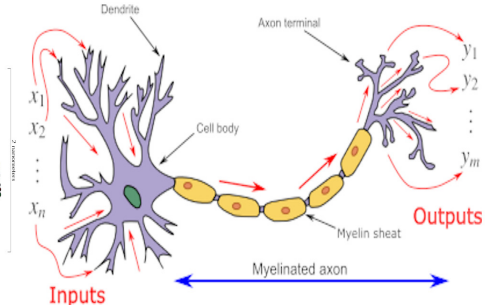
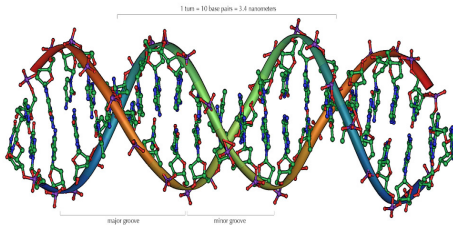


Reflection for GAs and EAS

Andreas Geyer-Schulz | April 22, 2025

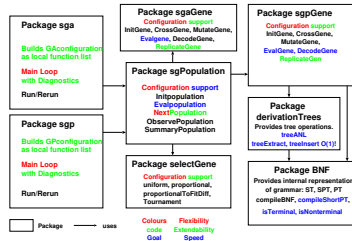
INFORMATION SERVICE AND ELECTRONIC MARKETS ANDREAS.GEYER-SCHULZ@KIT.EDU



1. Reflection
2. Mechanisms
3. Genetic Operators
4. Fitness Scaling and Selection
5. Representations
6. Conclusion
7. Control

Motivation

Context: The Development of the R-Packages sgX for GA/EA/EP/GP/GE.
Architecture 2021 [Geyer-Schulz, 2021]:



Problem: Parameter tuning, automatic algorithm selection, and self-adaptivity: Focus of research in evolutionary algorithms [Lobo et al., 2007].

Support for parameter tuning and self-configuration misses in current R standard GA/GP packages. Examples:

- No support: E.g. R-Package GA [Scrucca, 2021])
- Limited support: E.g. R-Package Adana ([Cebeci and Tekeli, 2022])

Goals:

- Increase adaptivity for parameter tuning and algorithm selection.

Definition

Computational reflection is the activity performed by a computational system when doing computation about (and by that possibly affecting) its own computation [Maes, 1987, p. 147].

Definition

The concept of reflection describes “a mechanism for changing structure and behavior of software systems dynamically” [Buschmann et al., 1996].

Definition

A reflective system is a system which incorporates structures (aspects of) itself. We call the sum of these structures the **self-representation** of the system [Maes, 1987, p. 148].

Example: Simulated Annealing.

In simulated annealing algorithms, the self-representation of the system consists of 4 elements: $f(s)$, $f(s_{new})$, the counter k , and the maximal number of trials k_{max} .

Self-adaptive behavior is

- The computation of the temperature $T(k, k_{max})$
- The computation of the probability of acceptance of a worse solution $P(f(s), f(s_{new}), T_k)$.

To record all parameters with which a system has been called.

```
a < — RunSGA(F7, popsize=400, generations=200, crossrate=0.25, mutrate=0.001)
names(a)
```

```
[1] "popStat" "solution" "evalFail" "GAconfig" "GAenv" "timer"
```

