GECCO 2020 Tutorial: Dynamic Control Parameter Choices in Evolutionary Computation

Gregor Papa[1] and Carola Doerr[2]

[1] Jožef Stefan Institute, Ljubljana, Slovenia
[2] Sorbonne University, Paris, France
gregor.papa@ijs.si & carola.doerr@mpi-inf.mpg.de

https://gecco-2020.sigevo.org/

GECCO '20 Companion, Cancun, Mexico

ACM ISBN 978-1-4503-7127-8/20/07
doi:10.1145/3377929.3389876

Instructors

Gregor Papa is a Senior researcher and a Head of Computer Systems Department at the Jožef Stefan Institute, Ljubljana, Slovenia, and an Associate Professor at the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia. He received the PhD degree in Electrical engineering at the University of Ljubljana, Slovenia, in 2002. His research interests include meta-heuristic optimisation methods and hardware implementations of high-complexity algorithms, with a focus on dynamic setting of algorithms' control parameters. He regularly organizes conferences and workshops in the field of nature-inspired algorithms from the year 2004 till nowadays. He led and participated in several national and European projects.

Carola Doerr is a permanent CNRS researcher at Sorbonne University in Paris, France. Her main research activities are in the mathematical analysis of randomized algorithms, with a strong focus on evolutionary algorithms and other black-box optimizers. She has been very active in the design and analysis of black-box complexity models, a theory-guided approach to explore the limitations of heuristic search algorithms. She has used knowledge from these studies to prove superiority of dynamic parameter choices in evolutionary computation. Carola has received several awards for her work on evolutionary computation, among them the Otto Hahn Medal of the Max Planck Society and four best paper awards at GECCO. She is/was programm committee chair of PPSN 2020, FOGA 2019, and the GECCO theory track in 2015 and 2017.

- **Version Management:** Not only parameters, but also these slides are dynamic :-) You can find the latest version on our homepages.
  - http://www-ia.lip6.fr/~doerr/DoerrGECCO20tutorial.pdf
  - http://cs.ijs.si/papa/files/GECCO2020tutorial.pdf
- **Terminology:** This tutorial was designed for GECCO attendees. If you are interested in the topic, but not familiar with the terminology used in evolutionary computation, please do not hesitate to contact us. We will be happy to discuss the ideas, methods, and results in a language that avoids terms like "*fitness*", "*mutation*", "*crossover*", "*selection*", etc.

Survey Articles

In 110 minutes we cannot discuss all existing works. Summaries of the state-of-the-art techniques and a good starting point for your research can be found in these surveys (see reference list at the end for details)

- **Empirical works**
  - Karafotias, Hoogendoorn, Eiben, 2015 [KHE15]
    (detailed survey of empirical works)
  - Aleti, Moser, 2016 [AM16]
    (systematic literature survey with additional pointers)
  - Eiben, Hinterding, Michalewicz, 1999 [EHM99]
    (classic seminal paper, introduced a now widely accepted classification scheme)
  - Lobo, Lima, Michalewicz, 2007 [LLM07]
    (book on parameter selection, includes chapters on tuning and control)
- **Theoretical works**
  - Doerr, Doerr, 2020 [DD20]
    (surveys theoretical works which prove performance bounds with mathematical rigor; introduces the revised classification scheme discussed below)

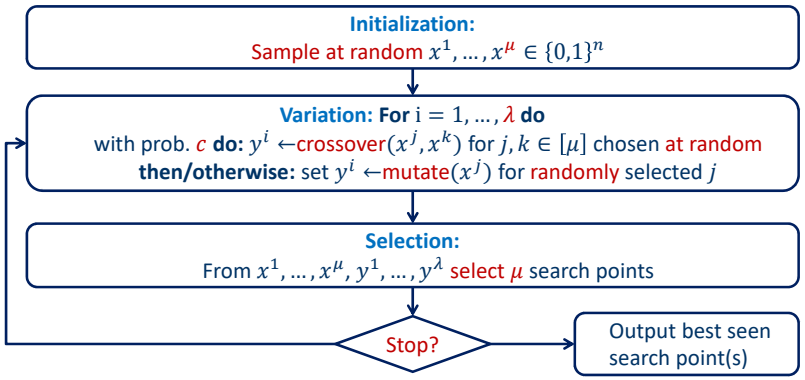
Outline of the Tutorial

- **Part 1:** Introduction
  What are the goals of parameter control?
- **Part 2:** Example: Controlling the Mutation Rate of the (1+1) EA
  Motovating example, 1/5-th success rule
- **Part 3:** Taxonomy of Parameter Control Mechanisms
  Which parameter control techniques exist, and how can we classify them?
- **Part 4:** Applications of Parameter Control
  Where is parameter control used in practice?
- **Part 5:** Wrap Up
  Let's stay in touch

Part 1: Introduction

All iterative optimization heuristics are parametrized

- Here is a "typical" evolutionary algorithm, a $(\mu + \lambda)$ GA with crossover



→ There are quite a few parameters that need to be decided upon: population size, crossover rate, mutation rate, selective pressure

Are Parameter Values Important?

- The very early days of EC:
  "*EAs are robust problem solvers*"
  → no need to tune parameters!
- However, it was soon realized that this hope does not (and, in fact, cannot, as the "no free lunch" theorems tell us) materialize. It is today widely acknowledged that the parameter values have a decisive influence on the performance of an EA.
- Big open question (to date!): *How to find good parameter values?*

Globally Good Parameter Values?

- "Sports" of the 70s/80s in EC: Finding good parameter values
  - good = "globally good", i.e., for a broad range of problems
  - Examples: De Jong [DJ75], Grefenstette [Gre86] give recommendations for parameters such as population size, mutation and crossover probabilities, selection strategies, etc.

    → these recommendations are *independent* of problem class, problem size, … (*absolute values*)
  - Mühlenbein [Müh92] and others suggest $1/n$ as mutation rate for problems of lengths n (*relative values*)
    - Note: we know today that this choice indeed works well for a broad range of problems, see discussion below. However, it is widely acknowledged today, that problem size is not the only feature that matters.
- "Modern view" of parameter selection: no globally optimal parameter values exist
  → parameters need to be adjusted to the problem at hand

Difficulty of Finding Good Parameter Choices

1. Even if we find "optimal" parameter values for one problem, these may (!, don't have to) be much different for similarly-looking problems
2. Small changes in one parameter can (!, don't have to) cause huge performance gaps
   - Many empirical works on this matter exist (again, check this year's GECCO talks to see if/how much effort has been put into finding the right parameters)
3. Finding optimal parameter values is far from being trivial
   - it is basically a meta-optimization problem, typically mixed-integer, noisy, non-smooth ☹
   - Those of you interested in theoretical results can find in [DJS+13] or [Len18] examples where changing the mutation rate by a small constant factor changes the expected running time from a small polynomial (e.g., $O(n \log n)$) to super-polynomial/exponential
4. Optimal parameter values can change during the optimization process
   - e.g., more exploration in the beginning, more exploitation towards the end (example: Simulated Annealing → increasing selective pressure)

Parameter Control

Goals of Parameter Control

- ✓ to **identify** good parameter values "on the fly" e.g., when prior training or tuning is not possible → integrate the tuning procedure into the optimization process
- ✓ to **track** good parameter values when they change during the optimization process
  Significant performance gains possible (not only constant factors!)

Part 2: Examples & 1/5 Success Rule

The LeadingOnes Problem

- Classic benchmark problem often studied in the theory of evolutionary computation (as one the simplest examples of a non-separable function)
- Original function:

$$\mathrm{LO}: \{0,1\}^n \to \mathbb{R}, x \mapsto \mathrm{LO}(x) = \max\{i \in [n] \mid \forall j \leq i : x_i = 1\}$$

1	1	0	1	1	0	1	1

LO-value: 2 (2 initial ones)

- Looks like a "stupid" problem? For most EAs, it is equivalent to this game:

$$\mathrm{LO}_{z,\sigma}: \{0,1\}^n \to \mathbb{R}, x \mapsto \mathrm{LO}_{z,\sigma}(x) = \max\{i \in [n] \mid \forall j \leq i : x_{\sigma(i)} = z_{\sigma(i)}\}$$

0	0	0	1	1	0	1	1

secret code z

4	8	5	1	7	2	6	3

secret permutation σ

1	1	0	1	1	0	1	1

$\mathrm{LO}_{z,\sigma}$-value: 3 (first 3 bits in the order prescribed by σ are coincide with those of z)

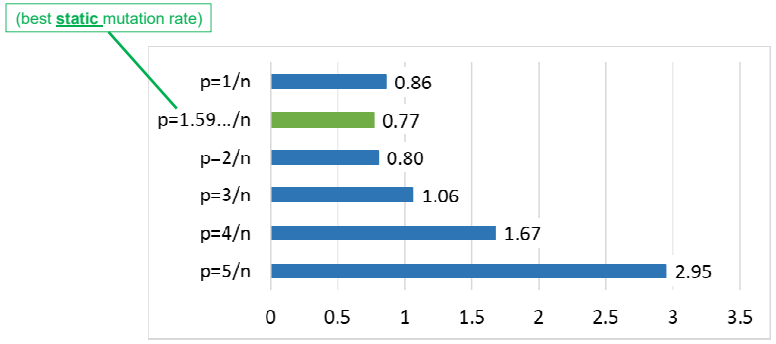
The (1+1) EA

- Initialization:
  Choose $x \in \{0,1\}^n$ uniformly at random (u.a.r.).
- Optimization: in iteration $t = 1,2,\ldots$ do
  1. create y from x by standard bit mutation \\ "mutation"
     i.e., flip each bit w/ probability p, independently of all other bits
  2. If $f(y) \geq f(x)$ replace x by y \\ "selection"

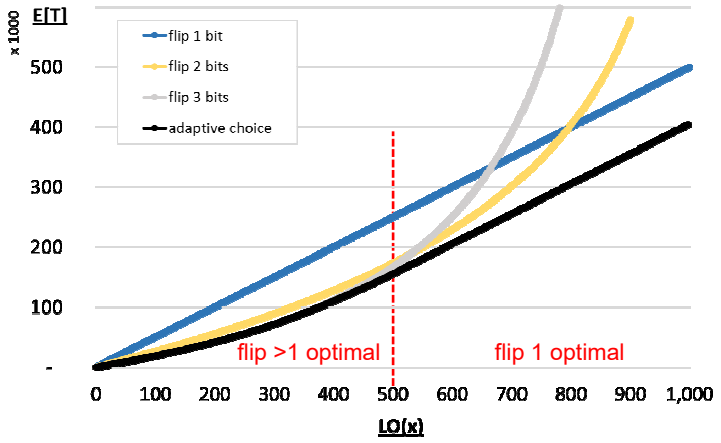
Critical parameter: the mutation rate p
(often recommended choice: $p = 1/n$)

Proven Optimization Times

- The expected optimization time of the (1+1) EA on LeadingOnes is $\frac{1}{2p^2}\left(\frac{1}{(1-p)^{n-1}} - 1 + p\right)$ [BDN10]

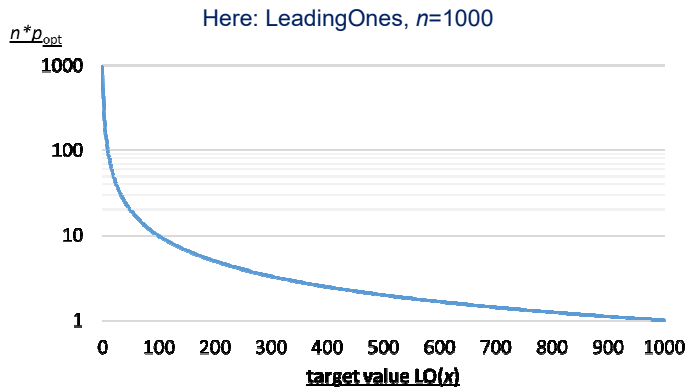


Fixed-Target Running Times



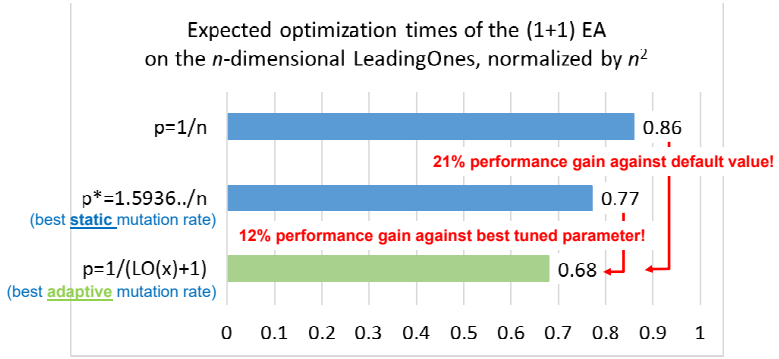
Expected fixed-target running times for dimension n=1000

