Paremeter Control in EC

- Motivation
- Educative examples
- Classification of paremeter control mechanisms
- “Real” examples
- Case studies
- Summary

More info:
Eiben, Hinterding, Michalewicz, Parameter Control in EAs,
IEEE Transactions on EC, vol 3, nr 2, july 1999, pp. 124-141

Motivation 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 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 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 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 $f(x_1,\ldots,x_n)$
- $L_i \le x_i \le U_i$ for i = 1,…,n bounds
- $g_i(x) \le 0$ for i = 1,…,q inequality constraints
- $h_i(x) = 0$ for i = q+1,…,m equality constraints

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

Varying mutation step size: option 1

Replace the constant σ by a function σ(t)

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

0 ≤ t ≤ T is the current generation number

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

Varying mutation step size: option 2

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

$$\sigma(t) = \begin{cases} \sigma(t-n)/c & \text{if } p_s > 1/5 \\ \sigma(t-n)\cdot c & \text{if } p_s < 1/5 \\ \sigma(t-n) & \text{otherwise} \end{cases}$$

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

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

Varying mutation step size: option 3

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

$$x'_i = x_i + N(0, \sigma')$$

$$\sigma' = \sigma \times e^{N(0,\tau)}$$

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

Varying mutation step size: option 4

Assign a personal σ to each variable in each individual
Incorporate σ's into the chromosomes: (x_1, …, x_n, σ_1, …, σ_n)
Apply variation operators to x_i's and σ_i's

$$\sigma'_i = \sigma_i \times e^{N(0,\tau)}$$

$$x'_i = x_i + N(0, \sigma'_i)$$

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

Example cont'd

Constraints
- $g_i(x) \le 0$ for i = 1,…,q inequality constraints
- $h_i(x) = 0$ for i = q+1,…,m equality constraints

are handled by penalties:

eval(x) = f(x) + W × penalty(x)

where

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

Varying penalty: option 1

Replace the constant W by a function W(t)

$$W(t) = (C \times t)^{\alpha}$$

0 ≤ t ≤ 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

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

$\beta < 1, \gamma > 1, \beta \times \gamma \neq 1$ 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:

eval ((x, W)) = *f* (x) + *W* × *penalty(x)*

while for mutation step sizes we had

eval ((x, σ)) = *f* (x)

this option is thus sensitive "cheating" ⇒ makes no sense

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)
- adaptive: feedback rule based on some measure monitoring search progress
- self-adaptative: parameter values evolve along with solutions; encoded onto chromosomes they undergo variation and selection

Scope/level

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

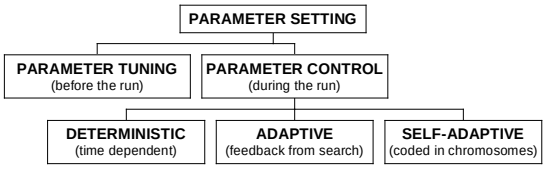
Note: given component (parameter) determines possibilities
Thus: scope/level is a derived or secondary feature in the classification scheme

Evidence/data

The parameter changes may be based on:
- time or nr. of evaluations (deterministic control)
- population statistics (adaptive control)
  - progress made
  - population diversity
  - gene distribution, 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

Taxonomy



Evaluation / Relevance

- Parameter control offers the possibility to use appropriate values in variuos 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)

"Real" examples

- Representation: delta coding (Whitley et al 1991)
- Evaluation function: SAW-ing (Eiben et al. 1998)
- Selection: PRSA (Mahfoud & Goldberg 1992)
- Mutation:
  - Deterministic p_m (Hesser & Männer 1991)
  - Deterministic vs. self-adaptive p_m (Bäck 1992)
- Crossover:
  - Adaptive pc (Davis 1989)
  - Self-adaptive choice of xovers (Spears 1995)
- Population size: GAVaPS (Arabas and Michalewicz 1994)

Representation: Delta coding

- For good balance between fast search and sustaining diversity
- Deterministic adjustment of nr. of bits coding object values
- Diversity:Hamming distance between best-worst chromosome
- Method:
  - k bits per object variable
  - Run GA till HD ≤ 1 ()
  - Save best solution as *PARTIAL*
  - New coding: k-1 bits as δ to the object value in *PARTIAL* & one sign bit
  - Run GA with δ encoded population till HD ≤ 1
  - Retreive new *PARTIAL* (best) and go to 3
  - Stop if global termination conditon is fulfilled
- Nr. of bits for can be increased if the same *PARTIAL* is found

Evaluation function: SAW-ing

- Applied for constraint satsifaction problems graph 3-coloring, satisfiability, etc.
- Evaluation function based on penalties, e.g.:

$$f_e(\bar{s}) = \sum_{i=1}^{m} w_i \cdot \chi(\bar{s}, c_i)$$

- Where c_i are constraints, χ = 1/0 for violation/satisfaction
- Adaptative control by Stepwise Adaptation of Weights (SAW) mechanism: weights of unsatisfied constraints in the best chromosome are raised periodically during a run
- Rationale: EA finds out what is difficult, heavier penalty makes EA "concentrate" on hard parts

Replacement (survivor selection)

- Parallel Recombinative Simulated Annealing (PRSA)
  - Initialize population
  - Initialize temerature T
  - Generate new offspring by
    - random parent selection
    - Crossover & mutation
  - Select parent to survive if

$$\frac{1}{1+\frac{\exp((f(parent)-f(child))}{T}} > random[0,1)$$

  - Adjust T
  - Stop if global termination condition is fulfilled
- A number of generations is performed with a given T
- T is decrease by deterministic schedule, e.g., T' = T x 0.95

