Biologically-inspired algorithms and models

1. Evolutionary algorithms and their mechanisms

Maciej Komosinski

Reminder from earlier studies (and, possibly, a supplement) #1

- What are the differences between *random search* and *random walk*?
- What two variants of the local search algorithm do you know?
- Is it known that one of them is better (ultimately produces better results – this question concerns quality, not running time)?
- What are the ways of intensification and diversification in *Tabu Search* and *Simulated Annealing*?

Reminder from earlier studies (and, possibly, a supplement) #2

- What crossover operators do you know in evolutionary/genetic algorithms?
- What selection techniques in evolutionary algorithms do you know?
- What methods do you know of increasing the diversity of solutions during evolution?
- What is the difference between the *steady state* and the *generational replacement* architectures?
- What ways do you know to deal with constraints?

Two scenarios of using optimization algorithms

Applications of optimization algorithms can be roughly divided into interactive ones (*on-line*) and batch ones (*off-line*).

In the off-line approach, we are interested in the best solution found during the entire running time of the algorithm. In the interactive ("on-line") approach, we are interested in making a given optimization algorithm yield results as good as possible all the time. To evaluate the behavior of the optimization algorithm in the *on-line* and *off-line* scenarios, De Jong proposed specific indicators [Gol02, pp. 107, 110]; come up with the two simple ones.

Variants of evolutionary algorithms

- Genetic algorithms (GA)
- Evolutionary strategies (ES)
- Evolutionary programming (EP)
- Genetic programming (GP)
- Classifier systems (CFS) and genetics-based machine learning (GBML)
- Coevolutionary architectures
- ...

GA: John Holland (1973, 1975), David Goldberg (1989)
EP: Lawrence Fogel (1963), David Fogel (1992)
ES: Ingo Rechenberg (1973), Thomas Bäck (1996)
GP: John Koza (1992)

Reminder: algorithm structure and parameters

Main loop:

$t := 0$
initialize $P(t)$
evaluate $P(t)$
while (**not** stopping-condition)
{
    $t := t + 1$
    select $P(t)$ from $P(t-1)$
    modify $P(t)$
    evaluate $P(t)$
}

Reminder: algorithm structure and parameters

Parameters:

- population size *POPSIZE*
- probability of crossing-over *PXOVER*
- probability of mutation *PMUT*

- choosing the stopping criterion
- choosing the selection mechanism (positive and possibly negative)
- adjusting parameter values of the selection mechanism

The role of selection

- What is selection needed for in an evolutionary algorithm?
- What would happen if selection were purely random?
- What would happen if selection were deterministic and gave every individual an equal chance?
- Can the strength of the selection pressure be expressed as a number?

The role of nondeterminism in algorithmics

Discussion on the types of nondeterminism and the role of nondeterminism in algorithms and in computer science.

The role of nondeterminism in algorithmics

Discussion on the types of nondeterminism and the role of nondeterminism in algorithms and in computer science.

Consider four generators of random sequences: one that is based on consecutive numbers and *modulo*, poor pseudo-random, good pseudo-random, and truly random.

The role of nondeterminism in algorithmics

Discussion on the types of nondeterminism and the role of nondeterminism in algorithms and in computer science.

Consider four generators of random sequences: one that is based on consecutive numbers and *modulo*, poor pseudo-random, good pseudo-random, and truly random.

```
int random1(int n)
    static int c=-1
    c++
    c=c%n
    return c
```

```
int random2(int n)
    static int c=0
    c=(c·...+...)%n
    return c
```

```
int random3(int n)
    //very
    //complicated
    //logic
    return ...
```

```
int random4(int n)
    return ☼%n
```

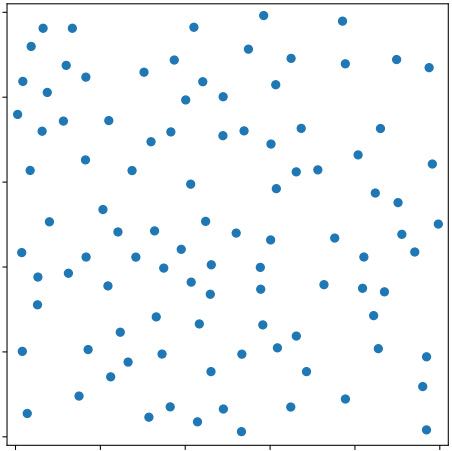
The role of nondeterminism in algorithmics