Optimal Mutation Rates



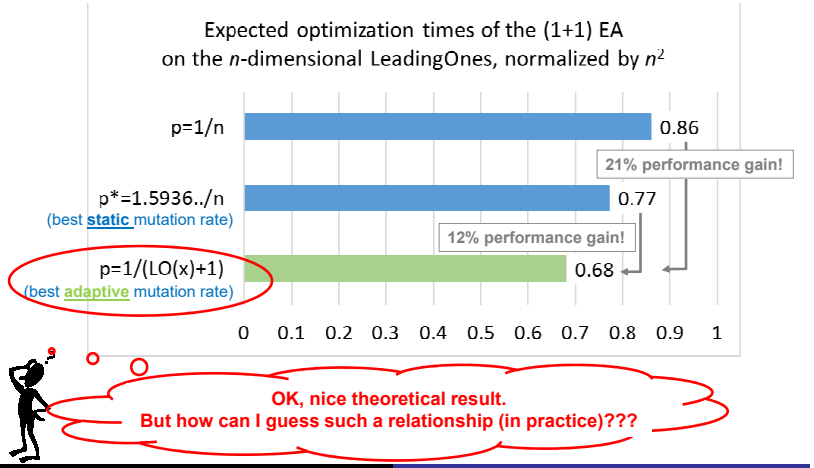
- $p_{\text{opt}} = \frac{1}{\text{LO}(x)+1}$ [BDN10,Sud13]
- $k_{\text{opt}} = \left\lfloor \frac{n}{\text{LO}(x)+1} \right\rfloor$ [DW18,Doerr19]

Measurable Gain of Dynamic Mutation Rates



These results were proven in [BDN10]

Measurable Gain of Dynamic Mutation Rates



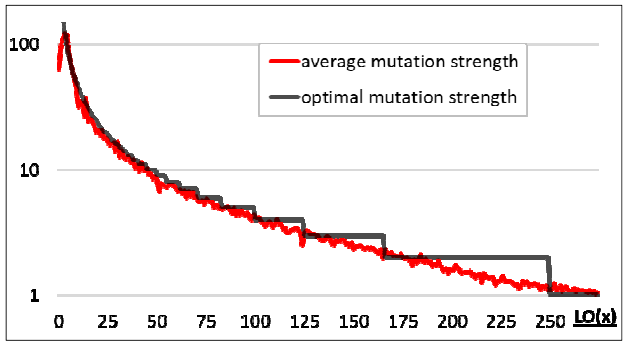
(1+1) EA with Adaptive Mutation Rates

- Initialization:
  1. Choose $x \in \{0,1\}^n$ uniformly at random (u.a.r.).
  2. Initialize $p = 1/n$
- Optimization: in iteration $t = 1,2,\ldots$ do
  1. create y from x by standard bit mutation w/ mutation rate p
  2. If $f(y) \geq f(x)$
     - replace x by y \\ selection
     - replace p by Ap \\ parameter update
  3. If $f(y) < f(x)$
     - replace p by bp \\ parameter update

based on a variant from [DW18]

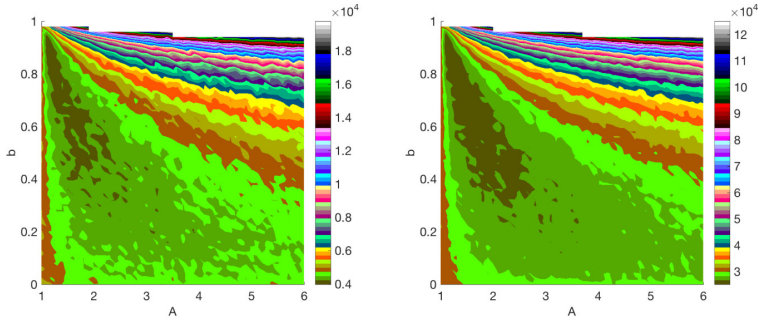
Illustration (taken from [DW18])

- Results for LeadingOnes, $n = 500$
- Update strengths: $A = 2, b = 1/2$
- Plot compares average number of bits flipped (red) vs. optimal number (black)
- Logarithmic scale, zoom into $\mathrm{LO}(x) \leq 250$:



Fitness Landscape of Tuning Problem (from [DW18])

The performance gain is not very sensitive with respect to the choice of the hyper-parameters A and b:

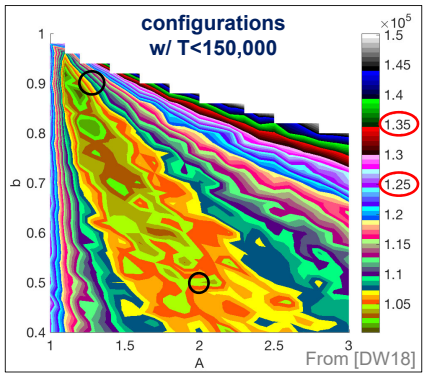


(d) LeadingOnes with $n = 100$ (e) LeadingOnes with $n = 250$

Fitness Landscape of Tuning Problem (from [DW18])

The performance gain is not very sensitive with respect to the choice of the hyper-parameters A and b:

Heatmap shows average optimization time for different combinations of A and b for the adaptive (1+1) EA on 500-dimensional LeadingOnes
(the static (1+1) $\mathrm{EA}_{>0}$ needs $\approx 135{,}000$ function evaluations, RLS 125,000)



From [DW18]

Typical values for A and b are

- $A = 2, b = \frac{1}{2}$
  (gives an avg runtime of $\approx 104{,}000$)
- better values exist, as we shall discuss next

The 1/5-Success Rule

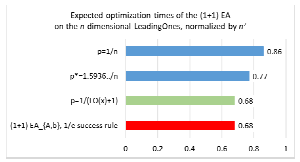
- Probably *the* most famous success-based parameter adaptation rule
- Rechenberg [Rec73]:
  - observed that for the sphere function and a corridor landscape the optimal success rate of the (1+1) ES is around 1/5 (i.e., there is some theoretical foundation of this rule)
  - Suggestion:
    - If (observed success rate > 1/5) → increase mutation rate
      Informal interpretation: we seem to be in an easy part of the optimization problem → increasing mutation rates might result in larger progress per step
    - If (observed success rate < 1/5) → decrease mutation rate
      Informal interpretation: we could be approaching an optimum and should focus our search → decrease mutation rate for a more conservative search
  - Note 1: similar rules have been proposed by Schumer, Steiglitz 68 [SS68] and Devroye [Dev72]
  - Note 2: the same idea can also be used to control other parameters, such as the population size, crossover probabilities, etc.

1/5 Success Rules (indirect interpretation)

- Indirect interpretation of the 1/5-success rule, suggested in [KMH+04] and later also used in [Aug09]
  - When the success rate is around $1/5$, the parameter value should be stable
  - In our algorithm:
    - Our update rule
      If $f(y) \geq f(x)$: $p \leftarrow \min\left\{Ap, \frac{1}{2}\right\}$
      else $\quad p \leftarrow \max\{bp, 1/n^2\}$
    - When $A = \left(\frac{1}{b}\right)^4$ and success rate is 1/5, then after 5 iterations the mutation rate p becomes $pAb^4 = p$, i.e., it is stable!
    - When $A = \left(\frac{1}{b}\right)^{s-1}$ and success rate is $1/s$, then after s iterations the mutation rate p becomes $pAb^{s-1} = p$, i.e., it is stable!

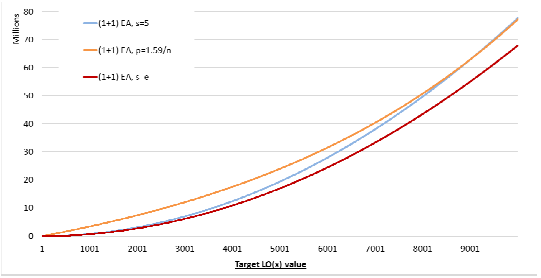
1/5 Success Rule Can be Optimal also in Discrete Optimization

- Theorem [DDL19]: The 1/e success rule yields optimal mutation rates for the (1+1) EA on LeadingOnes.



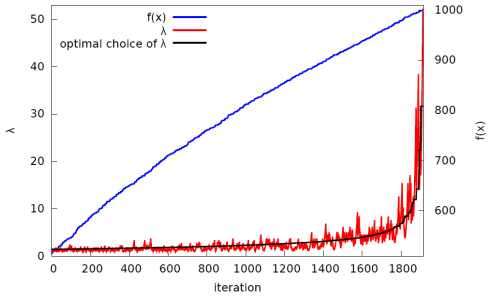
Average Fixed-Target Running Times for the (1+1) EA with mutation rates:

- best static
- 1/5 success rule
- 1/e success rule



Example 2: The (1+(λ, λ)) Genetic Algorithm

- Previous example (LeadingOnes): constant factor gain over best static mutation rate
- Now: asymptotic gain!
- The self-adjusting (1+(λ, λ)) GA uses the 1/5 success rule to control the population size λ. In [DDE15] the fit between the optimal value and the dynamically selected λ was observed to be very good:



Theoretical result proven in [DD18]: 1/5 success rule yields linear (and thus asymptotically optimal !) performance. No static choice of λ achieves linear time.

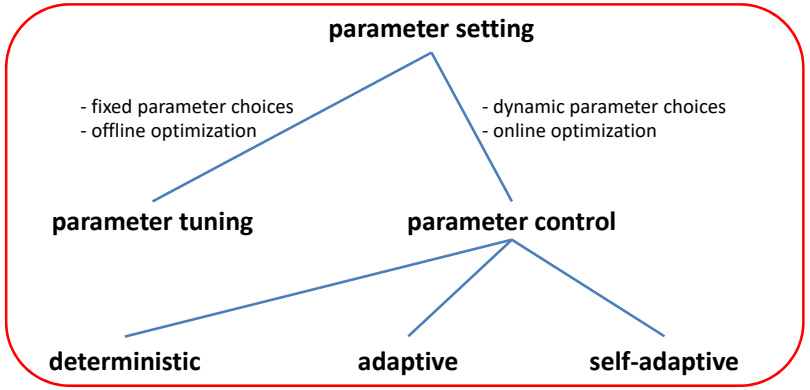
Part 3: Taxonomy from [EHM 99], [DD20]

Main Questions in Parameter Control

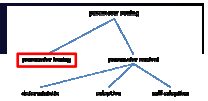
1. *Which parameter* is adapted?
   (and who is affected: 1 individual vs. whole population)
   1. Population size
   2. Mutation rate, Crossover probability
   3. Selection pressure
   4. Fitness function (e.g., penalty terms for constraints)
   5. Representation
   6. …
2. What is the *basis/evidence for the update*?
   1. time elapsed: number of fitness evaluations, generation count, CPU time
   2. progress, e.g., in terms of absolute or relative fitness gain
   3. diversity measures
   4. …
3. *How* do we select the parameter(s):
   1. multiplicative updates
   2. learning-inspired parameter selection
   3. endogenous/self-adaptive parameter selection: use EAs to find good values
   4. hyper-heuristics
   5. …

Classification Scheme of [EHM99]

- Many attempts to find unifying taxonomy for parameter choices exist (see page 168 in [KHE15] for a survey)
- To date, the most popular classification scheme is that of Eiben, Hinterding, Michalewicz [EHM99]:

