Evolutionary Computing

Chapter 7

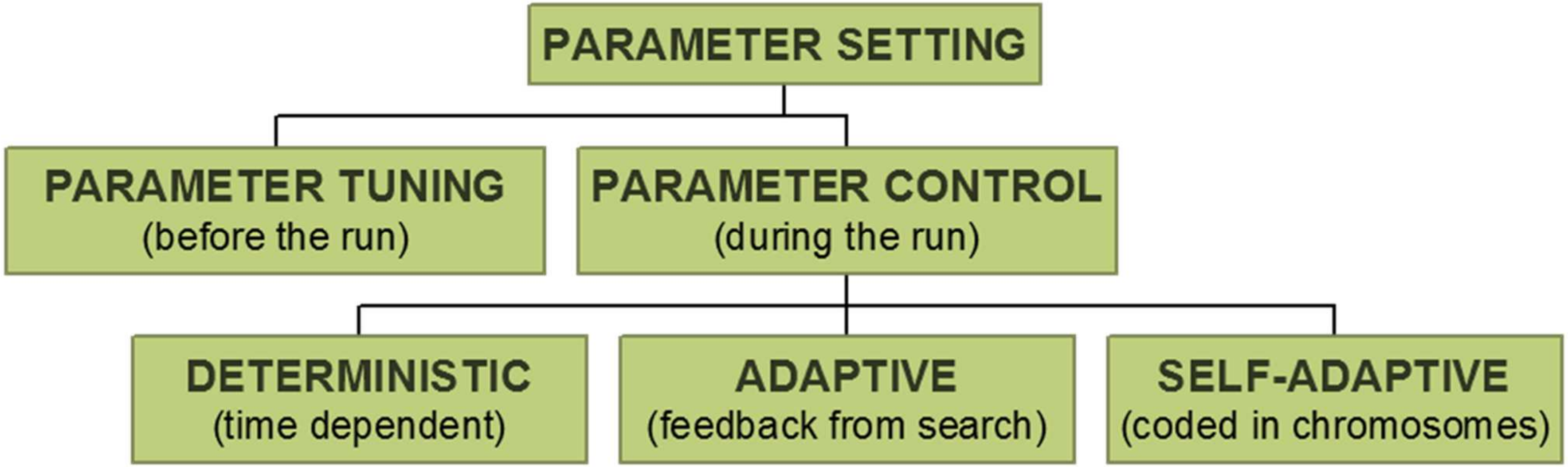
Chapter 7: Parameters and Parameter Tuning

- History
- Taxonomy
- Parameter Tuning vs Parameter Control
- EA calibration
- Parameter Tuning
  - Testing
  - Effort
  - Recommendation

Brief historical account

- 1970/80ies “GA is a robust method”
- 1970ies + ESs self-adapt mutation stepsize σ
- 1986 meta-GA for optimizing GA parameters
- 1990ies EP adopts self-adaptation of σ as ‘standard’
- 1990ies some papers on changing parameters on-the-fly
- 1999 Eiben-Michalewicz-Hinterding paper proposes clear taxonomy & terminology

Taxonomy


PARAMETER SETTING
PARAMETER TUNING
(before the run)
PARAMETER CONTROL
(during the run)
DETERMINISTIC
(time dependent)
ADAPTIVE
(feedback from search)
SELF-ADAPTIVE
(coded in chromosomes)

Parameter tuning

Parameter tuning: testing and comparing different values before the "real" run

Problems:

- users mistakes in settings can be sources of errors or sub-optimal performance
- costs much time
- parameters interact: exhaustive search is not practicable
- good values may become bad during the run

Parameter control

Parameter control: setting values on-line, during the actual run, e.g.

- predetermined time-varying schedule p = p(t)
- using (heuristic) feedback from the search process
- encoding parameters in chromosomes and rely on natural selection

Problems:

- finding optimal p is hard, finding optimal p(t) is harder
- still user-defined feedback mechanism, how to “optimize”?
- when would natural selection work for algorithm parameters?

Notes on parameter control

- Parameter control offers the possibility to use appropriate values in various stages of the search
- Adaptive and self-adaptive control can "liberate" users from tuning → reduces need for EA expertise for a new application
- Assumption: control heuristic is less parameter-sensitive than the EA

BUT

- State-of-the-art is a mess: literature is a potpourri, no generic knowledge, no principled approaches to developing control heuristics (deterministic or adaptive), no solid testing methodology

WHAT ABOUT AUTOMATED TUNING?