a: The persistent self-representation of RunSGA.

```
a$GAconfig
```

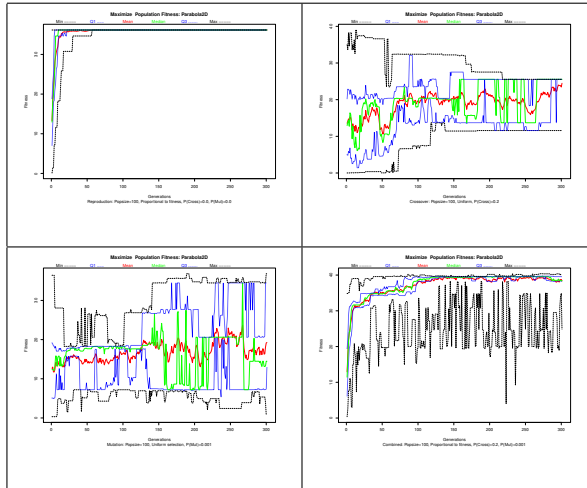
```
[[1]]
```

```
[1] "RunSGA(F7,evalmethod=\"EvalGeneU\",generations=200,popsize=400,
crossrate=0.25,crossrate2=0.3,ivcrossrate=\"Const\", crossover=\"Cross2Gene\",uCrossSwap=0.2,
mutrate=0.001,mtrate2=0.01,cutoffFit=0.5, mutation=\"MutateGene\", replication=\"Kid2\",
max=TRUE,offset=1,eps=0.01,tournamentSize=2,selectionBias=1.5, selection=\"SUS\",
mateselection=\"SUS\",selectionContinuation=TRUE, scaling=\"NoScaling\",
scalingThreshold=0,scalingExp=1, scalingExp2=1,rdmWeight=1,dispersionMeasure=\"var\",
scalingDelay=1,elitist=TRUE,verbose=1,allsolutions=FALSE,replay=0,
executionModel=\"Sequential\", workers=NA, master=NA, port=NA, homogeneous=TRUE,
profile=FALSE, batch=FALSE)"
```

Repeatability of computational experiments!

Examples of the Use of Reflection in sgX

Visualization of the development of the population statistics.



- To keep information for debugging, stepping, and tracing facilities.
- Reasoning about control.
- Self-Optimisation.
- Self-Modification.
- Self-Activation.

Reflection (193), [Buschmann et al., 1996, p. 193-219]

The **Reflection** architectural pattern provides a mechanism for changing structure and behavior of software systems dynamically. It supports the modification of functional aspects such as type structures and function call mechanisms.

In this pattern, an application is split into two parts:

1. A meta level provides information about selected system properties and makes the software self-aware.
2. A base level includes the application logic. Its implementation builds on the meta level. Changes to information kept in the meta level affect subsequent base-level behavior.
3. For the manipulation of meta objects, an interface (the **meta object protocol (MOP)**) is specified and implemented. It is responsible for checking the correctness of the change specification and for performing the change.

Uninformed. The choice of parameters depends on time (e.g. measured as the number of function evaluations). No additional information from the search process is used.

Example: Simulated Annealing.

Self-Adaptive. The search for optimal parameter values and the search for optimal solutions is combined. The meta-level information about the algorithm configuration is encoded into the genotype.

Example: Adaptive population size, mutation rate, and crossover rate [Bäck et al., 2000].

Adaptive. The search for optimal parameter values and the search for optimal solutions is separated. The meta-level information is collected by monitoring properties of the algorithm's run (e.g. the population performance statistics). Changes in the algorithm's properties are used to guide the update of the parameters.

Example: Credit assignment to genetic operators over a an adaptation window for participating in generating genes with improved fitness [Davis, 1989].

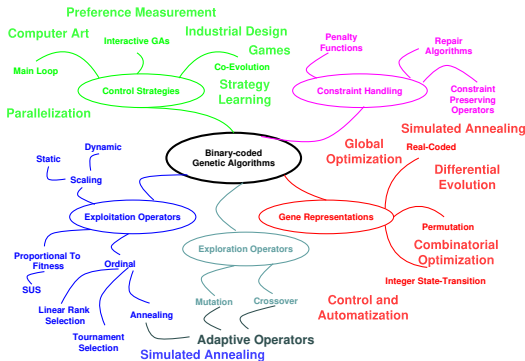
Benefits:

- No explicit modification of source code.
- Changing a software system is easy.
- Support for many kinds of changes.

Liabilities:

- Modifications at the meta level may cause damage.
- Increased number of components.
- Lower efficiency.
- Not all potential changes are supported.

A Partial Mind Map of Variants of Genetic Algorithms



How do we choose population size as well as the type (Which operators?) and amount (Which parameters?) of reproductive variation for a particular problem of interest?