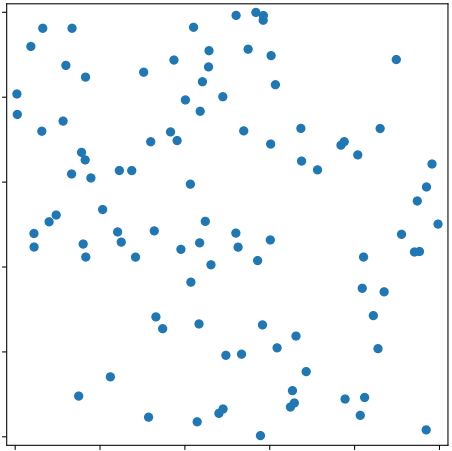
Discussion on the types of nondeterminism and the role of nondeterminism in algorithms and in computer science.

Consider four generators of random sequences: one that is based on consecutive numbers and *modulo*, poor pseudo-random, good pseudo-random, and truly random.

- an example of supplementation (e.g., 4 substances every 2 days each, interactions are unknown)
- an example of giving gifts
- an example of signal dithering (audio, video, ...) and rounding
- and finally, an example of an algorithm...

The role of nondeterminism in algorithmics – „randomness”





The role of nondeterminism in algorithmics – conclusions

Discussion on the types of nondeterminism and the role of nondeterminism in algorithms and in computer science.

The role of nondeterminism in algorithmics – conclusions

Discussion on the types of nondeterminism and the role of nondeterminism in algorithms and in computer science.

- you are inventing some algorithm, for example an optimization algorithm (or some other). What would prompt you to use the random() function in it?

The role of nondeterminism in algorithmics – conclusions

Discussion on the types of nondeterminism and the role of nondeterminism in algorithms and in computer science.

- you are inventing some algorithm, for example an optimization algorithm (or some other). What would prompt you to use the random() function in it?
  - examples: *Greedy* and *Steepest* with multiple neighbors with the same quality, the induction of decision trees with multiple attributes with equal entropy, ...

The role of nondeterminism in algorithmics – conclusions

Discussion on the types of nondeterminism and the role of nondeterminism in algorithms and in computer science.

- you are inventing some algorithm, for example an optimization algorithm (or some other). What would prompt you to use the random() function in it?
  - examples: *Greedy* and *Steepest* with multiple neighbors with the same quality, the induction of decision trees with multiple attributes with equal entropy, ...
  - try to completely "determinize" SA. Will there be any negative consequences?

The role of nondeterminism in algorithmics – conclusions

Discussion on the types of nondeterminism and the role of nondeterminism in algorithms and in computer science.

- you are inventing some algorithm, for example an optimization algorithm (or some other). What would prompt you to use the random() function in it?
  - examples: *Greedy* and *Steepest* with multiple neighbors with the same quality, the induction of decision trees with multiple attributes with equal entropy, ...
  - try to completely "determinize" SA. Will there be any negative consequences?
- in what situations is "unrestricted", full randomness beneficial?

The role of nondeterminism in algorithmics – conclusions

Discussion on the types of nondeterminism and the role of nondeterminism in algorithms and in computer science.

- you are inventing some algorithm, for example an optimization algorithm (or some other). What would prompt you to use the random() function in it?
  - examples: *Greedy* and *Steepest* with multiple neighbors with the same quality, the induction of decision trees with multiple attributes with equal entropy, ...
  - try to completely "determinize" SA. Will there be any negative consequences?
- in what situations is "unrestricted", full randomness beneficial?
- why are we worried about ...1080777777696980348..., but not about ...1080735172696980348... ?

The role of nondeterminism in algorithmics – conclusions

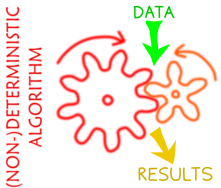
Discussion on the types of nondeterminism and the role of nondeterminism in algorithms and in computer science.

