Parameter Control in Evolutionary Algorithms: Trends and Challenges

Giorgos Karafotias, Mark Hoogendoorn, and A.E. Eiben

VU University Amsterdam

***Abstract*—More than a decade after the first extensive overview on parameter control, we revisit the field and present a survey of the state of the art. We briefly summarise the development of the field and discuss existing work related to each major parameter or component of an evolutionary algorithm. Based on this overview we observe trends in the area, identify some (methodological) shortcomings, and give recommendations for future research.**

***Index Terms*—Evolutionary algorithm, heuristic search, methodology, parameter control.**

I. Introduction

PARAMETER setting is one of the long standing grand challenges of the Evolutionary Computing (EC) field [55]. The essence of the problem can be summarized as follows.

- The performance of evolutionary algorithms (EA) greatly depends on the values of their parameters. For good performance, parameter values have to be carefully chosen. This is known as the 'parameter tuning problem'.
- The (near-)optimal parameter values may change over the run of an EA. For good performance, parameter values may need to be carefully varied over time. This is known as the 'parameter control problem'.

These two problems are clearly related, but bear important differences. Theoretically speaking, with some oversimplification we could say that the tuning problem is the stationary side, while the control problem is the non-stationary side of the same coin. From a practical perspective, tuning is an absolute 'must' for EA users, since no EA can be executed without giving some value to all of its parameters. Meanwhile, parameter control is more of a *neat-to-have* than a *need-to-have*, since EAs are certainly capable of running without changing the values of their parameters during the run. Finally, with a more severe oversimplification, we could say that the tuning problem has been solved by now. At least, there are very good parameter tuning methods developed and publicized over the last decade and the EC community is increasingly adopting them. For a good state of the art overview we recommend works by Eiben and Smit [60] and Smit [194].

For parameter control, the situation is quite different. The control problem is far from being solved although the possible advantages of control have been identified already in the past [47], [55], [65]:

- It allows the EA to use appropriate parameter values in different stages of the search process. (E.g., search by big leaps in the beginning, fine tune the near-optimal solutions by small steps in the last stage of the search.)
- It allows the EA to adjust to the changing fitness landscapes when facing dynamic problems.
- It allows the EA to collect information about the fitness landscape during the search and use the accumulating information to improve performance in later stages, (see [47], page 5).
- Using a parameter control mechanism liberates the user from the task of choosing parameter values. (That is, it implicitly also solves the tuning problem.)

While this latter argument is frequently articulated, in practice any controller must have parameters of its own, but these are frequently hidden behind design decisions. (We will return to this issue later in Section III and Section VII-B.)

Though there are very promising pieces of work in the literature on parameter control, there is also a certain lack of focus. The main purpose of this paper is to give a new impetus to parameter control research in Evolutionary Computing. We aim to achieve this goal through the following specific objectives:

1) Discuss the map of the field from a 'helicopter view'.
2) Provide an extensive review of related work.
3) Observe main clusters of work and analyze the corresponding research trends.
4) Identify the most prominent challenges for future developments.

The rest of the paper is organized as follows. The next section gives a short overview of the development of the field. In Section III we present a general framework for parameter setting and discuss the terminology used in this paper. The actual overview of the state of the art is given in Sections IV, V, and VI. Section IV discusses control methods for each of the major parameters / components of an EA. This is followed by Section V describing approaches that aim at controlling multiple parameters. Section VI is devoted to parameter independent methods. After the overview we identify the most important trends over the last two decades and discuss future directions in Section VII. Finally, Section VIII concludes the paper, expressing our hope that the coming period will be the 'decade of parameter control' with algorithmic as well as methodological advances.

Manuscript received March 6, 2013; revised August 17, 2013 and December 6, 2013. Corresponding author: G. Karafotias (email: g.karafotias@vu.nl).

G. Karafotias, M. Hoogendoorn and A.E. Eiben are with the Computational Intelligence Group at VU University Amsterdam, Netherlands.

II. Development of the Field

The problem of parameter control in EAs is as old as EAs themselves. However, it was not noticed in the early years of the field for a number of reasons. One of those reasons was the general attitude that perceived and advocated genetic algorithms (GAs) –that were the best known EAs back in the 70ies and 80ies– as robust search methods that work well for a wide range of parameter values. In other words, even the tuning problem was not recognized widely, although some authors discussed 'optimal' values for population size, mutation rate, and crossover rate, see e.g. [46], [91], [190]. Concerning the runtime adaptation of parameter values the research community was divided. Among GA researchers the issue remained largely unnoticed, with just a few exceptions [191], and the same was true for Evolutionary Programming. Meanwhile, the use of self-adaptation mechanisms to change mutation step sizes on-the-fly was standard practice in Evolution Strategies [177], [193].

Another problem hindering development was the fragmentation and inconsistency in terminology as illustrated by the notion of self-adaptation. The term was introduced in Evolution Strategies to denote the technique of including parameters of the EA (in particular, the mutation step sizes) into the chromosomes, thus co-evolving solutions and strategy parameters. However, some authors used a different name for the same technique, e.g. [78] used the term "meta-evolution" for self-adaptation, and some others used the same name for a different technique, e.g.[58] used the term self-adaptation to denote an adaptation mechanism that did not encode parameter values into the chromosomes. By the end of the 90ies the idea of changing parameter values on-the-fly gained traction even in the GA community, [196], [198], [211], but there was no unifying vision and generally adopted terminology. Some authors recognized the importance of these issues and offered a (partial) solution, see [9], [197], but these attempts did not receive the attention they deserved.

The situation changed in 1999 with the publication of [55]. This paper presented a unifying vision, a clear taxonomy by a list of essential features and the corresponding terminology that were quickly adopted and became the *de facto* standard in the field. It categorized parameter setting methods according to four aspects:

1) *What* is changed? (Which parameter?)
2) *How* the changes are made? (By what kind of mechanism?)
3) The *scope/level* of the change. (Population-level, individual level, sub-individual level.)
4) The *evidence* that guides the changes.

Furthermore, the types of mechanisms to make changes in parameter values were clearly defined, distinguishing *deterministic*, *adaptive* or *self-adaptive* methods as follows. Deterministic methods are uninformed, they follow a predetermined schedule for assigning new parameter values. Adaptive methods are informed as they receive feedback from the EA run and assign values based upon that feedback. Self-adaptive methods encode parameter values in the genome along the solutions and allow them to co-evolve with the problem solutions. Note that this notion of self-adaptation is generic. It can concern any parameter and its use is not limited to denote the specific strategy used in ES to control the mutation step sizes.

Apart from [55], several other frameworks and views of the field have been discussed. Here, we provide a list of relevant PhD Theses in this area, paying special attention to their contributions to classifying the field by alternative or complementary taxonomies.

- J.E. Smith, Self Adaptation in Evolutionary Algorithms, 1998 [197] presents its own taxonomy that is very similar to the one we presented above. However, the evidence that guides the changes is seen as "perhaps the most important" dimension (page 18). Later on in a joint publication, these views were merged [61] (page 142).
- R.K. Ursem, Models for Evolutionary Algorithms and Their Applications in System Identification and Control Optimization, 2003 [212] contains a discussion of methods for parameter control in EAs and a critical view on existing taxonomies therein. It proposes a novel taxonomy (page 50) that basically distinguishes "non-adaptive control" and "adaptive control", where the first one lumps together parameter tuning and deterministic control in the scheme of [55], while the second one is further divided into "measure-based", "self-adaptive", and "population-structure-based".
- O. Kramer, Self-adaptive heuristics for evolutionary computation, 2008 [130], [131] is primarily concerned with self-adaptive forms of parameter control. However, it also contains an "Extended Taxonomy of Parameter Setting Techniques" (page 31) that indeed extends the taxonomy of [55]. It does maintain the 3 types of control methods (deterministic, adaptive, self-adaptive) and identifies three types of parameter tuning: "by hand", by "Design of Experiments", and by "metaevolutionary" techniques.
- J. Maturana, General Control of Parameters for Evolutionary Algorithms, 2009 [155] (in French)[1] presents a good model of parameter control that distinguishes the EA, the controller, and their interactions, thus allowing for systematic developments (page 44).
- A. Fialho, Adaptive Operator Selection for Optimization, 2010 [76] does not introduce a new classification scheme, but uses the terms "off-line" or "external" tuning (page 35) and "on-line" or "internal" parameter control (page 38). This terminology emphasises the conceptual similarities with the reactive search perspective where off-line tuning and on-line tuning coincide with our parameter tuning and adaptive parameter control, respectively, cf. page 159 in [24].
- A. Aleti, An Adaptive Approach to Controlling Parameters of Evolutionary Algorithms, 2012 [6] focuses on adaptive parameter control and presents an interesting model (page 57) that distinguishes the optimization process and the control process. This latter is further divided into four principal steps or strategies for: (i) Feedback Collection, (ii) Parameter Effect Assessment, (iii) Param-

[1] see http://www.inf.uach.cl/maturana/files/presentation_WEB.pdf for a good English summary in the form of a presentation

eter Quality Attribution, (iv) Parameter Value Selection. These identify the essential components and provide a natural division of research, i.e., a sub-taxonomy within the adaptive branch of [55].

The theses listed above cover a very substantial amount of work related to parameter control and provide valuable insights about the most specialized experts' view on the field. One interesting idea is to shift the main 'water shed' from tuning vs. control to blind (tuning and deterministic) vs. informed (adaptive and self-adaptive) – rephrased after [212]. The other notable issue is the possibility to distinguish fine grade details within the adaptive parameter control category as offered in [197] and [6]. These details concern the fourth dimension above regarding the evidence that guides the changes and the way it is collected and handled.

III. A General Framework and Terminology for Parameter Setting

Before the actual survey we present a conceptual framework and a corresponding terminology for parameter control in EAs. To this end, let us recall the framework for parameter tuning as described in [60]. The essence of this framework is to distinguish three layers: the application layer (that contains a problem to be solved), the algorithm layer (that contains an EA), and the design layer (that contains a method to specify all details of the given EA –that is, its numeric as well as symbolic parameters).

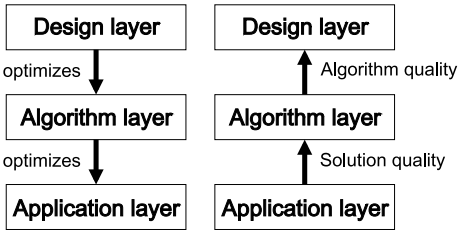

Fig. 1. Control flow (a) and information flow (b) through the three layers in the hierarchy of parameter tuning, after [60].

As Figure 1 indicates, the whole scheme can be divided into two optimization problems. The lower one belongs to the blocks Application + Algorithm. It consists of a problem on the application layer and an EA on the algorithm layer trying to find an optimal solution for this problem. The upper one belongs to the blocks Design + Algorithm and it consists of an algorithmic design method[2] that is trying to find optimal settings for the EA on the algorithm layer.

This framework is also suited to position parameter control. The key notion here is a method M that is assigning (good) values to one or more of EA parameters. Now the structure reflected in the tuning framework allows us to distinguish two options:

1) to position the control method M on the design layer, or
2) to position it on the algorithm layer.

[2]As opposed to a design method relying on human intuition.

The first option is in line with the taxonomy introduced in [55] where parameter tuning and parameter control are two alternative approaches to setting EA parameters. In case of tuning, M works in an off-line fashion and the parameter values it specifies remain fixed during the run of the given EA. In case of control, M works in an on-line fashion, i.e. parameters are given an initial value when starting the EA and M changes the values while the EA is running. Formally, the difference lies in the dependencies, or attributes, of the method M. In case of a tuning method, the output of M –an appropriate parameter vector for the given EA– only depends on the problem at hand and the users actual definition of algorithm performance.[3] For control methods, however, the output of M also depends on the actual state of the evolutionary search process. (Note that control methods solely relying on elapsed time are also covered here, since time can be seen as a state descriptor/identifier.) We could also say that a tuning method tries to solve a static optimization problem, while a control method is concerned with non-stationary optimization.

The second option amounts to considering the EA and the control mechanism(s) together as the search algorithm, where the EA forms the core and the controller is an extra runtime component to enhance the search. This view fits well with several algorithms where the bare bone EA and the parameter control mechanism(s) are deeply integrated; think for instance of evolution strategies and the self-adaptation of mutation step-sizes. The rationale behind this option is a temporal workflow perspective that separates the design time and the runtime of an algorithm. Adopting this view, parameter control is a part of the deployed runtime system while tuning is a procedure in the design phase to tailor the overall system to a specific problem (class). This view also brings forth another point: a controller does have parameters of its own (though these are frequently hidden behind design decisions). Therefore, the controller can be tuned as part of the integrated search algorithm (instead of tuning the EA parameters that are subjects to this controller). Whether or not parameter control and parameter tuning are used in an evolutionary problem solver can be decided independently from one another. Therefore, we obtain four possible combinations shown in Figure 2.

The combinations in the top row of Figure 2 employ tuning and have the advantage of tailoring to a specific problem class. They incur the cost of the time and resources required for the tuning process. The combinations in the left column employ control and offer the benefits of dynamically varying parameter values. The cost of control is more subtle: out-of-the-box control will have to perform some exploration thus may slow down the search or apply parameters harmful for the rest of the search; tuning a controller can increase the design space of the tuning process as compared to tuning the static parameters of the EA.

Regardless the positioning of the control mechanism M (at the design layer or the algorithm layer of Figure 1), the classic division of deterministic, adaptive, and self-adaptive control is applicable and we will use it in the following too.

[3]This is called *utility function* in [60].

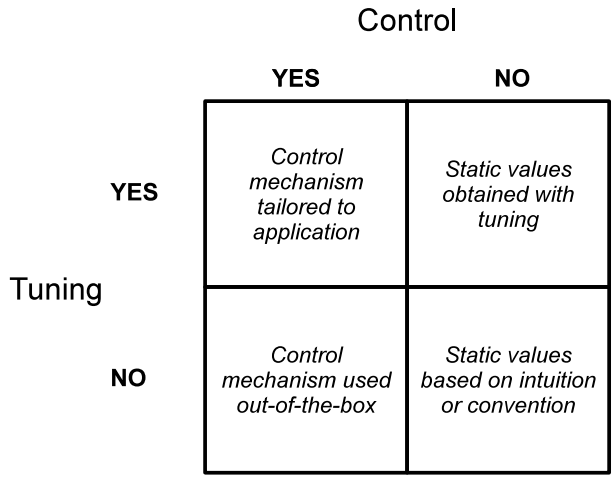

Fig. 2. The possible combinations of tuning and control.

Furthermore, we also classify control mechanisms as *parameter specific*, *ensembles* or *parameter independent*. Parameter specific strategies are designed for or only applicable to a specific parameter or component. Ensembles are combinations of heterogeneous control strategies, each separate strategy specific to a parameter; these controllers are combined to create an EA with overall control of multiple parameters. Parameter independent methods are designed without having a specific parameter in mind and can be applied to control any parameter (or at least any numeric parameter). The overview in the following sections is organized by this division.

