemes: in the latter the mechanisms for the credit assignment and updating of different strategy parameters are entirely implicit, i.e., they are the selection and variation operators of the evolutionary cycle itself.

2.3 What *Evidence* Informs the Change?

The third criterion for classification concerns the evidence used for determining the change of parameter value [49, 51]. Most commonly, the progress of the search is

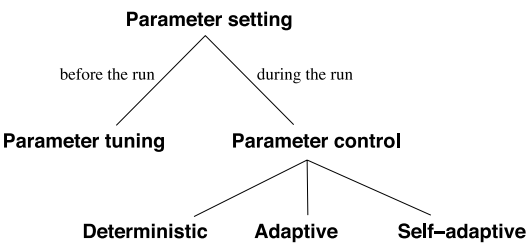

Fig. 1 Global taxonomy of parameter setting in EAs

monitored, e.g., by looking at the performance of operators, the diversity of the population, and so on. The information gathered by such a monitoring process is used as feedback for adjusting the parameters. From this perspective, we can make further distinction between the following two cases:

Absolute evidence We speak of absolute evidence when the value of a strategy parameter is altered by some rule that is applied when a predefined event occurs. The difference from deterministic parameter control lies in the fact that in deterministic parameter control a rule fires by a deterministic trigger (e.g., time elapsed), whereas here feedback from the search is used. For instance, the rule can be applied when the measure being monitored hits a previously set threshold – this is the event that forms the evidence. Examples of this type of parameter adjustment include increasing the mutation rate when the population diversity drops under a given value [38], changing the probability of applying mutation or crossover according to a fuzzy rule set using a variety of population statistics [37], and methods for resizing populations based on estimates of schemata fitness and variance [52]. Such mechanisms require that the user has a clear intuition about how to steer the given parameter into a certain direction in cases that can be specified in advance (e.g., they determine the threshold values for triggering rule activation). This intuition may be based on the encapsulation of practical experience, data-mining and empirical analysis of previous runs, or theoretical considerations (in the order of the three examples above), but all rely on the implicit assumption that changes that were appropriate to make on *another* search of *another* problem are applicable to *this* run of the EA on *this* problem.

Relative evidence In the case of using relative evidence, parameter values are compared according to the fitness of the offspring that they produce, and the better values get rewarded. The direction and/or magnitude of the change of the strategy parameter is not specified deterministically, but relative to the performance of other values, i.e., it is necessary to have more than one value present at any given time. Here, the assignment of the value of the strategy parameter involves credit assignment, and the action of the EA may determine whether or not the new value persists or propagates throughout the population. As an example, consider an EA using more crossovers with crossover rates adding up to 1.0 and being reset based on the crossovers performance measured by the quality of offspring they create. Such methods may be controlled adaptively, typically using "bookkeeping" to monitor performance and a user-supplied update procedure [11, 32, 45], or self-adaptively [5, 23, 35, 47, 50, 53] with the selection operator acting indirectly on operator or parameter frequencies via their association with "fit" solutions.

2.4 Summary

Our classification of parameter control methods is three-dimensional. The *component* dimension consists of six categories: representation, evaluation function, variation operators (mutation and recombination), selection, replacement, and population. The other dimensions have respectively three (deterministic, adaptive, self-adaptive) and two categories (absolute, relative). Their possible combinations are

given in Table 1. As the table indicates, deterministic parameter control with relative evidence is impossible by definition, and so is self-adaptive parameter control with absolute evidence. Within the adaptive scheme both options are possible and are indeed used in practice.

3 Related Work

We conducted a literature review to get an overview of the work that has been done on the various parameters of evolutionary algorithms of the last decade. Our aim was not to deliver a fully annotated bibliography, but rather to illuminate some examples from the literature on this topic. The literature spans the conference proceedings of three major EC conferences: GECCO (1999–2006), CEC (1999–2006) and PPSN (1990–2006). In total we found 235 papers that were concerned, in any way (thus not necessarily (self-)adaptive), with one of the parameters of EAs mentioned above: representation, initialisation, evaluation function, variation operators, selection and population size. (In addition, we found 76 papers about adaptive EAs in general.) We categorised the 235 papers, the result of which is shown in Fig. 2. We consider this a preliminary overview giving some indication of the distribution of research effort spent on these issues. The histogram clearly shows that much research

Table 1 Refined taxonomy of parameter setting in EAs: types of parameter control along the type and evidence dimensions. The – entries represent meaningless (nonexistent) combinations

	Deterministic	Adaptive	Self-adaptive
Absolute	+	+	–
Relative	–	+	+

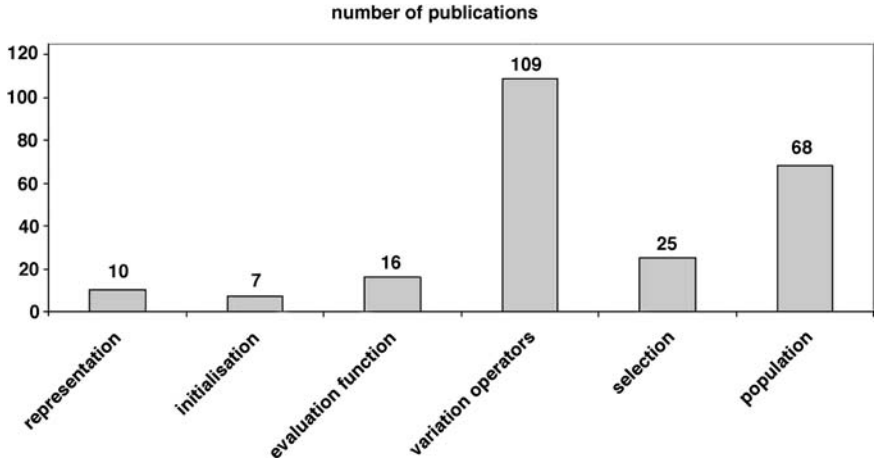

Fig. 2 Publication histogram

effort is spent on the variation operators (in general: 25, mutation: 54, crossover: 30). Also, the population parameter is well researched. However, we are aware of the fact that this number is biased, because it includes papers that are somewhat out of the scope of this chapter: for example, on population control in genetic programming, on the island-model of (sub)populations and on distributing (sub)populations in parallel evolutionary algorithms. We did not include papers on co-evolution.

We briefly discuss each EA parameter here, where we focus on the papers that explicitly look at (self-) adaptivity of the parameters. If possible, we make a distinction between deterministic, and self- adaptation within the discussion of a parameter.

3.1 Representation

Concerning representation, the *genome length* can be taken as a variable during an evolutionary run [28, 36, 43, 56]. Consider Ramsey et al. [43] who investigate a variable length genome under different mutation rates. To the suprise of the authors, the length of individuals self-adapts in direct response to the applied mutation rate. When tested with a broad range of mutation rates, the length tends to increase dramatically in the beginning and then decrease to a level corresponding to the mutation rate.

In earlier work, Harvey [28] presents an important property of variable-length genomes: "the absolute position of some symbols on the genotype can usually no longer be used to decide what feature those symbols relate to." Harvey sketches SAGA: a framework that was constructed to investigate the dynamics of a GA when genotype lengths are allowed to increase. The framework includes a particular crossover operator (SAGA cross) that has the requirement that the similarities are maximised between the two left segments that are swapped and between the two right segments that are swapped. This results in a computationally efficient algorithm where populations largely converge.