(Kenneth De Jong [?]) gives a historic account of the “parameter tuning problem” over the last 40 years.)

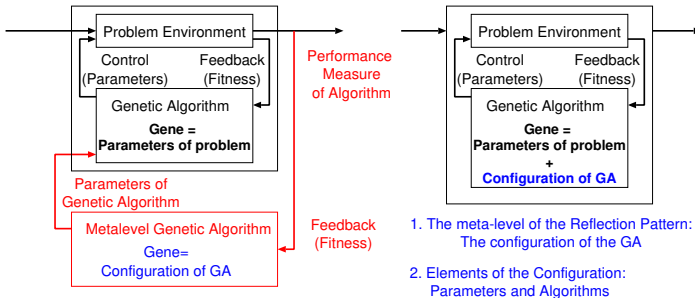
Roughly speaking, we show that for both static and time- dependent optimization problems, the average performance of any pair of algorithms across all possible problems is identical. [Wolpert and Macready, 1997, p. 67]

- Are there **optimal settings** for the evolutionary algorithms in general (**across all possible problems**)?

No!

- Are there ways of finding **improved parameter** values for **a specific (class of) problems**?

Yes!

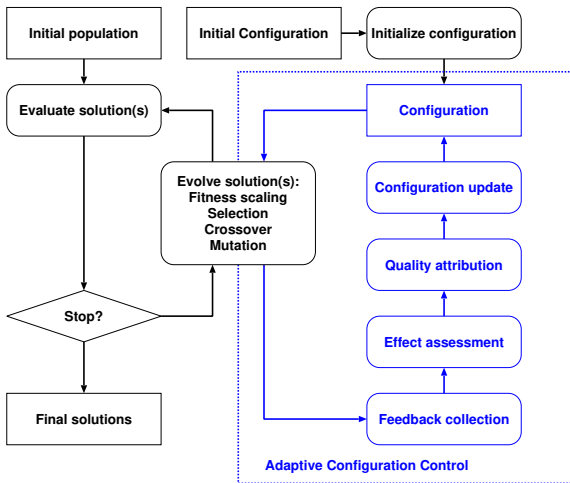


John G. Grefenstette (1986)
Optimization of Control Parameters
for Genetic Algorithms
[Grefenstette, 1986].

Hans P. Schwefel (1981) Numerical
Optimization of Computer Models
[Schwefel, 1981]

Reflection is essential for both types of algorithms.

A Conceptual Model of Adaptive Configuration Control for Genetic and Evolutionary Algorithms



(See [Aleti and Moser, 2016, p. 56:6])

What Has Been Done ... (1990-2015)

Feedback collection		Effect assessment	
Phenotype	86.2% (+)	Best	34.6%
Feasibility	15.8%	Current	29.7%
Genotype	15.1%	Ancestor	25.8% (+)
Genotype diversity	6.3%	Population	23.3%
Relative Phenotype	5.7%	Worst	9.4%
		Median	1.3%
Quality attribution		Parameter update	
Learned	42.1% (+)	Greedy	37.7% (+)
Immediate	34.6%	Quality-proportionate (QP)	34.0%
Average	13.2%	Deterministic	20.1%
Extreme	10.7%	QP minimum probability	13.2%

Number of papers reviewed: 152.

Sources: Google Scholar, IEEE Explore, ACM Digital Library, Springer Digital Library, Elsevier Science Direct

Percentage of papers indicates focus of work.

+ indicates “trend leader” in 2015. (See [Aleti and Moser, 2016, p. 56:9])

Feedback Collection, Effect Assessment, Quality Attribution, Parameter Update

Feedback. Feedback is a measure of properties indicating the algorithm's behavior.

Phenotype (quality of solution), *relative phenotype* (relative quality of solution), *genotype* (information from parts of a solution), *genotype diversity* in a population of solutions, *feasibility* (violations of constraints).