IV. Parameter and Component Specific Methods

In this section we present parameter specific strategies that were designed for a certain parameter or component. Though some may be considered applicable to other parameters as well, they are placed in the section dedicated to the parameter they were originally intended for in literature. The following subsections describe control designed for the population, selection, variation, fitness function and for parameters related to parallel EAs.

A. Population

Extensive work has been carried out regarding the population size of evolutionary algorithms. Here we present this work divided in four main categories:

1) Theoretical studies on population size and the benefit of dynamic population sizing.
2) Approaches that aim at disposing of the population size parameter all together by introducing a new operator or concept.
3) Strategies that attempt to approximate a good (minimal and sufficient) population size during the EA run.
4) Mechanisms that directly control the parameter on-the-fly.

A literature review on control of the population size is given by Lobo and Lima [148].

Theoretical studies: The population size as a parameter has been studied from a theoretical point of view within the Genetic Algorithms community employing schema and building block theory. There are two ways to approach the population size: (i) how large should the population be to have a sufficient supply of building blocks and (ii) how large should the population be to cope with errors in selection [103].

Reeves [178] calculated the minimum necessary population so that every allele is present for binary as well as n-ary representations. It is suggested that the population is initialized in a "smart" way so as to cover as much of the search space as possible. Though having all alleles present in the initial population is important, it is more crucial that actual building blocks are also available. Goldberg et al. [87] derive the probability that all building blocks are present in a population and, based on that, an equation that calculates a minimum necessary population size as a function of the alphabet cardinality, the size and the number of building blocks.

From the selection point of view, Goldberg et al. [85] used statistical decision theory to derive a formula for the necessary population size based on the permissible decision errors between building blocks, the noise caused by selection and variation and the complexity of the problem. Harik et al. [103] took an alternative approach to modeling population size requirements according to selection errors. They use the theory of random walks and the Gambler's Ruin model to approximate selection decisions of a GA with tournament selection and without mutation.

More recently, Laredo et al. [138] proposed a method for evaluating GA population control schemes, based on building block concepts and decomposable functions with known population dynamics [85]. The authors examine the assumption that larger populations are useful in the beginning of the run, while later on, the EA can converge with smaller population sizes. Their approach is theoretical and limited only to selectorecombinatorial[4] GAs and decomposable trap functions.

A similar theoretical study on dynamic population sizing for GAs is given by Lobo [146]. Population size is examined from the perspective of the building block concept [85] and the Gambler's Ruin model for population size [103]. The latter calculates the minimum requirement for population to solve a problem with a given probability. It uses the supply of building blocks and the probability that selection will correctly favor a building block instead of a competitor. The author derives two control strategies based on the observation that these factors of supply and selection do not remain constant during the run of the GA. The building block supply model provides an approximation method for the former while the latter is calculated using a heuristic. These values are used in each generation with the Gambler's Ruin model to derive the new population size.

Removing the population size as a parameter: The approaches described here remove the population size parameter mainly by replacing it with a maximum lifetime, by imposing

[4] A selectorecombinatorial GA uses only selection and recombination, i.e. it is a GA without mutation.

a limit on some resource that indirectly limits the population or by replacing the multi-individual population with a representative distribution.

Removing the population size as a parameter firstly appeared with the Genetic Algorithm with Varying Population Size (GAVaPS) by Arabas et al. [10]. Individuals are assigned a maximum lifetime which depends on their fitness and the average or best fitness of the population. When this maximum lifetime is reached, the individual is removed. The ratio of offspring to population size is kept constant, thus selective pressure also remains fixed while the population size varies. Deciding the maximum lifetime of an individual requires minimum and maximum bounds, consequently, two new parameters are introduced.

The lifetime concept of GAVaPS was extended using non-random mating: niGAVaPS by Fernandes et al. [70] uses an incest prevention mechanism (inbreeding control) , while nAMGAVaPS by Fernandes and Rosa [68] selects mating partners based on phenotypic similarity (assortative mating control). SRP-EA [69] further extended nAMGAVaPS by introducing (i) a dynamic threshold of similarity for the assortative mating mechanism and (ii) a probabilistic scheme to individual removal instead of the fully predetermined lifetimes in the original GAVaPS and nAMGAVaPS.

Another variation of the GAVaPS lifetime scheme was described by Bäck et al. [18]. The Adaptive Population GA (APGA) uses the same lifetime allocation but differs from GAVaPS in the reproduction cycle and in that, when incrementing the ages of individuals in each cycle, the best individual of that cycle remains unchanged. This adaptive population strategy is part of an ensemble and is described in more detail in Section V. This method was also applied to co-operative co-evolution by Iorio and Li [119]. An analysis of APGA by Lobo and Lima [147] shows theoretical and experimental results suggesting an upper bound and a converging population to a constant size that is determined by the minimum and maximum lifetime parameters. The authors conclude that the population is not adapted by the APGA but the size parameter is in fact replaced by the two lifetime parameters.

Cook and Tauritz [42] suggested two strategies for removing the population size parameter. FiScIS-EA removes individuals according to a survival probability, derived by linear scaling of the individual's fitness in the range between the minimum and maximum fitness values present in the population. GC-EA simply evades choosing a population size by maintaining a population as large as is allowed by memory limitations. This requires a well-chosen parent selection mechanism that scales well while a survival selection operator is still needed in case the memory boundary is reached during the run. Both methods aim at resolving the issue of the population size without introducing new meta-parameters.

Another method for removing the population size parameter that is particular to Genetic Programming was suggested by Wagner and Michalewicz [216], [217]. The population size and maximum tree depth parameters are replaced by two parameters limiting the maximum total number of nodes in the population (soft and hard limits) in order to bound resource consumption. As a result, the number of individuals in the population varies with time, allowing the natural growth of good quality complex solutions with the expense of having less individuals. This approach was used in the DyFor algorithm [218] that also controls additional parameters and is thus described in Section V.

Finally, we also mention here two genetic algorithms that replace the population with a self-adapting probability distribution. The Population-based Incremental Learning algorithm by Baluja and Caruana [22] maintains a probability vector to represent its population. Each component of this vector represents the proportion of alleles $0/1$ in the assumed population. A similar approach is the Compact Genetic Algorithm by Harik et al. [105].

Approximating a good population size: The following methods automate the process often performed by human designers: running the EA with progressively larger population sizes to determine how large a population would be sufficient for solving the problem at hand. They differ mainly in the number of subpopulations they use and the criteria for terminating subpopulations.

Harik and Lobo [104] first suggested the Parameter-less GA.[5] It creates and runs separate populations, each new population is double the size of the previous. Populations are run in parallel and raced: when a population is outperformed by a larger one, it is immediately deleted. Smaller populations are given more evaluations than larger ones. The target is to find as soon as possible the smallest population capable of solving the problem.

An adaptation of the Parameter-less GA above is the Greedy Population Sizing approach (GPS-EA) by Smorodkina and Tauritz [201]. New subpopulations again have double the size of the previous. The difference is that GPS-EA runs only two populations in parallel at any time. The smaller population expires either if its average fitness becomes worse than that of the larger or if its best fitness stops improving at a value better than that reached by its predecessor. When a population expires, the algorithm proceeds by creating a new one. A rough theoretical analysis suggests that the GPS-EA should be able to reach fitness levels comparable to that of a GA with an optimal static population in no more than double the time.

The IPOP-CMA-ES by Auger and Hansen [13] and the IPOP-αCMA-ES by Hansen and Ros [102] extend the (μ_w, λ)-CMA-ES [98]. The population control simply augments the existing restart strategy of the algorithm by doubling the population size with each restart.[6] A further extension is suggested with the BIPOP-CMA-ES by Hansen [97], [96]. With each restart, two interlaced methods for calculating two options for the new population size are used. A large population size is doubled every time while a small population size is proportional to the ratio of the last used large size to the default size modified by a random factor. The new population size is the smallest of the two options.

[5]The authors use the name "Parameter-less" for their algorithm. However, this name is somewhat misleading, since technically speaking their algorithm is not parameterless and only concerns population sizes.

[6]The text refers to the parameter controlled as "population size λ" which might be confusing since λ usually notes the number of offspring produced in each generation. This is due to the structure of the CMA-ES, which differs from the conventional EA loop, and also the fact that it is defined $\mu = \frac{\lambda}{2}$.

Controlling population size on-the-fly: Contrary to other parameters, control of the population size entails some additional design choices except for the standard parameter control components (frequency, amount of change etc.). The additional components are the procedures by which individuals are created or removed when the population grows or shrinks. Costa et. al. [43] investigate the potential of a simplistic deterministic control schedule of monotonous growth or shrinking. Different combinations of creation/removal strategies are tried, however, the presented experiments are too limited to draw meaningful conclusions.

Deterministic control can of course be more elaborate than a monotonous change. The saw-tooth GA by Koumousis and Katsaras [129] combines a linear decrease schedule with periodic reinitializations that restore the population to its maximum size (by inserting random new individuals). An inverse saw-tooth model of the population size is examined by Hu et al. [112]. Inspired by a parallel and asynchronous EA implementation, it gradually increases the population while mass extinctions occur periodically. Experiments suggest that these extinctions can trigger increases in the best fitness. De Vega et al. [50] also used extinctions that are triggered in every generation and remove a fixed number of the worst individuals.

The simplest form of adaptive population size is based on intuitive triggers or formulas. PRoFIGA by Eiben et al. [57] observes the improvement of the best fitness to make adjustments: increasing fitness triggers a proportional growth of the population, stagnation for a period longer than a predetermined threshold results in a fixed increase while in all other cases the population diminishes. The aim is to create appropriate exploration-exploitation levels in the beginning of the search, during the hill-climbing process and after the population is stuck at a (local) optimum respectively. Fernandez et al.[74] suggested using extinctions that are triggered by events/metrics defined as policies to be chosen by the algorithm designer.

Smith and Smuda [199] proposed dynamic adaptation of the population size of GAs based on a theoretical foundation rather than intuition. Using theory on schemata fitness, schemata competition and selection error [86], the adaptive control mechanism continuously estimates the expected fitness loss due to selection error by performing pairwise competitions of the common schemata of mating pairs and subsequently assigns the number of offspring to each mating pair so as to maintain a loss close to user defined target value. Though one parameter is replaced with another, the motivation of the authors is to replace obscure GA parameters with more intuitive and user-friendly ones.

Using a clustering algorithm, a GA can construct a linkage model and derive building blocks information (such as the DSMGA [225]). Yu et al. [226], [227] used this information, e.g. fitness variance of building blocks and signal size between competing building blocks, to estimate the necessary population size for the next generation. They suggested performing population growth with random individuals in the beginning and using crossover in the final stages.

Hinterding et al. [109] combined adaptation using fitness as feedback and testing multiple subpopulations of different sizes. The proposed scheme maintains three populations of different sizes. A run is divided in epochs and, at the end of each epoch, population sizes are changed according to a set of simple rules that move the population sizes towards the size that performed best in the last epoch or expand the range of used sizes if the subpopulations performed equally. Unlike the "optimal size approximating" methods of the previous subsection, this control strategy continuously adapts the population size based on feedback from the search and can both increase and decrease it. This strategy is combined with self-adaptive mutation in the Self-Adaptive Genetic Algorithm described in V.

Finally, population size has also been self-adapted. Since it is a global parameter, its value has to be derived from the individuals' values. Eiben et al. [66] calculated the population size of the next generation as the sum of the values of all individuals. Teo [207] derived the population size by averaging the individuals' values and also suggested encoding the rate of change for the parameter value in the genome instead of absolute values. We mention that the former study concluded that self-adaptation of the population size is not promising while the second had encouraging results. However, they differ in many aspects, most notably the EA variants used.

B. Selection

A number of control mechanisms have been proposed for dynamic selection in an EA.

Several deterministic methods draw inspiration from the Boltzmann distribution which also underlies Simulated Annealing as introduced by Kirkpatrick, Gelatt, and Vecchi [128] and originates from condensed matter physics. In the case of combinatorial optimization this distribution in combination with a thermal equilibrium guarantees to have asymptotic convergence to global optima. The idea is to control selection in a deterministic way using a function that increases the selectivity over time. Goldberg [84] was the first to create a tournament selection in genetic algorithms that results in the Boltzmann distribution, and as a consequence he was able to prove convergence to global optima. De la Maza and Tidor [49] focused on faster convergence and introduced a scheme which uses a Boltzmann weighting strategy for selection. They utilize a deterministic scheme which increases selectivity in a linear fashion. Dukkipati et al. [52] defined specific Cauchy criteria for a scheme to control the Boltzmann selection. Their main criterion is that the differences between successive selection pressures should decrease as the evolutionary process proceeds.

Next to approaches that focus on obtaining a Boltzmann distribution to guarantee a good outcome, several strategies have been introduced that use a similar scheme to deterministically adjust the selectivity as the evolutionary process progresses.

Marin and Sole [153] used a similar temperature-based schedule to control selection in combination with extinction dynamics. They create a graph-based space for the population in which individuals are placed and connected via weights representing differences between fitness values. Selection is based on the weights and replacement is done either at random or by selecting the best. The choice is made probabilistically

with the probability depending on the temperature of the process. The result is a more likely insertion of the best individual in the end of a run.

Next to deterministic control, adaptive mechanisms have also been suggested that monitor the state of the process and adapt the selection accordingly. Affenzeller [3] introduced The Segregative Genetic Algorithms (SEGA) that uses a fixed population size and the amount of offspring generated depends on the population diversity. It was extended by the same author to SESEGASA [4] adding a method to detect premature convergence and a more advanced selective pressure mechanism. McGinley et al. [15] used the diversity of the population to adjust the tournament size, resulting in a bigger tournament size in case of a high diversity and smaller in case of a low diversity.

Another category of adaptive methods utilize some kind of spatial structure to enable the adaptation of the selection mechanism. Mass extinction is applied to obtain adaptive selection using a spatial structure based on the concept of self-organized criticality and the sandpile model in the work by Krink and Thomsen [135]. They organize individuals in a grid corresponding to the lattice of the sandpile model (for more information see [21], [20]). As grains are dropped on the grid, avalanches determine which zones in the grid become extinct. In these zones new individuals are placed which are mutations of the currently best individual. Rudolph and Sprave [183] present an approach consisting of a genetic algorithm which is placed in a particular cellular space. Each individual has certain neighbors assigned and is only allowed to mate with those neighbors. In order for new individuals to be accepted into the population, they need to be better than a certain threshold which is dynamically controlled.

An adaptive approach inspired by ants was presented by Kaveh and Shahrouzi [124]. Their idea is to maintain a separate population (called the colony) of individuals that were shown to be promising in order to maintain population diversity. Individuals are selected and removed from the colony using a pheromone-based scheme and parents for the next generation are selected using a combination of the colony and the regular population.

Self-adaption of selection has been proposed as well. Eiben et al. [66] self-adaptated the selective pressure by encoding the tournament size in the genome. Because it is a global parameter, the value is determined by summing up all the individuals' values. In similar work, Maruo et al. [154] suggested using a majority vote instead of a summation.