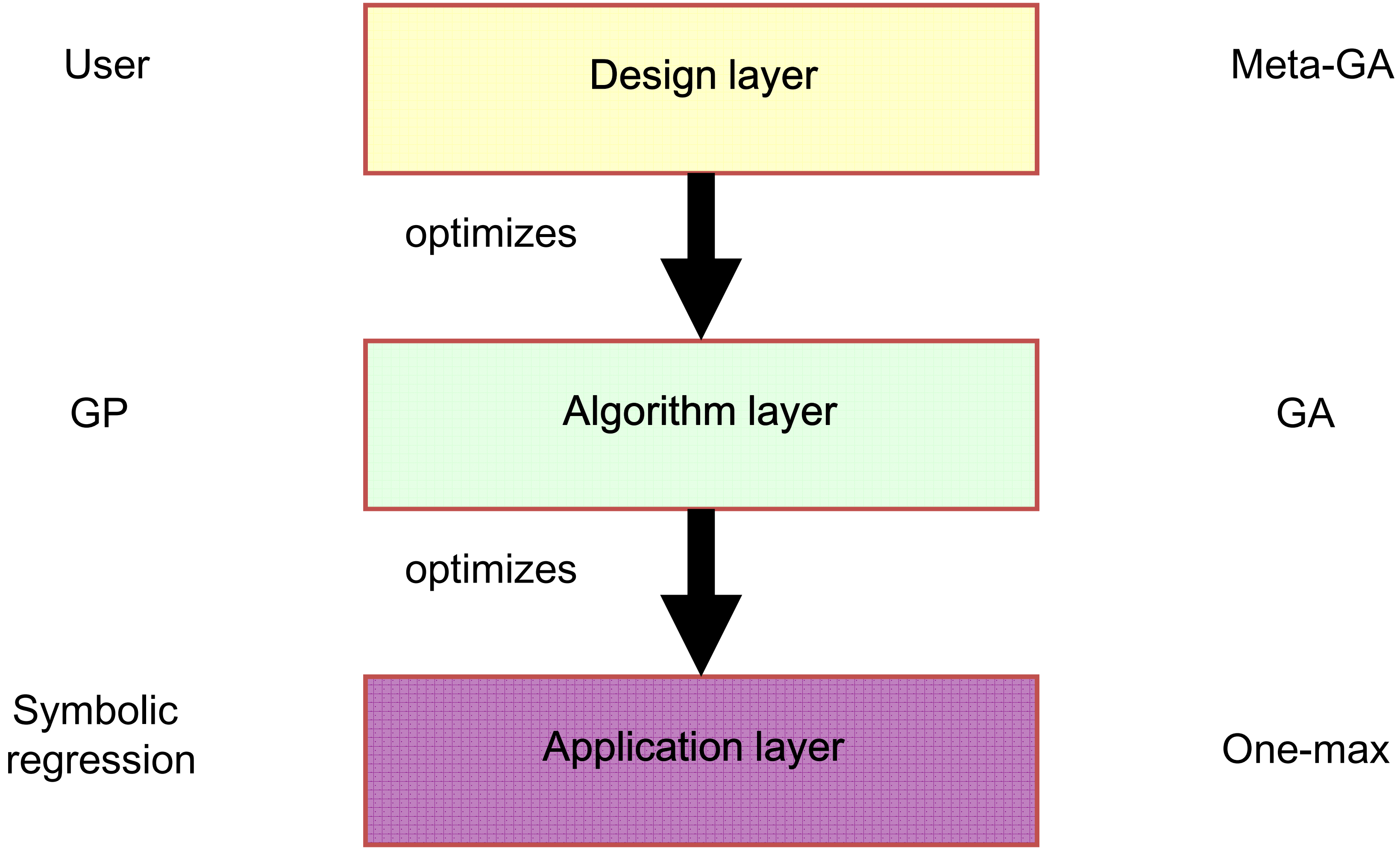
Historical account (cont'd)

Last decade:

- More & more work on parameter control
  - Traditional parameters: mutation and xover
  - Non-traditional parameters: selection and population size
  - All parameters ➔ "parameterless" EAs (name!?)

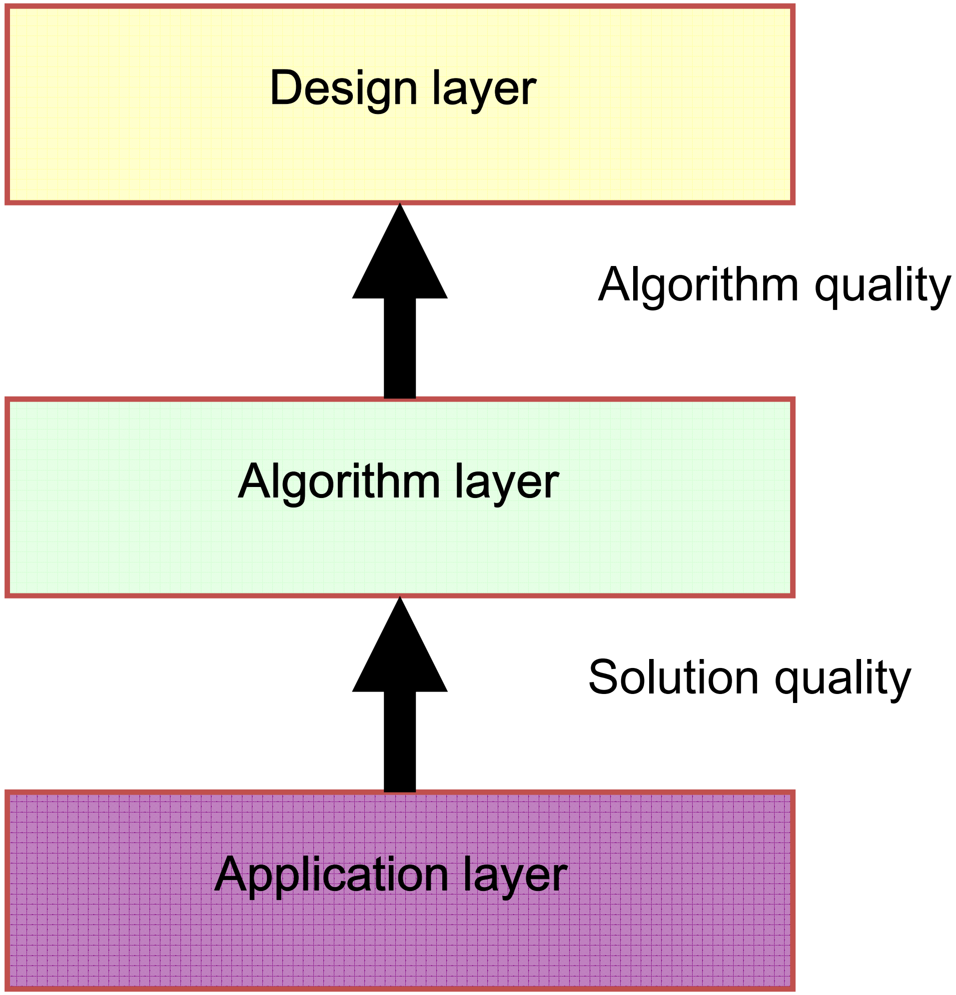
- Not much work on parameter tuning, i.e.,
  - Nobody reports on tuning efforts behind their EA published
  - A handful papers on tuning methods / algorithms