Stringer and Wu [56] show that a variable-length GA can evolve to shorter average size populations. This is observed when: (1) selection is absent from the GA, or (2) when selection focuses on some other property not influenced by the length of individuals. The model starts with an integer array of 100 elements, where each element represents an individual and the value denotes an individual's chromosome length. A simulated crossover produces children from two random parents, where the value of the first child equals the first (random) crossover point plus the value of the second parent less the second (random) crossover point; a similar procedure is used for the second child.

3.2 Initialisation

Although making the initialisation adaptive may seem contradictory (i.e., what should it adapt to initially?), there is a significant amount of literature dedicated to *dynamic restart strategies* for EAs [24,31,41,48]. This can be understood as (self-)adaptive initialisation.

Fukunga [24] shows how to find a good restart strategy in the context of resource-bounded scenarios: where the goal is to generate the best possible solution given a fixed amount of time. A new strategy is proposed that works based on a database of past performance of the algorithm on a class of problem instances. The resulting static restart strategy is competitive with those that are based on detection of lack of progress. This static strategy is an example of deterministic parameter control and it is surprising that it ourperforms the dynamic variant. According to the authors, one reason for this is that the dynamic variant can only consider local information of the current run.

Jansen [31] compares very simple deterministic strategies on a number of examples with different properties. The comparison is done in terms of a theoretical investigation on expected runtimes of various strategies. The strategies are more complex than fixing some point of time for a restart, but less complex than adaptive restart strategies. Two classes of dynamic restart strategies are presented and one additive and one multiplicative strategy is investigated in detail.

Finally, re-initialisation can also be considered in parallel genetic algorithms. Sekaj [48] researches this: the effect of re-initialisation is analysed with respect to convergence of the algorithm. In parallel genetic algorithms, (sub)populations may periodically *migrate* (as discussed later in the section about population). Sekaj lets re-initialisation happen when such migration took place. At re-initialisation, the current population was replaced by a completely new, randomly generated population. Additionally, two dynamic versions were presented: one where the algorithm after some number of generations compares the best individuals of each population and the worst population was re-initialised; and one which was based on the population diversity. Re-initialisation is able to remove differences between homogeneous and heterogeneous parallel GAs.

3.3 Evaluation Function

Regarding the evaluation function, some dynamic variants of this function are presented throughout the literature [19, 26, 34, 44].

The Stepwise Adaptation of Weights (SAW) technique has been introduced for problems that have a fitness function composed as a weighted sum of some atomic measure of quality. For instance, problems involving constraint satisfaction and data mining belong to this class, where the atomic measure can be the satisfaction of one given constraint or the correct classification of one given data record [77,80]. SAWing is an adaptive technique that adjusts the weights in the sum periodically at predefined intervals by increasing those that belong to "wrong" items, that is, to unsatisfied contraints or ill-classified records. Hereby SAWing effectively changes the fitness function and allows the algorithm to "concentrate" on the difficult cases. SAWing has been shown to work well on constraint satisfaction problems and data mining [78, 79, 81].

Reis et al. [44] propose and analyse a fractional-order dynamic fitness function: a fitness function based on fractional calculus. Besides the "default" fitness, the function has a component that represents the gain of the dynamical term. This dynamic

function is compared with a static function that includes a discontinuity measurement. The comparison has been done in the domain of electronic circuit design. Both variants outperform the standard fitness algorithm.

Within the area of constraint optimisation problems, Kazarlis and Petridis [34] propose a technique where the problem constraints are included in the fitness function as penalty terms. During the GA evolution these terms are varied, such that the location of a global optimum is facilitated and local optima are avoided. In addition to a static penalty assigment method (which is more often used in these types of problems), an increasing function is introduced that depends on the evolution time. This function can be linear, exponential, square, cubic, quartic or 5-step. For the particular problems under research (Cutting and Packing and Unit Commitment Problem), it was found that the square function was the optimum increase rate of the penalty term (and not the linear function that was expected by the authors).

For the satisfiability problem, Gottlieb and Voss [26] compare three approaches based on adapting weights, where weights indicate the relative importance of a constraint in a particular satisfiability problem. Adaptation takes place after a fixed number of fitness evaluations. One of the three approaches yielded overall optimal performance that exploits SAT-specific knowledge.

It is noteworthy to mention that the dynamics of a fitness function can also be understood the other way around: where the fitness function is taken as being a dynamic one (because the problem is dynamic) and the EA has to deal with this. In such a case, the fitness function is thus not (self-)adaptive. For example, Eriksson and Olsson [19] propose a hybrid strategy to locate and track a moving global optimum.

3.4 Variation Operators

By far, most research effort in (self-)adaptive parameter control is spent on the variation operators: mutation and crossover. Although there are many papers about tuning the parameter values of the operator rates, a significant number look into (self-)adaptive parameter value control for mutation and crossover.

There are approximately a dozen papers that discuss the (self-)adaptive parameter control for *both* operators. Smith and Fogarty [50] use Kauffman's NK model to self-adapt the crossover and mutation rates. The method puts a mutation rate in the chromosome itself and lets the (global) mutation rate be based on some aggregated value of the mutation rates of the individuals. The authors compare their new method to a number of frequently used crossover operators with standard mutation rates. The results are competitive on simple problems, and significantly better on the most complex problems. Ho et al. [30] present a probabilistic rule-based adaptive model in which mutation and crossover rates are adapted during a run. The model works based on subpopulations that each use different (mutation or crossover or both) rates and good rates emerge based on the performance of the subpopulations. (Although called 'self-adaptive' by the authors, in our earlier mentioned terminology, this model is adaptive and not self-adaptive.) Finally, Zhang et al. [62] let the crossover and mutation rate adapt based on the application of K-means clustering. The population is

clustered and rates are inferred based on the relative sizes of the clusters containing the best and worst chromosomes.

Regarding *mutation*, we see that the mutation operator is undoubtedly one of the most researched parameters according to our overview: 54 papers have been published about it. The topics of these papers range from introducing new mutation operators to the convergence of fitness gain effects of particular operators. Of these papers, 26 are specifically about (self-)adaptive mutation operators. We give some examples of these mutation operator studies. One of the earliest references in our overview is Bäck [4] who presents a self-adaptation mechanism of mutation rates. The rates are included in the individual itself. The methods enables a near-optimal schedule for the mutation rate. More recently, Katada et al. [33] looked at the domain of evolutionary robotics, where they tested a GA whose effective mutation rate changed according to the history of the genetic search. The individuals are neural networks that evolve over time. The control task was motion pattern discrimination. The variable mutation rate strategy shows better performance in this task, and this benefit was more pronounced with a larger genetic search space. Finally, Arnold [3] shows that *rescaled* mutations can be adaptively generated yielding robust and nearly optimal performance in problems with a range of noise strengths. Rescaling mutation has been suggested earlier in the literature, but this paper specifically discusses an adaptive approach for determining the scaling factor.

Concerning *crossover*, we briefly describe two studies. White and Oppacher [59] use automata to allow adaptation of the crossover operator probability during the run. The basic idea is to identify groups of bits within an individual that should be kept together during a crossover. An automaton encoded the probability that a given bit will be exchanged with the other parent under the crossover operators. First experiments show that the new crossover yields satisfactory results. The second study was undertaken by Vekaria and Clack [57] who investigate a number of biases to characterise adaptive recombination operators: directional – if alleles are either favoured or not for their credibility; credit – degree at which an allele becomes favoured; initialisation – if alleles are favoured without knowing their credibility; and hitchhiker – if alleles become favoured when they do not contribute to the fitness increase. Some experiments show, among other results, that initialisation bias (without mutation) does improve genetic search. Overall, the authors conclude that the biases are not always beneficial.