On the other hand, Smorodkina and Tauritz [200] proposed self-adaptive mate selection. Each individual encodes its own preferences as a selection mechanism in the form of a tree (similar to GP). The first parent is selected via a regular selection method; it then uses its own strategy to select its mate.

C. Variation

Controlling the various aspects of the variation operators (i.e. in general crossover and mutation in all their namings, forms and flavors) is probably the most popular subject and focus point found in literature on parameter control in EAs. In this section we present work carried out in the following main categories:

1) Theoretical studies on the influence of variation operators' parameters and attempts to determine theoretic optimal values for these parameters.
2) Removing variation parameters by using novel mutation and crossover operators.
3) Control of mutation and crossover rate (commonly noted as p_m and p_c respectively - or F and CR in Differential Evolution).
4) (Self-)Adaptive Operator Selection (AOS). Unlike (3), studies in this category control the selection of available operators including several choices, treating them as alternatives (e.g. choosing among several types of crossover).
5) Control of the distribution of offspring sampling, a subject that is particular only to real number encodings but has been the focus of extensive work.

1) Theoretical studies: Theoretical results on the values and effect of the mutation probability in Genetic Algorithms were given by Hesser and Männer [107]. They suggest that mutation is only necessary for finite populations to compensate with the loss of building blocks due to errors of the selection process. In the absence of that risk, i.e. in the hypothetical situation of infinite population, mutation would act disruptively by destroying already accumulated information and, thus, reducing the convergence speed. Furthermore, it is suggested that the probability of mutation should converge to 0 as the genome length l increases because the probability to destroy good value combinations increases proportionally with l. Using a time-requirement model and based on the assumption of a sufficiently large population and certain absorption probabilities, a formula is derived for calculating optimal mutation rates. A further heuristic for calculating a good combination for mutation and crossover rates is proposed but it requires estimating values that are very difficult to derive. Böttcher et al. [27] presented another analysis limited to a $(1+1)$-GA and the LeadingOnes problem showing that the optimal mutation rate is not the conventional $\frac{1}{n}$. They suggest an equation for adaptive mutation rate that reduces the expected time to solution .

Jansen and De Jong [120] presented a study on the influence of the number of offspring λ on the number of evaluations required to reach the optimum for a $(1+\lambda)$-EA. Analysis with two simple functions (OneMax and LeadingOnes) suggests that a $(1+\lambda)$-EA cannot outperform a $(1+1)$-EA when solving unimodal functions, though keeping λ within certain bounds (proportional to $\log N$ - where N is the population size) will not incur a significant slowdown. Further analysis with artificial multimodal landscapes shows that a value of λ larger than one can greatly increase performance when solving multimodal problems. Since keeping within the bound of $\log N$ does not slow evolution significantly while larger λ values are beneficial for multimodal landscapes, the authors conclude with the generic guideline of defining $\lambda = \log N$ when the characteristics of the fitness function are not known.

In the field of GAs, crossover is often seen as the main search operator while mutation is considered as a secondary operator [111], useful only for the recovery of lost allele values [107]. Subsequently, traditional practice often involves setting mutation rate to very small values [46]. Mühlenbein [167] discussed a different view: mutation is itself a search operator, especially beneficial when combined with high selective pressure [17]. This latter line of thought suggests that mutation rate should be set to relatively high values. Bäck [16] used mutation as the only search operator and studied the success probability and optimal mutation rate for a simplistic problem type.

De Jong and Spears [48] examined the disruptiveness of crossover in Genetic Algorithms and its dependence on the number of crossover points. They suggested that crossover disruption is beneficial late in the search when the population is converging and when the population size is too small and that an appropriate adaptive crossover operator should increase its disruptive potential when population homogeneity increases.

In the area of Differential Evolution (DE), it was originally suggested that DE is quite robust and that its variation parameters (scaling factor and crossover rate) can be easily set to some "standard" values [205]. However, like with the other variants of EAs, later studies showed that the performance of DE is also highly influenced by the values of its parameters [234], [28], [81]. Reynoso-Meza et al. [179] conducted experiments to find good settings for these parameters when DE is applied to a multi-objective problem class and concluded that the choice for the value of the scaling factor is difficult and case specific.

2) Removing variation parameters: Fernandes et al. [73] introduced a novel mutation operator based on the nature-inspired concept of self-organized criticality and the sandpile model (for more information see [21], [20]). At each generation, individuals are mapped to a matrix (each row being an individual and each column a gene). This matrix is used as the lattice for a sandpile model; whenever an avalanche occurs, the affected cells (i.e. genes) are mutated. To avoid mutating highly fit individuals, an avalanche is interrupted at a cell if a randomly generated value is higher than the normalized fitness of the individual that the specific cell belongs to. Since the sandpile mutation must work on evaluated individuals and the subsequent selection must have up-to-date (i.e. after mutation) fitness values, that would require two cycles of evaluations per generation. To avoid this, the sandpile mutation uses the fitness values of the parents of an individual to derive an expected normalized fitness, thus requiring evaluations of individuals only after the mutation procedure is complete [71], [72].

3) Controlling mutation and crossover rates: An adaptive approach to controlling both mutation and crossover rates was presented by Srinivas and Patnaik [204]. The goal is to maintain diversity by increasing variation when the population converges (which is detected by the mean fitness approaching the best) while also maintaining good solutions. To achieve this, every time an offspring is produced, crossover rates are defined by a linear function proportional to the difference in parent's fitness with the best and inversely proportional to the grade of convergence. A schema analysis suggests that this adaptive method is better than a static approach in terms of promoting schemata with higher fitness and rapidly increasing the fitness of schemata.

Bäck [17] applied self-adaptation to the GA mutation rate parameter. Bit string genotypes are extended with one or multiple sections encoding mutation rates. These sections are decoded into probabilities and used to mutate themselves; the resulting new mutation rates are then used to mutate the objective values. This approach differs from the original ES self-adaptation where mutation parameters are not self-mutated but instead a static distribution is used. In following work [19] the same author suggested that the binary encoding of the mutation rate limits precision and fine-tuning. As an improvement, mutation rate is attached as a real number to the genotype. Furthermore, self-mutation of p_m is performed using a lognormal distribution similar to the common practice in Evolution Strategies. In a different direction, Smith and Fogarty [198] applied GA self-adaptation to a steady state GA and investigated the influence of selection and crossover on the success of self-adaptation.

More recently, Vafaee [213] and Nelson suggested transforming the mutation procedure of a GA into a sampling process in the $\{0,1\}^n$ space and, subsequently, controlling this distribution by adapting a vector of probabilities . The Site-Specific Rate GA (SSRGA) calculates a probability vector using a fitness weighted sum of a set of individuals. The population is split into several subpopulations that occupy different peaks and the probability vector for each subpopulation is calculated. Individuals are then mutated by setting their alleles according to the probabilities of the corresponding vector. Though it is described as mutation by the authors it is in fact a sampling process that resembles the method of the CMA-ES [95] (see Section IV-C5). A similar method was presented by Nadi and Khader [169] with the difference that the sampling process is merged with crossover: two parents create two offspring; alleles that are common to both parent are maintained while the rest are set according to a probability vector derived from the whole population.

Extensive work has been carried out for the control of the scaling factor and crossover rate of Differential Evolution (DE). The simplest methods are based on formulas or rules according to the authors' intuition of how the parameters should vary. Ghosh et al. [82] [83] suggested a strategy where F decreases (linearly or logarithmically) as the distance of the target vector's fitness to the current best fitness decreases while CR increases as the fitness of the donor vector is closer to the current best fitness. A formula specific to multi-objective optimization for adapting F using the numbers of non dominated solutions and crowding metrics was proposed by Qian and Li [175]. Liu and Lampinen [144] used fuzzy logic controllers with fitness and diversity measures as inputs for the FADE algorithm. The fuzzy memberships are used to evaluate rules that are hand coded by the authors.

For a more adaptive approach, Zielinski and Laur [231], [232] used the Design of Experiments method: for the two variation parameters of DE, a two-by-two factorial design is used and all combinations are applied for equal numbers

of trial vectors; when significant differences are detected the two points of the affected parameter(s) are moved in the appropriate direction by a predetermined interval. A popular approach for the adaptive control of the DE variation parameters is sampling their values from random distributions and adapting the characteristics of these distributions. The JADE algorithm by Zhang and Sanderson [230], [229] samples the values for the two parameters from two corresponding normal distributions for each new trial vector created. The values that lead to a vector better than the parent are saved and used to calculate the means of the distributions for the next generation. A similar approach is followed by the ILSDEMO [223] and SaDE [176] algorithms (which are ensembles and are described in Section V).

Self-adaptation of the DE variation parameters has also been explored by multiple authors, the differences lying in the manner the offspring's parameter values are calculated. Abbass [1] used the typical mutation formula of DE, Brest et al. [28] gave the trial vector the value of the target vector with 90% chance (otherwise random) while the SA-DE algorithm of Brest et al. [29] averages the parameter values of the target vector and the three differential vectors and mutates this average with a factor from a lognormal distribution (following ES practice).

4) Operator selection: Adaptive Operator Selection (AOS) deals with the symbolic parameter of the variation operators. Several alternatives for such operators exist in all EA variants and AOS aims at the concurrent and adaptive use of several operators.

Adaptive AOS mechanisms mostly model the problem as a multi-armed bandit (MAB), a simplified version of the reinforcement learning problem [206]. Maturana et al. [157] discuss a comprehensive view of AOS, identifying the components of an adaptive operator selection framework. A credit assignment component uses feedback provided by the EA to calculate scores of operators. These scores are maintained in a credit registry and are used by the selection component in order to select the next operator to be used by the EA. In a more generic view, the AOS mechanism can also be complemented by a component that creates and adds new available operators to the AOS or removes existing ones.

The simplest operator selection method is probability matching, i.e. *softmax* selection of operators based on assigned scores. An operator's probability to be selected is calculated as the proportion of the operator's current score to the total sum of scores of all operators. This method was applied by Thierens [210] to GAs, by Gong et al. [89] to DE (the PM-AdapSS-DE algorithm) and by Qin and Suganthan [176], Mallipeddi et al. [151] and Xie et al. [223] as part of the SaDE, EPSDE and ILSDEMO ensembles described in Section V.

A disadvantage of the probability matching approach stems from the allocation of probabilities directly proportionally to the rewards; it results in the best known operator not being exploited maximally because sub-optimal operators are also applied. The smaller the difference between scores the more often sub-optimal operators will be applied instead of the best known. Adaptive pursuit was used by Thierens [209], [210] to solve this problem. It increases the selection probability of the best known operator while decreasing all other probabilities.

A third selection method is the Dynamic Multi-Armed Bandits (D-MAB) suggested by DaCosta et al. [44]. The Upper Confidence Bound (UCB) algorithm is used, which will usually select the option that has the highest expected reward while still maintaining a small probability of selecting worse options for exploration purposes. However, UCB is not appropriate for a dynamic environment: if an option becomes less efficient than another it will take a lot of time for the corresponding probabilities to adjust accordingly. For this reason, a Page-Hinkley test is used to detect changes in the EA and restart the UCB with all operators being equal.

Credit assignment is usually performed by counting the number of successful applications of an operator, where success is defined as creating offspring that are fitter than the parent(s). This method is followed by most AOS mechanisms in literature.

The EXtreme value-based Adaptive Operator Selection (ExAOS) by Fialho et al. [75] uses a credit assignment method based on a sliding window. Whenever an operator is applied, the fitness improvement is calculated and added to the window. The credit assigned to the operator is the maximum value found in the window. This approach aims at rewarding operators that contribute with large improvements, even if they only do so once. It is interesting to note that this contradicts Gong et al. [89] who concluded that averaging past rewards is preferable. The ExAOS credit assignment method is paired with D-MAB selection [44] described above. A sliding window is also used by Fialho et al. [77]. They suggest increasing the reward with the time elapsed since the last application of this operator and decreasing the reward with the number of times the operator has been applied within the window. The aim of this method is to adapt quickly to (even subtle) changes of the dynamic environment. Li et al. [141] suggested a sliding window that stores the rate of improvement in the fitness of the offspring as compared to the parent. The sum of all these rewards in the window is used by a ranking mechanism to assign credit to the operators.

A different credit assignment mechanism is Compass suggested by Maturana and Saubion [159]. Based on the concepts found in [160] (see Section VI), an operator's impact is evaluated using measures of both fitness and diversity in order to calculate the exploration-exploitation balance achieved by the operator. The assigned credit reflects how closely the achieved balance is to an exploration-exploitation balance that is required by a user-defined schedule. Other tested credit assignment methods are based on domination between operators and Pareto fronts. Compass was paired with probability matching selection but was also combined with D-MAB in [156].

Except for the widely used multi-armed bandit approach discussed so far, AOS has also been treated as a full reinforcement learning problem by Sakurai et al. [188], Chen et al. [39] and Pettinger and Everson [174]. Unlike the previous approaches, these methods include the notion of state that is defined using feedback from the EA. For each distinct state, separate preferences are learned for each operator and selection of the operator to apply is based on the current state

of the search.

An important issue with operator selection (as with all adaptive parameter control) is the feedback used for control. Veerapen et al. [215] presented and compared various utility measures for variation operators. These measures are based on a combination of exploration and exploitation measures and use Pareto-dominance to evaluate operator utility. In another study, Whitacre et al. [221] make a distinction between the source of feedback and any further statistical treatment (a notion further elaborated in [122]). Several feedback sources are suggested (including whether the offspring survives or the number of generations it remains in the population). The data received by these sources can be treated by averaging them or by detecting the outliers in the sample. The latter method intends to award operators that produce "extraordinary" individuals. Experiments show that the choice of feedback source has a significant impact on performance while the outlier detection method is shown to be very promising.

Operator selection has also been approached in a self-adaptive manner. Spears [202] suggested a simple self-adaptive operator selection mechanism for GAs where chromosomes are extended with one bit that selects between two operators. A local approach is followed, i.e. every time a new individual is created the operator bits of the parents determine which crossover is applied. Experiments showed that the performance of the self-adaptive GA is, in most cases, similar to the best performing static GA (using one of the two operators). Control experiments, however, revealed a very interesting fact: the mere availability of multiple operators was adequate while actually self-adapting the choice of operator did not have any additional benefit. In similar work, Riff and Bonnaire [180] extend each individual with a gene denoting the operator used to create it. When a new offspring is produced, the operator of the fittest parent is employed.

Instead of including the operator identifier in the genome, an alternative is to include probabilities for all available operators. Gomez [88] implemented self-adaptive operator selection in this manner with the Hybrid Adaptive Evolutionary Algorithm (HaEa). He argued that centralized adaptive operator selection has the disadvantage of the complexity related to calculating the operators' global value while self-adaptive operator selection suffers from the need to define meta-operators for evolving genes that encode operator rates. The latter problem of self-adaptation is tackled here by avoiding meta-operators for the genes that encode operator rates. Instead, random rewards (or penalties) are added to operator rates when the offspring is better (or worse) than the parent (in case the offspring is worse, the parent genome is kept but with the modified operator rates). Operator rates are used by a roulette selection scheme to decide which operator is used every time a child is produced.