- you are inventing some algorithm, for example an optimization algorithm (or some other). What would prompt you to use the random() function in it?
  - examples: *Greedy* and *Steepest* with multiple neighbors with the same quality, the induction of decision trees with multiple attributes with equal entropy, ...
  - try to completely "determinize" SA. Will there be any negative consequences?
- in what situations is "unrestricted", full randomness beneficial?
- why are we worried about ...1080777777696980348..., but not about ...1080735172696980348... ?

The role of nondeterminism in algorithmics – conclusions

Discussion on the types of nondeterminism and the role of nondeterminism in algorithms and in computer science.

- you are inventing some algorithm, for example an optimization algorithm (or some other). What would prompt you to use the random() function in it?
  - examples: *Greedy* and *Steepest* with multiple neighbors with the same quality, the induction of decision trees with multiple attributes with equal entropy, ...
  - try to completely "determinize" SA. Will there be any negative consequences?
- in what situations is "unrestricted", full randomness beneficial?
- why are we worried about ...1080777777696980348..., but not about ...1080735172696980348... ? The first one is straight from a RNG, the second one is "corrected" (is it?)

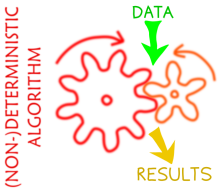


- when is true randomness preferable to good pseudo-randomness?

The role of nondeterminism in algorithmics – conclusions

Discussion on the types of nondeterminism and the role of nondeterminism in algorithms and in computer science.

- you are inventing some algorithm, for example an optimization algorithm (or some other). What would prompt you to use the random() function in it?
  - examples: *Greedy* and *Steepest* with multiple neighbors with the same quality, the induction of decision trees with multiple attributes with equal entropy, ...
  - try to completely "determinize" SA. Will there be any negative consequences?
- in what situations is "unrestricted", full randomness beneficial?
- why are we worried about ...1080777777696980348..., but not about ...1080735172696980348... ? The first one is straight from a RNG, the second one is "corrected" (is it?)



- when is true randomness preferable to good pseudo-randomness?
- and finally: should he get this funding and why??

Selection – popular techniques

f_i – the fitness of i-th individual ($i = 1..POPSIZE$)
e_i – the number of its expected copies in the new (consecutive) population,
$e_i = POPSIZE \cdot f_i / \sum f_j$

- Fitness proportionate random selection with replacement, commonly called the roulette wheel technique: individuals are assigned fields on the roulette wheel, the sizes of which are proportional to their fitness f_i. Then the roulette wheel is spun $POPSIZE$ times, selecting the drawn individual. The same principle, but better properties: *stochastic universal sampling* method*.
- Stochastic remainder selection without replacement: each individual gets as many copies in the new population as the integer part of its e_i. The remaining free places are filled by randomly deciding, for each individual with the probability being the fractional part of its e_i, whether it should go to the new population. Example: 4 individuals, $\boldsymbol{f} = [1,3,5,6]$.
- Selection according to random tournaments (parameter: k – tournament size). A more careful variant of this technique ensures that each individual participates in the same number of tournaments.

*https://en.wikipedia.org/wiki/Stochastic_universal_sampling

Selection – other techniques

- Deterministic remainder-based selection: each individual gets as many copies in the new population as the integer part of its e_i, and the remaining free places in the population are filled in order of decreasing fractional parts of individual e_i.
- Stochastic remainder selection with replacement: each individual gets as many copies in the new population as the integer part of its expected number of copies (e_i). The remaining places are filled according to the roulette principle proportionally to the fractional part of e_i.
- Ordinal selection: individuals are assigned integer ranks that correspond to their position in ranking, from best to worst. The selection is based on the probability function that depends not on raw fitness values, but on individual positions in the ranking. Various probability functions are used – linear and non-linear, and the parameters of these functions allow one to adjust selective pressure.

Exercise: classify these 6 techniques into two categories – depending on how they use the values of the fitness function.

Selection – additional properties

- Elitism (elitist model): fulfills the expectation that the selection process should not cause the loss of the best individual found so far. If such an individual does not find its way to the next population in a natural way (resulting from the selection method used), it is included in it and thus the information about the best solution so far is always preserved.
- Crowding factor model: similar to nature, where species filling the ecological niche must fight for limited resources – in the crowding model, new individuals replace old individuals (from the previous population) taking into account their similarities, i.e., new individuals take the place of the old individuals most similar to them. The crowding factor (a parameter) affects the way individuals are replaced [DJ75; Mah92].

