Genetic Algorithms

Parameters and Parameter Tuning

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

Brief historical account

- 1970s/80s: "GA is a robust method"
- 1970s+: ESs self-adapt mutation stepsize $\sigma$
- 1986: meta-GA for optimizing GA parameters
- 1990s: EP adopts self-adaptation of $\sigma$ as ‘standard’
- 1990s: papers on changing parameters on-the-fly

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 – part of development

Problems:
- user “mistakes” in settings can be sources of errors or suboptimal performance
- takes significant time
- parameters interact: exhaustive search is not practical (or even possible, in some cases)
- good values may become bad during the run (at different stages of evolutionary development in the population)

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

Historical account (cont’d)

Last 20 years:
- More & more work on parameter control
  - Traditional parameters: mutation and xover
  - Non-traditional parameters: selection and population size
  - All parameters → “parameterless” EAs (what to call these?)
  - Some theoretical results (e.g. Carola Doerr)
- Not much work on parameter tuning, i.e.,
  - Nobody reports on tuning efforts behind their published EAs (common refrain: “values were determined empirically”)
  - A handful of papers on tuning methods / algorithms
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 likely to be complex e.g., multimodal
- If there is some structure in the utility landscape, then perhaps we can do better than random or exhaustive search

The Tuning Problem

- Parameter values determine the success and efficiency of a genetic algorithm
- Parameter tuning is a method in which parameter values determined before a run and remain fixed during
- Common approaches:
  - Convention, e.g. mutation rate should be low; xover rate = 0.9
  - Ad hoc choices, e.g. let’s use population size of 100
  - Limited experimentation, e.g. let’s try a few values

The Tuning Problem

Problems

- Problems with convention and ad hoc choices are obvious
  - Were choices ever justified?
  - Do they apply in new problem domains?
- Problems with experimentation
  - Parameters interact – cannot be optimized one-by-one
  - Time consuming; 4 parameters with 5 values each yields 625 parameter combinations. 100 runs each = 62500 runs just for tuning – to be fair, any tuning method will be time consuming
  - Best parameter values may not be in test set

The Tuning Problem

Goal

- Think of design of a GA as a separate search problem
- Then a tuning method is a search algorithm
- Such a tuning method can be used to:
  - Optimize a GA by finding parameters that optimize its performance
  - Analyze a GA by studying how performance depends on parameter values and the problems to which it is applied
- So tuning problem solutions depend on problems to be solved, GA used, and utility function that defines how GA quality is measured
The Tuning Problem

Terminology

<table>
<thead>
<tr>
<th>METHOD</th>
<th>Problem Solving</th>
<th>Algorithm Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEARCH SPACE</td>
<td>Solution vectors</td>
<td>Parameter vectors</td>
</tr>
<tr>
<td>QUALITY</td>
<td>Fitness</td>
<td>Utility</td>
</tr>
<tr>
<td>ASSESSMENT</td>
<td>Evaluation</td>
<td>Test</td>
</tr>
</tbody>
</table>

- Fitness = objective function value
- Utility = ?
  - Mean Best Fitness
  - Average number of Evaluations to Solution
  - Success Rate
  - Robustness, ...
  - Combination of some of these

Defining Algorithm Quality

- GA quality generally measured by a combination of solution quality and algorithm efficiency
- Solution quality – reflected in fitness values
- Algorithm efficiency
  - Number of fitness evaluations
  - CPU time
  - Clock-on-the-wall time

Defining Algorithm Quality

- Three generally used combinations of solution quality and computing time for single run of algorithm
  - Fix computing time and measure solution quality
    - Given maximum runtime, quality is best fitness at termination
  - Fix solution quality and measure computing time required
    - Given a minimum fitness requirement, performance is the runtime needed to achieve it
  - Fix both and measure success
    - Given maximum runtime and minimum fitness requirement, run is successful if it achieves fitness requirement within runtime limit

Tuning Methods

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 Method

**Tuning by generate-and-test**

- Generate-and-test is a common search strategy
- Since EA tuning is a search problem itself...
- Straightforward approach:
  - Generate parameter vectors
  - Test parameter vectors
  - Terminate

All tuning methods are a form of generate-and-test.