Along the same lines, Montero and Riff [165], [166] propose operator selection self-adaption where the genome is extended to include the probabilities of the operators. The probability of the applied operator is given a reward/penalty which is either random (the "light-weight" version) or depending on the ratio of the difference between child and parent's fitness to the best/worse operator result in the last generations. An adaptive version was also suggested that maintains one global probability per operator and, at each generation, one single operator is picked to produce all offspring. Success is measured as the average success over all offspring produced and the rewards/penalties are calculated as with the self-adaptive (though here there is no "light-weight" version).

5) Offspring sampling distribution for real encodings: One of the best known control strategies is 'Rechenberg's 1/5 rule' [177] for controlling the σ parameter, the variation of the Gaussian distribution used for mutation. Based on theoretical results with the sphere and corridor landscapes, Rechenberg concluded that the optimal success ratio should be 1 out of 5. If greater, σ should be increased and, if less, σ should be decreased. Rudolph [181] presented an analysis of this adaptation method suggesting that when used by an elitist algorithm it will lead to premature convergence.

An extension of the 1/5 rule with reinforcement learning was suggested by Muller et al. [168]. The temporal difference learning SARSA algorithm is employed to learn the appropriate action (increase, decrease or maintain the mutation step size) given the success ratio. Two possible methods of reward calculation are tested (based on the change of the success rate or the fitness). The controller showed poor results in the evaluation of this paper.

Though the 1/5th rule was important as it immediately introduced the notion of parameter control to the very first $(1 + 1)$-ES, the most significant innovation of Evolution Strategies was the concept of self-adaptation: encoding a parameter of the algorithm in the genome and allowing it to undergo evolution [193]. Though self-adaptation has been applied to several other parameters since (see Section VI), ES self-adaptation was initially used to control the variance σ of the Gaussian distribution used to perform mutation.

Ostermeier et al. [170] argued that the original self-adaptation of mutation step sizes does not work well for small populations because of the potentially extreme values of the random mutation and proposed a "derandomized" approach. The concept of an efficient evolution path motivated the use of correlation between individual mutation step sizes of the dimensions of the problem and the adaptation of a covariance matrix [171], [100]. Based on this, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) and its variants [101], [99], [98], [95], [102], [26], [14], [118] have become very successful numeric optimizers.

A deterministic control of the mutation step size is described by Krink et al. [134]. The parameter takes values from a sequence acquired by a nature-inspired method (the sandpile model). This is part of a combined control method and is presented in detail in Section V.

While discussing control of the mutation step sizes for ES, we mention the works by Rudolph [182] and by Arnold and MacLeod [12]; both contain comprehensive reviews of such methods along with their mathematical bases.

Though the parameter addressed in this subsection concerns mostly Evolution Strategies, some examples are found in other EA variants as well. Real-coded Evolutionary Programming involves an identical parameter that regulates the variance of the normal distribution used for mutation. Self-adaptation

of this strategy parameter was introduced to Evolutionary Programming by Fogel et al. [78]. An application of Gaussian noise to a binary encoded GA with self-adaptation of the mutation step size was suggested by Hinterding [108]. Gene values are first decoded into numbers, noise is added and then encoded back into bit strings. Individuals are extended with one gene to encode the mutation step size and the sequence of mutations (first the step size gene is mutated with a fixed deviation then the new step size is used to mutate the other genes) is similar to Evolution Strategies.

D. Fitness function

Although not common in all application domains dynamic fitness functions have been successfully used for various problems. Here, we present methods for controlling aspects of the fitness function in two categories:

1) Methods that adapt the weights of penalties in constrained optimization.
2) Other strategies applicable to unconstrained problems.

1) Constrained problem approaches: A straightforward strategy is to gradually increase the weights of unsatisfied contraints with time. Joines and Houck [121] suggested a mechanism that increases penalty weights as generations elapse. The Genecop II by Michaelewicz and Attia [163] makes such increases only with every restart of the search. Similar schemes were presented by Kazarlis and Petridis [125] and Ben Hamida and Schoenauer [94].

Alternating increase and decrease of penalty weights was suggested by Bean and Hadj-Alouane [25]. A weight vector determines the penalty of certain candidate solutions. The values of this vector start very small, increase at first, and then decrease and increase in an alternating scheme (where the rate of increase is higher than the rate of decrease).

The Stepwise Adaptation of Weights (SAW) mechanism by Eiben and van Hemert [63], [62] adjusts the weights of the penalties in periods of a certain number of fitness evaluations. The constraints that are violated by the best individual in the population are identified and the corresponding weights in the fitness function are increased. Lemonge et al. [140] proposed a similar scheme but instead of considering only the best individual they take into account the entire population.

Smith and Coit [195] presented an approach which adjusts the penalty factors for constraints based on feedback regarding the severity of the constraint and the overall success of finding feasible solutions. Wang et al. [220] suggested a fitness measure which is highly dependent on the composition of the population. If the population consists only of individuals that are infeasible (i.e. that do not satisfy all the constraints), the fitness function only considers the level of satisfaction of the constraints. If all individuals fulfill the constraints, only the objective function is used as an evaluation measure. Finally, in case both feasible and infeasible candidates are present, a combination is taken. Similar methods were proposed by Tessema and Yen [208], Montemurro, Vicenti and Vannucci [164] and Farmani and Wright [67].

Co-evolving of the weights along with the solutions was suggested by Coello Coello [41]. Two separate populations are used to represent the set of candidate solutions and the weights that are used by the objective fitness function. Individuals that represent weights are applied for several generations and are evaluated by the number of feasible solutions present in the candidate solution population. Similar co-evolution but with a predator-pray flavor was used by Paredis [172].

2) Non-constrained problem approaches: Besides adapting weights of penalties for constrained problems, fitness functions have also been controlled to enhance performance for general unconstrained problems. Eggermont and Hemert [54] apply the SAW-ing adaptive fitness function (with some extensions) to a Genetic Programming application tasked to solve simple regression problems. The fitness measure is a weighted sum of the prediction errors for all points. The weight of each sample point is updated based on the difficulty of finding the correct value for that specific sample. The SAW-ing strategy was also applied to the field of data mining by Eggemont et al. [53].

Sakanashi et al. [187] proposed an adaptive fitness function that tries to improve the performance of GAs for the so-called GA hard problems. The fitness function adapts to make sure that the algorithm explores the search space sufficiently. This adaptation is based on the average and best performances compared with a number of generations in the past.

Efficient search is also the purpose of using multiple auxiliary fitness functions. Approaches that focus on choosing the best function using reinforcement learning have been proposed by Afanasyeva and Buzdalov [2] and Buzdalova and Buzdalov [31].

E. Parallel EAs

Parallel EAs can either be implemented as several distinct subpopulations running in parallel with occasional migrations (island model) or as several distributed evolutionary operations acting upon a common population asynchronously. In this section we present work related to:

- controlling the parameters that are particular to distributed EAs and
- control methods or ensembles for standard parameters that are designed especially for distributed EAs.

1) Parameters of Parallel EAs: The parameters that are specific to distributed EAs (the island model in particular) concern migration (rate, number of immigrants, policies for selection and replacement) and the topology of the islands network. Cantu-Paz [32] showed that selection and replacement policies for migration have significant effects in the convergence and takeover times of parallel GAs. The same author [33] presented a theoretical analysis of the island size, the connectivity degree between islands, the migration rate, the network topology and their interrelations. Arnaldo et al. [11] also examined the influence of the topology of distributed EAs. Using $\beta-$graphs and NK landscapes, they concluded that certain topologies (in terms of characteristic path length and clustering coefficient) are more appropriate depending on the complexity of the problem.

The simplest approach to varying the migration was suggested by Hiroyasu et al. [110]. Migration occurs in fixed time intervals but the number of individuals migrating from

each island is random. The number of immigrants was also examined by Maeda et al. [150] with an adaptive approach. Again migration occurs in fixed time intervals but the number of immigrants each island sends is adapted separately using fitness based feedback as input to a fuzzy controller. The membership functions, parameter levels and inference rules are predetermined and fixed. Instead of fixing the migration period, Lardeux and Goëffon [137] adapted the probability of occurrence of migration events. They modeled a distributed EA as a fully connected directional graph, with each edge labelled with the probability of migration occurring in the direction of that edge. When an immigrant makes an improvement in the destination island then the probability of the edge it followed is increased otherwise it is decreased.

In addition to the amount or rate of immigrants, Zhan and Zhang [228] also adapted the paths (source and destination islands), thus in effect controlling the topology of the island network. Their method sorts islands according to their mean fitness and all islands send one immigrant to every island that is higher in the ranking.

2) Methods for Parallel EAs: For the island model, the existence of several subpopulations running in parallel offers two important advantages: (a) there are several parameter configurations available at the same time and (b) several parameter configurations can be evaluated concurrently. The former is not directly related to control but Gong and Fukunaga [90] used it by simply setting the parameters of each subpopulation to different (even random) values. The rationale is that, at each moment during the search process, there will be at least one parameter configuration that will be somewhat favorable for further advance. Along with the effects of migration of individuals, this approach may allow for more efficient search than having only one parameter setting. The ability to evaluate several parameter vectors concurrently was used by Lis and Lis [143] to control numeric parameters of subpopulations of a parallel GA. A master is responsible for distributing populations and parameter vectors to the processors. Each parameter has a number of allowed value "levels" and at each point time only three such levels exist in any parameter vector of all islands. A run is divided in epochs of equal length (number of evaluations) for each island; at the end of each epoch, the master receives the best individuals from each subpopulation and calculates the average best fitness achieved by each level of each parameter. Then every parameter is updated by shifting values towards the best (or keeping them as are if the best was the middle one).

Control of parameters for a distributed EA with asynchronous operations acting on a common population was explored by Budin at al. [30].

A simple mechanism for controlling the crossover and mutation probabilities for a parallel GA using fitness based feedback was presented by Wang at al. [219], however, this approach is not specific to the features of a distributed EA.

V. Control ensembles

In this section we present work on evolutionary algorithms with combined (heterogeneous) control mechanisms put together as an ensemble that controls several parameters concurrently. These ensembles generally fall in the main categories of combining variation and population control and combining variation and selection control in order to balance exploration and exploitation. Furthermore, extensive work has been carried out for the control of the numeric variation parameters combined with operator selection for Differential Evolution (DE).

Hinterding et al. [109] combined self-adaptation of mutation with adaptive control of the population for a GA. Mutation applies Gaussian noise by first decoding the genes into numerical values and then encoding the new values back to bit strings. A gene encoding the mutation step size is added to the chromosome and mutated as in Evolution Strategies. The size of the population is controlled adaptively by maintaining three subpopulations of different sizes and dividing the run into epochs. At the end of each epoch populations' sizes are adjusted according to a set of simple rules that move sizes towards the size that performed the best during the last epoch or expand the range if the subpopulations had equal performance.

Bäck et al. [18] also combined self-adaption of crossover and mutation probabilities and dynamic adjustment of the population size. Self-adaptation is typically achieved by encoding of the operator probability values in the genome. Individual mutation rates are first mutated and the new values are used to mutate the rest of the genome. Individual crossover rates represent the probability of that individual mating. These are evaluated by random tests separately and if a selected parent is not willing to mate it creates one offspring by mutation. The population size adaptation is removed by assigning lifetimes to individuals at creation. These lifetimes are calculated based on the new individual's fitness and the population's best and average fitness. This ensemble was applied to co-operative co-evolution by Iorio and Li [119] with the difference that the crossover rates of potential parents are averaged and a common decision for mating is made instead of treating them separately.

Simple deterministic control of mutation and selection was presented by Krink et al. [134]. Both controls are based on a predetermined sequence of numbers acquired based on concepts of self-organized criticality [20]. Mutation control is performed simply by using this sequence to set the variance of Gaussian mutation. The control for selection is based on extinctions, removing a percentage of the population at each generation. These percentages follow the same sequence of numbers described earlier. Individuals to be removed are chosen randomly while the remaining individuals seed the next population. It is argued that tournament and roulette selection, near the end of the run, do not allow new (recombined and/or mutated) individuals to enter the population, while extinctions, which have strong evidence in biological evolution, do allow for this introduction of novelty.

Herrera and Lozano [106] used fuzzy logic controllers to adapt the choice of the crossover operator and the selective pressure. The aim to to maintain a good balance between exploration and exploitation; for that reason the EA employs two crossover operators, one with exploitative properties and

one with explorative, with the frequency of application defined by a real-valued parameter. Selection is preformed with linear ranking and selective pressure can be controlled by a real valued parameter as well. These two parameters are controlled using two fuzzy logic controllers. The inputs are diversity-based observables (genotypic and phenotypic diversity) while no fitness-based feedback is used. The output of each controller is a delta factor to increase/decrease the corresponding parameter value.

ACROMUSE by McGinley et al. [161] is a complete ensemble with adaptive crossover, mutation and selection. Its purpose is to maintain a population that is both diverse and fit. It uses specifically designed crossover and selection operators and divides the population into exploration and exploitation sections based on diversity measures. An offspring can be created in exploration or exploitation mode according to a probability (this probability is dynamically adapted). Parent selection uses a tournament with a dynamically adapted size. The variation and selection controls are combined so that the exploring (and probably less-fit) individuals created under exploration mode with aggressive mutation are given fair chance to survive and reproduce.

In the area of DE, the SaDe algorithm by Qin and Suganthan [176] combines adaptive control of the amplification factor and crossover rate with adaptive selection between two operators. While F values are sampled uniformly from a predefined range, the CR values are sampled from a normal distribution and assigned to individual indexes to be used for a number of generations. During that period, values that create good offspring are recorded and are then used to calculate the new mean for the normal distribution used in the following epoch. Operators are assigned credit according to the number of surviving offspring they create and are selected for each new offspring with a softmax process. SaDE was extended with multiple operators in [115] and was applied to multi-objective optimization in [113] and with separate objective-wise parameters in [114]. A study by Zielinski et al. [233] suggested that the operator selection component of SaDE is crucial to the algorithm's success and can also be beneficial when applied to two other control methods for DE.

A very similar control ensemble is used by the ILSDEMO algorithm by Xie et al. [223]. A slightly varied approach was taken by Mallipeddi et al. [151] for the EPSDE. Individuals are assigned random variation strategies and parameters in the beginning. When an offspring survives, the associated parameters are retained and added to a pool. Otherwise, new parameters are drawn from the pool or are assigned randomly. Finally, an ensemble for DE was created by Li et al. [142] by combining JADE [230] (see Section IV-C3) and PM-AdapSS-DE [89] (see section IV-C4).

Finally, an application specific ensemble is the Dynamic Forecasting Genetic Program (DyFor GP) model designed by Wagner et al. [218] to cope with forecasting in dynamic environments. It controls the size of the training set used and removes the population size parameter (rather replaces it with a new one). A sliding window is used for training with the historical data. The size of the sliding window is controlled on-line using two values at each step. The best model produced with each window size is evaluated using a number of future data points. The size with the best accuracy is kept while the other is moved symmetrically around the first. The population size parameter is removed using Natural Non-static Population Control (NNPC) [216], i.e. imposing a maximum limit for the sum of the nodes of all solutions in the population. This allows complex solutions of good quality to grow while keeping resource consumption within limits.