Meta-schemes of selection

In the following selection methods, parts of the population (subpopulations) can be independently processed – these methods can therefore also act as a distribution and parallelization scheme for evolution.

- Island model: a population is split into subpopulations in which the chosen selection scheme operates (for example tournament, roulette or other). Evolution proceeds on each island independently, with periodic migration of some genotypes between islands. What effects does this have?
- Convection selection: unlike in the traditional island model, the division into subpopulations follows the similarity of the value of the objective function of solutions. What effects does this have?

Meta-schemes of selection

In the following selection methods, parts of the population (subpopulations) can be independently processed – these methods can therefore also act as a distribution and parallelization scheme for evolution.

- Island model: a population is split into subpopulations in which the chosen selection scheme operates (for example tournament, roulette or other). Evolution proceeds on each island independently, with periodic migration of some genotypes between islands. This model increases exploration capabilities.
- Convection selection: unlike in the traditional island model, the division into subpopulations follows the similarity of the value of the objective function of solutions. Convection selection improves the exploration ability of an EA by properly balancing selective pressure [KU17; KM18]. The way this selection method works is illustrated in animations here.
  Discussion: how will this meta-scheme perform in *EqualNumber* and *EqualWidth* variants [KM18, Fig. 3] when the selection in subpopulations is random (e.g., tournament size $= 1$), compared to the island model and to the standard, single-population EA with random selection?

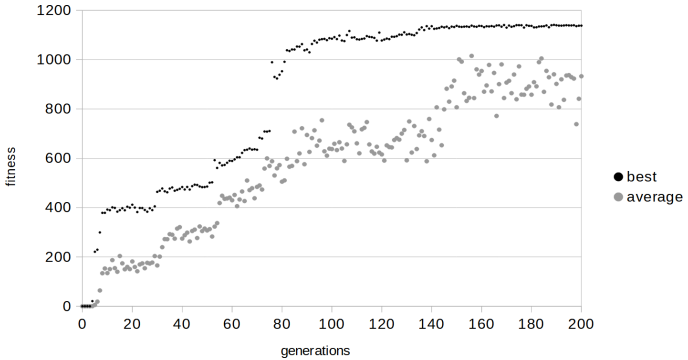
Negative selection

Sometimes (depending on the adopted GA architecture), in addition to using a positive selection, it is also necessary to employ a negative selection.

Its role is to make room in the population for new genotypes – negative selection decides which genotypes to remove from the population. Similar mechanisms as for the positive selection can be used; two examples of naive methods are deleting the worst genotype and a random one.

You ran an evolutionary algorithm with roulette selection

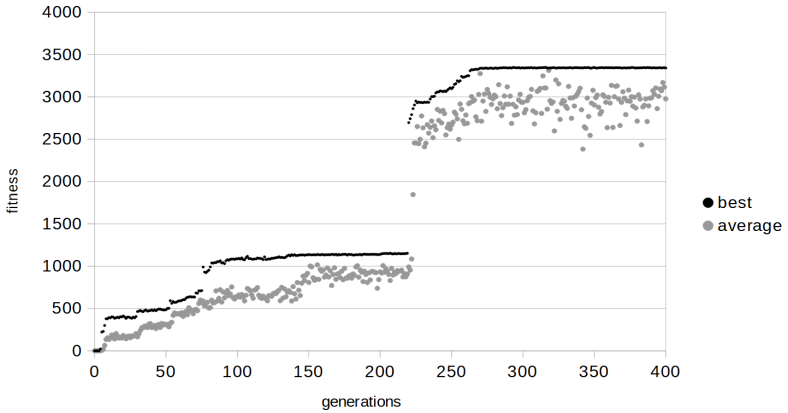
and got this outcome:



Question 1: Is it a good moment to stop optimization?