Deterministic mutation rates

- Theoretically derived optimal schedules to change p_m for the counting-ones function

$$p_m(t) = \sqrt{\frac{a}{b}} \cdot \frac{\exp\left(\frac{-ct}{2}\right)}{p \cdot \sqrt{L}}$$

- Where:
  - a,b,c, are constants
  - P is the population size
  - L is the chromosme length
  - T is the time (generation counter)
- Effect: if t increases optimal p_m decreases

Deterministic vs. self-adaptive p_m

- Bäck theoretical analysis for the counting-ones function: optimal p_m decreases if fitness (nr. of correct bits) increases
- Self-adaptive p_m by extra bits in chrom. to encode p_m
- Mutation mechanism:
  - Decode the extra bits to get p'_m
  - Mutate in the extra bits with this p'_m
  - Decode the new values of extra bits to get p_m
  - Mutate the rest of the chromosome with this p_m
- Experiments:
  - Observed p_m followed optimal schedule closely
  - GA with optimal p_m schedule and s-a p_m behave similarly

Adaptive crossover rates

- Use more xover ops simultaneously with own $p_c(xo_i)$
- Goal is to find "optimal" p_c's
- $v = (v_1,\ldots,v_n) = (p_c(xo_1), \ldots, p_c(xo_n))$
- $d = (d_1,\ldots,d_n)$ local deltas
  - d_i: advantage of child wrt parent created by xo_i
  - updated after each xover application
- Adaptation mechanism sketch:
  - Redistribute 15% of probabilities after K generations
  - Normalize d so that it totals 15
  - V_i (new) = v_i (old) + d_i (normalized)

Self-adaptive choice of crossovers

- Two crossovers: uniform and 2-point
- Add one extra bit to chromosomes to indicate crossover to be used:
  - Both parents have xover bit 1: 2-point xover
  - Both parents have xover bit 0: uniform xover
  - Parents disagree: random choice
- Experiments:
  - s-a did not improve GA
  - the usage of two xovers did

Adaptive population size

- GAVaPS system:
  - no survivor selection or replacement
  - individuals get maximum age at birth based on fitness
  - age of indiv's grows at each generation
  - if age reaches individuals max lifetime, it is removed from pop
- Experiments:
  - Population size
    - increases in the beginning
    - Decreases after a point
- GA gets better on some problems

Case study 1: aptive crossover arity

- Eiben, Sprinkhuizen, Thijssen, ICEC'97
- 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?

Approach

- constant size population divided into variable size subpopulations
- each subpopulation uses one xover and evolves on its own
- periodic migration of individuals between subpops
  - good subpopulations grow
  - migration applied frequently
- periodic redistribution of individuals between subpops
  - bad subpopulations regain size
  - redistribution applies seldomly (gives 2nd chance to xover)

Experiments

- steady state GA, tourn. sel + worst deletion
- crossovers:
  - n-point (n=2,3,4,5,6)
  - diagonal with k parents (k=2,3,4,5,6)
- 10 subpops of size 50
- 7 test functions: onemax, twin peaks, trap, trap-d, plateau, plateau-d, royal road
- 3 test series:
  - A: 1 xover before replacement (different generational gaps)
  - B: 1 one-child xover before replacement (equal gen. gaps)
  - C: diff. Nr. of xovers before replacement (equal gen. gaps)
- control experiments: usual GAs with above xovers

Outcomes

- Control experiments: more parents xovers better
- Series A: better xovers get larger subpopulations
- Series B,C: random variations in subpopulation sizes
- Best usual GA ≅ adpative GA

Conclusions / answers to questions:

- "fair" adaptive GA (setup A) could not identify best xovers
- adapive GA is a good idea: no performance loss, no tuning

Case study 2: "parameterless" GA

- Bäck, Eiben, vd Vaart, PPSN 2000
- Research objectives:
  - try new self-adaptive crossover rate mechanism
  - study self-adaptive p_m, p_c, and adaptive pop. size separately
  - study all these features together: "parameterless" GA
- Questions:
  - "Parameterless" GA feasible?
  - Self-adaptive crossover good?

Approach

- self-adaptive p_m, a la Bäck'92
- self-adaptive p_c:
  - individual p_c compared with random threshold
  - both OK: uniform xover
  - both not OK: both mutated to generate 2 offspring
  - 1 OK, 1 not: not OK parent mutates, OK parent waits
- adaptive pop. size, a la Arabas, Michalewicz'94:
  - no survivor selection or replacement
  - individuals get maximum age at birth based on fitness
  - age of indiv's grows at each generation
  - if age reaches individuals max lifetime, it is removed from pop.

Experiments

- Test suite: sphere, Rosenbrock, Ackley, Rastrigin, deceptive
- GA's:
  - Traditional: T
  - self-adaptive mutation alone: SAM
  - self-adaptive xover alone: SAX
  - adaptive population size alone: AP
  - all in combination: SAMXP

Outcomes

GAs ranked by speed/mean best fitness

	TGA	SAM	SAX	AP	SAMXP
Points	12.5	22.5	18.5	11	10.5
Rank	**3**	**5**	**4**	**2**	**1**

- "Parameterless" GA is the best tested
- Self-adaptive xover no good (mutation even worse!?)
- Population size matters most

Summary

- Parameter control has great promises:
  - same or better performance
  - less hand-work for tuning
  - self-adaptivity: let the EA do the work
- Paremeter control has caveats:
  - determinstic: scheme is still hand-made
  - traditional wisdoms may be misleading on "what"
  - no general guidelines for how to do it
- MORE RESEARCH IS NEEDED