VI. Parameter independent methods

In this section we describe generic control methods that were not designed for a specific parameter and can be applied to any (numeric) EA parameter.

A simple generic adaptive method can use a small set of values, adjust them based on simple rules and use feedback only to calculate rewards for each value (and not to make informed decisions taking into account the current situation). Wong et al. [222] proposed a probabilistic rule-driven adaptive model that maintains three possible values for each parameter. An EA run is divided into pairs of periods. During the first period values are chosen randomly (and their effect in terms of fitness improvement is recorded). In the second period values are chosen with probabilities proportional to the scores they achieved in the first period. In the end of the second period the three values are updated so as to move towards the best performing one or expand the range if none achieved performance above a threshold.

Improving the above idea, Aleti and Moser [7] suggested predicting which value would be best next based on time series forecasting, instead of using the last known winner. Continuous parameters are discretized giving a finite amount of possible values. A run is divided to iterations, at the end of each iteration the success ratio of every parameter value is calculated and added to the history list of that parameter value. Instead of simply translating these success ratios to selection probabilities for the next iteration, a line is fit to the recent history of each parameter to predict what its success ratio will be in the next iteration, thus making the selection probabilities more relevant. Credit and selection probabilities are maintained and updated for all values, making this strategy a multi-armed bandit (MAB) approach. This method was extended by adapting the discretization ranges on-the-fly: after each iteration, the best performing value (interval) is split into two while the worst performing one is merged with its worst neighbor [8].

Reinforcement learning is another problem independent approach that can be utilized here [206]. The idea is to use feedback from the EA that describes the state of the search and implement actions as changes to parameter values. Eiben et al. [64] used temporal difference learning to map states of the EA (described by fitness based metrics) to actions, i.e. parameter values. The controller algorithm uses a table that maps pairs of states and actions to estimated rewards. Learning this table is done using a combination of the Q-learning and the SARSA algorithms. When doing exploration, random (and potentially harmful) actions are chosen. For this reason, areas explored are chosen to be close to the known optimal actions. When

doing exploitation, the best action for the current state is found with a separate underlying genetic algorithm that optimizes the expected reward.

A special case of generic adaptive controllers are designed to be problem specific and need to be tuned to the problem at hand to learn a mapping from feedback to parameter values (they can operate only within the top left box in Figure 2). Such an approach was suggested by Kee et al. [126] where the mapping is learned during a training phase while the exact mechanism implementing this mapping is a module/design choice. Two alternative methods (table-based and rule-based) are used to map a state vector composed of three metrics (fitness change and variance and population variance) to parameter values. Karafotias et al. [122] employed neural networks to map diversity and fitness based feedback to parameter values. The weights of the network are calibrated off-line using a parameter tuner. Aine et al. [5] suggested "profiling" algorithms for specific problem types by an off-line training phase. This training results in tables that map the state of the EA (described by fitness and diversity measures) to not only parameter vectors but also the time until the next update of parameter values. Lee and Takagi [139] used fuzzy logic controllers instantiated by a meta-GA. Rules are represented by encoding the centers of the membership functions (which are fully overlapping) and the ID of a rule (i.e. the combination of inputs-outputs). Fuzzy logic was also used by Maturana and Saubion [160], [158] to achieve a balance between exploration and exploitation. The correlation between parameter values and the resulting diversity/fitness is learned during an initial learning phase. After that the parameters are controlled so as to maintain a certain exploration-exploitation balance which is dictated by a user-defined schedule (thus this method, besides the automatic calibration, also requires a hand-tuning process).

Instead of designing a concrete control strategy or a method for calibrating one, Liu et al. [145] presented a platform for expressing ad-hoc control strategies through a scripting-like language (PPCea). The user is provided with several measures of exploitation and exploration derived from ancestry trees that record the whole evolutionary process as parents - children branches.

Finally, we consider self-adaptation as a generic parameter control method. It entails incorporating parameter values in the genome and allowing them to undergo evolution, relying on selection to promote good values that are piggybacked on the solutions they helped create [65]. Though self-adaptation was initially coined for controlling the mutation parameter of Evolution Strategies, it could be applied to any parameter (i.e. it is at least feasible to implement such a control). As an example, a fully self-adaptive evolutionary algorithm was presented by Maruo et al. [154]. Probabilities of mutation and crossover, the mutation step size (for real coded representations), the crossover operator and the size of the selection tournament are self-adapted. Values of global parameters (crossover type and tournament size) are decided by the majority. Self-adaptation of global parameters was also investigated by Eiben et al. [66], though here, instead of majority voting, global parameter values are calculated as the sums of all votes. To keep the votes within defined bounds, self-adaptive mutation is not used (instead the static scheme from [19] is employed). Literature reviews on self-adaptation can be found in the works by Kramer [132] and Meyer-Nieberg and Beyer [162]. These reviews convincingly demonstrate the power of self-adaptation (mainly mutation parameters), especially in real-coded search spaces and in environments with uncertain or noisy fitness information. Meanwhile, they also point out that self-adaptation techniques may suffer from premature convergence and a (near-)optimal mutation strength is not always realized. This latter effect has also been observed in Artificial Life settings for digital organisms with discrete genotype spaces [40].

VII. Some Trends and Suggestions for Further Development

In this section we make some observations about the trends we identified in the literature and put forward some suggestions we deem relevant for future work in the field.

A. Trends

Figure 3 presents the number of publications in a histogram form in bins of 4 year periods split by the categories parameter specific (that is further split by parameter), ensembles, and generic. A plot of the total amount of publications on a cumulative scale is shown in Figure 4. It seems that the interest in parameter control modestly increases, while the drop at the end can be attributed to the dissemination delay of new publications.

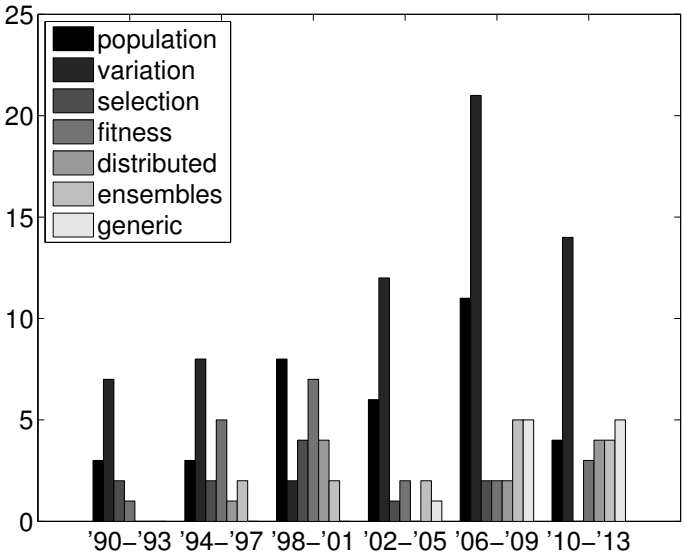

Fig. 3. The number of publications in 4 year periods for each parameter category described in this paper. For each period, bars denote (from left to right) population, variation, fitness, selection, distributed EAs, ensembles, generic. NB. Data for 2013 concerns the January-May period only.

As we can see parameters belonging to variation operators have received by far the most attention. Population related parameters are much less popular and parameters related to selection have received even less attention. Furthermore, ensembles for combined control of multiple parameters and generic control methods that can be applied to any parameter

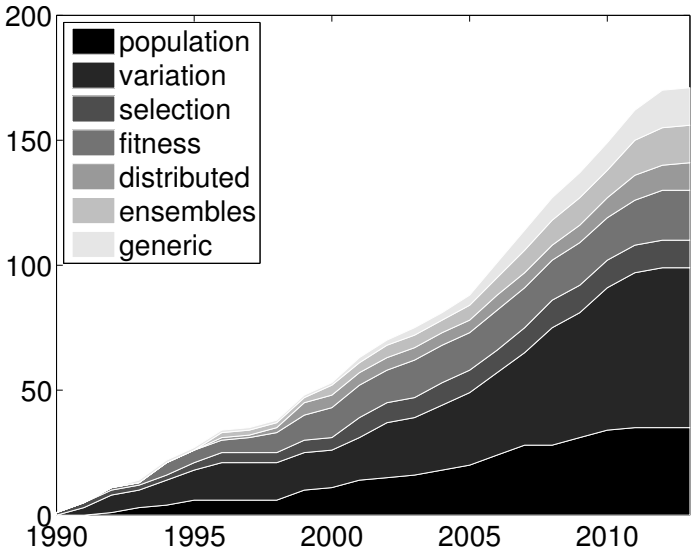

Fig. 4. The total number of publications on parameter control on a cumulative scale after 1990. Greyscale levels indicate the parameter addressed in the papers.

have been explored to very little extent (but they seem to gain traction lately).

The modest number of generic control methods implies what we call 'the patchwork problem': if one is to control more (all) parameters of an EA, then for each parameter she has to choose from a parameter specific set of existing methods and mix them into one system. Unfortunately, little is known about the joint effects of control mechanisms, thus there are no good guidelines about how to create good combinations. Therefore, the resulting mix is necessarily ad hoc and likely suboptimal. The work of Bäck et al. [18] is a good illustration for this as it mixes 3 different mechanisms to control 3 different parameters in an eclectic system without much justification. An encouraging exception is the ensemble by McGinley et al. [161] which is well orchestrated with controllers designed to work in cooperation.

Considering the methods and mechanisms used for adaptive parameter control, we can identify several categories that are relatively frequent in the literature:

- *Formulas* that calculate a parameter value using certain feedback from the EA. Such formulas may be based on theoretical results (e.g. [199], [84]) or the intuition of the designer (e.g. [204]).
- *Rules* triggered by certain events/thresholds concerning feedback from the search. They specify responding actions to these events, usually according to the designer's intuition (e.g. [57]).
- *Fuzzy Logic controllers* whose input is feedback from the search. Their output may be used to trigger hand-coded rules (in which case there is an overlap with the above category, e.g. [144], [150]) or they may directly output parameter values (e.g. [106]).
- *Value Sets* that are adapted, i.e. multiple (two or three) values are tested and the best one is used in the next phase while the others are adjusted accordingly. The testing phase is then repeated again (e.g. [143] [222] [218]).
- *Multi-Armed Bandit* strategies that treat each value as a separate arm and adapt their selection probabilities by learning expected rewards. This approach is very popular with Adaptive Operator Selection (e.g. [210], [44]).
- The full *Reinforcement Learning* approach that uses feedback from the search as descriptors to define states and maps actions to parameter values (e.g. [64], [168], [2]).

Regarding the field of parameter control in EAs as a whole, we can note a disappointing lack of impact. To be specific, many of the related papers report promising experimental results, however, no parameter control method or strategy has been widely adopted by the community or has became part of the common practice toolkit of evolutionary computing – a problem that was also noted by De Jong [47]. Perhaps this is caused by the lack of convincing evidence of the added value of control techniques. Putting it differently, despite the existence of important pieces of related work, it seems that most of the papers published make but a limited contribution to the field. All too often, such a paper (including ours!) has a limited scope and focuses on a simplistic control of a single specific parameter based on the author's intuition of how the parameter should vary. Typically, the paper does not analyze and explain the resulting behavior of the parameter and the EA, nor does it properly evaluate the added value of the control strategy based on good benchmarks.

B. Suggestions for Further Development

In this section we make suggestions for research directions that can help parameter control prove its value on a larger scale and become more relevant for evolutionary computing. Needless to say, we do not have clear recipes to this end, but we identify a number of issues (i.e., the *'what'*) and, where possible, give some hints regarding the *'how'*.

Our first suggestion is perhaps the easiest: work on reducing the patchwork problem mentioned above. For instance, carry out investigations about the combination of different control mechanisms and try to understand their joined effects. Additionally, more research could be done on controlling parameters that have received relatively little attention in the past. Furthermore, developing generic control mechanisms that work on any –or at least many– parameters could be helpful here.

As a second suggestion we note that identifying relevant EA behavior descriptors (sometimes called observables or monitors) can offer direct advantages for developing better parameter control mechanisms. In particular, it can help designing mechanisms for adaptive control, because these are based on using information (feedback) from the evolutionary search process. Being able to identify the most relevant information (EA behavior descriptors) can obviously improve the decisions made about new parameter values based on that information. This research line could benefit from data mining techniques that disclose the correlations between various EA behavior descriptors over time and the links between such descriptors and (the online dynamics of) solution quality.

Furthermore, we advocate research concerning the niche for parameter control in general, and that of specific parameter

control mechanisms in particular. In other words, we should try to understand for which problems and EA performance requirements is a given method advantageous. A related question we frequently receive is: "Is parameter control better than parameter tuning?", or "When should I do parameter tuning and when should I do parameter control?". For a solid answer based on quantitative evidence there is a lack of experimental data (with very few exceptions e.g. [80]) and –even more importantly– a lack of a decent research methodology.[7] A possibly useful angle to this end is to consider the application scenario as an important dimension. Using the categories of Eiben and Smith in [61], Chapter 13, it could be argued that for repetitive problems parameter tuning is worth the extra effort, whereas parameter control is the logical choice for one-off problems, applications with dynamic fitness functions, and on-line adaptation, e.g. robotics [51], [92]. Nevertheless, extra research and discussions are needed to back up solid answers to this and similar questions.

Progress towards a better research methodology could be made along several lines. For instance, the EC community should agree on sound measures for 'effort' in order to make fair comparisons between different techniques. This is a nontrivial problem, because tuning effort occurs during the design stage before the real EA run, while the computational overhead caused by control mechanisms manifests itself during a run. Furthermore, while there are software and hardware independent measures for tuning effort, these are based on counting fitness evaluations, see e.g. [194]. However, the extra work that parameter control mechanisms perform may be hidden from such a counter. Using computing time for this seems an easy solution, but that may raise other issues, as discussed by Eiben and Jelasity in [56].

Another essential prerequisite for sound assessments of control techniques is a good benchmarking practice. This issue reaches further than the composition of a good test suite. The added value of a parameter control technique will prove different when comparing it to different alternatives. This is a relevant and nontrivial question, because there are several justifiable options here, including:

- Comparison with the same EA without parameter control
  - with commonly used ('default') parameter values, or
  - with optimized (tuned) parameter values.
- Comparison with the same EA with a blind control mechanism, i.e. randomly varying the parameter values, see Karafotias et al. [123].
- Comparison with the same EA using another mechanism to control the same parameters.

Which of these and other possible options is appropriate for a good assessment should be investigated and widely discussed.