3.5 Selection

The majority of the 25 papers that we found with general reference to the selection mechanism of EAs are not about (self-)adaptive selection, but address rather a wide range of topics: e.g., without selection, effects of selection schemes, types of selection (clonal, anisotropic), and so forth. A stricter search shows that most studies that we categorised as being about (self-)adaptive selection actually refer to another EA parameter, for example, the mutation rate or the evaluation function. We only found one paper about (self-)adaptive survivor selection as defined in the terminology above.

Gorges-Schleuter [25] conducts a comparative study of global and local selection in evolution strategies. Traditionally, selection is a global parameter. In the so-called *diffusion model* for EAs, the individuals *only* exhibit local behaviour and the selection of partners for recombination and the selection individuals for survival are restricted to geographically nearby individuals. Local selection works then as follows: the first generated child is included in the next population whatsoever, each next child has to be better than its parent in order to be included in the next population.

3.6 Population

The population (size) parameter scores second-highest in our chapter overview. We already mentioned that this number is somewhat biased, because a number of the papers are about topics that are outside the scope of this chapter. General research streams that we identified regarding the population parameters with the 68 papers are: measuring population diversity, population size tuning, island model and migration parameter, ageing individuals, general effects of population size, population control in genetic programming, adaptive populations in particle swarm optimisation. For some topics, e.g., multi-populations or parallel populations, it is actually the *evaluation function* that is variable and not the population (size) itself – although at first sight the population size seems the varied parameter. This can also be said about co-evolutionary algorithms.

Approximately 10 of the 68 papers are specifically about (self-)adaptive population size. Later in this chapter, we discuss a number of these papers. Here, we briefly mention two other such papers. First, Lu and Yen [39] propose a dynamic multi-objective EA in which population growing and decline strategies are designed including a number of indicators that trigger the adaptive strategies. The EA is shown to be effective with respect to the population size and the diversity of the individuals. Secondly, Fernandes and Rosa [22] combine a dynamic reproduction rate based on population diversity, an ageing mechanism and a particular type of (macro-)mutation into one mechanism. The resulting mechanism is tested in a range of problems and shown to be superior in finding global optima.

4 Case Studies

We include three case studies that illustrate the benefits of (self-)adapting EA parameters. The first case study considers self-adaptive mutation and crossover and adaptive population size. The second study looks at on-the-fly population size adjustment. The third case study considers (self-)adaptive selection.

Throughout all case studies, we consider three important performance measures that reflect algorithm speed, algorithm certainty, and solution quality. The speed of optimization is measured by the Average number of Evaluations on Success (AES), showing how many evaluations it takes on average for the successful runs to find the optimum. The Success Rate (SR) shows how many of the runs were successful in

finding the optimum. If the GA is somewhat unsuccessful ($SR < 100\%$), the measurement MBF (Mean Best Fitness) shows how close the GA can come to the optimum. If $SR = 100\%$, then the MBF will be 0, because every run found the optimum 0. (The MBF includes the data of all runs in it, the successful and the unsuccessful ones.)

4.1 An Empirical study of GAs "Without Parameters"

An empirical study on GAs "without parameters" by Bäck, Eiben and van der Vaart [7] can be considered as the starting point of the research this very chapter is based upon. The research it describes aims at eliminating GA parameters by making them (self-)adaptive while keeping, or even improving, GA performance. The quotes in the title indicate that this aim is only partly achieved, because the mechanisms for eliminating GA parameters can have parameters themselves. The paper describes methods to adjust

- the mutation rate (self-adaptive, by an existing method after [4]),
- the crossover rate (self-adaptive, by a newly invented method),
- the population size (adaptive, by an existing method after [2], [42, pp. 72–80])

on-the-fly, during a GA run.

The method to change mutation rate is self-adaptive. The mutation rate is encoded in extra bits at the tail of every individual. For each member in the starting population the rate is completely random within a given range. Mutation then takes place in two steps. First only the bits that encode the mutation rate are mutated and immediately decoded to establish the new mutation rate. This new mutation rate is applied to the main bits (those encoding a solution) of the individual. Crossover rates are also self-adaptive. A value between 0 and 1 is coded in extra bits at the tail of every individual (initialised randomly). When a member of the population is selected for reproduction by the tournament selection, a random number r below 1 is compared with the member's p_c . If r is lower than p_c, the member is ready to mate. If both selected parents are ready to mate two children are created by uniform crossover, mutated and inserted into the population. If it is not lower, the member will only be subject to mutation to create one child which undergoes mutation and survivor selection immediately. If both parents reject mating, the two children are created by mutation only. If one parent is willing to mate and the other one does not, then the parent that is not in for mating is mutated to create one offspring, which is inserted in the population immediately. The willing parent is on hold and the next parent selection round only picks one other parent. Population size undergoes changes through an adaptive scheme, based on the maximum-lifetime idea. Here every new individual is allocated a remaining lifetime (RLT) between the allowable minimum and maximum lifetime (MinLT and MaxLT) at birth. The RLT depends on the individual's fitness at the time of birth, related to other individuals in the population. In each cycle (roughly: generation), the remaining lifetime of all the members in the population is decremented by one. There is only one exception for the fittest member, whose

RLT is left unchanged. If the RLT of an individual reaches zero it is removed from the population.

The three methods to adjust parameters on-the-fly are then added to a traditional genetic algorithm and their effect on GA performance is investigated experimentally. The experimental comparison includes 5 GAs: a simple GA as benchmark, three GAs featuring only one of the parameter adjusting mechanisms and a GA that applies all three mechanisms and is therefore almost "parameterless". The experimental comparison is based on a test suite of five functions composed to comform to the guidelines in [8,60]: the sphere model, the generalised Rosenbrock function, the generalised Ackley function, the generalised Rastrigin function, and the fully deceptive six-bit function. All test functions are used with $n = 10$ dimensions and are scaled to have an optimal value of 0. We performed 30 runs for each condition.

In order to give a clear and compact overview of the performance of all GA variants we show the outcomes by ranking the GAs for each function in Table 2. To obtain a ranking, we award the best GA (fastest or closest to the minimum) one point, the second best GA two points, and so on, so the worst performing GA for a given function gets five points. If, for a particular function, two GAs finish very close to each other, we award them equally: add the points for both those rankings and divide that by two. After calculating these points for each function and each GA variant we add the points for all the functions to form a total for each GA. The GA with the least points has the best overall performance.

The overall competition ends in a close finish between the all-in GA as number one and AP-GA, the GA with adaptive population size, right on its heels. In this respect, the objective of the study has been achieved, using the all-in GA the user has fewer parameters[1] to set and gets higher performance than using the simple GA.

An unexpected outcome of this study is that adaptive population sizes proved to be the key feature to improve the benchmark traditional GA, TGA. Alone, or in combination with the self-adaptive variation operators, the mechanism to adjust the population size during a run causes a consequent performance improvement w.r.t. the benchmark GA. These outcomes give a strong indication that, contrary to past

Table 2 Ranking of the GAs (the labelling is obvious from the text)

	TGA	SAM-GA	SAX-GA	AP-GA	all-in GA
Sphere	2	5	3	1	4
Rosenbrock	3	5	4	4	2
Ackley	1	2.5	5	4	2.5
Rastrigin	2.5	5	4	2.5	1
Deceptive	4	5	2.5	2.5	1
Points	12.5	22.5	18.5	11	10.5
End rank	**3**	**5**	**4**	**2**	**1**

[1] This is not entirely true, since (1) *initial* values for those parameters must be given by the user and (2) the population adaptation method introduces two new parameters MinLT and MaxLT.

and present practice (where quite some effort is devoted to tuning or online controlling of the application rates of variance operators), studying control mechanisms for variable population sizes should be paid more attention.

