Parameter Control in Evolutionary Computing

A.E. Eiben
Computational Intelligence Group
Free University Amsterdam
http://www.cs.vu.nl/~gusz/

More info:
A.E. Eiben and J.E. Smith,
Introduction to Evolutionary Computing, Springer, 2003

Contents
- EC in a nutshell
- Motivation for parameter control
- Educative examples
- Classification and evaluation
- Case studies
- Summary and recommendations

The Basic EC Metaphor

The main evolutionary cycle

The two pillars of evolution

Motivation for parameter control 1

An EA has many strategy parameters, e.g.
- mutation operator and mutation rate
- crossover operator and crossover rate
- selection mechanism and selective pressure (e.g. tournament size)
- population size

Good parameter values facilitate good performance

Q1 How to find good parameter values ?
Motivation for parameter control 2

EA parameters are rigid (constant during a run) BUT an EA is a dynamic, adaptive process THUS optimal parameter values may vary during a run

Q2: How to vary parameter values?

Parameter tuning

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

Problems:
- users mistakes in settings can be sources of errors or suboptimal 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 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 strategy parameters?

Example

Task to solve:
- \( \min f(x_1, \ldots, x_n) \)
- \( l_i \leq x_i \leq u_i \) for \( i = 1, \ldots, n \) bounds
- \( g_i(x) \leq 0 \) for \( i = 1, \ldots, q \) inequality constraints
- \( h_i(x) = 0 \) for \( i = q+1, \ldots, m \) equality constraints

Algorithm:
- EA with real valued representation \( (x_1, \ldots, x_n) \)
- arithmetic averaging crossover
- Gaussian mutation: \( x' = x + N(0, \sigma) \)
  standard deviation \( \sigma \) is called mutation step size

Varying mutation step size: option 1

Replace the constant \( \sigma \) by a function \( \sigma(t) \)

\[
\sigma(t) = 1 - 0.9 \times \frac{t}{T}
\]

Features:
- changes in \( \sigma \) are independent from the search progress
- strong user control of \( \sigma \) by the above formula
- \( \sigma \) is fully predictable
- a given \( \sigma \) acts on all individuals of the population

Varying mutation step size: option 2

Replace the constant \( \sigma \) by a function \( \sigma(t) \) updated after every \( n \) steps by Rechenberg’s 1/5 success rule:

\[
\sigma(t) = \begin{cases} 
\sigma(t - 1) / c & \text{if } p_c > 1/5 \\
\sigma(t - 1) \cdot c & \text{if } p_c < 1/5 \\
\sigma(t - 1) & \text{otherwise}
\end{cases}
\]

\( p_c \) is the % of successful mutations, \( c \) is a parameter \((0.8 < c < 1)\)

Features:
- changes in \( \sigma \) are based on feedback from the search progress
- some user control of \( \sigma \) by the above formula
- \( \sigma \) is not predictable
- a given \( \sigma \) acts on all individuals of the population
Varying mutation step size: option 3

Assign a personal $\sigma$ to each individual
Incorporate this $\sigma$ into the chromosome: $(x_1, \ldots, x_n, \sigma)$
Apply variation operators to $x_i$'s and $\sigma$

$$\sigma' = \sigma \times e^{\frac{1}{(0, t)}}$$
$$x_i' = x_i + N(0, \sigma')$$

Features:
- changes in $\sigma$ are results of natural selection
- (almost) no user control of $\sigma$
- $\sigma$ is not predictable
- a given $\sigma$ acts on one individual

Varying mutation step size: option 4

Assign a personal $\sigma$ to each variable in each individual
Incorporate $\sigma$'s into the chromosomes: $(x_1, \ldots, x_n, \sigma_1, \ldots, \sigma_n)$
Apply variation operators to $x_i$'s and $\sigma_i$'s

$$\sigma'_i = \alpha \times e^{\frac{1}{(0, t)}}$$
$$x_i' = x_i + N(0, \sigma'_i)$$

Features:
- changes in $\sigma_i$ are results of natural selection
- (almost) no user control of $\sigma_i$
- $\sigma_i$ is not predictable
- a given $\sigma_i$ acts on all individuals of the population