There is also much to gain in insights and algorithm performance through studying the parameters of parameter control mechanisms. There is a widespread opinion among parameter control researchers that even though a parameter control mechanism introduces new parameters of its own, the extended system (EA + parameter controller) is less sensitive to these parameters than the original EA to its own parameters. The skeptical observation, however, is that there is little real evidence to back up this belief, with the σ's and τ's in evolution strategies being the only well-known exception. A natural way to address this issue is to study parameters of parameter control mechanisms. For instance, using sensitivity analysis [189] can help verify or refute the aforementioned fundamental assumption. Furthermore, these studies can lead to better control mechanisms, simply because their working *does* depend on their parameters, even if a control mechanism is more robust than the bare-bone EA it regulates.

Last but not least, let us mention the admittedly rather general issue of a deeper understanding of parameter control mechanisms. Similarly to the main body of work within EC, most publications about a parameter control mechanism simply show *that* it works. Usually, there is no or very little discussion about *why* it works and *how* it works. In this respect, major improvements are possible through developing the know-how of monitoring the dynamics of the controlled parameter(s) and the whole EA over time. This requires the identification of relevant behavior descriptors for EAs, such as the extent of exploration and exploitation (a concept that is itself difficult to define but very relevant [214], [59]), population diversity, solution quality, and many more. Such improvements will also contribute to the design of more powerful adaptive controllers as was explained in our second suggestion above.

VIII. Concluding remarks

In this paper we presented an extensive survey of work concerning parameter control in evolutionary algorithms. This overview revealed a great number of interesting publications with promising results. Meanwhile, we also noted a disappointing discrepancy. In theory, parameter control mechanisms have a great potential to improve evolutionary problem solvers. In practice, however, the evolutionary computing community did not adopt parameter control techniques as part of the standard machinery – controlling EA parameters on-the-fly is still a rather esoteric option, with self-adaptation in evolution strategies being the exception that confirms the rule.

In an attempt to help close this gap we also discussed a number of ideas about important issues for further research. We certainly do not claim that we cover all possibly interesting ideas. But we hope to initiate a fruitful discussion in the community and on the long run, to improve the quality of control methods and to collect convincing evidence regarding their added value. This will hopefully help adoption of control methods on a larger scale, thus realizing their full potential.

As a final remark, let us note that the EC community could draw inspiration from other fields as well. For instance, there are significant advancements in the field of local search using techniques that automatically adjust the parameters of the search methods on-the-fly [24], [93].

References

[1] H.A. Abbass. The self-adaptive Pareto differential evolution algorithm. In CEC-2002 [34], pages 831–836.

[7] The weakness of experimental methodology is in fact a problem for the whole EC field, as noted by Eiben and Jelasity [56].

[2] A. Afanasyeva and M. Buzdalov. Choosing best fitness function with reinforcement learning. In *Proceedings of he 2011 10th International Conference on Machine Learning and Applications*, pages 354–357, 2011.

[3] M. Affenzeller. A new approach to evolutionary computation: Segregative genetic algorithms (SEGA). In *Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence*, volume 2084 of *Lecture Notes in Computer Science*, pages 594–601. Springer Verlag, 2001.

[4] M. Affenzeller. Segregative genetic algorithms (SEGA): A hybrid superstructure upwards compatible to genetic algorithms for retarding premature convergence. *International Journal of Computers, Systems and Signals (IJCSS)*, 2:18–32, 2001.

[5] S. Aine, R. Kumar, and P. P. Chakrabarti. Adaptive parameter control of evolutionary algorithms to improve quality-time trade-off. *Appl. Soft Comput.*, 9:527–540, 2009.

[6] A. Aleti. *An Adaptive Approach to Controlling Parameters of Evolutionary Algorithms*. PhD thesis, Swinburne University of Technology, http://users.monash.edu.au/ aldeidaa/papers/thesis.pdf, 2012.

[7] A. Aleti and I. Moser. Predictive parameter control. In Krasnogor et al. [133], pages 561–568.

[8] A. Aleti, I. Moser, and S. Mostaghim. Adaptive range parameter control. In *Proceedings of the 2012 IEEE Congress on Evolutionary Computation*, pages 1–8, Brisbane, Australia, 2012. IEEE Press.

[9] P.J. Angeline. Adaptive and self-adaptive evolutionary computations. In *Computational Intelligence: A Dynamic Systems Perspective*, pages 152–163. IEEE Press, 1995.

[10] J. Arabas, Z. Michalewicz, and J. J. Mulawka. GAVaPS - a genetic algorithm with varying population size. In ICEC-94 [116], pages 73–78.

[11] I. Arnaldo, I. Contreras, D. Milln-Ruiz, J.I. Hidalgo, and N. Krasnogor. Matching island topologies to problem structure in parallel evolutionary algorithms. *Soft Computing*, 17(7):1209–1225, 2013.

[12] D. V. Arnold and A. MacLeod. Step length adaptation on ridge functions. *Evol. Comput.*, 16(2):151–184, 2008.

[13] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In CEC-2005 [35], pages 1769–1776.

[14] A. Auger, M. Schoenauer, and N. Vanhaecke. LS-CMA-ES: A second-order algorithm for covariance matrix adaptation. In Yao *et al* [224], pages 182–191.

[15] F. Morgan B. McGinley and C. O'Riordan. Genetic algorithms with dynamic niche sharing for multimodal function optimization. In ICEC-96 [117], pages 1127–1128.

[16] T. Bäck. The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In Männer and Manderick [152], pages 85–94.

[17] 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.

[18] T. Bäck, A. E. Eiben, and N. A. L. van der Vaart. An empirical study on GAs "without parameters". In Schoenauer *et al* [192], pages 315–324.

[19] T. Bäck and M. Schütz. Intelligent mutation rate control in canonical genetic algorithms. In Z. Ras and M. Michalewicz, editors, *Foundations of Intelligent Systems*, volume 1079 of *Lecture Notes in Computer Science*, pages 158–167. Springer Berlin / Heidelberg, 1996.

[20] P. Bak and K. Sneppen. Punctuated equilibrium and criticality in a simple model of evolution. *Phys. Rev. Lett.*, 71:4083–4086, 1993.

[21] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality: An explanation of the $1/f$ noise. *Physical Review Letters*, 59(4):381–384, 1987.

[22] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. Technical report, Carnegie Mellon University, Pittsburgh, PA, USA, 1995.

[23] W Banzhaf *et al*, editor. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*. Morgan Kaufmann, San Francisco, 1999.

[24] R. Battiti, M. Brunato, and F. Mascia. *Reactive Search and Intelligent Optimization*. Springer Verlag, 2008.

[25] J.C. Bean and A. Ben Hadj-Alouane. A dual genetic algorithm for bounded integer programs. Technical Report 92-53, University of Michigan, 1992.

[26] H.G. Beyer and B. Sendhoff. Covariance matrix adaptation revisited — the CMSA evolution strategy —. In Rudolph *et al* [184], pages 123–132.

[27] S. Böttcher, B. Doerr, and F. Neumann. Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph, editors, *PPSN*, volume 6238–6239 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2010.

[28] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *Evolutionary Computation, IEEE Transactions on*, 10(6):646–657, 2006.

[29] J. Brest, A. Zamuda, B. Bokovi, S. Greiner, and V. Zumer. An analysis of the control parameters adaptation in DE. In U.K. Chakraborty, editor, *Advances in Differential Evolution*, volume 143 of *Studies in Computational Intelligence*, pages 89–110. Springer Berlin Heidelberg, 2008.

[30] L. Budin, M. Golub, and D. Jakobovic. Parallel adaptive genetic algorithm. In *International ICSC/IFAC Symposium on Neural Computation NC98*, pages 157–163, 1998.

[31] A. Buzdalova and M. Buzdalov. Increasing efficiency of evolutionary algorithms by choosing between auxiliary fitness function with reinforcement learning. In *Proceedings of he 2012 11th International Conference on Machine Learning and Applications*, pages 150–155, 2012.

[32] E. Cantú-Paz. Migration policies and takeover times in genetic algorithms. In Banzhaf *et al* [23], page 775.

[33] E. Cantú-Paz. Topologies, migration rates, and multi-population parallel genetic algorithms. In Banzhaf *et al* [23], pages 91–98.

[34] *Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC'2002)*, Honolulu, USA, 12-17 May 2002. IEEE Press, Piscataway, NJ.

[35] *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, Edinburgh, UK, 2-5 September 2005. IEEE Press, Piscataway, NJ.

[36] *Proceedings of the 2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, Vancouver, Canada, 16-21 July 2006. IEEE Press, Piscataway, NJ.

[37] *Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC'2007)*, Singapore, 25-28 September 2007. IEEE Press, Piscataway, NJ.

[38] *Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC'2009)*, Trondheim, Norway, 18-21 May 2009. IEEE Press, Piscataway, NJ.

[39] F. Chen, Y. Gao, Z.Q. Chen, and S.F. Chen. Scga: Controlling genetic algorithms with sarsa(0). In *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, volume 1, pages 1177–1183, 2005.

[40] Jeff Clune, Dusan Misevic, Charles Ofria, Richard E. Lenski, Santiago F. Elena, and Rafael Sanjun. Natural selection fails to optimize mutation rates for long-term adaptation on rugged fitness landscapes. *PLoS Comput Biol*, 4(9):e1000187, 09 2008.

[41] C.A. Coello Coello. Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2):113–127, 2000.

[42] J.E. Cook and D.R. Tauritz. An exploration into dynamic population sizing. In Pelikan and Branke [173], pages 807–814.

[43] J.C. Costa, R. Tavares, and A. Rosa. An experimental study on dynamic random variation of population size. In *Systems, Man, and Cybernetics, 1999. IEEE International Conference on*, volume 1, pages 607–612, 1999.

[44] L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag. Adaptive operator selection with dynamic multi-armed bandits. In Ryan and Keijzer [186], pages 913–920.

[45] 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.

[46] K.A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975.

[47] K.A. De Jong. Parameter setting in EAs: a 30 year perspective. In Lobo et al. [149], pages 1–18.

[48] K.A. De Jong and W.M. Spears. An analysis of the interacting roles of population size and crossover in genetic algorithms. In *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, PPSN I, pages 38–47. Springer-Verlag, 1991.

[49] M. de la Maza and B. Tidor. Boltzmann weighted selection improves performance of genetic algorithms. Technical report, MIT A.I. LAB, 1991.

[50] F.F. de Vega, E. Cant-Paz, J. I. Lpez, and T. Manzano. Saving resources with plagues in genetic algorithms. In Yao *et al* [224], pages 272–281.

[51] Cristian M. Dinu, Plamen Dimitrov, Berend Weel, and A. E. Eiben. Self-adapting fitness evaluation times for on-line evolution of simulated robots. In Christian Blum et al., editors, *GECCO '13: Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 191–198, Amsterdam, The Netherlands, 6-10 July 2013. ACM.

[52] A. Dukkipati, M.N. Murty, and S. Bhatnagar. Cauchy annealing schedule: An annealing schedule for boltzmann selection scheme in evolutionary algorithms. In *Proceedings of the 2004 Congress on Evolutionary Computation : CEC 2004, June 19-23, 2004, Portland, OR, USA*, pages 55–62, Piscataway, NJ, 2004. IEEE.

[53] J. Eggermont, A. E. Eiben, and J. I. van Hemert. A comparison of genetic programming variants for data classification. In D.J. Hand, J.N. Kok, and M.R. Berthold, editors, *Advances in Intelligent Data Analysis, Third International Symposium, IDA-99*, volume 1642 of *Lecture Notes in Computer Science*, pages 281–290. Springer, 1999.

[54] J. Eggermont and J. van Hemert. Adaptive genetic programming applied to new and existing simple regression problems. In J.F. Miller *et al*, editor, *Proceedings of the 4th European Conference on Genetic Programming*, number 2038 in LNCS, pages 23–35. Springer, Berlin, Heidelberg, New York, 2001.

[55] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter Control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[56] A. E. Eiben and M. Jelasity. A critical note on Experimental Research Methodology in EC. In CEC-2002 [34], pages 582–587.

[57] A. E. Eiben, E. Marchiori, and V. A. Valko. Evolutionary algorithms with on-the-fly population size adjustment. In Yao *et al* [224], pages 41–50.

[58] A. E. Eiben and Z. Ruttkay. Self-adaptivity for constraint satisfaction: Learning penalty functions. In *International Conference on Evolutionary Computation*, pages 258–261, 1996.

[59] A. E. Eiben and A. Schippers. On evolutionary exploration and exploitation. *Fundamenta Informaticae*, 35(1-4):35–50, 1998.

[60] A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[61] A. E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin Heidelberg, 2003.

[62] A. E. Eiben, J.K. van der Hauw, and J.I. van Hemert. Graph coloring with adaptive evolutionary algorithms. *Journal of Heuristics*, 4(1):25–46, 1998.

[63] 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*, Advanced Topics in Computing, chapter 26, pages 389–402. McGraw-Hill, 1999.

[64] A.E. Eiben, M. Horvath, W. Kowalczyk, and M.C. Schut. Reinforcement learning for online control of evolutionary algorithms. In Brueckner, Hassas, Jelasity, and Yamins, editors, *Proceedings of the 4th International Workshop on Engineering Self-Organizing Applications (ESOA'06)*, volume 4335, pages 151–160. Springer, 2006.

[65] A.E. Eiben, Z. Michalewicz, M. Schoenauer, and J.E. Smith. Parameter control in evolutionary algorithms. In Lobo et al. [149], pages 19–46.

[66] A.E. Eiben, M.C. Schut, and A.R. de Wilde. Is self-adaptation of selection pressure and population size possible? - a case study. In PPSN-2006 [185], pages 900–909.

[67] R. Farmani and J.A. Wright. Self-adaptive fitness formulation for constrained optimization. *IEEE Transactions on Evolutionary Computation*, 7(5):445–455, 2003.

[68] C. Fernandes and A. Rosa. A study on non-random mating and varying population size in genetic algorithms using a royal road function. In *2001 Congress on Evolutionary Computation (CEC'2001)*, pages 60–66. IEEE Press, Piscataway, NJ, 2001.

[69] C. Fernandes and A. Rosa. Self-regulated population size in evolutionary algorithms. In Runarsson *et al* [185], pages 920–929.

[70] C. Fernandes, R. Tavares, and A.C. Rosa. niGAVaPS - outbreeding in genetic algorithms. In *Proceedings of the 2000 ACM symposium on Applied computing - Volume 1*, SAC '00, pages 477–482. ACM, 2000.

[71] C. M. Fernandes, J. L. J. Laredo, A. M. Mora, A. C. Rosa, and J. J. Merelo. The sandpile mutation operator for genetic algorithms. In *Proceedings of the 5th international conference on Learning and Intelligent Optimization*, LION'05, pages 552–566. Springer-Verlag, 2011.

[72] C.M. Fernandes, J.L.J. Laredo, A.M. Mora, A.C. Rosa, and J.J. Merelo. A study on the mutation rates of a genetic algorithm interacting with a sandpile. In *Proceedings of the 2011 international conference on Applications of evolutionary computation - Volume Part I*, EvoApplications'11, pages 32–42. Springer-Verlag, 2011.

[73] C.M. Fernandes, J. J. Merelo, V. Ramos, and A.C. Rosa. A self-organized criticality mutation operator for dynamic optimization problems. In Ryan and Keijzer [186], pages 937–944.