After having published this paper, this conclusion has been generalised by distinguishing variation and selection as the two essential powers behind an evolutionary process [18, Chapter 2]. Here, variation includes recombination (crossover) and mutation; for selection we can further distinguish parent selection and survivor selection (replacement). Clearly, the population size is affected by the latter. From this perspective, the paper gives a hint that further to studying mechanisms for on-the-fly calibration of variation operators, the EC community should adopt a research agenda for on-the-fly selection control mechanisms, including those focusing on population size management.

4.2 Evolutionary Algorithms with On-the-Fly Population Size Adjustment

The investigation in [15] is a direct follow-up to [7] discussed in the previous section. As noted by Bäck et al. the population size is traditionally a rigid parameter in evolutionary computing. This is not only true in the sense that for the huge majority of EAs the population size remains constant over the run, but also for the EC research community that has not spent much effort on EAs with variable population sizes. However, there are biological and experimental arguments to expect that this would be rewarding. In natural environments, population sizes of species change and tend to stabilise around appropriate values according to factors such as natural resources and carrying capacity of the ecosystem. Looking at it technically, population size is the most flexible parameter in natural systems: it can be adjusted much more easily than, for instance, mutation rate.

The objective of this study is to perform an experimental evaluation of a number of methods for calibrating population size on-the-fly. Note that the paper does not consider (theory-based) strategies for *tuning* population size [65–68, 70, 74]. The inventory of methods considered for an experimental comparison includes the following algorithms from the literature. The Genetic Algorithm with Variable Population Size (GAVaPS) from Arabas [2], [42, pp. 72–80] eliminates population size as an explicit parameter by introducing the age and maximum lifetime properties for individuals. The maximum lifetimes are allocated at birth depending on fitness of the newborn, while the age (initialised to 0 at birth) is incremented at each generation by one. Individuals are removed from the population when their ages reach the value of their predefined maximum lifetime. This mechanism makes survivor selection unnecessary and population size an observable, rather than a parameter. The Adaptive Population size GA (APGA) is a variant of GAVaPS where a steady-state GA is used, and the lifetime of the best individual remains unchanged when individuals grow older [7]. In [27, 69] Harik and Lobo introduce a parameterless GA (PLGA) which evolves a number of populations of different sizes simultaneously. Smaller populations get more function evaluations, where population i is allowed to run four times more generations than the population $i+1$. If, however, a smaller population converges, the algorithm drops it. The Random Variation of the Population

Size GA (RVPS) is presented by Costa et al. in [63]. In this algorithm, the size of the actual population is changed every N fitness evaluations, for a given N. Hinterding, Michalewicz and Peachey [29] presented an adaptive mechanism, in which three sub-populations with different population sizes are used. The population sizes are adapted at regular intervals (*epochs*) biasing the search to maximise the performance of the group with the mid-most size. The criterion used for varying the sizes is fitness diversity. Schlierkamp-Voosen and Mühlenbein [72] use a competition scheme between sub-populations to adapt the size of the sub-populations as well as the overall population size. There is a quality criterion for each group, as well as a gain criterion, which dictates the amount of change in the group's size. The mechanism is designed in such a way that only the size of the best group can increase. A technique for dynamically adjusting the population size with respect to the probability of selection error, based on Goldberg's research [67], is presented in [73]. Finally, the following methods have been selected for the experimental comparison.

1. GAVaPS from [2],
2. GA with adaptive population size (APGA) from [7],
3. the parameterless GA from [27],
4. the GA with Random Variation of Population Size (RVPS) from [63],
5. the Population Resizing on Fitness Improvement GA (PRoFIGA), newly invented for this paper.

The new method is based on monitoring improvements of the best fitness in the population. On fitness improvement the EA is made more explorative by increasing the population size. If the fitness is not improving (for a short while) the population is made smaller. However, if stagnation takes too long, then the population size is increased again. The intuition behind this algorithm is related to (a rather simplified view on) exploration and exploitation. The bigger the population size, the more it supports explorative search. Because in the early stages of an EA run fitness typically increases, population growth, hence exploration, will be more prominent in the beginning. Later on it will decrease gradually. The shrinking phase is expected to "concentrate" more on exploitation of fewer individuals after reaching the limits of exploration. The second kind of population growth is supposed to initiate renewed exploration in a population that ist stuck in local optima. Initial testing has shown that GAVaPS was very sensitive for the *reproduction ratio* parameter and the algorithm frequently increased the size of the population over several thousand individuals, which resulted in unreliable performance. For this reason it was removed from further experimentation.

When choosing the test suite for experimentation popular, but ad hoc collections of objective functions are deliberately avoided for reasons outlined in [14] and [18, Chapter 14]. Instead, the multimodal problem generator of Spears [54] is used for it has been designed to facilitate systematic studies of GA behavior. This generator creates random problem instances, i.e., fitness landscapes over bit-strings, with a controllable size (chromosome length) and degree of multi-modality (number of peaks). The test suite consists of 10 different landscapes for 100 bits, where the number of peaks ranges from 1 to 1000 through 1, 2, 5, 10, 25, 50, 100, 250, 500, and 1000.

We performed 100 runs for each condition. Here again, the performance of the algorithms is assessed by Success Rate (SR), Mean Best Fitness (MBF), and the Average number of Evaluations to a Solution (AES). SR is an effectivity measure that gives the percentage of runs in which the optimum (the highest peak) was found. MBF is also an effectivity measure showing the average of the best fitness in the last population over all runs. AES is a measure of efficiency (speed): it is the number of evaluations it takes on average for the successful runs to find the optimum. If a GA has no success ($SR = 0$) then the AES measure is undefined.

The main results are given in the graphs with a grid background in Fig. 3 and the left-hand side of Fig. 4.

The AES plots are shown in Fig. 3 (left). These graphs show clear differences between the algorithms. There are, however, no significant differences between the problem instances when only looking at the speed curves (except for the parameterless GA). Apparently, finding a solution does not take more evaluations on a harder problem that has more peaks. (Although it should be noted that for harder problems the averages are taken over fewer runs, cf. the SR figures below, which reduces the reliability of the statistics.) This is an interesting artefact of the problem generator that needs further investigation. The increasing problem hardness, however, is clear from

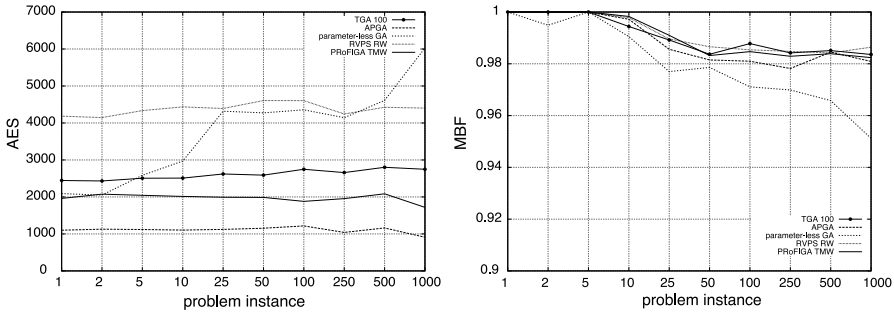

Fig. 3 AES (left) and MBF (right) of TGA, APGA, the parameterless GA, RVPS and PRoFIGA with max-eval = 10,000