Control flow of EA calibration / design

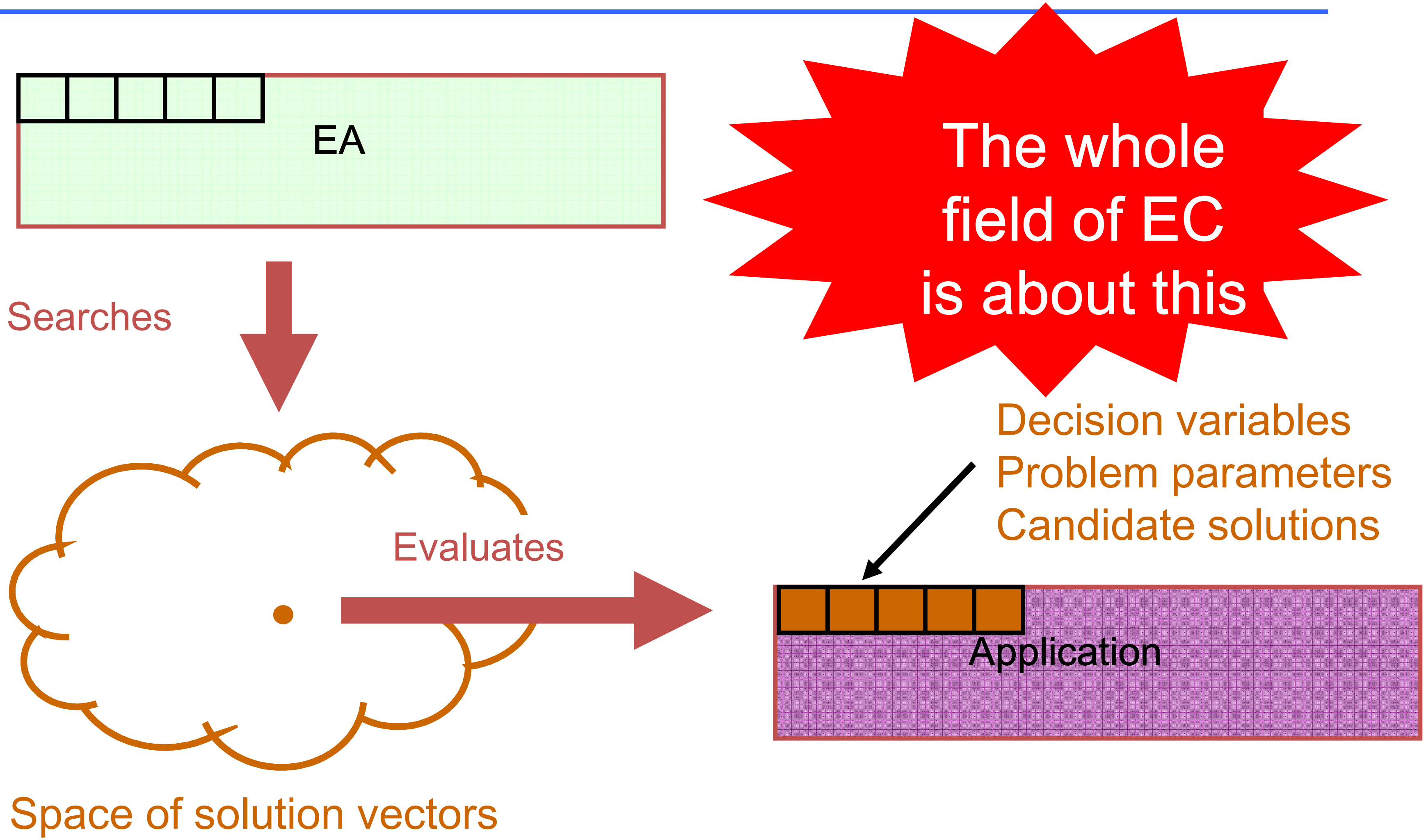


Adapted from A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing 2014

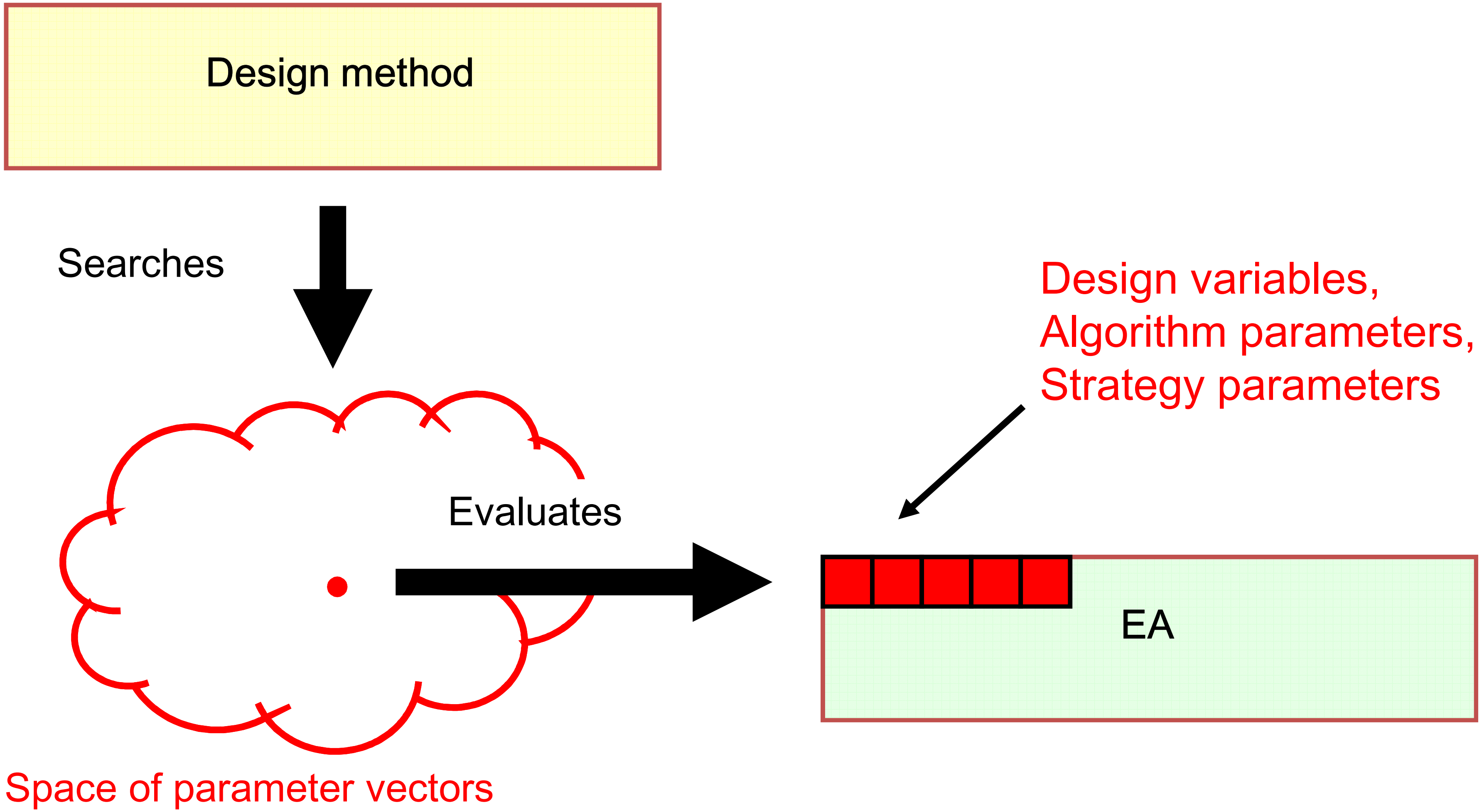
Information flow of EA calibration / design