[74] F. Fernandez, M. Tomassini, and L. Vanneschi. Saving computational effort in genetic programming by means of plagues. In *2003 Congress on Evolutionary Computation (CEC'2003)*, pages 2042–2049. IEEE Press, Piscataway, NJ, 2003.

[75] Á. Fialho, L. Costa, M. Schoenauer, and M. Sebag. Extreme value based adaptive operator selection. In Rudolph *et al* [184], pages 175–184.

[76] A.R.S. Fialho. *Adaptive Operator Selection for Optimization*. PhD thesis, Universite Paris-Sud, http://tel.archives-ouvertes.fr/docs/00/57/84/31/PDF/thesisAlvaro.pdf, 2010.

[77] . Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence*, 60(1-2):25–64, 2010.

[78] D.B. Fogel, L.J. Fogel, and J.W. Atmar. Meta-evolutionary programming. In *Signals, Systems and Computers, 1991. 1991 Conference Record of the Twenty-Fifth Asilomar Conference on*, pages 540 –545 vol.1, 1991.

[79] S Forrest, editor. *Proceedings of the 5th International Conference on Genetic Algorithms*. Morgan Kaufmann, San Francisco, 1993.

[80] G. Francesca, P. Pellegrini, T. Sttzle, and M. Birattari. Off-line and on-line tuning: A study on operator selection for a memetic algorithm applied to the qap. In Peter Merz and Jin-Kao Hao, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 6622 of *Lecture Notes in Computer Science*, pages 203–214. Springer Berlin Heidelberg, 2011.

[81] R. Gämperle, S.D. Müller, and P. Koumoutsakos. A parameter study for differential evolution. In *WSEAS Int. Conf. on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, pages 293–298, 2002.

[82] A. Ghosh, A. Chowdhury, R. Giri, S. Das, and S. Das. A fitness-based adaptation scheme for control parameters in differential evolution. In Pelikan and Branke [173], pages 2075–2076.

[83] A. Ghosh, S. Das, A. Chowdhury, and R. Giri. An improved differential evolution algorithm with fitness-based adaptation of the control parameters. *Information Sciences*, 181(18):3749 – 3765, 2011.

[84] D.E. Goldberg. A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems*, 4:445–460, 1990.

[85] D.E. Goldberg, K. Deb, and J.H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362, 1992.

[86] D.E. Goldberg, K. Deb, and J.H. Clark. Accounting for noise in the sizing of populations. In L.D Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 127–140. Morgan Kaufmann, San Francisco, 1993.

[87] D.E. Goldberg, K. Sastry, and T. Latoza. On the supply of building blocks. In Spector *et al* [203], pages 336–342.

[88] J. Gomez. Self adaptation of operator rates in evolutionary algorithms. In Kalyanmoy Deb *et al*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, volume 3102 of *Lecture Notes in Computer Science*, pages 1162–1173. Springer, 2004.

[89] W. Gong, Á. Fialho, and Z. Cai. Adaptive strategy selection in differential evolution. In Pelikan and Branke [173], pages 409–416.

[90] Y. Gong and A. Fukunaga. Distributed island-model genetic algorithms using heterogeneous parameter settings. In *IEEE Congress on Evolutionary Computation*, pages 820–827, 2011.

[91] J.J. Grefenstette. Optimization of control parameters for genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 16(1):122 –128, 1986.

[92] Evert Haasdijk, S.K. Smit, and A.E. Eiben. Exploratory analysis of an on-line evolutionary algorithm in simulated robots. *Evolutionary Intelligence*, 5(4):213–230, 2012.

[93] Y. Hamadi, F. Saubion, and E. Monfroy, editors. *Autonomous Search*. Springer, 2012.

[94] S. Ben Hamida and M. Schoenauer. An adaptive algorithm for constrained optimization problems. In Schoenauer *et al* [192], pages 529–538.

[95] N. Hansen. The CMA evolution strategy: a comparing review. In J.A Lozano and P Larranaga, editors, *Towards a New Evolutionary Computation : Advances in Estimation of Distribution Algorithms*, pages 75–102. Springer, Berlin, Heidelberg, New York, 2006.

[96] N. Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 2389–2396. ACM, 2009.

[97] N. Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 noisy testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 2397–2402. ACM, 2009.

[98] N. Hansen and S. Kern. Evaluating the cma evolution strategy on multimodal test functions. In Yao *et al* [224], pages 282–291.

[99] N. Hansen, S.D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11(1):1–18, 2003.

[100] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In ICEC-96 [117], pages 312 –317.

[101] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[102] N. Hansen and R. Ros. Benchmarking a weighted negative covariance matrix update on the BBOB-2010 noiseless testbed. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, GECCO '10, pages 1673–1680. ACM, 2010.

[103] G.R. Harik, E. Cantú-Paz, D.E. Goldberg, and B.L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evol. Comput.*, 7:231–253, 1999.

[104] G.R. Harik and F.G. Lobo. A parameter-less genetic algorithm. In Banzhaf *et al* [23], pages 258–265.

[105] G.R. Harik, F.G. Lobo, and D.E. Goldberg. The compact genetic algorithm. *Evolutionary Computation, IEEE Transactions on*, 3(4):287 –297, 1999.

[106] F. Herrera and M. Lozano. Adaptation of genetic algorithm parameters based on fuzzy logic controllers. In *Genetic Algorithms and Soft Computing*, pages 95–125. Physica-Verlag, 1996.

[107] J. Hesser and R. Männer. Towards an optimal mutation probability for genetic algorithms. In H.-P Schwefel and R Männer, editors, *Proceedings of the 1st Conference on Parallel Problem Solving from Nature*, number 496 in Lecture Notes in Computer Science, pages 23–32. Springer, Berlin, Heidelberg, New York, 1991.

[108] R. Hinterding. Gaussian mutation and self-adaption for numeric genetic algorithms. In *Proceedings of the 1995 IEEE Conference on Evolutionary Computation*, pages 384–389. IEEE Press, Piscataway, NJ, 1995.

[109] R. Hinterding, Z. Michalewicz, and T. Peachey. Self-adaptive genetic algorithm for numeric functions. In 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, pages 420–429. Springer, Berlin, Heidelberg, New York, 1996.

[110] T. Hiroyasu, M. Miki, and M. Negami. Distributed genetic algorithms with randomized migration rate. In *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, volume 1, pages 689 –694, 1999.

[111] J.H Holland. *Adaption in Natural and Artificial Systems*. MIT Press, Cambridge, MA, 1992. 1st edition: 1975, The University of Michigan Press, Ann Arbor.

[112] T. Hu, S. Harding, and W. Banzhaf. Variable population size and evolution acceleration: a case study with a parallel evolutionary algorithm. *Genetic Programming and Evolvable Machines*, 11(2):205–225, 2010.

[113] V. L. Huang, A.K. Qin, P.N. Suganthan, and M.F. Tasgetiren. Multi-objective optimization based on self-adaptive differential evolution algorithm. In CEC-2007 [37], pages 3601–3608.

[114] V. L. Huang, S-Z Zhao, R. Mallipeddi, and P.N. Suganthan. Multi-objective optimization using self-adaptive differential evolution algorithm. In CEC-2009 [38], pages 190–194.

[115] V.L. Huang, A.K. Qin, and P.N. Suganthan. Self-adaptive differential evolution algorithm for constrained real-parameter optimization. In CEC-2006 [36], pages 17–24.

[116] *Proceedings of the First IEEE Conference on Evolutionary Computation*. IEEE Press, Piscataway, NJ, 1994.

[117] *Proceedings of the 1996 IEEE Conference on Evolutionary Computation*. IEEE Press, Piscataway, NJ, 1996.

[118] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evol. Comput.*, 15(1):1–28, 2007.

[119] A. Iorio and X. Li. Parameter control within a co-operative co-evolutionary genetic algorithm. In J.J Merelo Guervos *et al*, editor, *Proceedings of the 7th Conference on Parallel Problem Solving from Nature*, number 2439 in Lecture Notes in Computer Science, pages 247–256. Springer, Berlin, Heidelberg, New York, 2002.

[120] T. Jansen and K.A. De Jong. An analysis of the role of offspring population size in EAs. In Langdon *et al* [136], pages 238–246.

[121] J.A. Joines and C.R. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In ICEC-94 [116], pages 579–584.

[122] G. Karafotias, S.K. Smit, and A.E. Eiben. A generic approach to parameter control. In C. Di Chio *et al*, editor, *Proceedings of EvoApplications 2012: Applications of Evolutionary Computation*, number 7248 in Lecture Notes in Computer Science, pages 366–375. Springer, Berlin, Heidelberg, New York, 2012.

[123] Giorgos Karafotias, Mark Hoogendoorn, and A.E. Eiben. Why parameter control mechanisms should be benchmarked against random variation. In *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*, pages 349–355, Cancun, Mexico, 2013. IEEE Press.

[124] A. Kaveh and M. Shahrouzi. Dynamic selective pressure using hybrid evolutionary and ant system strategies for structural optimization. *Int J Numer Meth Eng*, 73:544–563, 2008.

[125] S. Kazarlis and V. Petridis. Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. In 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, pages 211–220. Springer, Berlin, Heidelberg, New York, 1998.

[126] E. Kee, S. Airey, and W. Cyre. An adaptive genetic algorithm. In Spector *et al* [203], pages 391–397.

[127] M. Keijzer, editor. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2006)*. Morgan Kaufmann, San Francisco, 2006.

[128] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[129] V.K. Koumousis and C.P. Katsaras. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE Transactions on Evolutionary Computation*, 10(1):19–28, 2006.

[130] O. Kramer. *Self-adaptive heuristics for evolutionary computation*. PhD thesis, University of Paderborn, http://link.springer.com/book/10.1007/978-3-540-69281-2/page/1, 2008.

[131] O. Kramer. *Self-adaptive heuristics for evolutionary computation*, volume 147 of *Studies in Computational Intelligence*. Sptinger, 2008.

[132] O. Kramer. Evolutionary self-adaptation: a survey of operators and strategy parameters. *Evolutionary Intelligence*, 3(2):51–65, 2010.

[133] N. Krasnogor et al., editors. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2011)*, Dublin, Ireland, 12-16 July 2011. ACM.

[134] T. Krink, P. Rickers, and R. Thomsen. Applying self-organised criticality to evolutionary algorithms. In Schoenauer *et al* [192], pages 375–384.

[135] T. Krink and R. Thomsen. Self-organized criticality and mass extinction in evolutionary algorithms. In *Proceedings of the 2001 Congress on Evolutionary Computation*, volume 2, pages 1155–1161. IEEE, 2001.

[136] W B. Langdon *et al*, editor. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. Morgan Kaufmann, San Francisco, 9-13 July 2002.

[137] F. Lardeux and A. Goëffon. A dynamic island-based genetic algorithms framework. In *Proceedings of the 8th international conference on Simulated evolution and learning*, SEAL'10, pages 156–165. Springer-Verlag, 2010.

[138] J.L.J. Laredo, C. Fernandes, J.J. Merelo, and C. Gagné. Improving genetic algorithms performance via deterministic population shrinkage. In Franz Rothlauf, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2009)*, pages 819–826. ACM, 2009.

[139] M.A. Lee and H. Takagi. Dynamic control of genetic algorithms using fuzzy logic techniques. In Forrest [79], pages 76–83.

[140] A. Lemonge, C.C. Afonso, and H.J.C. Barbosa. An adaptive penalty scheme for genetic algorithms in structural optimization. *International Journal for Numerical Methods in Engineering*, 59(5):703–736, 2003.

[141] K. Li, A. Fialho, S. Kwong, and Q. Zhang. Adaptive operator selection with bandits for multiobjective evolutionary algorithm based decomposition. *Evolutionary Computation, IEEE Transactions on*, 2013 to appear.

[142] K. Li, . Fialho, and S. Kwong. Multi-objective differential evolution with adaptive control of parameters and operators. In C.A.Coello Coello, editor, *Learning and Intelligent Optimization*, volume 6683 of *Lecture Notes in Computer Science*, pages 473–487. Springer Berlin Heidelberg, 2011.

[143] J. Lis and M. Lis. Self-adapting parallel genetic algorithm with the dynamic probability, crossover rate and population size. In J. Arabas, editor, *Proceedings of the First Polish Evolutionary Algorithms Conference*, pages 79–86, 1996.

[144] J. Liu and J. Lampinen. A fuzzy adaptive differential evolution algorithm. *Soft Computing*, 9(6):448–462, 2005.

[145] S.-H. Liu, M. Crepinsek, M. Mernik, and A. Cardenas. Parameter control of EAs using exploration and exploitation measures: Preliminary results. In B. Filipic and J. Silc, editors, *Bioinspired Optimization Methods and their Applications, International Conference on*, pages 141–150. Jozef Stefan Institute, 2012.

[146] F.G. Lobo. Idealized dynamic population sizing for uniformly scaled problems. In Krasnogor et al. [133], pages 917–924.

[147] F.G. Lobo and C.F. Lima. Revisiting evolutionary algorithms with on-the-fly population size adjustment. In Keijzer [127], pages 1241–1248.

[148] F.G. Lobo and C.F. Lima. Adaptive population sizing schemes in genetic algorithms. In Lobo et al. [149], pages 185–204.

[149] F.G. Lobo, C.F. Lima, and Z. Michalewicz, editors. *Parameter Setting in Evolutionary Algorithms*. Springer, 2007.

[150] Y. Maeda, M. Ishita, and Q. Li. Fuzzy adaptive search method for parallel genetic algorithm with island combination process. *Int. J. Approx. Reasoning*, 41(1):59–73, 2006.

[151] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Appl. Soft Comput.*, 11(2):1679–1696, 2011.

[152] R Männer and B Manderick, editors. *Proceedings of the 2nd Conference on Parallel Problem Solving from Nature*. North-Holland, Amsterdam, 1992.

[153] J. Marin and R.V. Sole. Evolutionary optimization through extinction dynamics. In Banzhaf *et al* [23], pages 1344–1349.

[154] M.H. Maruo, H.S. Lopes, and M.R. Delgado. Self-adapting evolutionary parameters: Encoding aspects for combinatorial optimization problems. In G.. Raidl and J. Gottlieb, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3448 of *Lecture Notes in Computer Science*, pages 154–165. Springer Berlin Heidelberg, 2005.

[155] J. Maturana. *Controle Generique de Parametres pour les Algorithmes Evolutionnaires (General Control of Parameters for Evolutionary Algorithms)*. PhD thesis, University of Angers, http://tel.archives-ouvertes.fr/docs/00/45/91/85/PDF/These_JorgeMATURANA.pdf, 2009.

[156] J. Maturana, A. Fialho, F. Saubion, M. Schoenauer, and M. Sebag. Extreme compass and dynamic multi-armed bandits for adaptive operator selection. In CEC-2009 [38], pages 365 –372.

[157] J. Maturana, F. Lardeux, and F. Saubion. Autonomous operator management for evolutionary algorithms. *Journal of Heuristics*, 16:881–909, 2010.