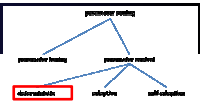


Parameter Tuning



- Typical tuning approach:
  - run some initial tests and observe how the performance depends on the chosen parameter values
  - choose the parameter values that seem most promising
  - requires a (large) budget for the training
- **Quite sophisticated tools** for parameter tuning are available:
  - irace [LDC+16], SPOT [BBFKK10], GGA [AMS+15], ParamILS [HHLBS09], SMAC [HHLB11], HyperBand [LJD+17]], BOHB [FKH18], …
  - Advantage of these tools: automated identification of reasonable parameter values → supports human and reduces bias
  - Disadvantage: recommended parameter values are static!
  - Note: even when focusing on dynamic parameter choices, parameter tuning can be very essential to select good hyper-parameters, see [BDSS17] for an example

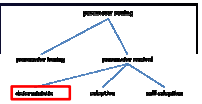
"Deterministic" Parameter Control



Key intuition:

- Belief that optimal parameters often follow a similar pattern
- Example pattern: "first allow for exploration, then for exploitation"
- to stimulate or enforce such a pattern, time-dependent parameter settings can be used (where time = number of generations, fitness evaluations, wall-clock time, etc.)
  - **Examples:**
    1. cooling schedule of the selective pressure ("temperature") in selective pressure of Simulated Annealing. Often used update scheme: $T(t) = \alpha^t T(0)$ (multiplicative updates)
    2. start with some (large) mutation rate $p(0)$, decrease p after every 10,000 fitness evaluations
    3. after each 1,000 iterations, draw a random mutation probability

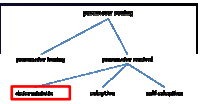
“Deterministic” Parameter Control



- The last example on the previous slide shows that---as already acknowledged in [EHM99]---the term “deterministic” is not very well chosen
  → the choice can be random!
  → the only important feature is that it depends only on the time elapsed so far, and not on any other feedback of the optimization process
- More suitable terms could be
  - “*time-dependent*”, “*scheduled*” update scheme, or
  - “*feedback-free*”, “*progress-independent*” update scheme

  but in lack of a widely acknowledged alternative, “deterministic update rule” is still the predominantly used term
- Also note that finding the optimal deterministic update rules requires tuning, i.e., while they bypass the disadvantage of the non-flexible static parameter values, they do not allow the algorithm to identify the good parameter values by itself

“Deterministic” Parameter Control



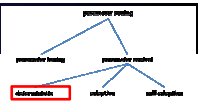
- Some selected examples, theory works:
  - Hesser and Männer (PPSN’90) [HM90] suggested the following rule for the mutation strength of a GA with population size λ for OneMax:
    $p_m(t) := \frac{\sqrt{\frac{\alpha}{\beta}}\exp\left(-\frac{\gamma t}{2}\right)}{\lambda\sqrt{n}}$ where α, β, γ are constants
  - Jansen Wegener [JW06]: mutation rate changes in every iteration
    - $p_t(n) := 2^i/n$ where $i \equiv (t-1) \bmod \log(n) - 1$

    +/- very frequent changes → non-stable algorithm

    - worse performance on simple functions like OneMax, linear functions, LeadingOnes, etc.

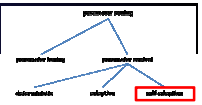
    + examples where better performance than any static choice can be proven
  - Doerr, Doerr, Kötzing [DDK18]: in every iteration, a random step size is used for a multi-valued OneMax-type problem (this problem will be discussed in more detail in the next section, along with a self-adjusting parameter choice. the algorithm that we refer to here is the one using a static probability distribution from which the step sizes are sampled)

“Deterministic” Parameter Control



- Random Variation of the Population Size GA (RVPS) by Costa, Tavares, and Rosa [CTR99]
  - size of the actual population is changed every *N* fitness evaluations, for a given *N* (according to some monotonous rule)
  - Both shrinking and increasing the population size are considered
- Saw-tooth like population size growth considered by
  - Koumousis and Katsaras in [KK06] (TEC 2006): linear decrease of population size with eventual re-initialization of the population size by adding randomly selected individuals
  - Hu, Harding, Banzaf [HHB10]: inverse saw-tooth like population sizes

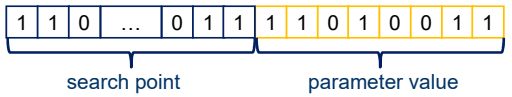
Self-Adaptive Parameter Control



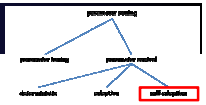
- Parameter Control Idea 2:

	Finding good parameter values is difficult
+	EAs are good problem solvers
=	Use an EA to determine parameter values

- Many different ways to do this. Examples (sketched, much room for creativity here!):
  1. Create a new population of parameter values, choose from this parameter values, possibly apply variation to them, and employ them in your EA, select based on progress made
  2. append to the solution candidates a string which encodes the parameter value, first mutate the parameter value part, then use this parameter to change the search point, selection as usual

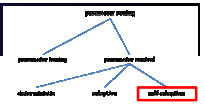


Self-Adaptive Parameter Choices



- We won't discuss this in much detail, but if you are interested in such mechanisms, you can start your investigations with the following works
  - Bäck (PPSN'92) [Bäc92] and follow-up works: extends the chromosome by 20 bits. Mutation works as follows:
    1. Decoding the 20 bits to the individual's own mut. rate p_m
    2. Mutating the bits encoding p_m with mutation probability p_m
    3. Decoding these changed bits to p'_m
    4. Mutating the bits that encode the solution with mutation probability p'_m
  - Dang, Lehre (PPSN'16) [DL16] and B. Doerr, Witt, Yang [DWY18] : theoretical works on a self-adaptive choice of the mutation strength in a non-elitist population

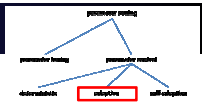
Comment on Self-Adaptiveness



- Essential feature is not that parameters are encoded in genome
- Essential: each individual carries its own parameter
- Key working principle:
  survival of the fittest individuals → survival of best parameters
- Observe: improve parameter selection and fitness at the same time:

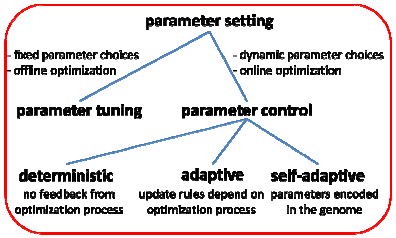
  1 original optimization problem
  + 1 parameter selection problem
  = 1 integrated problem

Adaptive Parameter Control



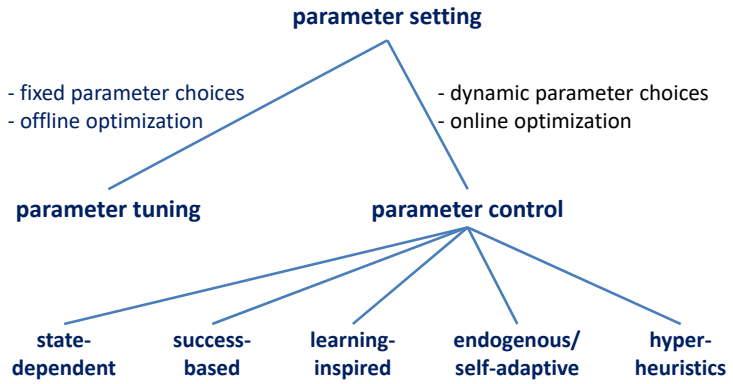
- Parameter Control Idea 3:
  - "global" estimate for parameter quality, i.e., not individual-based
  - use feedback from the optimization process
  - change the parameters according to some pre-described rule
- Relevant feedback includes:
  - *function values* of the search points in the population
  - *diversity* of the search points
  - *absolute or relative progress* obtained within the last τ iterations
  - …
- Examples:
  - 1/5-success rule (introduced above)
  - CMA-ES update of covariance, step size, population size,…
  - many more examples will follow…

Comment on Classification Scheme of [EHM99]



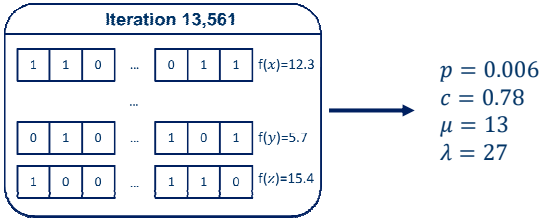
- The terms "deterministic", "adaptive", and "self-adaptive" have not been formally defined
  → be aware that they are not used very consistently in the literature
- Since [EHM99] almost 20 years have passed.
  → The field has advanced considerably
  (but maybe not to the extend it should have, as also noted in [KHM15])
  → we feel that time has come to introduce a different taxonomy

Revised Classification from [DD20]



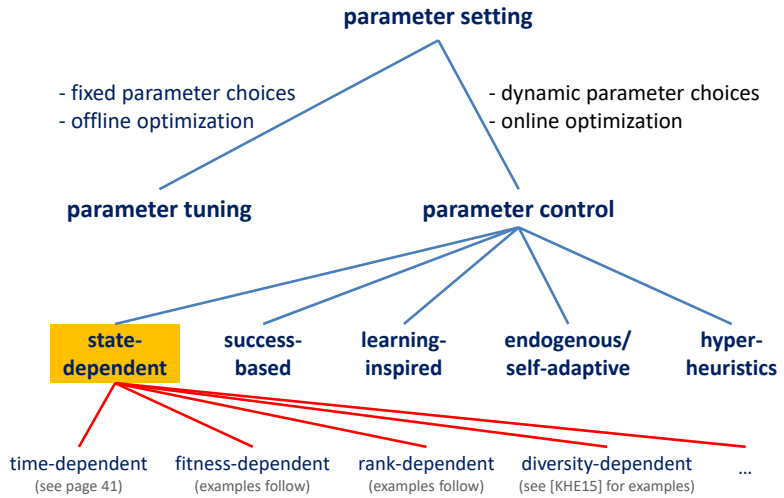
State-Dependent Parameter Selection

- State-dependent parameter selection mechanisms do not depend on the history of the optimization process, but only on the current state
- Analogy for this functional dependence: take a "screenshot" of the current population and map it to parameter values



- Most commonly used indicators for the state of the algorithm:
  - time elapsed so far (# fitness evaluations, iteration counter, CPU time, ...)
    → corresponds to "deterministic" parameter setting in the classification [EHM99]
  - function values (absolute values, diversity, ranks,…)
  - genotypic properties (e.g., diversity of the population)

Revised Classification from [DD20]



Fitness-Dependent Parameter Selection

- Requires a good understanding of how the parameters should depend on the function values
- Has been looked at
  - empirically, e.g., Bäck [Bäc92,Bäc96], Fialho, Da Costa, Schoenauer, Sebag PPSN'08 [FCSS08] and follow-up works for OneMax
  - theoretically, e.g.,
    [BD19, DDY16b,BLS14] for OneMax and
    [Doe19,BDN10,Sud13,DW18,LOW17] for LeadingOnes
    → surprise: good understanding for LeadingOnes,
    not so good understanding for OneMax!