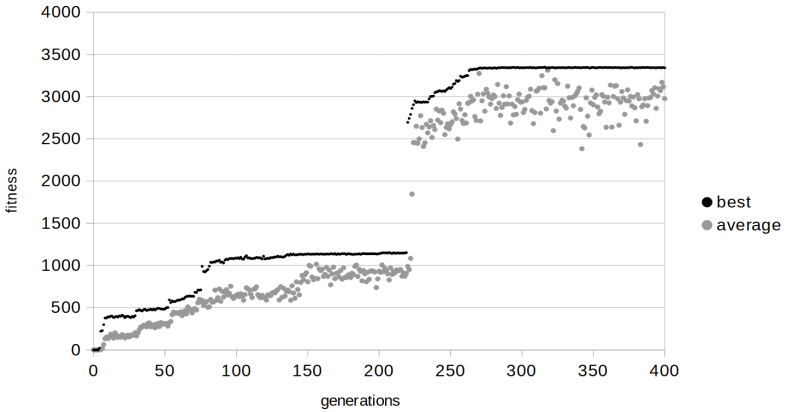
Question 2. Is it correct to conclude from this run that the diminishing (and eventually zero) improvements are due to the fact that it is becoming increasingly difficult to find better solutions in the surroundings of the population?

Answering question 1: the future is unknown!



What other criteria (apart from stabilized fitness) can be used to develop a better stopping condition and avoid the situation above?

Answering question 1: the future is unknown!



Measures of solution diversity in the population and the relationship with selection pressure and with the potential for operators to change solutions. Possibly also some knowledge (even a cursory one) of the fitness landscape.

Scaling

Discussion: where does the need for scaling come from? Analysis of the behavior of roulette selection at the beginning of evolution and in later stages (cf. previous plot).

- Linear scaling: $f' = af + b$. The coefficients a and b are adjusted so that the fitness f' of the best individual is a given multiple (for example $2\times$) of the fitness of the "average" individual. After scaling, negative fitness values may appear – you can then reset them to zero or perform another linear transformation.
- Power law scaling: $f' = f^k$. The coefficient k depends on the specific optimization task, and thus this method is not particularly useful.
- σ-truncation scaling (truncation at the level dependent on the standard deviation). Fitness values depend not only on the values of the original fitness of individuals, but also on the distribution of fitness in the population. The average fitness of the population μ and the standard deviation of the fitness in the population σ are determined, and the fitness (in the case of maximization) becomes $f' = f - (\mu - c \cdot \sigma)$. Negative values of f' are replaced by zero. The coefficient c determines the selection pressure: the larger the c, the lower the pressure.

σ-truncation scaling – example

For example 10 individuals with fitness 57, 57, 58, 58, 59, 59, 61, 62, 62, 65. The average, μ, is 59.8, and the standard deviation $\sigma = 2.6$.

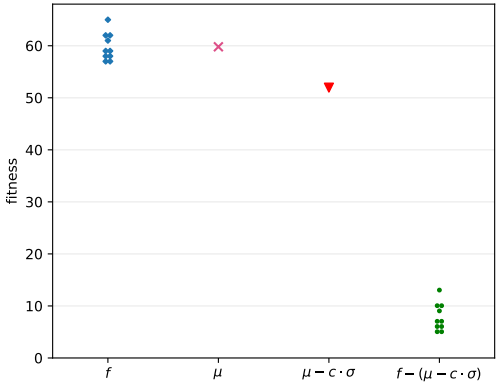


Figure: Scaling plot (truncation at the level dependent on the standard deviation) for coefficient $c = 3$.

A calculation (and interpretation) exercise

Before scaling, the probability of selecting the i-th individual, p_i, is the quotient of the individual's fitness value (f_i) and the total fitness of all individuals ($j = 1..n$).

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j} = \frac{f_i}{n\mu}$$

How much will p'_i be if $f'_i = f_i - (\mu - c\sigma)$?

Express p'_i as a function of p_i and of other possibly easily interpretable expressions.

A calculation (and interpretation) exercise – solution

$$p'_i = \frac{f_i - \mu}{nc\sigma} + \frac{1}{n} \quad = \quad \frac{\mu}{c\sigma}\left(p_i - \frac{1}{n}\right) + \frac{1}{n}$$

Interpret the above formula and relate it to the preceding scaling plot.

- What is $\frac{1}{n}$?
- Consider the situation where μ is small and σ is large (the beginning of evolution) and the opposite, when evolution is already at an advanced stage.
- Does this scaling have a point of neutrality where it has no effect?