Effect Assessment. Estimation of effect of parameter values on algorithmic performance from feedback. Goal: Maximize the cumulative effect by adapting the probability distribution of parameters. Success of a parameter: Ancestor effect, population effect, best, worst, median effect, current effect.

Quality Attribution. Each parameter value has a quality estimate which is updated from the observed effect whenever a parameter value is used.
Immediate quality attribution (current solution), *average quality attribution* (the average change of a population), *extreme quality attribution* (outliers, best) *learned quality attribution* (e.g. reinforcement learning).

Parameter Update. Update mechanisms differ in the trade-off between exploitation and exploration: Quality proportionate update (the probability distribution of selecting a parameter values is updated based on estimated quality), with minimum probability, *greedy update* (best parameter value in next iteration), *deterministic update* (a predefined rule).

1. Parameters: Population size (23.3%, mutation rate (27.7%), crossover rate (22.6%), fitness (21.4
2. Algorithms: Mutation operator (18.2%), crossover operator (13.2%), parent selection (9.4%), ..
3. Representations: Encoding (1.9%), grammars (TBD), ...

The percentage rates indicate the percentage of papers in the survey of Aleti and Moser [Aleti and Moser, 2016]

Current Architectural Support for Adaptivity of Genetic Operators in the R-Packages sgX

The Reflection Pattern.

- **The base level:** For fitness scaling, selection, mutation, and crossover the basic algorithms are implemented and encapsulated in function factories.

Each function factory generates functions with the same abstract interface. Each interface includes a parameter for passing a list of local functions.

- **The meta level:** The list of local functions contains a part of the internal system representation of the algorithm: It is used both for parameters and algorithms. Dynamic changes are possible.

The internal system representation of the algorithm run is a vector of 8 population fitness statistics (mean, min, Q1, median, Q3, max, variance, mean absolute deviation) per generation.

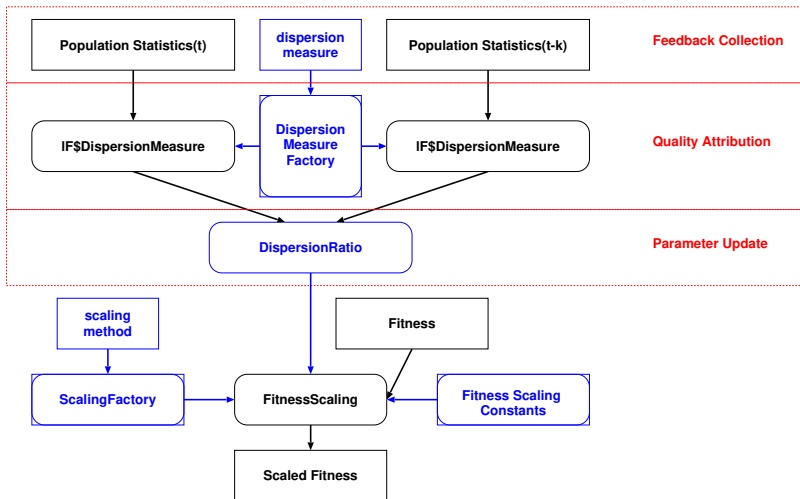
- **The meta object protocol:** The list of local functions `lf` is changed by assigning functions generated by function factories and by assigning parameters as constant functions.

The use of a local function is by function dispatch. E.g. `lf$mutrate()`.

The Principle of Orthogonality: All algorithms are independent of each other.

- Adaptation strategies: Threshold, Ratio-Based, Relative Difference, ...
- Type of Measure: e.g. Location and spread.
- Time-Lag in measures.
- Probabilities of operators and operator mixes.
- Learning?
- Local operator-specific parameters.

- General operator parameters are:
 - Exponent of fitness scaling by power function.
 - Crossover Rate.
 - Mutation Rate.
- Adaptation of these parameters is independent of the gene representation of the evolutionary algorithm.
Example: Threshold rules work for GP algorithms (see [Stanhope and Daida, 1996]) as well as for GAs.