Example cont’d

Constraints
- $g_i(x) \leq 0$ for $i = 1, \ldots, q$ inequality constraints
- $h_i(x) = 0$ for $i = q+1, \ldots, m$ equality constraints
are handled by penalties:

$$\text{eval}(x) = f(x) + W \times \text{penalty}(x)$$

where

$$\text{penalty}(x) = \sum_{j=1}^{m} \begin{cases} 1 & \text{for violated constraint} \\ 0 & \text{for satisfied constraint} \end{cases}$$

Varying penalty: option 1

Replace the constant $W$ by a function $W(t)$

$$W(t) = (C \times t)^\gamma$$

$0 \leq t \leq T$ is the current generation number

Features:
- changes in $W$ are independent from the search progress
- strong user control of $W$ by the above formula
- $W$ is fully predictable
- a given $W$ acts on all individuals of the population

Varying penalty: option 2

Replace the constant $W$ by $W(t)$ updated in each generation

$$W(t+1) = \begin{cases} \beta \times W(t) & \text{if last k champions all feasible} \\ \gamma \times W(t) & \text{if last k champions all infeasible} \\ W(t) & \text{otherwise} \end{cases}$$

$\beta < 1, \gamma > 1, \beta \times \gamma > 1$ champion: best of its generation

Features:
- changes in $W$ are based on feedback from the search progress
- some user control of $W$ by the above formula
- $W$ is not predictable
- a given $W$ acts on all individuals of the population

Varying penalty: option 3

Assign a personal $W$ to each individual
Incorporate this $W$ into the chromosome: $(x_1, \ldots, x_n, W)$
Apply variation operators to $x_i$'s and $W$

Alert:

$$\text{eval}((x, W)) = f(x) + W \times \text{penalty}(x)$$

while for mutation step sizes we had

$$\text{eval}((x, \sigma)) = f(x)$$

this option is thus sensitive for “cheating”
Lessons learned from examples

Various forms of parameter control can be distinguished by:

- primary features:
  - what component of the EA is changed
  - how the change is made

- secondary features:
  - level/scope of change
  - evidence/data backing up changes

What

Practically any EA component can be parameterized and thus controlled on-the-fly:

- representation
- evaluation function
- variation operators
- selection operator (parent or mating selection)
- replacement operator (survival or environmental selection)
- population (size, topology)

How

Three major types of parameter control:

- deterministic: some rule modifies strategy parameter without feedback from the search (based on some counter, typically time or number of search steps)
- adaptive: feedback rule, i.e., heuristic, based on some measure monitoring search progress
- self-adaptive: parameter values evolve along with solutions, encoded onto chromosomes they undergo variation and selection

Taxonomy

PARAMETER TUNING (before the run)
PARAMETER CONTRELL (during the run)

PARAMETER SETTING

DETERMINISTIC (time dependent)
ADAPTIVE (feedback from search)
SELF-ADAPTIVE (encoded in chromosomes)

Note on self-adaptation

Self-adaptation has an essential “logic” that is often not well understood (cf. α–first σ–last discussion)

\[
\sigma' = \sigma \times e^{\sigma (0, t)}
\]
\[
x' = x + N(0, \sigma')
\]

It is crucial that \( \alpha \) is used for mutating \( x \) (α–first)

Reason: a (new) value for \( \sigma \) is evaluated implicitly by evaluating the new \( x \) it creates

A value for \( \sigma \) only survives if its “hosting” \( x \) survives i.e., eval(x, σ) = f(x)

Scope/level

The parameter may take effect on different levels:

- environment (fitness function)
- population
- individual
- sub-individual

Note: given component (parameter) determines possibilities
Thus: scope/level is a derived or secondary feature in the classification scheme
Evidence/data

The parameter changes may be based on:
- time or nr. of evaluations (deterministic control)
- population statistics (adaptive control)
  - progress made
  - population diversity
  - success of an operator, etc.
- relative fitness (self-adaptive control)

Note: borders of this division coincide with the type ("how")
Thus: evidence/data is a secondary feature in the classification scheme

Evidence/data cont’d

In adaptive control, a further distinction can be made about the evidence used for changing the parameter
- Absolute evidence: predefined event triggers change, e.g. increase \( p_m \) by 10% if population diversity falls under threshold \( x \)
- Direction and magnitude of change is fixed
- Relative evidence: compare values through solutions created with them, e.g. increase \( p_m \) if top quality offspring came by high mut. Rates
- Direction and magnitude of change is not fixed