A calculation (and interpretation) exercise – solution

$$p'_i = \frac{f_i - \mu}{nc\sigma} + \frac{1}{n} \quad = \quad \frac{\mu}{c\sigma}\left(p_i - \frac{1}{n}\right) + \frac{1}{n}$$

Interpret the above formula and relate it to the preceding scaling plot.

- What is $\frac{1}{n}$?
- Consider the situation where μ is small and σ is large (the beginning of evolution) and the opposite, when evolution is already at an advanced stage.
- Does this scaling have a point of neutrality where it has no effect?

Reminder: what was one of the ideas to represent "selective pressure" as a single value?

The rationale behind fitness scaling

The scaling mechanism helps keep the selection pressure constant throughout the entire course of evolution, regardless of the properties of the function being optimized. Controlling the selection pressure is very important – without it, efficient optimization is impossible.

Employing scaling is reasonable if the selection method that is used...

The rationale behind fitness scaling

The scaling mechanism helps keep the selection pressure constant throughout the entire course of evolution, regardless of the properties of the function being optimized. Controlling the selection pressure is very important – without it, efficient optimization is impossible.

Employing scaling is reasonable if the selection method that is used depends on the ratios of fitness values (and, for example, tournament and ranking selection do not).

Average fitness as the “neutral point” of the population

Is the average the best option?

Average fitness as the "neutral point" of the population

Is the average the best option?

Another (better than σ-truncation?) scaling function: the probability of selecting an individual,

$$p'_i = \frac{1}{1 + \exp(\frac{f_i - M}{\sigma})}$$

f_i – the fitness value of an individual,
M – the **median** value of fitness of all individuals in the population,
σ – the standard deviation of fitness values in the population.

For maximization, reverse the sign of the argument of exp().

- Minimal influence of extremely good/poor individuals (throughout the entire course of evolution), so the problem of premature convergence is eliminated. Minimal – because of the median (the extreme individual will only cause a moderate increase in σ).
- Effective separation of moderately fit individuals into two groups (above and below the median).
- At the beginning of evolution – usually large σ, so a low diversity of p'_i of individuals. At the end – convergence thus small σ, so better individuals promoted much more strongly.

Sample fitness distribution in a population during evolution

See [KU17, the top plot in Fig. 9].

How can you explain stable horizontal lines (groups of individuals with similar, specific values of the objective function) over a long period of time? Try to offer at least two possible reasons.

Crossover

Reminder: single-point, multiple-point, uniform, ...

Discussion: is crossover required in an evolutionary algorithm?

Crossover

Reminder: single-point, multiple-point, uniform, ...

Discussion: is crossover required in an evolutionary algorithm?

Discussion: is EA without crossover the same as multiple independently run LS?

Crossover

Reminder: single-point, multiple-point, uniform, ...

Discussion: is crossover required in an evolutionary algorithm?

Discussion: is EA without crossover the same as multiple independently run LS?

Discussion: mutation and crossover vs. exploration and exploitation.

Crossover – advanced techniques

- *Shuffle crossover*: an additional mechanism [ECS89] used for some types of crossover. It involves randomly reordering the locations of bits in the parent sequences (in both of them identically), performing the crossover, and restoring the original bit order in the descendants (think about why such a procedure is used and for which crossover operators it makes sense?)
- Adaptive crossover: the genotype, in addition to the bit values of the solution, can store the information about crossover cut points. Poor solutions disappear (along with the information about crossover points), while good solutions (and the information about beneficial crossover points) persist and improve. In this way, cut points are undergoing evolution along with the population of solutions. An extension of this idea is recording the fitness (quality) of various genetic operators and selecting an operator taking into account its fitness.
- Crossover with multiple ancestors: the recombination concerns genes drawn from the gene pool of selected parents, including the case of the so-called "orgies", in which the descendant solution may have more than two parents [EKK95; TYH99; Ahm15; AA19] (consider what effect the number of parents has?)

Discussion: is mutation required in an evolutionary algorithm?

Mutation

Discussion: is mutation required in an evolutionary algorithm?

As a general rule, it is advantageous for an individual after mutation to be similar to its parent, and for each mutation to result in changes of similar magnitude. Is this the case when we use the standard binary encoding of numbers? No. Mutations will occur sometimes in the less significant bits, sometimes in the more significant bits, causing sometimes small changes, sometimes huge ones.