Rank-Dependent Parameter Selection

- **Basic idea:**
  - bad search points should undergo large variation (→ large mutation rates)
  - good individuals should be modified only moderately (→ small mutation rates)
- **Example:**
  - Cervantes, Stephens IEEE TEC [CS09]:
    - rank search points in the current population
    - each search point is assigned a mutation rate that depends on its rank:
      - rank 1: mutation rate $p_{\min}$ // best individual of population
      - ... (linear interpolation)
      - rank s: mutation rate $p_{\max}$ // worst individual of population
    - the rank-based GA first selects an individual from the population and then modifies it with the mutation rate given by this ranking
  - Theoretical study of this algorithm are available in [OLN09]

Success-Based Parameter Selection

- **Basic idea:** after each (or after every τ) iteration(s) adjust the current parameter value depending on whether or not the last (τ) iteration(s) have been successful
- Examples for "success": finding
  - a strictly better search point // this is probably the most common measure
  - a search point that is integrated into population // used by the adaptive (1+1) $\text{EA}_{>0}$ from [DW18]
  - a fitness-increase of at least x%
  - point(s) that increase the diversity of the population
  - …
- Success-based parameter selection is classified as "adaptive parameter control" in the taxonomy of [EHM99]
- Example: 1/5 success rule

Example ≠ 1/5 success rule: 2-rate adaptation

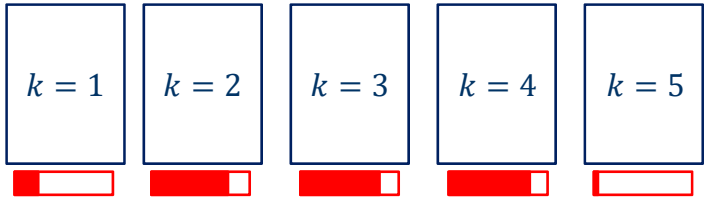
In [DGWY19] the following mechanism is suggested for controlling the mutation rate in a $(1+\lambda)$ EA:

- let p be the current mutation rate
  - in each iteration do:
    - create $\lambda/2$ offspring with mutation rate $2p$
    - create $\lambda/2$ offspring with mutation rate $p/2$
  - update p as follows (capping at $2/n$ and 1/4, respectively)
    - with probability 1/2 set it to the value for which the best offspring has been found
    - with probability 1/2, independently of the last iteration, randomly decide whether to replace p by either $p/2$ or by $2p$
  - Main result: this simple mechanism achieves the asymptotically optimal[1] $T^{\text{gen}} = \Theta\left(\frac{n}{\log \lambda} + \frac{n \log n}{\log \lambda}\right)$ performance on OneMax.

[1]Optimality follows from [BLS14]

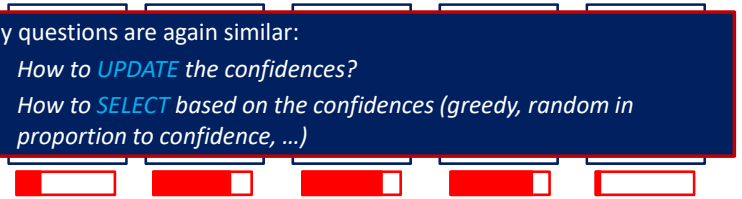
Learning-Inspired Parameter Control - Key Ideas

- The main idea for learning-/reward-type adjustment rules is
  - have a set/"portfolio" S of possible parameter values
  - according to some rule, test one (or some) of these values
  - based on the feedback from the optimization process, update the likelihood to employ the tested value
- Picture to have in mind: multi-armed bandits (MAB)
  - K experts
  - in each round, you have to chose one of them and you follow his advice
  - you update your confidence in this expert depending on the quality of his forecast



Learning-Inspired Parameter Control - Key Ideas

- The main idea for learning-/reward-type adjustment rules is
  - have a set/"portfolio" S of possible parameter values
  - according to some rule, test one (or some) of these values
  - based on the feedback from the optimization process, update the likelihood to employ the tested value
- Picture to have in mind: multi-armed bandits (MAB)
  - K experts
  - in each round, you have to chose one of them and you follow his advice
  - you update your confidence in this expert depending on the quality of his forecast
- Key questions are again similar:
  - *How to UPDATE the confidences?*
  - *How to SELECT based on the confidences (greedy, random in proportion to confidence, …)*



(Another) Exploration/Exploitation Trade-Off

- **Main difficulty: exploitation vs. exploration trade off**

exploitation: we want, of course, to use an optimal parameter value as often as possible

exploration: we want to test each parameter value sufficiently often, to make sure that we select the "optimal" one (in particular when the quality of its "advice" changes, which is the typical situation that we face in evolutionary optimization)

Learning-Type Updates, Remarks

- Frequently found feature: *time-discounted methods*. That is, *a good advice in the past is worth less than a good advice now*
  - different update mechanisms and "forgetting rates" have been experimented with, see discussion below
  - note that such mechanisms are in particular useful when the quality of advice (in our setting, this could be the expected fitness gain, the expected decrease in distance to the optimum, or some other quantity) changes over time
- Note: such learning mechanisms are referred to as "*operator selection*" in [KHE15]. Another keywords to search for is "*credit assignment*". It may also be worth to look into literature from learning, in particular on *multi-armed bandit* algorithms (main goal: maximize reward "on the go", i.e., while learning) and on *reinforcement learning* (possibly have dedicated "learning" iterations, a notion of state is introduced and the hope is to learn for each state which operator maximizes expected progress). Some *hyper-heuristics* are also learning-based.
- Again we have to focus on a few selected works here. Much more work has been done, see Section IV.C.4 in [KHE15] for a survey. There is still much room for further creativity and much research is needed to understand which mechanisms are most useful in which situations!

Dynamic Multi-Armed Bandits View

- K different parameter values
  - p_t^i probability to chose operator i in iteration t $(p_t^1, p_t^2, \dots, p_t^K)$
  - c_t^i confidence in operator i at iteration t $(c_t^1, c_t^2, \dots, c_t^K)$
- Main questions: how to update probabilities? how to updates confidence?
  - well-studied questions in *machine learning*!
  - But: main focus in ML is for static "*rewards*"
  - main difference to EC: our "rewards" (success rate, fitness increase, etc) change over time.
- 2 first ideas:
  - **1. Probability Matching:**
    - $c_{t+1}^i = (1-\alpha)\, c_t^i + \alpha r^t$, where i is the operator selected in iteration t and r^t is the reward of that iteration
    - $p_{t+1}^i = p_{\min} + (1 - K p_{\min}) \frac{c_{t+1}^i}{\sum_{j=1,\dots,K} c_{t+1}^j}$,
    - p_t^i is proportional to c_t^i while maintaining a minimal amount of exploration

α controls the speed of confidence adaptation

minimal level of exploration

Dynamic Multi-Armed Bandits View

- K different parameter values
  - p_t^i probability to chose operator i in iteration t $(p_t^1, p_t^2, \dots, p_t^K)$
  - c_t^i confidence in operator i at iteration t $(c_t^1, c_t^2, \dots, c_t^K)$
- Main questions: how to update probabilities? how to updates confidence?
  - well-studied questions in *machine learning*!
  - But: main focus in ML is for static "*rewards*"
  - main difference to EC: our "rewards" (success rate, fitness increase, etc) change over time.
- 2 first ideas:

2. Adaptive Pursuit [Thi05]**:**

  - $c_{t+1}^i = (1-\alpha)\, c_t^i + \alpha r^t$, where i is the operator selected in iteration t and r^t is the reward of that iteration
  - $p_{t+1}^i = (1-\beta) p_t^i + \beta p_{\max}$, for current best "arm" $i = i^*$
  - $p_{t+1}^i = (1-\beta) p_t^i + \beta p_{\min}$, for other arms $i \neq i^*$
  - "winner takes it all"

controls speed of selection adaptation

UCB-Based Dynamic Multi-Armed Bandits

- Da Costa, Fialho, Schoenauer, Sebag (GECCO'08) [CFSS08] and follow-up works suggest a parameter control mechanism that hybridizes
  - a multi-armed bandit algorithm (Upper Confidence Bound *UCB*-type, see next slide) with
  - the statistical Page-Hinkley test (which triggers a restart of the UCB mechanism if positive, indicating a change in the time series)

UCB = Upper Confidence Bound

- **Upper Confidence Bound**, aka *UCB*-mechanisms are well known in learning theory, see work by Auer, Cesa-Bianchi, Fischer ML'02 [ACBF02]
- Main ideas:
  - cUCB greedily selects the operator (the "arm") maximizing the following expression:

$$\text{expected reward} + \sqrt{c \log \frac{\sum_k n_{k,t}}{n_{j,t}}},$$

    where
    - $n_{k,t}$ is the number of times the k-th arm has been pulled in the first t iterations and
    - c is a parameter that allows to control the exploration likelihood (vs. exploitation, which is controlled by the first summand)
  - tuned and other variants of this algorithm exist, see [ACBF02] for details and empirical evaluations
- These ideas can be used in operator selection, but note that in contrast to the classical setting in multi-armed bandit theory the rewards change over time (dynamic multi-armed bandit scenario)

Extreme Value-Based Adaptive Operator Selection

- In [FCSS08], Fialho, Da Costa, Schoenauer, and Sebag argue that, for many problems,
  - rare large fitness improvements are often better than
  - many small fitness improvements
- They suggest to distribute confidence based on the largest fitness improvement that an operator has produced in the last W iterations in which it has been used (*sliding window* of size W)
  - Sizing W is again non-obvious, too small W makes it difficult for an operator with rare but large fitness improvements to be chosen, while too large W makes it more difficult to adjust the search to the current state of the optimization process
- In [FCSS10] the authors suggest the following changes:
  - increase the reward with the time elapsed since the last application of the operator
  - decrease it with the number of times the operator has been used in the last iterations

Randomized Local Search w/ ε-greedy parameter control (1/2)

Theoretical result for learning-inspired parameter control available in [DDY16a]:

- studies *dynamic ε-greedy parameter selection:*
  - Fix a small number of possible mutation strengths $[r] := \{1, 2, \dots, r\}$
  - Estimate the expected fitness gain $v_{t-1}[k]$ from using k-bit flips (using data from the past, see next slide)
  - In iteration t
    - with probability ε, use a random $k \in [r]$ "exploring mut. strengths"
    - with prob. $1-\varepsilon$, use a k that maximized $v_{t-1}[k]$ "take the most efficient k"
  - Update the expected fitness gain estimations
- It is shown in [DDY16a] that this adaptive mechanism yields almost optimal running time on OneMax (the difference is tiny).
  It performs better than any static parameter choice.
- This algorithm with the same budget computes a solution that asymptotically is 13% closer to the optimum than Randomized Local Search
  (provided that the budget is at least $0.2675n$).

Randomized Local Search w/ ε-greedy parameter control (2/2)

- Expected fitness gain estimation for using a k-bit flip:
$$v_t[k] := \frac{\sum_{s=1}^{t} 1_{r_s=k} \, (1-\delta)^{t-s} \big(f(x_s) - f(x_{s-1})\big)}{\sum_{s=1}^{t} 1_{r_s=k} \, (1-\delta)^{t-s}}$$
- $1/\delta$: "*forgetting rate*", determines the decrease of the importance of older information. $1/\delta$ is (roughly) the information half-life
- The "velocity" can be computed iteratively in constant time by introducing a new parameter $w_t[r] := \sum_{s=1}^{t} \mathbf{1}_{r_s=r} (1-\delta)^{t-s}$
- This mechanism seems to work well also for other problems
  - So far, no other theoretical results available
  - A few experimental results for LeadingOnes and the Minimum Spanning Tree problem exist, see next 2 slides (these results were also presented in [DDY16a])
  - Again, much more work is needed to see how the algorithm performs on other problems and how to set the parameters δ and ε

Comments on Hyper-Heuristics

- Not covered in detail here, because there is a separate tutorial at GECCO dedicated to hyper-heuristics [TW19]
- Several ideas presented above can also be found in the hyper-heuristics literature. In fact, many hyper-heuristics could be easily integrated in the above.
- Why do they have their own category in the classification?
  historical reasons, 2 "sub-communities" with similar ideas
- Note that hyper-heuristics covers much more than controlling parameters
  → the main idea is to control the whole algorithm, in the sense of dynamically choosing which heuristic is best at a given state
- Surveys: [BGH+13,TW19]
- Recent theoretical works: [LOW19,DLOW18,LOW17]

Controlling Multiple Parameters

- Most EAs have several parameters
- Intuitively, there is no reason to not control more than one or even all of them
- A few works on controlling more than 1 parameter exists, see [KHE15]
- The problem how to best control several parameters is, however, widely open (given the non-conclusive state-of-the-art in controlling one parameter, this is perhaps not very surprising)

Controlling Mutation Rate and Variance

Example for controlling more than 1 parameter, taken from [YDB19]:

- Instead of choosing mutation strength from $\mathrm{Bin}(n,p)$ [as done by standard bit mutation], sample from $N(\mu,\sigma^2)$
  - $\sigma^2 = 0$: deterministic k,
    → Randomized Local Search with variable neighborhood size
  - $\sigma^2 = np(1-p)$: very similar to standard bit mutation [central limit theorem]
    → Evolutionary Algorithm
- Introducing variance yields a meta-algorithm
  Online Parameter Selection → Online Algorithm Selection
- quite efficient on OneMax and LeadingOnes key tool:
  - use $f(y) \geq f(x)$ to control mean mutation rate μ
  - use # of iterations without parameter change to control variance

Part 4: Applications

Real-world optimization

Real-world optimization

- Real-world optimization problems occur in many applications
  - Engineering design,
  - Scientific modelling,
  - Image processing,
  - Production,
  - Transportation,
  - Bioinformatics,
  - Finances, etc.
- Real-world systems are, in general, large and very complex. They need to process a large amount of data, to perform complex optimization and make decisions fast [KBP13].