1. No scaling: Identity function.
2. Base algorithm: Power scaling: fit^{exp} Parameter: $exp \in R$
3. Threshold based adaptation strategy:

d_t, d_k dispersion of population fitness

$r = \frac{d_t}{d_k}$ dispersion ratio.

If $r \in [1 - \epsilon, 1 + \epsilon]$ then no scaling.

If $r < (1 - \epsilon)$ then power scaling with $exp1$.

If $r > (1 + \epsilon)$ then power scaling with $exp2$.

4. Continuous adaptation strategy: $exp = r \cdot w$ where w is a weight parameter.

Configuration choices: $2 + k \cdot m \cdot a$ where k number of time differences, $m = 6$ number of dispersion measures (variance, standard deviation, median absolute deviation, coefficient of variation, range, inter quartile range) and $a = 2$ number adaptation strategies.

For $k = 2$, we currently have $24 + 2$ configuration variants for scaling (without considering the parameters exp , $exp2$, ϵ , and w .)

Fitness scaling does not influence selection methods which are order based:

- Linear Rank Selection
- Deterministic Tournament Selection

Individually variable mutation (and crossover) rates are based on the following intuitive ideas:

- If a gene with low fitness which has been selected for reproduction wants to increase its future chances of survival, a higher mutation, respectively crossover, rate will help.
- Low fitness is a stress factor which increases mutation and crossover rate.
- From the schema theorem of GAs it is clear that reproducing a low fitness individual does not contribute to improve average population fitness. However, applying mutation or crossover to a low fitness individual will lead with relatively high probability to an improved offspring.

Individually Variable Mutation and Crossover Rates

Stanhope and Daida have introduced the following simple deterministic adaptation rule for individually variable rates [Stanhope and Daida, 1996]:

- A threshold k for classifying genes as high or low fitness genes:

$$k = c \cdot \text{fit}_{\text{best}}$$

where $0 < c < 1$ is a cutoff fitness score.

- Let r_1 be the rate for genes with high fitness and r_2 the rate for genes with low fitness:

$$p_{\text{operator}}(i) = \begin{cases} r_1 & \text{if } \text{fit}_i > k \\ r_2 & \text{if } \text{fit}_i \leq k \end{cases}$$

- Stanhope and Daida have used this deterministic rate adaptation strategy for mutation rates only.
- In the R-packages sgX the strategy is used both for crossover and mutation rates.
- Obvious generalizations are basing the classification on other population location measures and making the adaptation of the rate proportional to the fitness difference or ratio.

Shekel's Foxholes: A Small Experiment (Mutation)

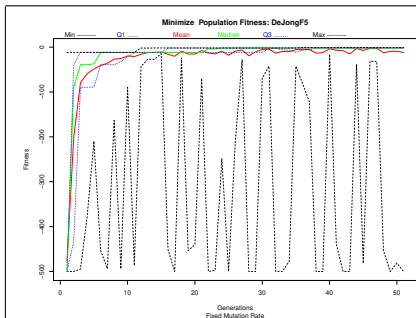
F5 is a 2-dimensional multimodal function suggested by Shekel (1971) called Shekel's foxholes. It is a continuous, non-convex, multimodal, non-quadratic function with 25 local minima defined by the 25 columns of the matrix A. The minimum is at (-32, -32) with a minimum of 0.998.

- We compare 100 runs of a standard simple genetic algorithm
RunSGA(penv=F5, max=FALSE, popsize=100, generations=50, crossrate=0.2, mutrate=0.001)
- with individually adaptive mutation active:
RunSGA(penv=F5, max=FALSE, popsize=100, generations=50, crossrate=0.2, mutrate=0.001, mutrate2=0.01, cutoffFit=0.6, mutation="IVM")

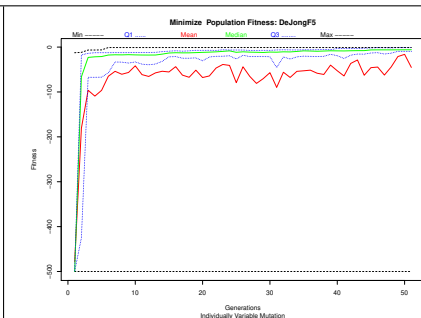
How often does the algorithm reach the correct hole with restricted resources in 100 or 1000 trials?