Do you recall any special code that could help here? A code related to neighboring sequences differing in the value of only one bit. Would such a code help here?

Mutation – smart encoding?

If we use the Gray code, in which adjacent numbers differ by only one bit (the Hamming distance), then the adjacent numbers will always be reachable by a single mutation. But will this really help?

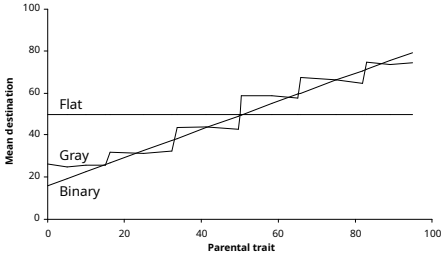


Figure: Severe discontinuities of the mutation operator have a continuous influence (a *bias*) on evolution [Bul99, p. 69]. Flat: mutation is simply randomly drawing a gene value from the allowed range. The Gray code does not eliminate the disadvantages of mutating encoded numeric values.

Mutation – smart encoding?

If we use the Gray code, in which adjacent numbers differ by only one bit (the Hamming distance), then the adjacent numbers will always be reachable by a single mutation. But will this really help?

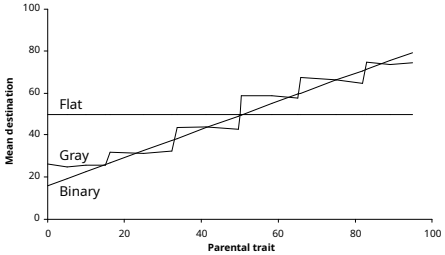


Figure: Severe discontinuities of the mutation operator have a continuous influence (a *bias*) on evolution [Bul99, p. 69]. Flat: mutation is simply randomly drawing a gene value from the allowed range. The Gray code does not eliminate the disadvantages of mutating encoded numeric values.

Discussion: which encoding would satisfy the requirement that each mutation changed the encoded value by ± 1?

Parameterization

The introduction of various improvements and extensions to GAs (for example: adaptive mutation probabilities) sometimes requires new parameters. On the other hand, some of these modifications enable automatic tuning of parameter values. The new parameters are often more easily interpretable and/or their impact on the algorithm performance and achieved results becomes more predictable.

Discussion: suppose you have some concrete optimization problem. What parameter values do you set and what mechanisms do you choose to use? What criteria do you use to set these particular values and not others?

Parameterization – population size and mutation

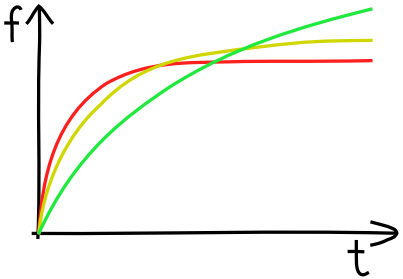
The size of the population affects primarily the inertia of the algorithm. With larger populations, the algorithm reacts more slowly, so for *on-line* applications, relatively small populations are recommended. Large populations, especially in the beginning of evolution, need more time to reach good solutions. On the other hand, for *off-line* applications, large populations are advantageous because they carry more information and allow for a more thorough search of the solution space. At the same time, a large number of individuals reduces the risk of getting stuck in a local optimum. However, it is necessary to adjust the size of the population so that it manages to converge to the region of good solutions in the available time.

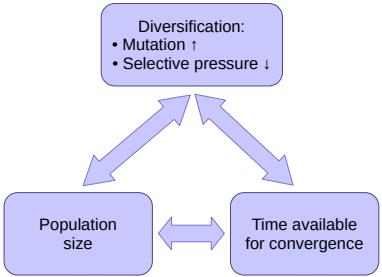
In GAs, the probability of mutation accounts for the number of random changes in the bits of individuals. This probability is recommended to be the inverse of the number of decision variables (if the decision variables are binary, how many genes are expected to be mutated in each generation?). Sometimes one can encounter the probability of mutation that is the inverse of the number of individuals in the population (how many genes are expected to be mutated in each generation?)

Parameterization – selection, stopping condition and interactions

A selection technique that has better properties than roulette is stochastic remainder selection (described earlier) with or without replacement, or tournament techniques. The recommended scaling mechanism is the truncation at the level dependent on the standard deviation, possibly accompanied by linear scaling.