Real-world optimization

- Real-world problems in general
  - Contain non-linear objective functions of mixed design variables (i.e. continuous and discrete)
  - Contain linear as well as non-linear constraints
  - Might have several local optima
- For a wide range of real-world optimization problems, a near-optimal or a better-than-known solution is considered a satisfactory result of an optimization problem.

Real-world optimization

- There are several characteristics that increase the complexity of the optimum solution search and for which parameter control could be advantageous
  - **Number and type of variables**: a large number of decision variables, including mixed-integer problems, where different types of variables are optimised
  - **Dynamic problems**: problems that are changing over time
  - **Problems under uncertainty**: the variables of the problem have some uncertainty
  - **Number of objectives**: problems that require optimizing more than one objective function simultaneously and need to be solved by a multi/many-objective approach
  - **Nested problems**: multi/bi-level optimization, where one optimization problem has another optimization problem as a constraint
- Some problems have combination of these characteristics

Real-world optimization

- Properly defined control parameters play a crucial role in effectively handling the mentioned characteristics and solving such problems.
  - For example: with increasing dimensionality of the problem its landscape complexity grows and the search space increases exponentially.

But

An optimization algorithm must still be able to explore the entire search space efficiently

Large scale global optimization

Large scale global optimization

- LSGO [OLML19] - the problem dimension D (the number of variables to be optimised) has an order of magnitude of up to several thousand (for real values) or even billions (for integer or binary values)
- An active research field due to the growing number of large-scale optimization problems in engineering, manufacturing and economy applications (such as bio-computing, data or web mining, scheduling, vehicle routing, etc.) [Cab16], [LTSY15]
  - Advances in machine learning and the wide use of deep artificial neural networks result in optimization problems with over a billion variables [HS06]

Large scale global optimization

A major challenge of large-scale optimization

Most engineering problems have an exponential increase in the number of required decision variables [OYM+17], [Van02]

- The challenges motivated the design of many kinds of efficient, effective, and robust kinds of metaheuristic algorithms to solve LSGO problems with high-quality solutions and high convergence performance as well as with low computational cost [MA17]

Large scale global optimization

- To achieve acceptable results even for the same problem, different parameter settings along with different reproduction schemes at different stages of optimization process are needed
- Several techniques (e.g., [ZBBv08], [DMS16]) have been designed to adjust control parameters in an adaptive or self-adaptive manner (instead of a trial-and-error procedure)

Large scale global optimization

- Some examples of LSGO
  - Data analytic and learning problems [ZCJW14]
  - Shape design optimization for aircraft wings and turbine blades [YSTY16]
  - Satellite layout design [TCZ+10]
  - Parameter calibration of water distribution system [WHD+13]
  - Seismic waveform inversion [WG12]

Dynamic optimization

Dynamic optimization

- Real-world optimization problems are usually subject to changing conditions over time
- The effects of these changes could influence several aspects of the problem, such as the objective function, the problem instance, its constraints, etc.
- The optimal solution of the problem might change over time.

Dynamic optimization

Changing problems, when solved by an adaptive optimization algorithm on-the-fly, are called dynamic optimization problems (DOPs) [NYB12]

Dynamic optimization

- The algorithm is expected to be able to track the current optimal solution as well as the changing optimal solution over time
- The optimization procedure has to be able to detect these changes and react quick enough
  - This also requires dynamic change of the ratio for exploration and exploitation parts of the search
  - Both adaptive [WWY09] and self-adaptive [BZB+09] parameter control can be used

Dynamic optimization

- Based on a comprehensive survey [Bra01], four different strategies can be used to help evolutionary/population-based algorithms to adapt in dynamical environments:
  - **Increasing diversity** of the population after a change is detected, (e.g., by increasing mutation rate every N generations)
  - **Maintaining diversity** throughout the run, to avoid convergence of the population on one point
  - **Memory based approaches**, taking into consideration older solutions and sometimes making predictions based on historical data
  - **Multi-population approaches**, where many small populations track their own peaks as the environment changes

Dynamic optimization

- Some examples of dynamic optimization
  - Production scheduling [Yil13]
  - Energy demand optimization [GZ15]
  - Transportation [YLLF18], [LEC12]
  - Financial optimization [HJS11]

Optimization under uncertainty

Optimization under uncertainty

- The presence of (a range of) uncertainties has to be taken into account for solving many real-world applications with evolutionary algorithms
- [JB05] categorize the uncertainties that influence EA performance into four types
  - When there is some **noise in the fitness function**
  - When there are changes of **design and environmental parameters after the optimization**
  - When **fitness function is an approximation**
  - When the **optimum changes over time** (as in dynamic optimization).

Optimization under uncertainty

- Methodologies for addressing noisy fitness function
  - *Explicit averaging* by calculating the average of the fitness values over a number of randomly sampled disturbances [Gre96], [Mil97]
  - *Implicit averaging* sample size as an inverse function of the population size [FG88]
  - *Fitness inheritance* where the offspring inherits also the mean and standard deviation of the objective value [BAE05]
  - *Selection modification* [Tei01]
- These methods assume that the search space follows a homogeneous noise distribution, such as a uniform or a normal distribution [VC16]

Optimization under uncertainty

- Some examples of optimization under uncertainty
  - Financial optimization [HJS11]
  - Transportation in unknown environments [ZHA16]
  - Space applications [VML11], [BS06]

Multi-objective optimization

Multi-objective optimization

- Multi-objective (and also many-objective) optimization approaches are used for optimization problems where several criteria need to be optimised, but they are equally treated and not merged (e.g., by weights) into one single objective
- The output of multi-objective optimization is a set of solutions that approximates the Pareto front
- There is no unique measure that would indicate how good a current approximation of the Pareto front is

In multi-objective cases

Adaptive parameter control is a bit more complicated to design and additional considerations are needed to design phenotype feedback collection part

Multi-objective optimization

- Possible assessment of the optimization process stage
  - Monitor the proportion of non-dominated solutions in the population [YJG09]
  - Convergence detection [NT09]
- The most common indicators that are also used as input to parameter control are the crowding distance and the contributing hyper-volume [IHR07], [BNE07]

Multi-objective optimization

- Other metrics can also be applied
  - ε-dominance
  - Generational distance
  - Delta indicator
  - Two set coverage, and so on [RVLB15].
- Compared to adaptive control, self-adaptive control is easier to design and implement because less modifications are needed to upgrade an existing multi-objective optimization algorithm [WWY10] [CLW07].

Multi-objective optimization

- Some examples of multi-objective optimization
  - Engineering design [GBX+18]
  - Transportation [SBD19], [LMR+18]
  - Production [GNB+19], [AYGL18]

Multilevel optimization

Multilevel optimization

- In many real-world processes there is a hierarchy of decision-makers and decisions are taken at different levels [MPV12]
  - The constraint domain associated with a multilevel problem is implicitly determined by a series of optimization problems which must be solved in a predetermined sequence
- The simplest form of a multilevel problem has two levels (i.e., bi-level optimization problem)
  - The optimization of such problem aims to achieve the optimum solution of the upper level, while the optimum of the lower optimization level is also taken into account
  - Since the lower level landscape changes for every upper level vector, parameter control seems to be useful approach

Multilevel optimization

- An interesting application of bilevel optimization is connected to parameter tuning of EAs as bilevel optimization problem
  - [SMXD14] propose the parameter tuning problem as an inherently bilevel programming problem involving algorithmic performance as the objective(s)
  - [And18] created a bilevel framework for parallel tuning of optimization control parameters, and compared it to irace proving that it can be competitive
  - Bilevel control parameter tuning can be used to design a parameter control mechanism [ABNS15]

Examples

Image processing: Feature selection

LS

- Feature selection for reducing the dimensionality in classification of hyperspectral images [DGG11]

Self-adaptive differential evolution SADE is used

- SADE is used in combination with Fuzzy kNN classifier
- Compared to GA-based and ACO-based approaches [SP10]

- Significant improvement for overall classification accuracy and Kappa coefficient

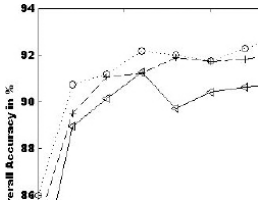


Image processing: Animated Tree Reconstruction

LS UN

- Feature extraction to reconstruct three dimensional procedural models of trees; to lower problem dimensionality needed for encoding local parameters [ZB14]
- The reconstruction is iteratively optimized using DE, which samples procedural tree model parameters to obtain a parameterized procedural model for instantiating a geometrical model

jDE is used

- DE with self-adaptive control parameter settings [BGB+06]

- Examples of reconstructed model animation are shown, such as simulation of its growth, sway in the wind, or adding leaves

Production: Scheduling

DY

- Optimal operations scheduling in the production of different components considering various constraints [PVK12]

PLES algorithm is used

- PLES is based on general GA with modified implementation of functions that allow varying population size, mutation and crossover [Pap13]
- Compared to standard non-adapting GA and GA with customized local search [KPV10]

- Faster convergence of PLES and comparable results to GA for various problem instances

Transport: Scheduling

DY

- Solving a constrained transportation scheduling problem, for transporting goods in emergency situations [KP13]

PLES algorithm is used

- PLES is based on general GA with modified implementation of functions that allow varying population size, mutation and crossover [Pap13]
- Compared to non-adaptive Ant-stigmergy algorithm [K12]

- The satisfying performance in finding solutions and escaping from local optima, for different transportation modes

Power systems: Scheduling

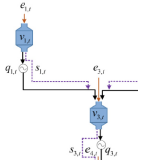
- Addressing total fuel costs and emissions minimization by appropriate hydro and thermal generation schedules [GZ15]

LS DY

NPdynϵjDE and PSADEs algorithms are used

- NPdynϵjDE is based on jDE self-adaptation [BZM06], population size reduction [BM08], and ϵ level adjustment [ZBBZ09]
- Surrogate parallel self-adaptive DE (PSADEs) is based on self-adaptation [GGK+14], with pre-computed surrogate model

- The satisfied 24-h system demand is obtained by using a new DE architecture



Water pipeline system: Parameter calibration

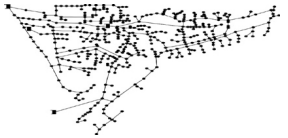
- Parameter calibration strengthens the model accuracy of the water distribution system. Many factors influence the reliability of WDS simulation [WHD+13]

LS

ensemble optimization evolutionary algorithm (EOEA) is used

- Combining global shrinking stage (to shrink the searching scope to the promising area) [BM08] and local exploration [YTY08] stage with self-adaptive group sizing
- Different problems were constructed/tested: 100D, 200D, 300D and 454D

- Results show good scalability of EOEA on this real-world application



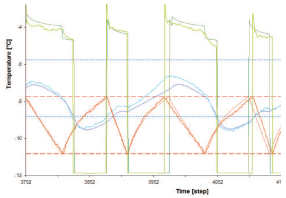
Appliance controller: Setting configuration

- Optimal performance of a refrigerator/freezer to cool to the desired temperature using the lowest possible power consumption [PM10] UN

PLES algorithm is used

- PLES is based on general GA with modified implementation of functions that allow varying population size, mutation and crossover [Pap13]

- The results show correlation between simulated and measured duty cycle of the compressor and with energy consumption



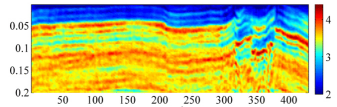
Seismic waveform inversion: Configuration