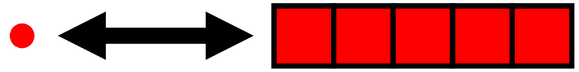
Lower level of EA calibration / design



Upper level of EA calibration / design



Parameter – performance landscape

- All parameters together span a (search) space 
- One point – one EA instance 
- Height of point = performance of EA instance on a given problem
- Parameter-performance landscape or utility landscape for each { EA + problem instance + performance measure }
- This landscape is unlikely to be trivial, e.g., unimodal, separable
- If there is some structure in the utility landscape, then we can do better than random or exhaustive search

Ontology - Terminology

	LOWER PART	UPPER PART
METHOD	EA	Tuner
SEARCH SPACE	Solution vectors	Parameter vectors
QUALITY	Fitness	Utility
ASSESSMENT	Evaluation	Test

- Fitness ≈ objective function value
- Utility = ?
  - **M**ean **B**est **F**itness
  - **A**verage number of **E**valuations to **S**olution
  - **S**uccess **R**ate
  - Robustness, …
  - Combination of some of these

Off-line vs. on-line calibration / design

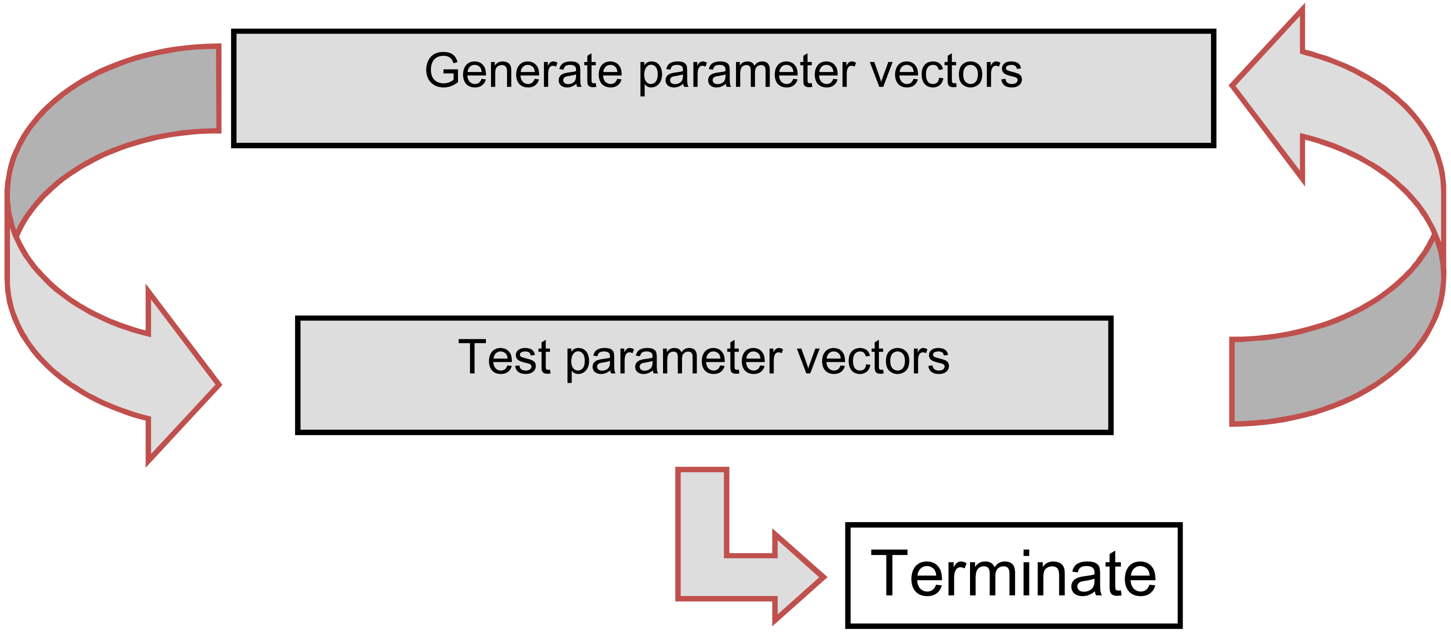
Design / calibration method

- Off-line → parameter tuning
- On-line → parameter control

- Advantages of tuning
  - Easier
  - Most immediate need of users
  - Control strategies have parameters too → need tuning themselves
  - Knowledge about tuning (utility landscapes) can help the design of good control strategies
  - There are indications that good tuning works better than control

Tuning by generate-and-test

- EA tuning is a search problem itself
- Straightforward approach: generate-and-test