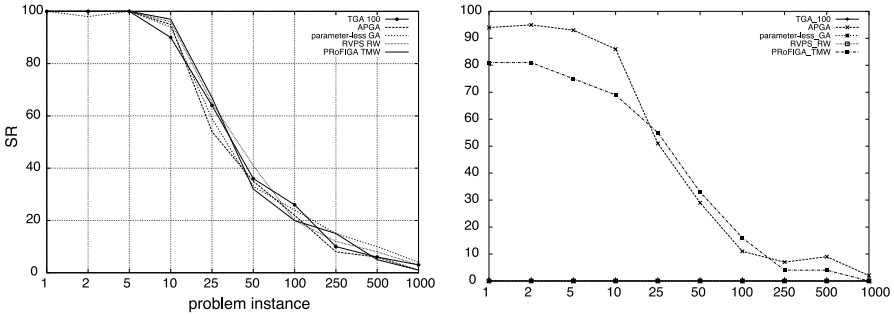

Fig. 4 SR of TGA, APGA, the parameterless GA, RVPS and PRoFIGA with max-eval = 10,000 (left) and with max-eval = 1500 (right)

the decreasing average quality of the best solution found (MBF), cf. Fig. 3 (right) and the decreasing probability of finding a solution (SR), cf. Fig. 4 (left).

We can rank the population (re)sizing methods based on the AES plots: APGA is significantly faster than the other methods, followed by PRoFIGA. The traditional GA comes third. The parameterless GA is only competitive for easy problems and the RVPS RW is clearly inferior to the other methods.

The SR and MBF results are quite homogeneous, with only one negative outlier, the parameterless GA. It seems that we cannot rank the algorithms by their effectivity. However, this homogeneity is a consequence of our choice of the maximum number of fitness evaluations in the termination criterion. Apparently it is "too" high allowing all contestants to reach the performance of the champions – be it slower. As a control experiment, we repeated all runs with the maximum number of fitness evaluations set to 1500. The resulting success rates are given in Fig. 4 (right), showing great differences. APGA and PRoFIGA obtain somewhat worse, but comparable SR results as before, but the other algorithms never find a solution yielding SR = 0 over all peaks.

Looking at the results we can conclude that adapting population sizes in an EA can certainly pay off. The gains in terms of efficiency, measured by the number of fitness evaluations needed to find a solution, can be significant: the winner of our comparison (APGA) achieves the same success rate and mean best fitness as the traditional GA with less than half of the work, and even the second best (PRoFIGA) needs 20% fewer evaluations. The second series of experiments shows that such an increase in speed can be converted into increased effectivity, depending on the termination condition. Here again, the winner is APGA, followed by PRoFIGA. It should be noted that two GAs from this comparison (the parameterless GA and RVPS RW) are much slower than the traditional GA. Hence, on-the-fly population (re)sizing is not necessarily better than traditional hand-tuning of a constant population size. The added value depends on the actual implementation, i.e., on *how* the population size is adjusted. Another observation made here is that the lifetime principle used in APGA eliminates explicit survivor selection and makes population size an observable instead of a user parameter. However, it should also be noted that using this idea does not mean that the number of EA parameters is reduced. In fact, it is increased in our case: instead of the population size N in the TGA, the APGA introduces two new parameters, *MinLT* and *MaxLT*.

These results can be naturally combined with those of Bäck et al. confirming the superiority of APGA on another test suite. Of course, highly general claims are still not possible about APGA. But these results together form a strong indication that incorporating on-the-fly population (re)sizing mechanisms based on the lifetime principle in EAs is a very promising design heuristic definitely worth trying and that APGA is a successful implementation of this general idea.

4.3 Boosting Genetic Algorithms with (Self-) Adaptive Selection

The paper [17] seeks an answer to the question whether it is feasible (i.e., possible and rewarding) to self-adapt selection parameters in an evolutionary algorithm? Note that the idea seems quite impossible considering that

- Self-adaptation manipulates parameters defined within an *individual*, hence the given parameter will have different values over different members of the population.
- Parameters regarding selection (e.g., tournament size in tournament selection or the bias in ranked biased selection) are inherently *global*, any given value holds for the whole population, not only for an individual.

This explains why existing approaches to controlling such parameters are either deterministic or adaptive.

The paper investigates self-adaptation of tournament size in a purely self-adaptive fashion and a variant that combines self-adaptation with a heuristic. The approach is based on keeping tournament size K as a globally valid parameter, but decomposing it. That is, to introduce local parameters k at the level of individuals that can be self-adapted and establish the global value through aggregating the local ones. Technically, this means extending the individual's chromosomes by an extra gene resulting in $\langle x, k \rangle$. Furthermore, two methods are required. One, to specify how to aggregate local k values to a global one. Two, a mechanism for variation (crossover and mutation) of the local k values.

The aggregation mechanism is rather straightforward. Roughly speaking, the global parameter will be the sum of the local votes of all individuals. Here we present a general formula applicable for any global parameter P and consider tournament size K as a special case.

$$P = \lceil \sum_{i=1}^{N} p_i \rceil \tag{1}$$

where $p_i \in [p_{\min}, p_{\max}]$, $\lceil\ \rceil$ denotes the ceiling function, and N is the population size.

Concerning variation of the extended chromosomes, crossover and mutation are distinguished. Crossover works on the whole $\langle x, k \rangle$, by whichever mechanism the user wishes. Mutation, however, is split. The x part of $\langle x, k \rangle$ is mutated by any suitable mutation operator, but for the k part a specific mechanism is used. A straightforward option would be the standard self-adaptation mechanism of σ values from Evolution Strategies. However, those σ values are not bounded, while tournament size is obviously bounded by zero and the population size. A possible solution is the self-adaptive mechanism for mutation rates in GAs as described by Bäck and Schütz [9]. This mechanism is introduced for $p \in (0, 1)$ and it works as follows:

$$p' = \left(1 + \frac{1 - p}{p} \cdot e^{-\gamma \cdot N(0,1)}\right)^{-1} \tag{2}$$

where p is the parameter in question and γ is the learning rate, which allows for control of the adaptation speed. This mechanism has some desirable properties:

1. Changing $p \in (0, 1)$ yields a $p' \in (0, 1)$.
2. Small changes are more likely than large ones.

3. The expected change of p by repeatedly changing it equals zero (which is desirable, because natural selection should be the only force bringing a direction in the evolution process).
4. Modifying by a factor c occurs with the same probability as a modification by $1/c$.

The concrete mechanism for self-adaptive tournament sizes uses individual k values $k \in (0, 1)$ and the formula of Equation (2) with $\gamma = 0.22$ (as recommended in [9]). Note that if a GA uses a recombination operator then this operator will be applied to the tournament size parameter k, just as it is applied to other genes. In practice this means that a child created by recombination inherits an initial k value from its parents and the definitive value k is obtained by mutation as described by Equation (2).