- Waveform inversion for whole Earth geophysics and exploration geophysics, to develop an accurate Earth model and for understanding of subsurface structures [WG12] LS UN

cooperative coevolutionary DE (CCDE) algorithm is used

- All subcomponents are cooperatively evolved to solve high-dimensional optimization problems through decomposition
- The next generations are selected according to the global fitness values
- The parameter adaptation scheme of jDE [BGB$^+$06] is used

- The CCDE results are very effective and have significant advantages over some other methods. CCDE is not sensitive to the size of the parameters



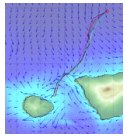
Underwater glider: Path planning

- Optimization of a short-term sea trajectory, with opportunistic sampling of dynamic mesoscale ocean structures (eddies), which offer short-term opportunities for underwater glider path optimization [ZH14] [ZHA16] DY UN

jDE is used

- Slightly modified jDE [BGB$^+$06], combining DE and underwater glider path planning (UGPP)

- Gliders operational capabilities benefit from improved path planning, especially when dealing with opportunistic short-term missions focused on dynamic structures



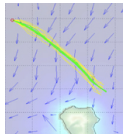
Underwater glider: Path planning 2

- Optimization of a sea trajectory for underwater glider path optimization [ZS19] DY UN

Success-History Based Adaptive Differential Evolution Algorithm (SHADE) including Linear population size reduction (L-SHADE) is used

- $L - SHADE_5$ was used including different population sizes and population sizing strategies

- Increased opportunity for mission scenario re-tests or in very hard scenarios



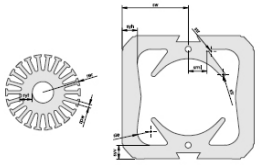
Electrical motor: Geometry selection

- Optimization of geometrical parameters of the electrical motor rotor and stator geometry [Pap08] UN

PLES algorithm is used

- PLES is based on general GA with modified implementation of functions that allow varying population size, mutation and crossover [Pap13]
- Compared to generational evolutionary algorithm (GEA) [TKP+07] and multilevel ant stigmergy algorithm (MASA) [Kv05]

- The results show fast convergence of the PLES but is not always able to find global optimum



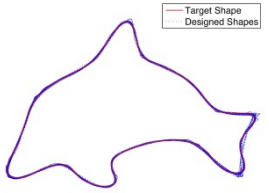
Target shape design optimization

- To tackle the target shape design optimization problems (TSDOPs) with B-spline as the geometry representation [YSTY16] LS MO

CMA-ES-CC algorithm is used

- CMA-ES with Cooperative Coevolution was implemented
- Compared with CMA-ES, iES [PXL+07], RCGA [DAJ02]

- The performance of CMA-ES-CC was stable, and the results of CMA-ES-CC were significantly better than with other EAs for TSDOPs



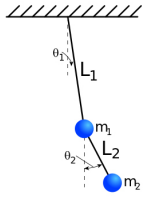
Pendulum: Reinforcement learning

- Considers modeling of inverted pendulum and double-pendulum swing-up [DVC19] DY

CMA-ES algorithm is used

- CMA-ES uses reproducing kernel Hilbert space (RKHS)
- Compared to standard CMA-ES and adaptive CMA-ES direct policy search CMA-ES-A

- The results show that CMA-ES-RKHS is able to avoid local optima and clearly outperformes other methods



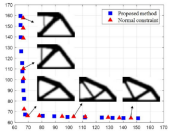
Topology optimization

- Topology optimization was performed with two different compliances (structure in different load cases are used as two different objective functions to be minimized) were considered as conflicting objective functions [RM18] MO

Adaptive weight multi-objective algorithm is used

- Conflicting objective functions are converted to a single objective function by applying weights, and these weights are adaptively updated to find evenly distributed solutions on the Pareto front.

- The results confirm that optimized solutions, obtained by using the proposed method, are evenly distributed on the Pareto front



Urban mass rapid transit: Transport

- Multi-objective simulation-based headway optimization for complex urban mass rapid transit systems in Vienna [SBD19]

DY
MO

CMA-ES algorithm is used

- multi-objective version of CMA-ES (MO-CMA-ES) is compared to single objective version (SO-CMA-ES) and NSGA-II

- The results show similar performance of all tested algorithms

Aerobic fermentation process: Production

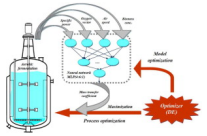
- The optimization of the oxygen mass transfer coefficient in stirred bioreactors in the presence of n-dodecane as oxygen vector, where the oxygen transfer in the fermentation broths has a significant influence on the growth of cultivated microorganism [DCGC13]

DY
MO

SADE-NN-1 algorithm is used

- An improved, simple, and flexible self-adaptive variant of DE, in combination (hybridized) with neural networks

- The improvements (hybridization) of the algorithm resulted in higher efficiency of the whole methodology



Summary

Summary

The distribution of presented cases according to problem characteristics

large-scale	dynamic	uncertain	multi-objective	nested
6	8	6	4	–

Summary

- You are welcome to contribute to this overview of the real-world cases of parameter control implementations
- Send your suggestions/cases to: gregor.papa@ijs.si

Part 5: Wrap Up

Wrap Up

- Our hope was
  To inspire and to enable you to test parameter control mechanisms
- We hope that you are (now) convinced that
  - Dynamic parameter choices can help to significantly improve the performance of your EA
  - Already quite simple mechanisms can be surprisingly efficient
  - Research on parameter control can be fun ☺
  - non-static parameter values should be the new standard in the field ☺
- As mentioned in the tutorial, a lot needs to be done to make this change happen
  - enjoy! ☺
  - don't get frightened by the fact that quite some work has been done already. There is still much room for creativity and we are just starting to understand how good mechanisms look like!
  - … and, last but not least, keep in touch ☺

→ If you get to work on parameter control, we would be very much interested in your results, positive and negative!

Gregor.Papa@ijs.si and Carola.Doerr@lip6.fr

Acknowledgements

Gregor Papa:

- This work was supported by the Slovenian Research Agency (research core funding No. P2-0098).
- This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 692286. (SYNERGY)
- This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 722734. (UTOPIAE)

Carola Doerr:

- I am very grateful to Benjamin Doerr, Thomas Bäck, Markus Wagner, Johannes Lengler, Dirk Sudholt, Pietro S. Oliveto, Carsten Witt, Johann Dreo, and to the participants of the Dagstuhl seminars 16412, 17191, and 19431 for many insightful discussions on parameter control mechanisms
- This work was financially supported by the Paris Ile-de-France Region and by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH
- This work was also supported by COST action 15140 on Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO)

References I

[ABNS15] Martin Andersson, Sunith Bandaru, Amos Ng, and Anna Syberfeldt, *Parameter tuning of MOEAs using a bilevel optimization approach*, Evolutionary Multi-Criterion Optimization (Cham) (António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos Coello Coello, eds.), Springer, Springer International Publishing, 2015, pp. 233–247.

[ACBF02] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer, *Finite-time analysis of the multiarmed bandit problem*, Machine Learning **47** (2002), 235–256.

[AM16] Aldeida Aleti and Irene Moser, *A systematic literature review of adaptive parameter control methods for evolutionary algorithms*, ACM Computing Surveys **49** (2016), 56:1–56:35.

[AMS+15] Carlos Ansótegui, Yuri Malitsky, Horst Samulowitz, Meinolf Sellmann, and Kevin Tierney, *Model-based genetic algorithms for algorithm configuration*, Proc. of International Conference on Artificial Intelligence (IJCAI'15), AAAI Press, 2015, pp. 733–739.

[And18] Martin Andersson, *A Bilevel Approach to Parameter Tuning of Optimization Algorithms Using Evolutionary Computing*, Ph.D. thesis, University of Skövde, 2018, p. 233.

[Aug09] Anne Auger, *Benchmarking the (1+1) evolution strategy with one-fifth success rule on the BBOB-2009 function testbed*, Companion Material for Proc. of Genetic and Evolutionary Computation Conference (GECCO'09), ACM, 2009, pp. 2447–2452.

[AYGL18] Elnaz Asadollahi-Yazdi, Julien Gardan, and Pascal Lafon, *Multi-objective optimization of additive manufacturing process*, IFAC-PapersOnLine **51** (2018), no. 11, 152 – 157, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

[Bäc92] Thomas Bäck, *The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm*, Proc. of Parallel Problem Solving from Nature (PPSN'92), Elsevier, 1992, pp. 87–96.

[Bäc96] ———, *Evolutionary algorithms in theory and practice - evolution strategies, evolutionary programming, genetic algorithms*, Oxford University Press, 1996.

[BAE05] Lam T Bui, Hussein A Abbass, and Daryl Essam, *Fitness inheritance for noisy evolutionary multi-objective optimization*, Proceedings of the 7th annual conference on Genetic and evolutionary computation, ACM, 2005, pp. 779–785.

[BBFKK10] Thomas Bartz-Beielstein, Oliver Flasch, Patrick Koch, and Wolfgang Konen, *SPOT: A toolbox for interactive and automatic tuning in the R environment*, Proc. of the 20. Workshop Computational Intelligence, Universitätsverlag Karlsruhe, 2010, pp. 264–273.

[BD19] Nathan Buskulic and Carola Doerr, *Maximizing drift is not optimal for solving onemax*, Proc. of Genetic and Evolutionary Computation Conference (GECCO'19, Companion), ACM, 2019, Full version available online at http://arxiv.org/abs/1904.07818. See also https://github.com/NathanBuskulic/OneMaxOptimal for more project data.

[BDN10] Süntje Böttcher, Benjamin Doerr, and Frank Neumann, *Optimal fixed and adaptive mutation rates for the LeadingOnes problem*, Proc. of Parallel Problem Solving from Nature (PPSN'10), Lecture Notes in Computer Science, vol. 6238, Springer, 2010, pp. 1–10.

[BDSS17] Nacim Belkhir, Johann Dréo, Pierre Savéant, and Marc Schoenauer, *Per instance algorithm configuration of CMA-ES with limited budget*, Proc. of Genetic and Evolutionary Computation (GECCO'17), ACM, 2017, pp. 681–688.

[BGB+06] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, *Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems*, IEEE Transactions on Evolutionary Computation **10** (2006), no. 6, 646–657.

[BGH+13] Edmund K. Burke, Michel Gendreau, Matthew R. Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu, *Hyper-heuristics: a survey of the state of the art*, Journal of the Operational Research Society **64** (2013), 1695–1724.

References II

[BLS14] Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt, *Unbiased black-box complexity of parallel search*, Proc. of Parallel Problem Solving from Nature (PPSN'14), Lecture Notes in Computer Science, vol. 8672, Springer, 2014, pp. 892–901.

[BM08] Janez Brest and Mirjam Sepesy Maučec, *Population size reduction for the differential evolution algorithm*, Appl. Intell. **29** (2008), no. 3, 228–247.

[BNE07] Nicola Beume, Boris Naujoks, and Michael Emmerich, *SMS-EMOA: Multiobjective selection based on dominated hypervolume*, European Journal of Operational Research **181** (2007), no. 3, 1653 – 1669.

[Bra01] Jürgen Branke, *Evolutionary approaches to dynamic optimization problems - updated survey*, GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, 2001, pp. 27–30.

[BS06] Christopher R. Bessette and David B. Spencer, *Optimal space trajectory design: A heuristic-based approach*, Spaceflight Mechanics 2006 - Proceedings of the AAS/AIAA Space Flight Mechnaics Meeting, Advances in the Astronautical Sciences, 2006, Spaceflight Mechanics 2006 - AAS/AIAA Space Flight Mechnaics Meeting ; Conference date: 22-01-2006 Through 26-01-2006, pp. 1611–1628 (English (US)).

[BZB+09] Janez Brest, Aleš Zamuda, Borko Bošković, Mirjam Sepesy Maučec, and Viljem Žumer, *Dynamic optimization using self-adaptive differential evolution*, IEEE Congress on Evolutionary Computation, 2009. CEC'09, IEEE, 2009, pp. 415–422.