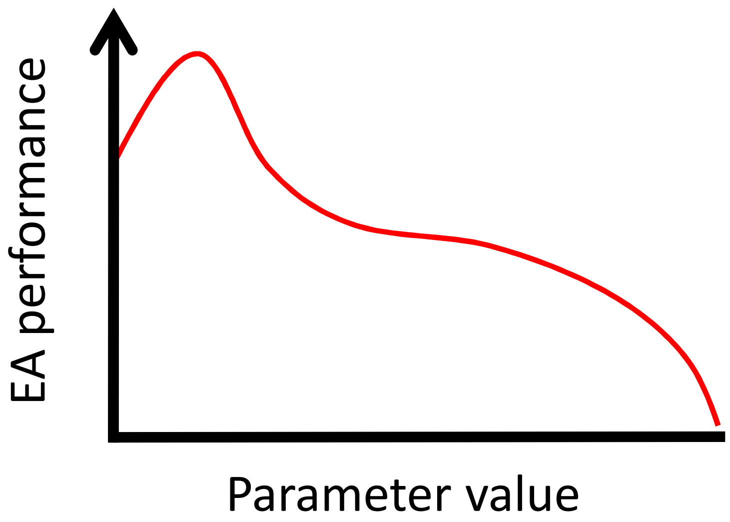


Testing parameter vectors

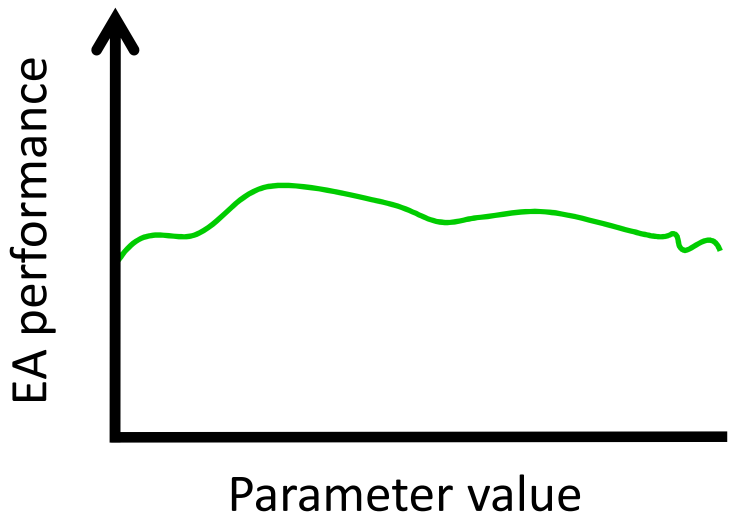
- Run EA with these parameters on the given problem or problems
- Record EA performance in that run e.g., by
  - Solution quality = best fitness at termination
  - Speed ≈ time used to find required solution quality
- EAs are stochastic → repetitions are needed for reliable evaluation → we get statistics, e.g.,
  - Average performance by solution quality, speed (MBF, AES, AEB)
  - Success rate = % runs ending with success
  - Robustness = variance in those averages over different problems
- Big issue: how many repetitions of the test

Numeric parameters

- E.g., population size, xover rate, tournament size, …
- Domain is subset of R, Z, N (finite or infinite)
- Sensible distance metric → searchable



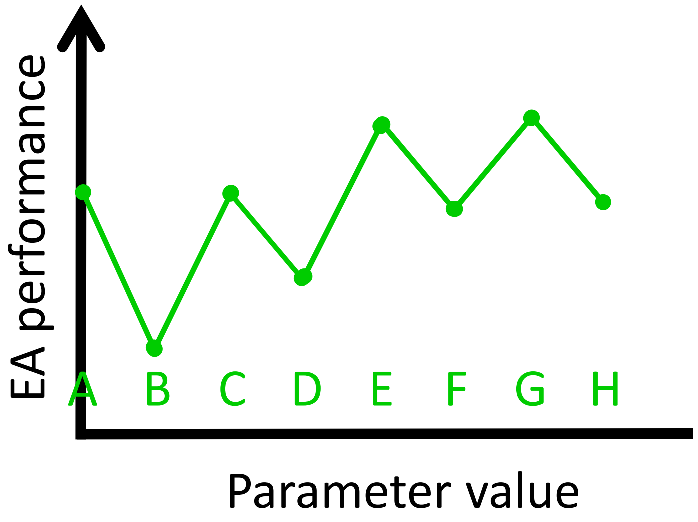
Relevant parameter



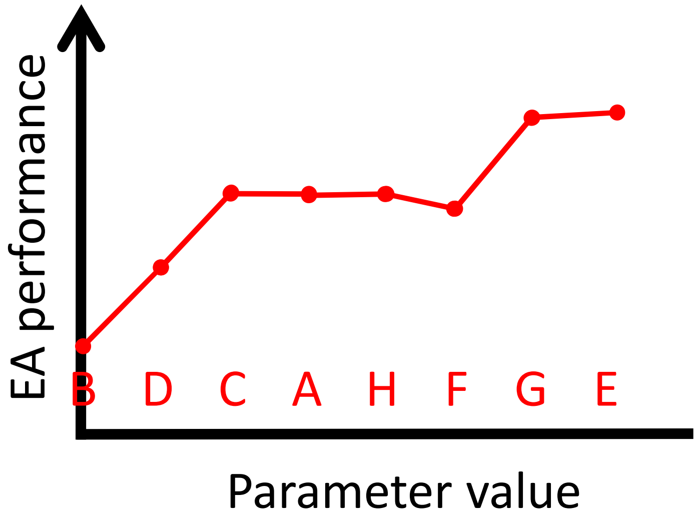
Irrelevant parameter

Symbolic parameters

- E.g., xover_operator, elitism, selection_method
- Finite domain, e.g., {1-point, uniform, averaging}, {Y, N}
- No sensible distance metric → non-searchable, must be sampled



Non-searchable ordering



Searchable ordering

Notes on parameters

- A value of a symbolic parameter can introduce a numeric parameter, e.g.,
  - Selection = tournament → tournament size
  - Populations_type = overlapping → generation gap
- Parameters can have a hierarchical, nested structure
- Number of EA parameters is not defined in general
- Cannot simply denote the design space / tuning search space by

$$S = Q_1 \times \ldots Q_m \times R_1 \times \ldots \times R_n$$

with Q_i / R_j as domains of the symbolic/numeric parameters

What is an EA? (1/2)

	ALG-1	ALG-2	ALG-3	ALG-4
SYMBOLIC PARAMETERS				
Representation	Bit-string	Bit-string	Real-valued	Real-valued
Overlapping pops	N	Y	Y	Y
Survivor selection	–	Tournament	Replace worst	Replace worst
Parent selection	Roulette wheel	Uniform determ	Tournament	Tournament
Mutation	Bit-flip	Bit-flip	N(0,σ)	N(0,σ)
Recombination	Uniform xover	Uniform xover	Discrete recomb	Discrete recomb
NUMERIC PARAMETERS				
Generation gap	–	0.5	0.9	0.9
Population size	100	500	100	300
Tournament size	–	2	3	30
Mutation rate	0.01	0.1	–	–
Mutation stepsize	–	–	0.01	0.05
Crossover rate	0.8	0.7	1	0.8