Refined taxonomy

- Combinations of types and evidences
  - Possible: +
  - Impossible: -

<table>
<thead>
<tr>
<th></th>
<th>Deterministic</th>
<th>Adaptive</th>
<th>Self-adaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Relative</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Scope/level

The parameter may take effect on different levels:
- environment (fitness function)
- population
- individual
- sub-individual

Note: given component (parameter) determines possibilities
Thus: scope/level is a derived or secondary feature in the classification scheme

Evaluation / Relevance

- Parameter control offers the possibility to use appropriate values in various stadia of the search
- Adaptive and self-adaptive parameter control
  - offer users "liberation" from parameter tuning
  - delegate parameter setting task to the evolutionary process
- EAs with (self-)adaptive parameter control are:
  - solving a given problem
  - calibrating themselves to the given problem (overhead)

Case study 1: adaptive crossover arity

- Eiben, Sprinkhuizen, Thijssen, ICEC’98
- Motivation:
  - multi-parent crossovers preferable on many functions
  - optimal arity to be fine-tuned
- Questions:
  - Adaptive GA able to identify better crossovers (arities)?
  - Adaptive GA better than non-adaptive?

Conclusions / answers to questions:
- "fast" adaptive GA could not identify best xovers
- adaptive GA is a good idea: no performance loss, no tuning
Case study 2: GA “without parameters”

- Baeck, Eiben, vd Vaart, PPSN 2000
- Research objectives:
  - Try new self-adaptive crossover rate mechanism
  - Study self-adaptive pm, pc, and adaptive pop. size separately
  - Study all these features together: “parameterless” GA
- Questions:
  - “Parameterless” GA feasible?
  - Self-adaptive crossover good?

Case study 3: adaptive fitness function (SAW-ing)

- Eiben, van Hemert ’99, Craenen, Eiben, Van Hemert 2003, etc.
- Problem:
  - Penalties for constraint violation should reflect how hard/easy constraints are
  - Needs knowledge, insights in problem (hard to obtain)
  - Easy/ness can change by time as search proceeds
- Solution:
  - EA adapts penalties, thus fitness function by simple scheme
  - After N steps increase penalty for violated constraints
- Findings:
  - SAW-ing can make EA superior (graph coloring, 3 SAT, CSP)
  - SAW-ing won’t find THE perfect weights, changing is the key

Outcomes

GA’s ranked by speed/mean best fitness

<table>
<thead>
<tr>
<th></th>
<th>TGA</th>
<th>SAM</th>
<th>SAX</th>
<th>AP</th>
<th>SAMX</th>
<th>APX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points</td>
<td>12.5</td>
<td>22.5</td>
<td>16.5</td>
<td>11</td>
<td>10.5</td>
<td>10.5</td>
</tr>
<tr>
<td>Rank</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

- GA “without parameters” is the best tested
- Self-adaptive xover no good (mutation even worse?)
- Population size matters most

Concluding remarks

- Parameter control offers the possibility to use appropriate values in various stadia of the search
- Adaptive and self-adaptive parameter control
  - Offer users “liberation” from parameter tuning
  - Delegate parameter setting task to the evolutionary process
  - The latter implies a double task for an EA: problem solving + self-calibrating (overhead)
- Parameter control has great promises:
  - Same performance less effort, better performance same effort
  - Less hand work for tuning
  - Reduces for need EA expertise for a new application, pushes back the complexity border

Some do’s and some don’t’s

- Do develop automated solutions, algorithms for creating deterministic schemes and adaptation heuristics
- Do concentrate more on selection, population, and fitness function parameters (beyond traditional wisdom)
- Do try to control more parameters together
- Do consider robustness of the method.
  - Regarding the “meta parameters” introduced in methods
  - If no. of parameters is increased by using (self)adaptation
- Do not try to self-adapt all parameters (not all fit the logic of self-adaptation)
- Do not validate your results on just an ad hoc test suite, do watch your methodology – cf. other tutorial here

and DO NOT PANIC

Slides of this presentation can be found at
http://www.cs.vu.nl/~gusz/