Besides the purely self-adaptive mechanism for adjusting tournament sizes the chapter also introduces a heuristic variant. In the self-adaptive algorithm as described above the direction (+ or −) as well as the extent (increment/decrement) of the change are fully determined by the random scheme. This is a general property of self-adaptation. However, in the particular case of regulating selection pressure we do have some intuition about the direction of change. Namely, if a new individual is better than its parents then it should try to increase selection pressure, assuming that stronger selection will be advantageous for him, giving a reproductive advantage over less fit individuals. In the opposite case, if it is less fit than its parents, then it should try to lower the selection pressure. Our second mechanism is based on this idea. Formally, we keep the aggregation mechanism from equation 1 and use the following rule. If $\langle x, k \rangle$ is an individual to be mutated (either obtained by crossover or just to be reproduced solely by mutation), then first we create x' from x by the regular bitflips, then apply

$$k' = \begin{cases} k + \Delta k & \text{if} \quad f(\mathbf{x}') \geq f(\mathbf{x}) \\ k - \Delta k & \text{otherwise} \end{cases} \tag{3}$$

where

$$\Delta k = \left| k - \left(1 + \frac{1-k}{k} e^{-\gamma N(0,1)} \right)^{-1} \right| \tag{4}$$

with $\gamma = 0.22$.

This mechanism differs from "pure" self-adaptation because of the heuristic rule specifying the direction of the change. However, it could be argued that this mechanism is not a clean adaptive scheme (because the initial k values are inherited), nor a clean self-adaptive scheme (because the final k values are influenced by a user defined heuristic), but some hybrid form. For this reason we perceive and name this mechanism *hybrid self-adaptive* (HSA). All together this yields two new GAs: the GA with self-adaptive tournament size (GASAT) and the GA with hybrid self-adaptive tournament size (GAHSAT).

The experimental comparison of these GAs and a standard GA for benchmark is based on exactly the same test suite as the study in the previous section. The GAs are tested on the same landscapes in $\{0,1\}^{100}$ with 1, 2, 5, 10, 25, 50, 100, 250, 500 and 1000 peaks obtained through the Multimodal Problem Generator of Spears [54]. We performed 100 runs for each condition. Also the performance measures are identical: the Mean Best Fitness (MBF) and its standard deviation (SDMBF), the Average number of Evaluations to a Solution (AES) and its standard deviation (SDAES), and the Success Rate (SR) are calculated, based on 100 independent runs. The results for the SGA, GASAT, and GAHSAT are shown in Table 3, Table 4, and Table 5, respectively.

The outcomes indicate that GASAT has better performance than SGA, but it is not as powerful as the hybrid self-adaptive mechanism. The initial research question about the feasibility of on-the-fly adjustment of K can be answered positively. It is interesting to remark here that the self-adaptive GAs are based on a simple mechanism (that was, nota bene, introduced for mutation parameters) and apply no sophisticated twists to it. Yet, they show very good performance that compares favorably with the best GA in [15]. The comparison between the former winner, APGA with adaptive population size, and GAHSAT is shown in Table 6. Note that the MBF results are omitted for they showed no significant difference. This comparison shows that the GAHSAT is very competitive, running out the APGA on the smoother landscapes.

Table 3 Results of SGA

Peaks	SR	AES	SDAES	MBF	SDMBF
1	100	1478	191	1.0	0.0
2	100	1454	143	1.0	0.0
5	100	1488	159	1.0	0.0
10	93	1529	168	0.9961	0.0142
25	62	1674	238	0.9885	0.0174
50	37	1668	221	0.9876	0.0140
100	22	1822	198	0.9853	0.0145
250	11	1923	206	0.9847	0.0137
500	6	2089	230	0.9865	0.0122
1000	5	2358	398	0.9891	0.0100

Table 4 End results of GASAT

Peaks	SR	AES	SDAES	MBF	SDMBF
1	100	1312	218	1.0	0.0
2	100	1350	214	1.0	0.0
5	100	1351	254	1.0	0.0
10	92	1433	248	0.9956	0.0151
25	62	1485	280	0.9893	0.0164
50	46	1557	246	0.9897	0.0128
100	21	1669	347	0.9853	0.0147
250	16	1635	336	0.9867	0.0130
500	3	1918	352	0.9834	0.0146
1000	1	1675	0	0.9838	0.0126

Table 5 End results of GAHSAT

Peaks	SR	AES	SDAES	MBF	SDMBF
1	100	989	244	1.0	0.0
2	100	969	206	1.0	0.0
5	100	1007	233	1.0	0.0
10	89	1075	280	0.9939	0.0175
25	63	1134	303	0.9879	0.0190
50	45	1194	215	0.9891	0.0127
100	14	1263	220	0.9847	0.0140
250	12	1217	166	0.9850	0.0131
500	7	1541	446	0.9876	0.0119
1000	4	1503	272	0.9862	0.0113

Table 6 Comparing GAHSAT and the winning APGA from [15]

	GAHSAT		APGA	
Peaks	SR	AES	SR	AES
1	100	989	100	1100
2	100	969	100	1129
5	100	1007	100	1119
10	89	1075	95	1104
25	63	1134	54	1122
50	45	1194	35	1153
100	14	1263	22	1216
250	12	1217	8	1040
500	7	1541	6	1161
1000	4	1503	1	910

5 Summary and Conclusions

The relevance of the above studies lies in the potential of on-the-fly adjustment of EA parameters that have not been widely considered in the past. The investigations reviewed here provide substantial evidence that on-line regulation of population size and selection can greatly improve EA performance. On the technical side, the case study on adjusting tournament size shows by example that global parameters can also be self-adapted, and that heuristic adaptation and pure self-adaptation can be successfully combined into a hybrid of the two.

On the general level, two things can be noted. First, we want to remark that parameter control in an EA can have two purposes. One motivation for controlling parameters on-the-fly is the assumption (observation) that in different phases of the search the given parameter should have different values for "optimal" algorithm performance. If this holds, then static parameter values are always inferior; for good EA performance one must vary this parameter. Another reason it can be done for is to avoid suboptimal algorithm performance resulting from suboptimal parameter values set by the user. The basic assumption here is that the algorithmic control mechanisms do this job better than the user could, or that they can do it approximately as good, but they liberate the user from doing it. Either way, they are beneficial.

The second thing we want to note is that making a parameter adaptive or self-adaptive does not necessarily mean that we have an EA with fewer parameters. For instance, in APGA the population size parameter is eliminated at the cost of introducing two new ones: the minimum and maximum lifetime of newborn individuals. If the EA performance is sensitive to these new parameters then such a parameter replacement can make things worse. But if the new parameters are less sensitive to accurate calibration, then the net effect is positive: the user can obtain a good algorithm with less effort spent on algorithm design. This latter is, however, hardly ever considered in evolutionary computing publications.

This phenomenon also occurs on another level. One could say that the mechanisms to adjust parameters are also (meta) parameters. For instance, the method that allocates lifetimes in APGA, or the function in Equation (2) specifying how the k values are mutated can be seen as high level parameters of the GA. It is in fact an assumption that these are well-chosen (smartly designed) and their effect is positive. Typically, there are more possibilities to obtain the required functionality, that is, there are possibly more well-working methods one can design. *Comparing different methods* implies experimental (or theoretical) studies very much like *comparing different parameter values* in a classical setting. Here again, it can be the case that algorithm performance is not so sensitive to details of this (meta) parameter, which can fully justify this approach.

Finally, let us place the issue of parameter control in a larger perspective. Over the last two decades the EC community has come to realise that EA performance, to a large extent, depends on well-chosen parameter values, which in turn may depend on the problem (instance) to be solved. In other words, it is now acknowledged that EA parameters need to be calibrated to specific problems and problem instances. Ideally, it should be the algorithm that performs the necessary problem-specific ad-

justments. Ultimately, it would be highly desirable to utilise the inherent adaptive power of an evolutionary process for calibrating *itself* to a certain problem instance, while solving that very problem instance. We believe that the extra computational overhead (i.e., solving the self-calibration problem additionally to the given technical problem) will pay off and hope to see more research on this issue.