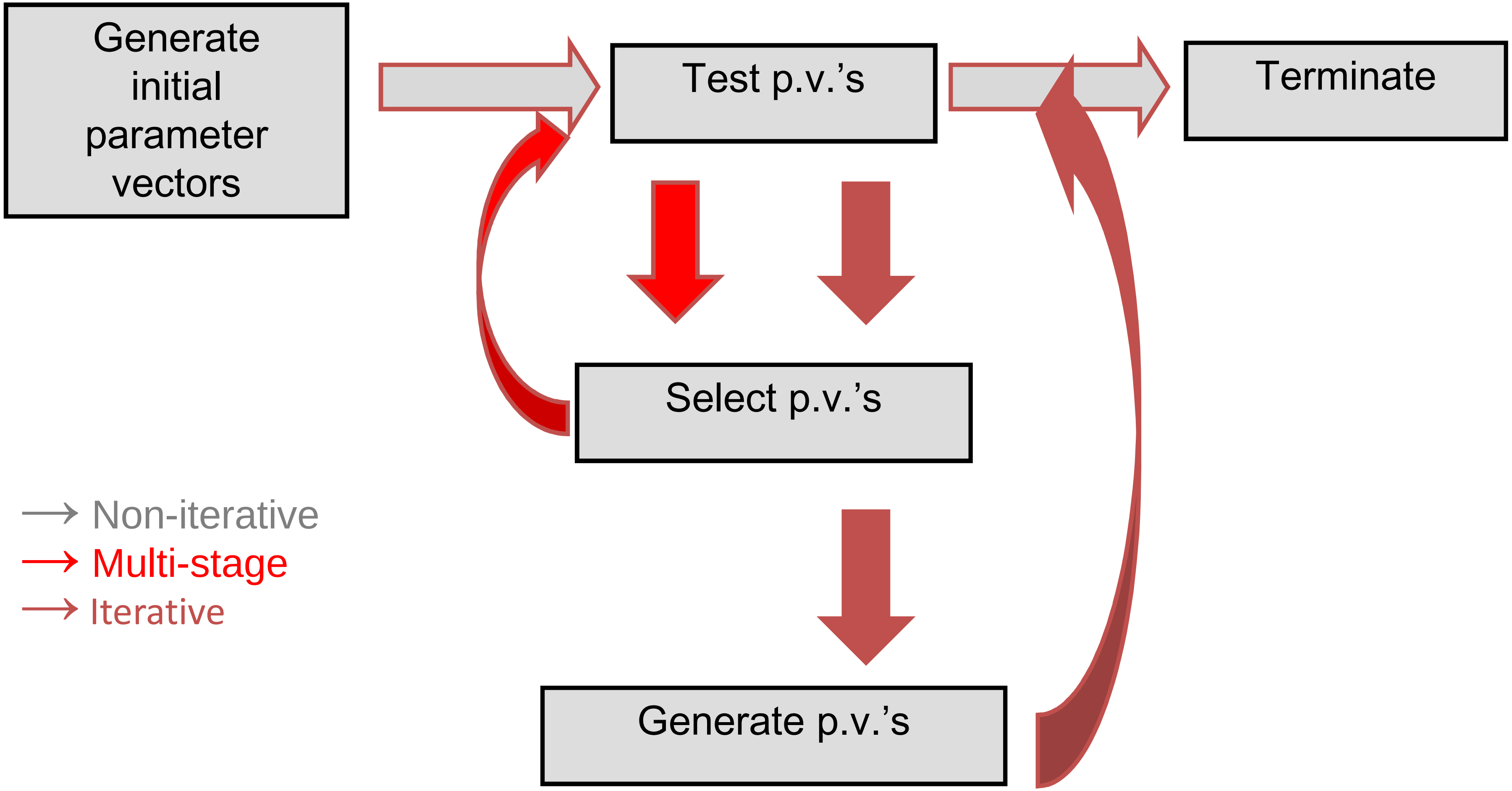
Adapted from A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing 2014

What is an EA? (2/2)

Make a principal distinction between EAs and EA instances and place the border between them by:

- Option 1
  - There is only one EA, the generic EA scheme
  - Previous table contains 1 EA and 4 EA-instances
- Option 2
  - An EA = particular configuration of the symbolic parameters
  - Previous table contains 3 EAs, with 2 instances for one of them
- Option 3
  - An EA = particular configuration of parameters
  - Notions of EA and EA-instance coincide
  - Previous table contains 4 EAs / 4 EA-instances

Generate-and-test under the hood



Tuning effort

- Total amount of computational work is determined by
  - A = number of vectors tested
  - B = number of tests per vector
  - C = number of fitness evaluations per test
- Tuning methods can be positioned by their rationale:
  - To optimize A (iterative search)
  - To optimize B (multi-stage search)
  - To optimize A and B (combination)
  - To optimize C (non-existent)
  - …