### 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)
  - Success rate = % runs ending with success
  - Robustness = variance in those averages over different problems
- Question: how many repetitions of the test (yet another "parameter")

### Definitions

- Because GAs are stochastic, single runs don’t tell us much about the quality of an algorithm
- Aggregate measures over multiple runs:
  - MBF: Mean Best Fitness
  - AES: Average evaluations to solution
  - SR: Success rate

### Generate-and-Test

**Numeric parameters**

- E.g., population size, xover rate, tournament size, ...
- Domain is subset of R, Z, N (finite or infinite)
- Values are well ordered → searchable
Generate-and-test
Symbolic parameters

• E.g., xover_operator, elitism, selection_method
• Finite domain, e.g., {1-point, uniform, averaging}, {Y, N}
• Values not well ordered → non-searchable, must be sampled

• A value of a symbolic parameter can introduce a numeric parameter, e.g.,
  – Selection = tournament → tournament size
  – Populations_type = overlapping → generation gap
  – Elitism = on → number of best members to keep

What is an EA?

<table>
<thead>
<tr>
<th>ALG-1</th>
<th>ALG-2</th>
<th>ALG-3</th>
<th>ALG-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYMBOLIC PARAMETERS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Representation</td>
<td>Bit-string</td>
<td>Bit-string</td>
<td>Real-valued</td>
</tr>
<tr>
<td>Overlapping pops</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Survivor selection</td>
<td>Tournament</td>
<td>Replace worst</td>
<td>Replace worst</td>
</tr>
<tr>
<td>Parent selection</td>
<td>Roulette-wheel</td>
<td>Uniform-deterministic</td>
<td>Tournament</td>
</tr>
<tr>
<td>Mutation</td>
<td>Bit-flip</td>
<td>Bit-flip</td>
<td>N(out)</td>
</tr>
<tr>
<td>Recombination</td>
<td>Uniform-solver</td>
<td>Uniform-solver</td>
<td>Discrete-recomb</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NUMERIC PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation-gap</td>
</tr>
<tr>
<td>Population size</td>
</tr>
<tr>
<td>Tournament size</td>
</tr>
<tr>
<td>Mutation rate</td>
</tr>
<tr>
<td>Mutation stepsize</td>
</tr>
<tr>
<td>Crossover rate</td>
</tr>
</tbody>
</table>

What is an EA?

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

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
Recommendations

- **DO TUNE** your evolutionary algorithm
- Think of the magic constants
- Decide: speed or solution quality?
- Decide: specialist or generalist EA?
- Measure and report tuning effort

Example study: ‘Best parameters’

- **Setup:**
  - Problem: Sphere Function (see next slide)
  - 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

- **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.

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.
Lower level of EA calibration / design

- EA
- Space of solution vectors
- Evaluates
- Decision variables
- Problem parameters
- Candidate solutions
- Application

Upper level of EA calibration / design

- Design method
- Space of parameter vectors
- Evaluates
- Design variables
- Algorithm parameters
- Strategy parameters

The whole field of EC is about this

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 (Grefenstette '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

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

<table>
<thead>
<tr>
<th>Tuned by</th>
<th>G-CMA-ES</th>
<th>SaDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-CMA-ES</td>
<td>Avg 0.77</td>
<td>St dev 0.2</td>
</tr>
<tr>
<td>REVAC++</td>
<td>0.85 0.24</td>
<td>12 %</td>
</tr>
<tr>
<td>SPOT</td>
<td>0.76 0.19</td>
<td>22 %</td>
</tr>
<tr>
<td>CEC-2005</td>
<td>0.97 0.32</td>
<td>-</td>
</tr>
</tbody>
</table>

- 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)

EA 1                                           EA 1

EA 2                                           EA 2

Performance

EA 1                                           EA 1

EA 2                                           EA 2

Performance