References

1. P.J. Angeline. Adaptive and self-adaptive evolutionary computations. In *Computational Intelligence*, pages 152–161. IEEE Press, 1995
2. J. Arabas, Z. Michalewicz, and J. Mulawka. GAVaPS – a genetic algorithm with varying population size. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 73–78. IEEE Press, Piscataway, NJ, 1994
3. D.V. Arnold. Evolution strategies with adaptively rescaled mutation vectors. In *2005 Congress on Evolutionary Computation (CEC'2005)*, pages 2592–2599. IEEE Press, Piscataway, NJ, 2005
4. T. Bäck. The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In Männer and Manderick [40], pages 85–94
5. T. Bäck. Self adaptation in genetic algorithms. In F.J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the 1st European Conference on Artificial Life*, pages 263–271. MIT Press, Cambridge, MA, 1992
6. T. Bäck. Self-adaptation. In T. Bäck, D.B. Fogel, and Z. Michalewicz, editors, *Evolutionary Computation 2: Advanced Algorithms and Operators*, Chapter 21, pages 188–211. Institute of Physics Publishing, Bristol, 2000
7. T. Bäck, A.E. Eiben, and N.A.L. van der Vaart. An empirical study on GAs "without parameters". In Schoenauer et al. [46], pages 315–324
8. T. Bäck and Z. Michalewicz. Test landscapes. In T. Bäck, D.B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, Chapter B2.7, pages 14–20. Institute of Physics Publishing, Bristol, and Oxford University Press, New York, 1997
9. Th. Bäck and M. Schütz. Intelligent mutation rate control in canonical genetic algorithms. In Zbigniew W. Ras and Maciej Michalewicz, editors, *Foundations of Intelligent Systems, 9th International Symposium, ISMIS '96, Zakopane, Poland, June 9-13, 1996, Proceedings*, volume 1079 of *Lecture Notes in Computer Science*, pages 158–167. Springer, Berlin, Heidelberg, New York, 1996
10. Y. Davidor, H.-P. Schwefel, and R. Männer, editors. *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, number 866 in Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, New York, 1994
11. L. Davis. Adapting operator probabilities in genetic algorithms. In J.D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 61–69. Morgan Kaufmann, San Francisco, 1989
12. A.E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors. *Proceedings of the 5th Conference on Parallel Problem Solving from Nature*, number 1498 in Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, New York, 1998
13. A.E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.
14. A.E. Eiben and M. Jelasity. A critical note on experimental research methodology in EC. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'2002)*, pages 582–587. IEEE Press, Piscataway, NJ, 2002

15. A.E. Eiben, E. Marchiori, and V.A. Valko. Evolutionary algorithms with on-the-fly population size adjustment. In X. Yao et al., editor, *Parallel Problem Solving from Nature, PPSN VIII*, number 3242 in Lecture Notes in Computer Science, pages 41–50. Springer, Berlin, Heidelberg, New York, 2004
16. A.E. Eiben, Z. Michalewicz, M. Schoenauer, and J.E. Smith. Parameter Control in Evolutionary Algorithms. In Lobo, Fernando G., Lima, Cláudio F. and Michalewicz, Zbigniew, editors, *Parameter Setting in Evolutionary Algorithms*, Studies in Computational Intelligence. Springer, 2007, pages 19–46
17. A.E. Eiben, M.C. Schut, and A.R. deWilde. Boosting genetic algorithms with (self-) adaptive selection. In *Proceedings of the IEEE Conference on Evolutionary Computation*, 2006, pages 1584–1589
18. A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin, Heidelberg, New York, 2003
19. R. Eriksson and B. Olsson. On the performance of evolutionary algorithms with life-time adaptation in dynamic fitness landscapes. In *2004 Congress on Evolutionary Computation (CEC'2004)*, pages 1293–1300. IEEE Press, Piscataway, NJ, 2004
20. L.J. Eshelman, editor. *Proceedings of the 6th International Conference on Genetic Algorithms*. Morgan Kaufmann, San Francisco, 1995
21. H.-G. Beyer et al., editor. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005)*. ACM, 2005
22. C. Fernandes and A. Rosa. Self-regulated population size in evolutionary algorithms. In Th.-P. Runarsson, H.-G. Beyer, E. Burke, J.-J. Merelo-Guervos, L. Darell Whitley, and X. Yao, editors, *Parallel Problem Solving from Nature – PPSN IX*, number 4193 in Lecture Notes in Computer Science, pages 920–929. Springer, Berlin, Heidelberg, New York, 2006
23. D.B. Fogel. *Evolutionary Computation*. IEEE Press, 1995
24. A.S. Fukunga. Restart scheduling for genetic algorithms. In Eiben et al. [12], pages 357–366
25. M. Gorges-Schleuter. A comparative study of global and local selection in evolution strategies. In Eiben et al. [12], pages 367–377
26. J. Gottlieb and N. Voss. Adaptive fitness functions for the satisfiability problem. In Schoenauer et al. [46], pages 621–630
27. Georges R. Harik and Fernando G. Lobo. A parameter-less genetic algorithm. In Wolfgang Banzhaf et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 258–265. Morgan Kaufmann, 1999
28. I. Harvey. The saga-cross: the mechanics of recombination for species with variable-length genotypes. In Männer and Manderick [40], pages 269–278
29. R. Hinterding, Z. Michalewicz, and T.C. Peachey. Self-adaptive genetic algorithm for numeric functions. In Voigt et al. [58], pages 420–429
30. C.W. Ho, K.H. Lee, and K.S. Leung. A genetic algorithm based on mutation and crossover with adaptive probabilities. In *1999 Congress on Evolutionary Computation (CEC'1999)*, pages 768–775. IEEE Press, Piscataway, NJ, 1999
31. T. Jansen. On the analysis of dynamic restart strategies for evolutionary algorithms. In J.J. Merelo Guervos, P. Adamidis, H.-G. Beyer, J.-L. Fernandez-Villacanas, and H.-P. Schwefel, editors, *Proceedings of the 7th Conference on Parallel Problem Solving from Nature*, number 2439 in Lecture Notes in Computer Science, pages 33–43. Springer, Berlin, Heidelberg, New York, 2002
32. B.A. Julstrom. What have you done for me lately?: Adapting operator probabilities in a steady-state genetic algorithm. In Eshelman [20], pages 81–87
33. Y. Katada, K. Okhura, and K. Ueda. An approach to evolutionary robotics using a genetic algorithm with a variable mutation rate strategy. In Yao et al. [61], pages 952–961