Optimize A = optimally use A

Applicable only to numeric parameters

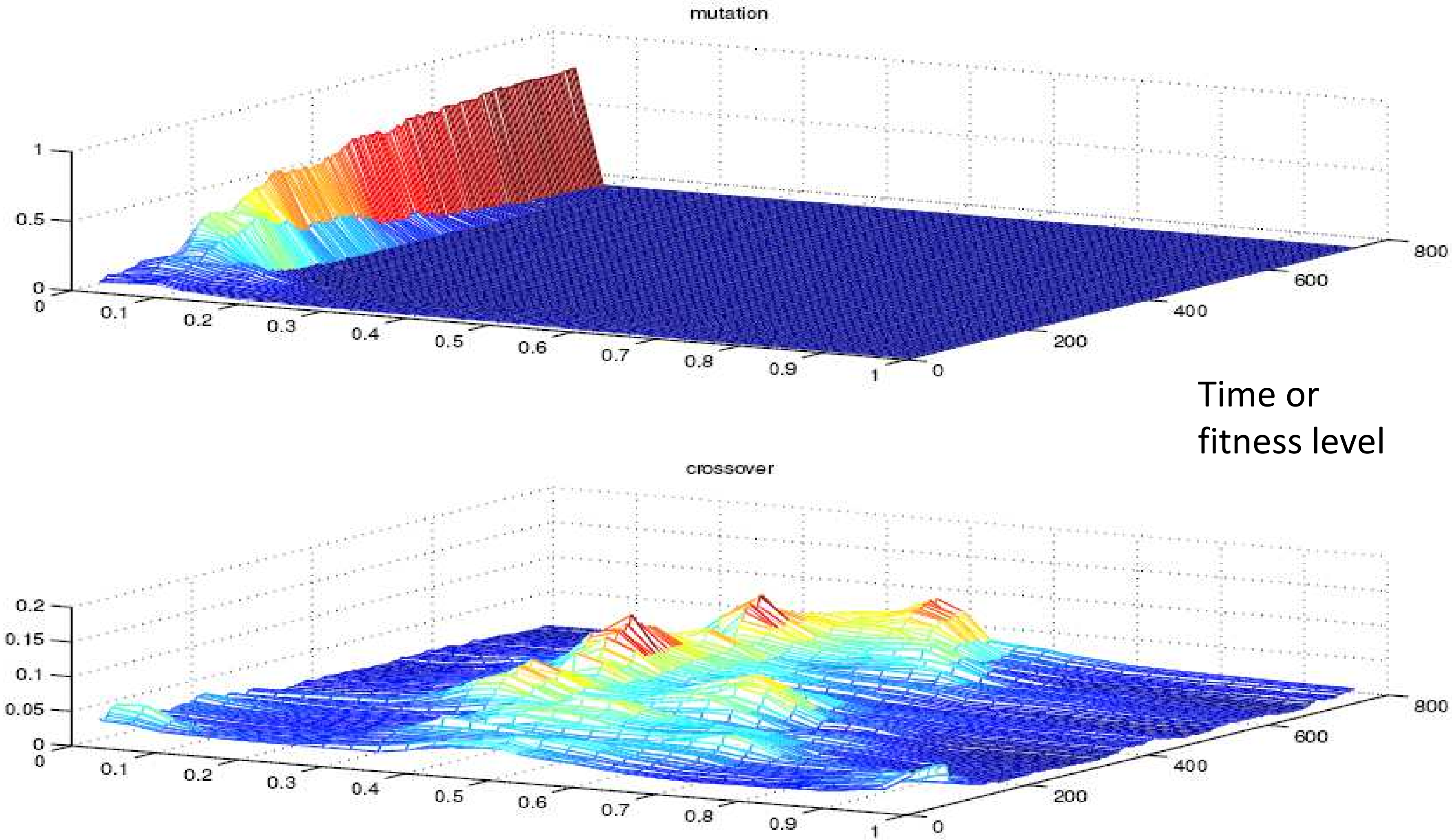
Number of tested vectors not fixed, A is the maximum (stop cond.)

Population-based search:

- **Initialize with N << A vectors and**
- **Iterate: generating, testing, selecting p.v.'s**

- Meta-EA (Greffenstette '86)
  - Generate: usual crossover and mutation of p.v.'s
- SPO (Bartz-Beielstein et al. '05)
  - Generate: uniform random sampling!!! of p.v.'s
- REVAC (Nannen & Eiben '06)
  - Generate: usual crossover and distribution-based mutation of p.v.'s

Evolution of Distributions for Schaffer's f_6



Time or fitness level

REVAC illustration

Optimize B = reduce B

Applicable to symbolic and numeric parameters
Number of tested vectors (A) fixed at initialization
Set of tested vectors can be created by

- regular method → grid search
- random method → random sampling
- exhaustive method → enumeration

Complete testing (single stage) vs. selective testing (multi-stage)

- Complete testing: nr. of tests per vector = B (thus, not optimizing)
- Selective testing: nr. of tests per vector varies, ≤ B
- Idea:
  - Execute **tests in a breadth-first fashion (stages)**, all vectors X < B times
  - Stop testing vectors with statistically significant poorer utility
- Well-known methods
  - ANOVA (Scheffer '89)
  - Racing (Maron & Moore '97)

Optimize A & B

Existing work:

- Meta-EA with racing (Yuan & Gallagher ‘04)

New trick: sharpening (Smit & Eiben 2009)

- Idea: test vectors X < B times and increase X over time during the run of a population-based tuner

Newest method:

- **REVAC with racing & sharpening = REVAC++**

Which tuning method?

- Differences between tuning algorithms
  - Maximum utility reached
  - Computational costs
  - Number of their own parameters – overhead costs
  - Insights offered about EA parameters (probability distribution, interactions, relevance, explicit model…)
- Similarities between tuning algorithms
  - Nobody is using them
  - Can find good parameter vectors
- Solid comparison is missing – ongoing

Tuning "world champion" EAs

Tuned by	G-CMA-ES Avg	G-CMA-ES St dev	G-CMA-ES CEC Δ	SaDE Avg	SaDE St dev	SaDE CEC Δ
G-CMA-ES	0.77	0.2	20 %	0.73	0.25	49 %
REVAC++	0.85	0.24	12 %	0.67	0.22	53 %
SPOT	0.76	0.19	22 %	0.73	0.20	49 %
CEC-2005	0.97	0.32	-	1.43	0.25	-