A good stopping criterion for *off-line* applications is the number of generations without improvement (although without introducing a threshold for the minimum difference, the algorithm may run for too long due to repeated, insignificantly small improvements). One can also monitor the average similarity in the population and stop the optimization when the diversity is lost.





References I

[AA19] Anas Arram and Masri Ayob. "A novel multi-parent order crossover in genetic algorithm for combinatorial optimization problems". In: *Computers & Industrial Engineering* 133 (2019), pp. 267–274. DOI: 10.1016/j.cie.2019.05.012.

[Ahm15] Zakir Hussain Ahmed. "A multi-parent genetic algorithm for the quadratic assignment problem". In: *Opsearch* 52 (2015), pp. 714–732. URL: https://www.researchgate.net/profile/Zakir-Ahmed-3/publication/275716879_A_multi-parent_genetic_algorithm_for_the_quadratic_assignment_problem/links/5bee70e54585150b2bba1f07/A-multi-parent-genetic-algorithm-for-the-quadratic-assignment-problem.pdf.

[Bul99] Seth Bullock. "Are artificial mutation biases unnatural?" In: *European Conference on Artificial Life*. Springer. 1999, pp. 64–73. DOI: 10.1007/3-540-48304-7_11. URL: https://eprints.soton.ac.uk/261452/1/10.1.1.40.2753.pdf.

[DJ75] Kenneth Alan De Jong. "Analysis of the behavior of a class of genetic adaptive systems". PhD thesis. University of Michigan, 1975. URL: https://deepblue.lib.umich.edu/bitstream/handle/2027.42/4507/bab6360.0001.001.pdf.

[ECS89] Larry J. Eshelman, Rich Caruana, and J. David Schaffer. "Biases in the crossover landscape". In: *Proceedings of the 3rd International Conference on Genetic Algorithms*. Ed. by J. David Schaffer. Morgan Kaufmann, 1989, pp. 10–19.

[EKK95] Agoston E. Eiben, Cees H. M. van Kemenade, and Joost N. Kok. "Orgy in the computer: Multi-parent reproduction in genetic algorithms". In: *European conference on artificial life*. Springer. 1995, pp. 934–945. URL: https://www.researchgate.net/profile/A-Eiben/publication/221530971_Orgy_in_the_Computer_Multi-Parent_Reproduction_in_Genetic_Algorithms/links/02bfe511b6708800d5000000/Orgy-in-the-Computer-Multi-Parent-Reproduction-in-Genetic-Algorithms.pdf.

References II

[Gol02] David Edward Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 2002.

[KM18] Maciej Komosinski and Konrad Miazga. "Comparison of the tournament-based convection selection with the island model in evolutionary algorithms". In: *Journal of Computational Science* 32 (2018), pp. 106–114. ISSN: 1877-7503. DOI: 10.1016/j.jocs.2018.10.001. URL: http://www.framsticks.com/files/common/ConvectionSelectionVsIslandModel.pdf.

[KU17] Maciej Komosinski and Szymon Ulatowski. "Multithreaded computing in evolutionary design and in artificial life simulations". In: *The Journal of Supercomputing* 73.5 (2017), pp. 2214–2228. ISSN: 1573-0484. DOI: 10.1007/s11227-016-1923-4. URL: http://www.framsticks.com/files/common/MultithreadedEvolutionaryDesign.pdf.

[Mah92] Samir W. Mahfoud. "Crowding and preselection revisited". In: *Parallel problem solving from nature*. Ed. by R. Männer and B. Manderick. Vol. 2. Elsevier, 1992, pp. 27–36. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.3943&rep=rep1&type=pdf.

[TYH99] Shigeyoshi Tsutsui, Masayuki Yamamura, and Takahide Higuchi. "Multi-parent recombination with simplex crossover in real coded genetic algorithms". In: *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation – Volume 1*. 1999, pp. 657–664. URL: https://www.researchgate.net/profile/Shigeyoshi-Tsutsui/publication/243776468_Multi-parent_recombination_with_simplex_crossover_in_real-coded_genetic_algorithms/links/00463531fa92bd4738000000/Multi-parent-recombination-with-simplex-crossover-in-real-coded-genetic-algorithms.pdf.