Number of trials	100	1000
Correct Holes standard SGA	61	664
Correct Holes SGA with IVM	95	965

Shekel's Foxholes: Visualization of Population Statistics



Fixed mutation rate



Individually variable mutation rate

- We compare 100 runs of a standard simple genetic algorithm
RunSGA(penv=F5, max=FALSE, popsize=100, generations=50, crossrate=0.2, mutrate=0.001)
- with individually adaptive crossover active:
RunSGA(penv=F5, max=FALSE, popsize=100, generations=50, crossrate=0.2, crossrate2=0.35, cutoffFit=0.7, ivcrossrate="IV",) mutrate=0.001)

How often does the algorithm reach the correct hole with restricted resources in 100 or 1000 trials?

Number of trials	100	1000
Correct Holes standard SGA	59	680
Correct Holes SGA with IVC	72	724

Self-Adaptive Representations: Grammars in GP (Sketch)

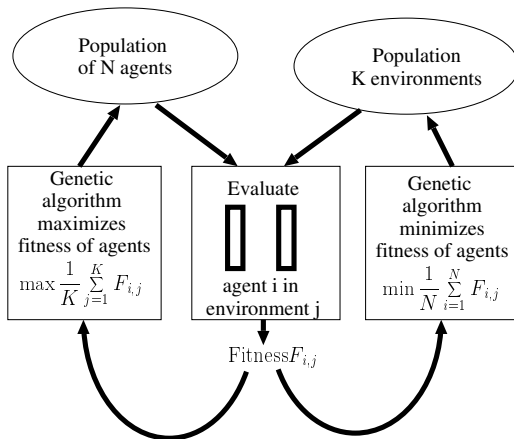
- Use 3 chromosomes:
 1. Chromosome 1: The derivation tree of the program.
 2. Chromosome 2: The production table of the grammar as a list of production rules (pairs of the form (nonterminal, symbol list)).
 3. Chromosome 3: Additional production rules.

Candidates for this list can be drawn by extracting random subtrees with small limited depth from chromosome 1. The root symbol of the subtrees and the frontier of the subtree is a pair which forms a valid new production rule.
- How do we select promising subtree candidates?
- How do we assess the performance contribution (fitness) of each of these production rules?

- The architectural pattern of reflection provides a unifying view on the integration of adaptation facilities into software packages for evolutionary algorithms in a configurable and general way.
- Many very specific and limited mechanisms of adaptation have been introduced and investigated in the literature. The reflection pattern helps in generalizing.
- Computational experiments are encouraged and facilitated. This makes the exploration of the design space of adaptive and self-adaptive systems easier.
- We have shown the first steps towards integrating such features into the R-packages sgX which are under development and first promising experimental results.

(Variants of the *Main Loop*)

- Genetic Algorithms with Fixed Size Population Generations (Done)
- Steady-State GAs, Overlapping Generations, ESS, $1 + \lambda$, ...
- Multi-Objective GAs
- Parallel and Distributed GAs
- **Co-Evolution**



- Game Playing from Zero Knowledge
- Designing Complex Environments and Their Solution
- Simulation of Complex Ecosystems

- Computational reflection: [Maes, 1987].
- Parameter control in evolutionary algorithms:
[Karafotias et al., 2015], [Eiben et al., 2007], [De Jong, 2007],
[Eiben et al., 1999], [Smith and Fogarty, 1997].
- Algorithm configuration: [Hutter et al., 2011],
[Leyton-Brown et al., 2003].
- Heuristics: [Burke et al., 2013]



Aleti, A. and Moser, I. (2016).

A systematic literature review of adaptive parameter control methods for evolutionary algorithms.




ACM Comput. Surv., 49(3(56)):1 – 35.



Bäck, T., Eiben, A. E., and van der Vaart, N. A. L. (2000).