Ranking at CEC 2005

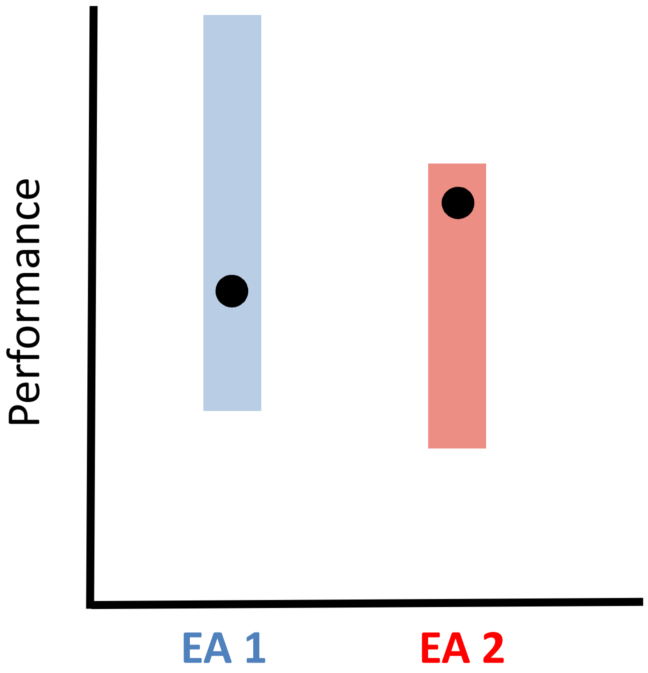
1. CMA-ES
2. SaDE

Ranking after tuning

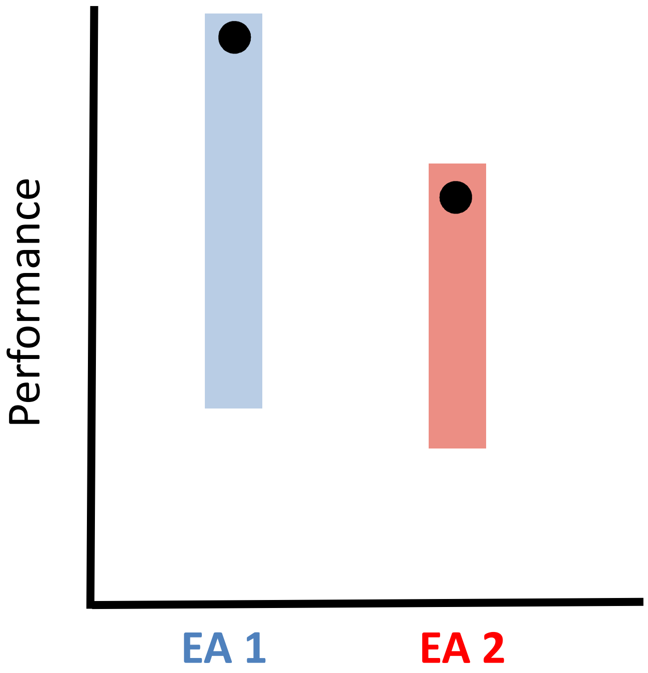
1. SaDE
2. CMA-ES

Main conclusion: if only they had asked us ….

Tuning vs. not tuning



EA as is (accidental parameters)



EA as it can be (“optimal” parameters)

Recommendations

- **DO TUNE** your evolutionary algorithm
- Think of the magic constants
- Decide: speed or solution quality?
- Decide: specialist of generalist EA?
- Measure and report tuning effort
- Try our toolbox: http://sourceforge.net/projects/mobat

Example study 'Best parameters'

- Setup:
  - Problem: Sphere Function
  - EA: defined by Tournament Parent Selection, Random Uniform Survivor Selection, Uniform Crossover, BitFlip Mutation
  - Tuner: REVAC spending X units of tuning effort, tuning for speed
  - A = 1000, B = 30, C = 10000
- Results: the best EA had the following parameter values
  - Population Size: 6
  - Tournament Size: 4
  - ...
- Conclusions: *for this problem* we need a high (parent) selection pressure. This is probably because the problem is unimodal.

Example study ‘Good parameters’

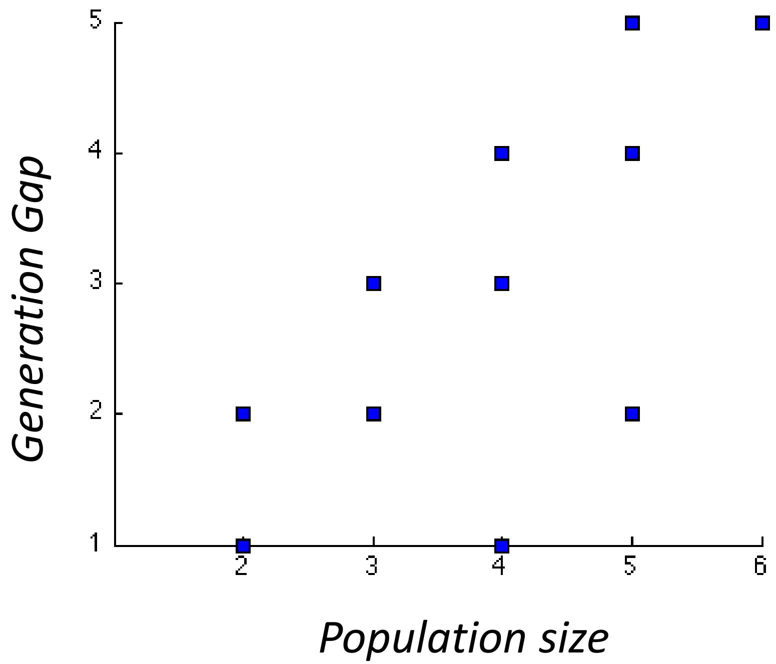
- Setup: same as before

- Results: The 25 best parameters vectors have their values within the following ranges
  - Mutation Rate: [0.01, 0.011]
  - Crossover Rate: [0.2, 1.0]
  - (..)

- Conclusions: *for this problem* the mutation rate is much more relevant than the crossover rate.

Example study 'interactions'

- Setup: same as before
- Results: plotting the pop. size and generation gap of the best parameter vectors shows the following



- Conclusions: *for this problem* the best results are obtained when (almost) the complete population is replaced every generation.

The (near) future of automated tuning

- Hybrid methods for A & B
- Well-funded EA performance measures, multi-objective formulation → multi-objective tuner algorithms
- (Statistical) models of the utility landscape → more knowledge about parameters
- Open source toolboxes
- Distributed execution
- Good testbeds
- Adoption by the EC community
- Rollout to other heuristic methods with parameters

Culture change?

- Fast and good tuning can lead to new attitude
- Past & present: robust EAs preferred
- Future: problem-specific EAs preferred
- Old question: what is better the GA or the ES?
- New question: what symbolic configuration is best?
- … given a maximum effort for tuning
- New attitude / practice:
  - tuning efforts are measured and reported
  - EAs with their practical best settings are compared, instead of unmotivated “magical”settings