[BZM06] J. Brest, V. Zumer, and M. S. Maucec, *Self-adaptive differential evolution algorithm in constrained real-parameter optimization*, 2006 IEEE International Conference on Evolutionary Computation, 2006, pp. 215–222.

[Cab16] Daniel Molina Cabrera, *Evolutionary algorithms for large-scale global optimisation: a snapshot, trends and challenges*, Progress in Artificial Intelligence **5** (2016), no. 2, 85–89.

[CFSS08] Luís Da Costa, Álvaro Fialho, Marc Schoenauer, and Michèle Sebag, *Adaptive operator selection with dynamic multi-armed bandits*, Proc. of Genetic and Evolutionary Computation Conference (GECCO'08), ACM, 2008, pp. 913–920.

[CLW07] Ruifen Cao, Guoli Li, and Yican Wu, *A self-adaptive evolutionary algorithm for multi-objective optimization*, Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence (Berlin, Heidelberg) (De-Shuang Huang, Laurent Heutte, and Marco Loog, eds.), Springer, 2007, pp. 553–564.

[CS09] Jorge Cervantes and Christopher R. Stephens, *Limitations of existing mutation rate heuristics and how a rank GA overcomes them*, IEEE Transactions on Evolutionary Computation **13** (2009), 369–397.

[CTR99] Carlos J. Costa, R. Tavares, and A. Rosa, *An experimental study on dynamic random variation of population size*, Proc. of Systems, Man, and Cybernetics (SMC'99), IEEE, 1999, pp. 607–612.

[DAJ02] K. Deb, A. Anand, and D. Joshi, *A computationally efficient evolutionary algorithm for real-parameter optimization*, Evolutionary Computation **10** (2002), no. 4, 371–395.

[DCGC13] Elena-Niculina Dragoi, Silvia Curteanu, Anca-Irina Galaction, and Dan Cascaval, *Optimization methodology based on neural networks and self-adaptive differential evolution algorithm applied to an aerobic fermentation process*, Applied Soft Computing **13** (2013), no. 1, 222 – 238.

[DD18] Benjamin Doerr and Carola Doerr, *Optimal static and self-adjusting parameter choices for the* $(1 + (\lambda, \lambda))$ *genetic algorithm*, Algorithmica **80** (2018), 1658–1709.

References III

[DD20] ———, *Theory of parameter control for discrete black-box optimization: Provable performance gains through dynamic parameter choices*, Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, Springer, 2020, pp. 271–321.

[DDE15] Benjamin Doerr, Carola Doerr, and Franziska Ebel, *From black-box complexity to designing new genetic algorithms*, Theoretical Computer Science **567** (2015), 87–104.

[DDK18] Benjamin Doerr, Carola Doerr, and Timo Kötzing, *Static and self-adjusting mutation strengths for multi-valued decision variables*, Algorithmica **80** (2018), 1732–1768.

[DDY16a] Benjamin Doerr, Carola Doerr, and Jing Yang, *k-bit mutation with self-adjusting k outperforms standard bit mutation*, Proc. of Parallel Problem Solving from Nature (PPSN'16), Lecture Notes in Computer Science, vol. 9921, Springer, 2016, pp. 824–834.

[DDY16b] ———, *Optimal parameter choices via precise black-box analysis*, Proc. of Genetic and Evolutionary Computation Conference (GECCO'16), ACM, 2016, pp. 1123–1130.

[Dev72] Luc Devroye, *The compound random search*, Ph.D. dissertation, Purdue Univ., West Lafayette, IN, 1972.

[DGG11] A. Datta, S. Ghosh, and A. Ghosh, *Wrapper based feature selection in hyperspectral image data using self-adaptive differential evolution*, 2011 International Conference on Image Information Processing, 2011, pp. 1–6.

[DJ75] Kenneth Alan De Jong, *An analysis of the behavior of a class of genetic adaptive systems*, Ph.D. thesis, University of Michigan, Ann Arbor, MI, USA, 1975.

[DJS+13] Benjamin Doerr, Thomas Jansen, Dirk Sudholt, Carola Winzen, and Christine Zarges, *Mutation rate matters even when optimizing monotonic functions*, Evolutionary Computation **21** (2013), 1–27.

[DL16] Duc-Cuong Dang and Per Kristian Lehre, *Self-adaptation of mutation rates in non-elitist populations*, Proc. of Parallel Problem Solving from Nature (PPSN'16), LNCS, vol. 9921, Springer, 2016, pp. 803–813.

[DL19] Benjamin Doerr and Carola Doerr Johannes Lengler, *Self-adjusting mutation rates with provably optimal success rules*, Proc. of Genetic and Evolutionary Computation Conference (GECCO'19), ACM, 2019, Full version available online at http://arxiv.org/abs/1902.02588, pp. 1479–1487.

[DLOW18] Benjamin Doerr, Andrei Lissovoi, Pietro S. Oliveto, and John Alasdair Warwicker, *On the runtime analysis of selection hyper-heuristics with adaptive learning periods*, Proc. of Genetic and Evolutionary Computation Conference (GECCO'18), ACM, 2018, pp. 1015–1022.

[DMS16] Swagatam Das, Sankha Subhra Mullick, and P.N. Suganthan, *Recent advances in differential evolution – an updated survey*, Swarm and Evolutionary Computation **27** (2016), 1 – 30.

[DVC19] Viet Hung Dang, Ngo Anh Vien, and Tae Choong Chung, *A covariance matrix adaptation evolution strategy in reproducing kernel hilbert space*, Genetic Programming and Evolvable Machines (2019), 1–23 (English).

[DW18] Carola Doerr and Markus Wagner, *On the effectiveness of simple success-based parameter selection mechanisms for two classical discrete black-box optimization benchmark problems*, Proc. of Genetic and Evolutionary Computation Conference (GECCO'18), ACM, 2018, pp. 943–950.

[DWY18] Benjamin Doerr, Carsten Witt, and Jing Yang, *Runtime analysis for self-adaptive mutation rates*, Proc. of Genetic and Evolutionary Computation Conference (GECCO'18), ACM, 2018, pp. 1475–1482.

References IV

[EHM99] Agoston Endre Eiben, Robert Hinterding, and Zbigniew Michalewicz, *Parameter control in evolutionary algorithms*, IEEE Transactions on Evolutionary Computation **3** (1999), 124–141.

[EMSS07] A. E. Eiben, Zbigniew Michalewicz, Marc Schoenauer, and James E. Smith, *Parameter control in evolutionary algorithms*, Parameter Setting in Evolutionary Algorithms, Studies in Computational Intelligence, vol. 54, Springer, 2007, pp. 19–46.

[FCSS08] Álvaro Fialho, Luís Da Costa, Marc Schoenauer, and Michèle Sebag, *Extreme value based adaptive operator selection*, Proc. of Parallel Problem Solving from Nature (PPSN'08), Lecture Notes in Computer Science, vol. 5199, Springer, 2008, pp. 175–184.

[FCSS10] ———, *Analyzing bandit-based adaptive operator selection mechanisms*, Annals of Mathematics and Artificial Intelligence **60** (2010), 25–64.

[FG88] J Michael Fitzpatrick and John J Grefenstette, *Genetic algorithms in noisy environments*, Machine learning **3** (1988), no. 2-3, 101–120.

[FKH18] Stefan Falkner, Aaron Klein, and Frank Hutter, *BOHB: Robust and efficient hyperparameter optimization at scale*, ICML, 2018, pp. 1436–1445.

[GBX+18] Abhijith M. Gopakumar, Prasanna V. Balachandran, Dezhen Xue, James E. Gubernatis, and Turab Lookman, *Multi-objective optimization for materials discovery via adaptive design*, Scientific Reports **8** (2018), 3738.

[GGK+14] A. Glotić, A. Glotić, P. Kitak, J. Pihler, and I. Tičar, *Parallel self-adaptive differential evolution algorithm for solving short-term hydro scheduling problem*, IEEE Transactions on Power Systems **29** (2014), no. 5, 2347–2358.

[GNB+19] G. Golkarnarenji, M. Naebe, K. Badii, A. S. Milani, A. Jamali, A. Bab-Hadiashar, R. N. Jazar, and H. Khayyam, *Multi-objective optimization of manufacturing process in carbon fiber industry using artificial intelligence techniques*, IEEE Access **7** (2019), 67576–67588.

[Gre86] John J. Grefenstette, *Optimization of control parameters for genetic algorithms*, IEEE Trans. Systems, Man, and Cybernetics **16** (1986), 122–128.

[Gre96] Horst Greiner, *Robust optical coating design with evolutionary strategies*, Applied Optics **35** (1996), no. 28, 5477–5483.

[GZ15] Arnel Glotić and Aleš Zamuda, *Short-term combined economic and emission hydrothermal optimization by surrogate differential evolution*, Applied Energy **141** (2015), 42 – 56.

[HHB10] Ting Hu, Simon Harding, and Wolfgang Banzhaf, *Variable population size and evolution acceleration: a case study with a parallel evolutionary algorithm*, Genetic Programming and Evolvable Machines **11** (2010), 205–225.

[HHLB11] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown, *Sequential model-based optimization for general algorithm configuration*, Proc. of Learning and Intelligent Optimization (LION'11), Springer, 2011, pp. 507–523.

[HHLBS09] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle, *ParamILS: An automatic algorithm configuration framework*, Journal of Artificial Intelligence Research **36** (2009), 267–306.

[HJS11] Nizar Hachicha, Bassem Jarboui, and Patrick Siarry, *A fuzzy logic control using a differential evolution algorithm aimed at modelling the financial market dynamics*, Information Sciences **181** (2011), no. 1, 79–91.

[HM90] Jürgen Hesser and Reinhard Männer, *Towards an optimal mutation probability for genetic algorithms*, Proc. of Parallel Problem Solving from Nature (PPSN'90), Lecture Notes in Computer Science, vol. 496, Springer, 1990, pp. 23–32.

[HS06] G. E. Hinton and R. R. Salakhutdinov, *Reducing the dimensionality of data with neural networks*, Science **313** (2006), no. 5786, 504–507.

References V

[IHR07] Christian Igel, Nikolaus Hansen, and Stefan Roth, *Covariance matrix adaptation for multi-objective optimization*, Evolutionary computation **15** (2007), no. 1, 1–28.

[JB05] Yaochu Jin and Jürgen Branke, *Evolutionary optimization in uncertain environments-a survey*, IEEE Transactions on evolutionary computation **9** (2005), no. 3, 303–317.

[JW06] Thomas Jansen and Ingo Wegener, *On the analysis of a dynamic evolutionary algorithm*, Journal of Discrete Algorithms **4** (2006), 181–199.

[KBP13] Peter Korošec, Uroš Bole, and Gregor Papa, *A multi-objective approach to the application of real-world production scheduling*, Expert Systems with Applications **40** (2013), no. 15, 5839–5853.

[KHE15] Giorgos Karafotias, Mark Hoogendoorn, and A.E. Eiben, *Parameter control in evolutionary algorithms: Trends and challenges*, IEEE Transactions on Evolutionary Computation **19** (2015), 167–187.

[KK06] 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** (2006), 19–28.

[KMH+04] Stefan Kern, Sibylle D. Müller, Nikolaus Hansen, Dirk Büche, Jiri Ocenasek, and Petros Koumoutsakos, *Learning probability distributions in continuous evolutionary algorithms - a comparative review*, Natural Computing **3** (2004), 77–112.

[KP13] Peter Korošec and Gregor Papa, *Metaheuristic approach to transportation scheduling in emergency situations*, Transport **28** (2013), no. 1, 46–59.

[KPV10] Peter Korošec, Gregor Papa, and Vida Vukašinović, *Application of memetic algorithm in production planning*, Proc. Bioinspired Optimization Methods and their Applications, BIOMA 2010, May 2010, pp. 163–175.

[Kv05] Peter Korošec and Jurij Šilc, *The multilevel ant stigmergy algorithm: An industrial case study*, Proceedings of the 8th Joint Conference on Information Sciences (Salt Lake City, UT), July 21-25 2005, pp. 475–478.

