Parameter-less Genetic Algorithm: Motivation

Motivation – to make the EA an algorithm that is robust, efficient and easy-to-use.

- Typically, the EAs require quite a bit of expertise in order to make them work well for a particular application.
- The user is not interested in tuning and fiddling the EA's parameters for each single application. He would be happy if he could get around somehow.

Parameter-less GA [Harik99] eliminates the following parameters when applying the algorithm to a particular problem:

- **population size**,
- **selection rate** s – the amount of bias towards better individuals; usually expressed by a ratio of sampling rates of individuals with the best and average fitness in the population.
- **crossover probability** p_c – the amount of mixing.

Parameter-less GA: Getting Rid of Selection Rate and Crossover Prob.

- **Schema** S – a template, which defines set of solutions from the search space with certain specific similarities. The schema consists of 0s, 1s and wildcard symbols * (any value).

  Schema properties – defining length, order, and fitness.

  Example: schema S ={11*0*} covers strings 11000, 11001, 11100, and 11101
- A **simplified growth ratio of schema S at generation** t, considering only $\phi(S,t)$ the effect of the selection operator on schema S at generation t and $\epsilon(S,t)$ the disruption factor on schema S due to the crossover operator is

$$\phi(S,t)\cdot[1-\epsilon(S,t)]$$

  Under the conservative hypothesis that a schema is destroyed during the crossover we get

$$s(1-p_c)$$

- **Schema theorem**: Short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm.

We just need to ensure that the GA will obey the schema theorem and the growth ratio of building blocks will be greater than 1.

- **Setting $s=4$ and $p_c=0.5$ gives a net growth ratio of 2.**

Standard GA: Population Sizing

Whether the building blocks will mix efficiently in a single optimal solution is now a matter of having an adequate population size

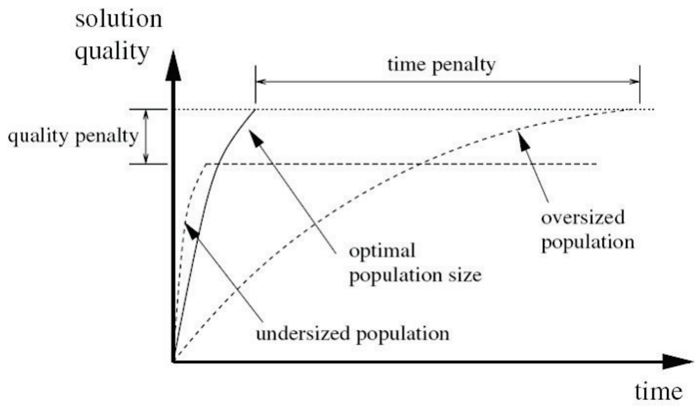
There are theoretical models for population sizing concluding that the **population size** should be **proportional to problem length and building blocks's signal-to-noise ratios**.

- For compact building blocks the required population size is reasonable.
- If the building blocks are not compact, then the population sizing requirements can be extremely large.

The models are difficult to apply in practice because they rely on parameters that are usually unknown and are hard to estimate for real world problems.

Effects of improperly set population size

- Too small population size $\Rightarrow$ quality penalty: The GA will converge to sub-optimal solutions.
- Too large population size $\Rightarrow$ time penalty: The GA will spend unnecessary computational resources.



Parameter-less GA: Getting Rid of Population Sizing

The idea is to let the algorithm do the experimentation with population sizes automatically by establishing a race among multiple populations of various sizes in a single GA's run:

- Each population $k > 1$ is twice as large as the population $k - 1$.
- The smaller populations are given more function evaluations, thus the different populations are at different stages of evolution.
- As time goes on, The smaller populations are eliminated and larger populations are created automatically based on observed average average fitness of the populations.

  If at any point in time, a larger population has an average fitness greater than that of a smaller population, then the smaller population is destroyed.

  – The rational for doing this is that in this situation it is very unlikely that the smaller population will produce a fitter individual than the larger one.
- The coordination of the array of populations is implemented with a counter of base m, which determines the proportion of fitness evaluations given to each of the simulated runs.

Parameter-less GA: Coordination of Populations

- At each generation, the counter of base $m = 4$ is incremented, and the position of the most significant digit changed during the increment operation is noted. That position indicates which population should be run.
- Since each population k is on the one hand half as large as the population $k+1$ and on the other hand is allowed 4 times more generations than population $k+1$ the population k is allowed to spend twice the number of fitness evaluations of population $k+1$.
- When some population converges or its average fitness is less than the average fitness of a larger population (due to a genetic drift – a population does not converges due to an insufficient selection pressure), it is removed together with all of the smaller populations.

  The counter is reset.

Counter base 4	Action
0	run 1 generation of population 1
1	run 1 generation of population 1
2	run 1 generation of population 1
3	run 1 generation of population 1
10	run 1 generation of population 2
11	run 1 generation of population 1
12	run 1 generation of population 1
13	run 1 generation of population 1
20	run 1 generation of population 2
21	run 1 generation of population 1
22	run 1 generation of population 1
23	run 1 generation of population 1
30	run 1 generation of population 2
31	run 1 generation of population 1
32	run 1 generation of population 1
33	run 1 generation of population 1
100	run 1 generation of population 3
101	run 1 generation of population 1
⋮	⋮