34. S. Kazarlis and V. Petridis. Varying fitness functions in genetic algorithms: studying the rate of increase of the dynamics penalty terms. In Eiben et al. [12], pages 211–220
35. N. Krasnogor and J.E. Smith. Emergence of profitable search strategies based on a simple inheritance mechanism. In Spector et al. [55], pages 432–439
36. C.-Y. Lee and E.K. Antonsson. Adaptive evolvability via non-coding segment induced linkage. In Spector et al. [55], pages 448–453
37. M. Lee and H. Takagi. Dynamic control of genetic algorithms using fuzzy logic techniques. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 76–83. Morgan Kaufmann, San Francisco, 1993
38. J. Lis. Parallel genetic algorithm with dynamic control parameter. In *Proceedings of the 1996 IEEE Conference on Evolutionary Computation*, pages 324–329. IEEE Press, Piscataway, NJ, 1996
39. H. Lu and G.G. Yen. Dynamic population size in multiobjective evolutionary algorithm. In *2002 Congress on Evolutionary Computation (CEC'2002)*, pages 1648–1653. IEEE Press, Piscataway, NJ, 2002
40. R. Männer and B. Manderick, editors. *Proceedings of the 2nd Conference on Parallel Problem Solving from Nature*. North-Holland, Amsterdam, 1992
41. K.E. Mathias, J.D. Schaffer, L.J. Eshelman, and M. Mani. The effects of control parameters and restarts on search stagnation in evolutionary programming. In Eiben et al. [12], pages 398–407
42. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin, Heidelberg, New York, 3rd edition, 1996
43. C.L. Ramsey, K.A. de Jong, J.J. Grefenstette, A.S. Wu, and D.S. Burke. Genome length as an evolutionary self-adaptation. In Eiben et al. [12], pages 345–353
44. C. Reis, J.A. Tenreiro Machado and J. Boaventura Cunha. Fractional dynamic fitness functions for ga-based circuit design. In Beyer et al. [21], pages 1571–1572
45. D. Schlierkamp-Voosen and H. Mühlenbein. Strategy adaptation by competing subpopulations. In Davidor et al. [10], pages 199–209
46. M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H.-P. Schwefel, editors. *Proceedings of the 6th Conference on Parallel Problem Solving from Nature*, number 1917 in Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, New York, 2000
47. H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, volume 26 of *ISR*. Birkhaeuser, Basel/Stuttgart, 1977
48. I. Sekaj. Robust parallel genetic algorithms with re-initialisation. In Yao et al. [61], pages 411–419
49. J.E. Smith. *Self Adaptation in Evolutionary Algorithms*. PhD Thesis, University of the West of England, Bristol, UK, 1998
50. J.E. Smith and T.C. Fogarty. Adaptively parameterised evolutionary systems: Self adaptive recombination and mutation in a genetic algorithm. In Voigt et al. [58], pages 441–450
51. J.E. Smith and T.C. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing*, 1(2):81–87, 1997
52. R.E. Smith and E. Smuda. Adaptively resizing populations: Algorithm, analysis and first results. *Complex Systems*, 9(1):47–72, 1995
53. W.M. Spears. Adapting crossover in evolutionary algorithms. In J.R. McDonnell, R.G. Reynolds, and D.B. Fogel, editors, *Proceedings of the 4th Annual Conference on Evolutionary Programming*, pages 367–384. MIT Press, Cambridge, MA, 1995
54. W.M. Spears. *Evolutionary Algorithms: the Role of Mutation and Recombination*. Springer, Berlin, Heidelberg, New York, 2000

55. L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. Morgan Kaufmann, San Francisco, 2001
56. H. Stringer and A.S. Wu. Behavior of finite population variable length genetic algorithms under random selection. In Beyer et al. [21], pages 1249–1255
57. K. Vekaria and C. Clack. Biases introduced by adaptive recombination operators. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, pages 670–677. Morgan Kaufmann, San Francisco, 1999
58. H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors. *Proceedings of the 4th Conference on Parallel Problem Solving from Nature*, number 1141 in Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, New York, 1996
59. T. White and F. Oppacher. Adaptive crossover using automata. In Davidor et al. [10], pages 229–238
60. D. Whitley, K. Mathias, S. Rana, and J. Dzubera. Building better test functions. In Eshelman [20], pages 239–246
61. X. Yao, E. Burke, J.A. Lozano, J. Smith, J.-J. Merelo-Guervos, J.A. Bullinaria, J. Rowe, P. Tino, A. Kaban, and H.-P. Schwefel, editors. *Parallel Problem Solving from Nature – PPSN-VIII*, number 3242 in Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, New York, 2004
62. J. Zhang, S.H. Chung, and J. Zhong. Adaptive crossover and mutation in genetic algorithms based on clustering technique. In Beyer et al. [21], pages 1577–1578
63. J. Costa, R. Tavares, and A. Rosa. An experimental study on dynamic random variation of population size. In *Proc. IEEE Systems, Man and Cybernetics Conf.*, volume 6, pages 607–612, Tokyo, 1999. IEEE Press
64. S. Forrest, editor. *Proceedings of the 5th International Conference on Genetic Algorithms*. Morgan Kaufmann, San Francisco, 1993
65. D.E. Goldberg. Optimal population size for binary-coded genetic algorithms. TCGA Report No. 85001, 1985
66. D.E. Goldberg. Sizing populations for serial and parallel genetic algorithms. In J.D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 70–79. Morgan Kaufmann, San Francisco, 1989
67. D.E. Goldberg, K. Deb, and J.H. Clark. Genetic Algorithms, Noise, and the Sizing of Populations. IlliGAL Report No. 91010, 1991
68. N. Hansen, A. Gawelczyk, and A. Ostermeier. Sizing the population with respect to the local progress in (1,λ)-evolution strategies – a theoretical analysis. In *Proceedings of the 1995 IEEE Conference on Evolutionary Computation*, pages 80–85. IEEE Press, Piscataway, NJ, 1995
69. F.G. Lobo. *The parameter-less Genetic Algorithm: rational and automated parameter selection for simplified Genetic Algorithm operation*. PhD Thesis, Universidade de Lisboa, 2000
70. C.R. Reeves. Using genetic algorithms with small populations. In Forrest [64], pages 92–99
71. J. Roughgarden. *Theory of Population Genetics and Evolutionary Ecology*. Prentice-Hall, 1979
72. D. Schlierkamp-Voosen and H. Mühlenbein. Adaptation of population sizes by competing subpopulations. In *Proceedings of the 1996 IEEE Conference on Evolutionary Computation*. IEEE Press, Piscataway, NJ, 1996
73. R.E. Smith. Adaptively resizing populations: An algorithm and analysis. In Forrest [64]

74. R.E. Smith. *Population Sizing*, pages 134–141. Institute of Physics Publishing, 2000
75. J. Song and J. Yu. *Population System Control*. Springer, 1988
76. V.A. Valkó. Self-calibrating evolutionary algorithms: Adaptive population size. Master's Thesis, Free University Amsterdam, 2003
77. B. Craenen and A.E. Eiben. Stepwise adaptation of weights with refinement and decay on constraint satisfaction problems. In L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 291–298. Morgan Kaufmann, 2001
78. B. Craenen, A.E. Eiben, and J.I. van Hemert. Comparing evolutionary algorithms on binary constraint satisfaction problems. *IEEE Transactions on Evolutionary Computation*, 7(5):424–444, 2003
79. J. Eggermont, A.E. Eiben, and J.I. van Hemert. Adapting the fitness function in GP for data mining. In R. Poli, P. Nordin, W.B. Langdon, and T.C. Fogarty, editors, *Genetic Programming, Proceedings of EuroGP'99*, Volume 1598 of *LNCS*, pages 195–204. Springer-Verlag, 1999
80. A.E. Eiben, B. Jansen, Z. Michalewicz, and B. Paechter. Solving CSPs using self-adaptive constraint weights: how to prevent EAs from cheating. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 128–134. Morgan Kaufmann, 2000
81. A.E. Eiben and J.I. van Hemert. SAW-ing EAs: adapting the fitness function for solving constrained problems. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, Chapter 26, pages 389–402. McGraw-Hill, London, 1999