[158] J. Maturana and F. Saubion. Towards a generic control strategy for evolutionary algorithms: an adaptive fuzzy-learning approach. In CEC-2007 [37], pages 4546–4553.

[159] J. Maturana and F. Saubion. A compass to guide genetic algorithms. In Rudolph *et al* [184], pages 256–265.

[160] J. Maturana and F. Saubion. On the design of adaptive control strategies for evolutionary algorithms. In *Artificial Evolution*, volume 4926 of *Lecture Notes in Computer Science*, pages 303–315. Springer, 2008.

[161] B. McGinley, J. Maher, C. O'Riordan, and F. Morgan. Maintaining healthy population diversity using adaptive crossover, mutation, and selection. *Evolutionary Computation, IEEE Transactions on*, 15(5):692 –714, 2011.

[162] S. Meyer-Nieberg and H.-G. Beyer. Self-adaptation in evolutionary algorithms. In Lobo et al. [149], pages 47–76.

[163] Z. Michalewicz and N. Attia. Evolutionary optimization of constrained problems. In A.V Sebald and L.J Fogel, editors, *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pages 98–108. World Scientific, 1994.

[164] M. Montemurro, A. Vincenti, and P. Vannucci. The automatic dynamic penalisation method (ADP) for handling constraints with genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 256:70–87, 2012.

[165] E. Montero and M.-C. Riff. Self-calibrating strategies for evolutionary approaches that solve constrained combinatorial problems. In *Proceedings of the 17th international conference on Foundations of intelligent systems*, ISMIS'08, pages 262–267. Springer-Verlag, 2008.

[166] E. Montero and M.-C. Riff. On-the-fly calibrating strategies for evolutionary algorithms. *Inf. Sci.*, 181:552–566, February 2011.

[167] H. Mühlenbein. How genetic algorithms really work: Mutation and hillclimbing. In Männer and Manderick [152], pages 15–26.

[168] S.D. Muller, N.N. Schraudolph, and P.D. Koumoutsakos. Step size adaptation in evolution strategies using reinforcement learning. In CEC-2002 [34], pages 151–156.

[169] F. Nadi and A.T. Khader. A parameter-less genetic algorithm with customized crossover and mutation operators. In Krasnogor et al. [133], pages 901–908.

[170] A. Ostermeier, A. Gawelczyk, and N. Hansen. A derandomized approach to self-adaptation of evolution strategies. *Evol. Comput.*, 2(4):369–380, 1994.

[171] A. Ostermeier, A. Gawelczyk, and N. Hansen. Step-size adaption based on non-local use of selection information. In Davidor et al. [45], pages 189–198.

[172] J. Paredis. Co-evolutionary constraint satisfaction. In Davidor et al. [45], pages 46–55.

[173] M. Pelikan and J. Branke, editors. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2010)*. ACM, 2010.

[174] J.E. Pettinger and R.M. Everson. Controlling genetic algorithms with reinforcement learning. In Langdon *et al* [136], pages 692–.

[175] W. Qian and A. li. Adaptive differential evolution algorithm for multiobjective optimization problems. *Applied Mathematics and Computation*, 201(12):431 – 440, 2008.

[176] A.K. Qin and P.N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In CEC-2005 [35], pages 1785–1791 Vol. 2.

[177] I Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Hozlboog Verlag, Stuttgart, 1973.

[178] C.R. Reeves. Using genetic algorithms with small populations. In Forrest [79], pages 92–99.

[179] G. Reynoso-Meza, J. Sanchis, X. Blasco, and M. Martinez. An empirical study on parameter selection for multiobjective optimization algorithms using differential evolution. In *Differential Evolution (SDE), 2011 IEEE Symposium on*, pages 1–7, 2011.

[180] M.-C. Riff and X. Bonnaire. Inheriting parents operators: A new dynamic strategy for improving evolutionary algorithms. In M.-S. Hacid, Z. Ras, D. Zighed, and Y. Kodratoff, editors, *Foundations of Intelligent Systems*, volume 2366 of *Lecture Notes in Computer Science*, pages 333–341. Springer Berlin / Heidelberg, 2002.

[181] G. Rudolph. Self-adaptive mutations may lead to premature convergence. *Evolutionary Computation, IEEE Transactions on*, 5(4):410 –414, 2001.

[182] G. Rudolph. Evolutionary strategies. In G. Rozenberg, T. Bäck, and J.N. Kok, editors, *Handbook of Natural Computing*, pages 673–698. Springer Berlin Heidelberg, 2012.

[183] G. Rudolph and J. Sprave. A cellular genetic algorithm with selfadjusting acceptance threshold. In *Proc. 1st IEE/IEEE Int. Conf. Genetic Algorithms in Eng. Sys.: Innovations and Appl*, pages 365–372, 1995.

[184] G. Rudolph *et al*, editor. *Parallel Problem Solving from Nature - PPSN X, 10th International Conference Dortmund, Germany, September 13-17, 2008, Proceedings*, volume 5199 of *Lecture Notes in Computer Science*. Springer, 2008.

[185] T.P. Runarsson *et al*, editor. *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference, Reykjavik, Iceland, September 9-13, 2006, Procedings*, volume 4193 of *Lecture Notes in Computer Science*. Springer, 2006.

[186] C. Ryan and M. Keijzer, editors. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2008)*. ACM, 2008.

[187] H. Sakanashi, K. Suzuki, and Y. Kakazui. Controlling dynamics of GA through filtered evaluation function. In Davidor et al. [45], pages 239–248.

[188] Y. Sakurai, K. Takada, T. Kawabe, and S. Tsuruta. A method to control parameters of evolutionary algorithms by using reinforcement learning. In *Signal-Image Technology and Internet-Based Systems (SITIS), 2010 Sixth International Conference on*, pages 74–79, 2010.

[189] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global Sensitivity Analysis. The Primer*. John Wiley & Sons, 2008.

[190] J.D. Schaffer, R.A. Caruana, L.J. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In J.D Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 51–60. Morgan Kaufmann, San Francisco, 1989.

[191] J.D. Schaffer and A. Morishima. An adaptive crossover distribution mechanism for genetic algorithms. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 36–40. L. Erlbaum Associates Inc., 1987.

[192] M Schoenauer *et al*, editor. *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.

[193] H.-P Schwefel. *Numerical Optimisation of Computer Models*. Wiley, New York, 1981.

[194] S.K. Smit. *Parameter Tuning and Scientific Testing in Evolutionary Algorithms*. PhD thesis, VU University Amsterdam, http://www.cs.vu.nl/en/Images/Selmar%20Smit17okt12_tcm75-310264.pdf, October 2012.

[195] A.E. Smith and D.W. Coit. Penalty functions. In *Handbook on Evolutionary Computation*, page C5.2. Oxford University Press, 1997.

[196] J. E. Smith and T. C. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 1:81–87, 1997.

[197] J.E. Smith. *Self Adaptation in Evolutionary Algorithms*. PhD thesis, University of West of England, Bristol, http://www.bit.uwe.ac.uk/jsmith/pdfs/thesis.pdf, 1998.

[198] J.E. Smith and T.C. Fogarty. Self adaptation of mutation rates in a steady state genetic algorithm. In ICEC-96 [117], pages 318–323.

[199] R.E. Smith and E. Smuda. Adaptively resizing populations: Algorithm, analysis, and first results. *Complex Systems*, 9:47–72, 1995.

[200] E. Smorodkina and D. Tauritz. Toward automating EA configuration: The parent selection stage. In CEC-2007 [37], pages 63–70.

[201] E. Smorodkina and D.R. Tauritz. Greedy population sizing for evolutionary algorithms. In CEC-2007 [37], pages 2181–2187.

[202] W.M. Spears. Adapting crossover in evolutionary algorithms. In *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 367–384. MIT Press, 1995.

[203] L Spector *et al*, editor. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. Morgan Kaufmann, San Francisco, 2001.

[204] M. Srinivas and L.M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4):656–667, 1994.

[205] R. Storn and K. Price. Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[206] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1998.

[207] J. Teo. Exploring dynamic self-adaptive populations in differential evolution. *Soft Computing*, 10(8):673–686, 2006.

[208] B. Tessema and G.G. Yen. A self adaptive penalty function based algorithm for constrained optimization. In CEC-2006 [36], pages 246–253.

[209] D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In Hans-Georg Beyer and Una-May O'Reilly, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005)*, pages 1539–1546. ACM, 2005.

[210] D. Thierens. Adaptive strategies for operator allocation. In Lobo et al. [149], pages 77–90.

[211] A. Tuson and P. Ross. Adapting operator settings in genetic algorithms. *Evol. Comput.*, 6(2):161–184, 1998.

[212] R.K. Ursem. *Models for Evolutionary Algorithms and Their Applications in System Identification and Control Optimization*. PhD thesis, University of Aarhus, http://www.brics.dk/DS/03/6/BRICS-DS-03-6.pdf, 2003.

[213] F. Vafaee and P.C. Nelson. An explorative and exploitative mutation scheme. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation*, pages 1 –8, Barcelona, Spain, 2010. IEEE Computational Intelligence Society, IEEE Press.

[214] M. Črepinšek, S.-H. Liu, and M. Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.*, 45(3):35:1–35:33, 2013.

[215] N. Veerapen, J. Maturana, and F. Saubion. A comparison of operator utility measures for on-line operator selection in local search. In *Proceedings of the 6th international conference on Learning and Intelligent Optimization*, LION'12, pages 497–502. Springer-Verlag, 2012.

[216] N. Wagner and Z. Michalewicz. Genetic programming with efficient population control for financial time series prediction. In E.D. Goodman, editor, *2001 Genetic and Evolutionary Computation Conference Late Breaking Papers*, pages 458–462, 2001.

[217] N. Wagner and Z. Michalewicz. Parameter adaptation for GP forecasting applications. In Lobo et al. [149], pages 295–309.

[218] N. Wagner, Z. Michalewicz, M. Khouja, and R. McGregor. Time series forecasting for dynamic environments: The dyfor genetic program model. *IEEE Transactions on Evolutionary Computing*, 11(4):433–452, 2007.

[219] R.J. Wang, Y. Ru, and Q. Long. Improved adaptive and multi-group parallel genetic algorithm based on good-point set. *Journal of Software*, 4(4):348–356, 2009.

[220] Y. Wang, Z. Cai, Y. Zhou, and Z. Fan. Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. *Structural and Multidisciplinary Optimization*, 37(4):395–413, 2009.

[221] J.M. Whitacre, T.Q. Pham, and R.A. Sarker. Use of statistical outlier detection method in adaptive evolutionary algorithms. In Keijzer [127], pages 1345–1352.

[222] Y.-Y. Wong, K.-H. Lee, K.-S. Leung, and C.-W. Ho. A novel approach in parameter adaptation and diversity maintenance for genetic algorithms. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 7:506–515, 2003.

[223] D. Xie, L. Ding, S. Wang, Z. Guo, Y. Hu, and C. Xie. Self-adaptive differential evolution based multi-objective optimization incorporating local search and indicator-based selection. In D-S. Huang, C. Jiang, V. Bevilacqua, and J.C. Figueroa, editors, *Intelligent Computing Technology*, volume 7389 of *Lecture Notes in Computer Science*, pages 25–33. Springer Berlin Heidelberg, 2012.

[224] Xin Yao *et al*, editor. *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Birmingham, UK, September 18-22, 2004, Proceedings*, volume 3242 of *Lecture Notes in Computer Science*. Springer, 2004.

[225] T.-L. Yu, D.E. Goldberg, A. Yassine, and Y.-P. Chen. Genetic algorithm design inspired by organizational theory: pilot study of a dependency structure matrix driven genetic algorithm. In E. Cantú-Paz *et al*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, volume 2723 of *Lecture Notes in Computer Science*, pages 1620–1621. Springer, 2003.

[226] T.-L. Yu, K. Sastry, and D.E. Goldberg. Online population size adjusting using noise and substructural measurements. In CEC-2005 [35], pages 2491–2498.

[227] T.-L. Yu, K. Sastry, and D.E. Goldberg. Population sizing to go: Online adaptation using noise and substructural measurements. In Lobo et al. [149], pages 205–223.

[228] Z.H. Zhan and J. Zhang. Co-evolutionary differential evolution with dynamic population size and adaptive migration strategy. In Krasnogor et al. [133], pages 211–212.

[229] J. Zhang and A.C. Sanderson. Jade: Self-adaptive differential evolution with fast and reliable convergence performance. In CEC-2007 [37], pages 2251–2258.

[230] J. Zhang and A.C. Sanderson. Jade: Adaptive differential evolution with optional external archive. *Evolutionary Computation, IEEE Transactions on*, 13(5):945–958, 2009.

[231] K. Zielinski and R. Laur. Parameter adaptation for differential evolution with design of experiments. In *Parameter Adaptation for Differential Evolution with Design of Experiments*, pages 212–217, 2006.

[232] K. Zielinski and R. Laur. Differential evolution with adaptive parameter setting for multi-objective optimization. In CEC-2007 [37], pages 3585–3592.

[233] K. Zielinski, X. Wang, and R. Laur. Comparison of adaptive approaches for differential evolution. In Rudolph *et al* [184], pages 641–650.

[234] K. Zielinski, P. Weitkemper, R. Laur, and K.-D. Kammeyer. Parameter study for differential evolution using a power allocation problem including interference cancellation. In CEC-2006 [36], pages 1857–1864.

Giorgos Karafotias graduated with a B.Sc. Computer Science degree in 2008 and received a M.Sc Artificial Intelligence degree in 2010. He is currently working on his Ph.D. in Evolutionary Computing with the Computational Intelligence Group at VU University Amsterdam. His research focuses on parameter control in evolutionary algorithms and self-adaptive systems.

Mark Hoogendoorn is an assistant professor within the Computation Intelligence group of the Department of Computer Science at the VU University Amsterdam. Within his research, he focuses on collective adaptive systems, both from a fundamental as well as a more applied perspective. For the latter, the main domains of application include health, wellbeing, and ambient intelligence. He has been involved in various research projects, on both a national and international level, and has been the joint coordinator of the EU FP7 funded ICT4Depression project. Furthermore, he served on numerous program committees and has been one of the program chairs of the IEA/AIE 2013 conference. Before starting as an assistant professor, he was a visiting research at the University of Minnesota. He obtained his PhD degree from the VU University Amsterdam in 2007 focusing on multi-agent organizations.

A.E. (Gusz) Eiben received his MSc in mathematics in 1985 and the PhD in Computer Science in 1991. Since 1999 he is full professor on the VU University Amsterdam. He is one of the European early birds of evolutionary computing. He published his first paper in 1990 and was one of the ”founding fathers” of EvoNet, the European EC Network that initiated, among other things, the EuroGP and EvoStar conferences. He is or has been editorial board member of several EC journals and has (co-)organized various EC conferences. His most cited publication is the best selling textbook ”Introduction to Evolutionary Computing” (co-authored with Jim Smith). His former research interest ranged from multi-parent reproduction, and constraint handling to evolutionary art. Parameter tuning and parameter control are classic subjects and recently his became active in evolutionary robotics.