[K12] Peter Korošec, Jurij Šilc, and Bogdan Filipič, *The differential ant-stigmergy algorithm*, Information Sciences **192** (2012), no. 1, 82–97.

[LDC+16] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle, *The irace package: Iterated racing for automatic algorithm configuration*, Operations Research Perspectives **3** (2016), 43–58.

[LEC12] T.W. Liao, P.J. Egbelu, and P.C. Chang, *Two hybrid differential evolution algorithms for optimal inbound and outbound truck sequencing in cross docking operations*, Applied Soft Computing **12** (2012), no. 11, 3683 – 3697.

[Len18] Johannes Lengler, *A general dichotomy of evolutionary algorithms on monotone functions*, Proc. of Parallel Problem Solving from Nature (PPSN'18), Lecture Notes in Computer Science, vol. 11102, Springer, 2018, pp. 3–15.

[LJD+17] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar, *Hyperband: A novel bandit-based approach to hyperparameter optimization*, J. Mach. Learn. Res. **18** (2017), no. 1, 6765–6816.

[LLM07] Fernando G. Lobo, Cláudio F. Lima, and Zbigniew Michalewicz (eds.), *Parameter setting in evolutionary algorithms*, Studies in Computational Intelligence, vol. 54, Springer, 2007.

[LMR+18] Laurent Lemarchand, Damien Massé, Pascal Rebreyend, , and Johan Håkansson, *Multiobjective optimization for multimode transportation problems*, Advances in Operations Research (2018), 13.

References VI

[LOW17] Andrei Lissovoi, Pietro S. Oliveto, and John Alasdair Warwicker, *On the runtime analysis of generalised selection hyper-heuristics for pseudo-Boolean optimisation*, Proc. of Genetic and Evolutionary Computation Conference (GECCO'17), ACM, 2017, Extended version available at https://arxiv.org/abs/1801.07546, pp. 849–856.

[LTSY15] X. Li, K Tang, P. N. Suganthan, and Z. Yang, *Editorial for the special issue of information sciences journal (ISJ) on 'nature-inspired algorithms for large scale global optimization*, Information Science **316** (2015), 437–439.

[MA17] Ali Wagdy Mohamed and Abdulaziz S. Almazyad, *Differential evolution with novel mutation and adaptive crossover strategies for solving large scale global optimization problems*, Applied Computational Intelligence and Soft Computing **2017** (2017), 1–18.

[Mit97] Brad L. Miller, *Noise, sampling, and efficient genetic algorithms*, Ph.D. thesis, University of Illinois at Urbana-Champaign, 1997.

[MPV12] A. Migdalas, Panos M. Pardalos, and Peter Vrbrand, *Multilevel optimization: Algorithms and applications*, 1st ed., Springer Publishing Company, Incorporated, 2012.

[Müh92] Heinz Mühlenbein, *How genetic algorithms really work: Mutation and hillclimbing*, Proc. of Parallel Problem Solving from Nature (PPSN'92), Elsevier, 1992, pp. 15–26.

[NT09] Boris Naujoks and Heike Trautmann, *Online convergence detection for multiobjective aerodynamic applications*, 2009 IEEE Congress on Evolutionary Computation, May 2009, pp. 332–339.

[NYB12] Trung Thanh Nguyen, Shengxiang Yang, and Juergen Branke, *Evolutionary dynamic optimization: A survey of the state of the art*, Swarm and Evolutionary Computation **6** (2012), 1–24.

[OLML19] Mohammad Nabi Omidvar, Xiaodong Li, Daniel Molina, and Antonio LaTorre, *Evolutionary large-scale global optimization*, 2019, CEC 2019 Tutorial.

[OLN09] Pietro Simone Oliveto, Per Kristian Lehre, and Frank Neumann, *Theoretical analysis of rank-based mutation - combining exploration and exploitation*, Proc. of Congress on Evolutionary Computation (CEC'09), IEEE, 2009, pp. 1455–1462.

[OYM+17] Mohammad Nabi Omidvar, Ming Yang, Yi Mei, Xiaodong Li, and Xin Yao, *DG2: A faster and more accurate differential grouping for large-scale black-box optimization*, IEEE Transactions on Evolutionary Computation **21** (2017), no. 6, 929–942.

[Pap08] Gregor Papa, *Parameter-less evolutionary search*, Proc. Genetic and Evolutionary Computation Conference (GECCO 2008), 2008, pp. 1133–1134.

[Pap13] ———, *Parameter-less algorithm for evolutionary-based optimization*, Computational Optimization and Applications **56** (2013), no. 1, 209–229.

[PM10] Gregor Papa and Peter Mrak, *Optimization of cooling appliance control parameters*, Proceedings of the 2nd International Conference on Engineering Optimization, EngOpt2010, 2010.

[PVK12] Gregor Papa, Vida Vukašinović, and Peter Korošec, *Guided restarting local search for production planning*, Engineering Applications of Artificial Intelligence **25** (2012), no. 2, 242–253.

[PXL+07] Pan Zhang, Xin Yao, Lei Jia, B. Sendhoff, and T. Schnier, *Target shape design optimization by evolving splines*, 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 2009–2016.

[Rec73] Ingo Rechenberg, *Evolutionsstrategie*, Friedrich Fromman Verlag (Günther Holzboog KG), Stuttgart, 1973.

References VII

[RM18] Namhee Ryu and Seungjae Min, *Multi-objective optimization with an adaptive weight determination scheme using the concept of hyperplane: Multi-objective optimization with an adaptive weight*, International Journal for Numerical Methods in Engineering **118** (2018).

[RVLB15] Nery Riquelme, Christian Von Lücken, and Benjamin Baran, *Performance metrics in multi-objective optimization*, Computing Conference (CLEI), 2015 Latin American, IEEE, 2015, pp. 1–11.

[SBD19] David Schmaranzer, Roland Braune, and Karl Franz Dörner, *Multi-objective simulation optimization for complex urban mass rapid transit systems*, Annals of Operations Research (2019).

[SMXD14] Ankur Sinha, Pekka Malo, Peng Xu, and Kalyanmoy Deb, *A bilevel optimization approach to automated parameter tuning*, Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (New York, NY, USA), GECCO '14, ACM, 2014, pp. 847–854.

[SP10] F. Samadzadegan and T. Partovi, *Feature selection based on ant colony algorithm for hyperspectral remote sensing images*, 2010 2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing, 2010, pp. 1–4.

[SS68] Michael A. Schumer and Kenneth Steiglitz, *Adaptive step size random search*, IEEE Transactions on Automatic Control **13** (1968), 270–276.

[Sud13] Dirk Sudholt, *A new method for lower bounds on the running time of evolutionary algorithms*, IEEE Transactions on Evolutionary Computation **17** (2013), 418–435.

[TCZ+10] Hong-Fei Teng, Yu Chen, Wei Zeng, Yan-Jun Shi, and Qing-Hua Hu, *A dual-system variable-grain cooperative coevolutionary algorithm: Satellite-module layout design*, IEEE Transactions on Evolutionary Computation **14** (2010), no. 3, 438–455.

[Tei01] Jürgen Teich, *Pareto-front exploration with uncertain objectives*, International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2001, pp. 314–328.

[Thi05] Dirk Thierens, *An adaptive pursuit strategy for allocating operator probabilities*, Proc. of Genetic and Evolutionary Computation Conference (GECCO'05), ACM, 2005, pp. 1539–1546.

[TKP+07] Tea Tušar, Peter Korošec, Gregor Papa, Bogdan Filipič, and Jurij Šilc, *A comparative study of stochastic optimization methods in electric motor design*, Applied Intelligence **27** (2007), no. 2, 101–111.

[Van02] G.N. Vanderplaats, *Very large scale optimization*, Nat. Aeronautics Space Admin., Washington, DC, 2002.

[VC16] Marta Vallejo and David W. Corne, *Evolutionary algorithms under noise and uncertainty: A location-allocation case study*, 2016 IEEE Symposium Series on Computational Intelligence (SSCI) (2016), 1–10.

[VML11] M. Vasile, E. Minisci, and M. Locatelli, *An inflationary differential evolution algorithm for space trajectory optimization*, IEEE Transactions on Evolutionary Computation **15** (2011), no. 2, 267–281.

[WG12] C. Wang and J. Gao, *High-dimensional waveform inversion with cooperative coevolutionary differential evolution algorithm*, IEEE Geoscience and Remote Sensing Letters **9** (2012), no. 2, 297–301.

[WHD+13] Yu Wang, Jin Huang, Wei Shan Dong, Jun Chi Yan, Chun Hua Tian, Min Li, and Wen Ting Mo, *Two-stage based ensemble optimization framework for large-scale global optimization*, European Journal of Operational Research **228** (2013), no. 2, 308 – 320.

References VIII

[WWY09] Hongfeng Wang, Dingwei Wang, and Shengxiang Yang, *A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems*, Soft Computing **13** (2009), no. 8-9, 763–780.

[WWY10] Yao-Nan Wang, Liang-Hong Wu, and Xiao-Fang Yuan, *Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure*, Soft Computing **14** (2010), no. 3, 193–209.

[YDB19] Furong Ye, Carola Doerr, and Thomas Bäck, *Interpolating Local and Global Search by Controlling the Variance of Standard Bit Mutation*, Proc. Conference on Evolutionary Computation (CEC'19), IEEE, 2019, pp. 2292–2299.

[Yil13] Ali R. Yildiz, *A new hybrid differential evolution algorithm for the selection of optimal machining parameters in milling operations*, Applied Soft Computing **13** (2013), no. 3, 1561 – 1566, Hybrid evolutionary systems for manufacturing processes.

[YJG09] Dongdong Yang, Licheng Jiao, and Maoguo Gong, *Adaptive multi-objective optimization based on nondominated solutions*, Computational Intelligence **25** (2009), no. 2, 84–108.

[YLLF18] Yan Ye, Jingfeng Li, Kaibin Li, and Hui Fu, *Cross-docking truck scheduling with product unloading/loading constraints based on an improved particle swarm optimisation algorithm*, International Journal of Production Research **56** (2018), no. 16, 5365–5385.

[YSTY16] Zhenyu Yang, B. Sendhoff, Ke Tang, and Xin Yao, *Target shape design optimization by evolving b-splines with cooperative coevolution*, Applied Soft Computing **48** (2016), no. C, 672–682.

[YTY08] Zhenyu Yang, Ke Tang, and Xin Yao, *Large scale evolutionary optimization using cooperative coevolution*, Information Sciences **178** (2008), no. 15, 2985 – 2999, Nature Inspired Problem-Solving.

[ZB14] Ales Zamuda and Janez Brest, *Vectorized procedural models for animated trees reconstruction using differential evolution*, Information Sciences **278** (2014), 1–21.

[ZBBv08] A. Zamuda, J. Brest, B. Bošković, and V. Žumer, *Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution*, IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), June 2008, pp. 3718–3725.

[ZBBZ09] Ales Zamuda, Janez Brest, Borko Boskovic, and Viljem Zumer, *Differential evolution with self-adaptation and local search for constrained multiobjective optimization*, Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2009, Trondheim, Norway, 18-21 May, 2009, IEEE, 2009, pp. 195–202.

[ZCJW14] Z. H. Zhou, N. V. Chawla, Y. Jin, and G. J. Williams, *Big data opportunities and challenges: Discussions from data analytics perspectives [discussion forum]*, IEEE Computational Intelligence Magazine **9** (2014), no. 4, 62–74.

[ZH14] Aleš Zamuda and José Daniel Hernández Sosa, *Differential evolution and underwater glider path planning applied to the short-term opportunistic sampling of dynamic mesoscale ocean structures*, Applied Soft Computing **24** (2014), 95 – 108.

[ZHA16] A. Zamuda, J. Daniel Hernández Sosa, and L. Adler, *Improving constrained glider trajectories for ocean eddy border sampling within extended mission planning time*, 2016 IEEE Congress on Evolutionary Computation (CEC), 2016, pp. 1727–1734.

[ZS19] Aleš Zamuda and José Daniel Hernández Sosa, *Success history applied to expert system for underwater glider path planning using differential evolution*, Expert Systems with Applications **119** (2019), 155 – 170.