An empirical study on gas “without parameters”.

In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature PPSN VI*, pages 315–324, Berlin, Heidelberg. Springer Berlin Heidelberg.

-  Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Qu, R. (2013).
Hyper-heuristics. a survey of the state of the art.
The Journal of the Operational Research Society, 64(12):1695 – 1724.
-  Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996).
Reflection.
In *Pattern-Oriented Software Architecture. A System of Patterns*, volume 1, chapter 2.5.2, pages 193 – 220. John Wiley, Chichester.
-  Cebeci, Z. and Tekeli, E. (2022).
R Package Adana. Adaptive Nature-Inspired Algorithms for Hybrid Genetic Optimization.



Davis, L. (1989).

Adapting operator probabilities in genetic algorithms.

In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61 – 69, Los Altos. George Mason University, Morgan Kaufmann Publishers.



De Jong, K. (2007).

Parameter setting in EAs: A 30 year perspective.

In Lobo, F. G., Lima, C. F., and Michalewicz, Z., editors, *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*, chapter 1, pages 1 – 18. Springer Berlin Heidelberg, Berlin, Heidelberg.






Eiben, A., Hinterding, R., and Michalewicz, Z. (1999).
Parameter control in evolutionary algorithms.
IEEE Transactions on Evolutionary Computation, 3(2):124 – 141.




Eiben, A. E., Michalewicz, Z., Schoenauer, M., and Smith, J. E.
(2007).
Parameter control in evolutionary algorithms.
In Lobo, F. G., Lima, C. F., and Michalewicz, Z., editors, *Parameter Setting in Evolutionary Algorithms*, chapter 2, pages 19 – 46.
Springer Berlin Heidelberg, Berlin, Heidelberg.



Geyer-Schulz, A. (2021).
Architectural design of a unified ga/gp-package for r.
Talk at DSSV-ECDA 2021.

-  Grefenstette, J. G. (1986).
Optimization of control parameters for genetic algorithms.
IEEE Transactions on Systems, Man, and Cybernetics, 16(1):122 – 128.
-  Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011).
Sequential model-based optimization for general algorithm configuration.
In Coello, C. A. C., editor, *Learning and Intelligent Optimization*, volume 6683 of *Lecture Notes in Computer Science*, pages 507 – 523. Springer Berlin Heidelberg, Berlin, Heidelberg.
-  Karafotias, G., Hoogendoorn, M., and Eiben, A. E. (2015).
Parameter control in evolutionary algorithms: Trends and challenges.
IEEE Transactions on Evolutionary Computation, 19(2):167 – 187.

-  Leyton-Brown, K., Nudelman, E., Andrew, G., McFadden, J., and Shoham, Y. (2003).


A portfolio approach to algorithm select.

In *Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI'03*, pages 1542 – 1543, San Francisco, CA, USA.
Morgan Kaufmann Publishers Inc.

-  Lobo, F. G., Lima, C. F., and Michalewicz, Z., editors (2007).





Parameter Setting in Evolutionary Algorithms.


Springer Berlin Heidelberg, Berlin, Heidelberg.

-  Maes, P. (1987).

Concepts and experiments in computational reflection.

SIGPLAN Not., 22(12):147 – 155.

-  Schwefel, H.-P. (1981).
Numerical Optimization of Computer Models.
Wiley, Chichester.
-  Scrucça, L. (2021).
R-Package GA: Genetic Algorithms.
-  Smith, J. and Fogarty, T. (1997).
Operator and parameter adaptation in genetic algorithms.
Soft Computing, 1(2):81 – 87.
-  Stanhope, S. A. and Daida, J. M. (1996).
An individually variable mutation-rate strategy for genetic algorithms.
In Koza, J. R., editor, *Late Breaking Papers at the Genetic Programming 1996 Conference*, pages 177 – 185, Stanford. Stanford University Bookstore.

-  Wolpert, D. and Macready, W. (1997).
No free lunch theorems for optimization.
IEEE Transactions on Evolutionary Computation, 1(1):67 – 82.