# Intelligent Systems Engineering

## EC4 Evolutionary Optimization

## Variation, Constraint handling, Self-adaptation

Petr Musilek

University of Alberta

# Outline



These notes are based on [Keller et al. 2017] chapter 11.

# Variation

Variation operators

- means for searching the solution space for improved solutions, or
- potentially for weaker solutions that could lead to improved solutions.
  - through a weak selection method that allows less-than best solutions to survive and be the basis of further exploration

Traditional variation operators (already discussed)

- binary mutation,
- Gaussian mutation, or
- one-point, n-point, or uniform crossovers

The choice of variation operators goes "hand in glove" with the choice of representation (not the other way round!)

# Real-Valued Variation

When searching $\mathbb{R}^n$, it is typical to use a Gaussian mutation operator defined by two parameters

- the mean $\mu$ (set typically to 0, for an unbiased search in the neighborhood of a parent)
- standard deviation $\sigma$, or variance $\sigma^2$ (defining the size of the neighborhood)

For any $\sigma > 0$, there is a nonzero probability of returning any value between -1 and 1

- the smaller the standard deviation, the smaller the search neighborhood
- but the probability of a large move away from 0 may be very small, e.g.
- the probability of a zero-mean Gaussian random variable returning a value greater than 1 or less than -1 is
  - approximately 0.32 for $\sigma = 1.0$,
  - approximately 0.05 for $\sigma = 0.5$

The progress that a search based on a Gaussian random mutation operator will make is highly dependent on the value(s) of $\sigma$ (in each dimension).

To have a larger probability of creating a greater distance between an offspring and its parent

- rather than increasing the value of $\sigma$ in a Gaussian mutation,
- a Cauchy-distributed random mutation can be employed

A Cauchy random variable is constructed by taking the ratio of two independent identically distributed Gaussian random variables. It has interesting mathematical properties

- it is symmetric, yet
- it has no explicit mean or standard deviation (i.e., the expected value of the random variable is undefined and so are all higher moments of the distribution).

The Cauchy distribution has "fatter tails" than a corresponding Gaussian distribution

- the probability of mutating a real-valued parameter to a greater extent is markedly greater
- this can be helpful for escaping from locally optimal solutions

# Multiparent Recombination Operators

The use of

- one-point, $n$-point, and uniform crossovers,
- blending recombination (averaging components of individuals)

was illustrated using two individuals.

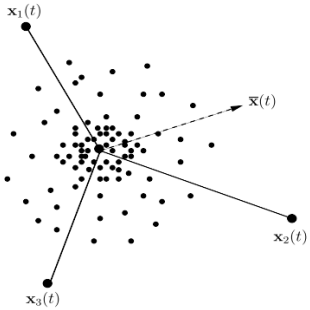
However, there is no limit to relying on two parents

- nature provides inspiration for evolutionary algorithms with two parents, but
- it may be beneficial to recombine elements or blend parameters of three or more solutions

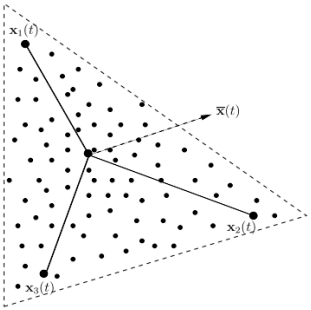
When applying blending recombination

- averaging components across multiple solutions estimates the mean of the population
- the greater the number of individuals, the better the offspring will estimate the mean
- when a population is contained in a locally optimal region of the search space, it can accelerate convergence toward the local optimum (sacrificing searching outside the region)

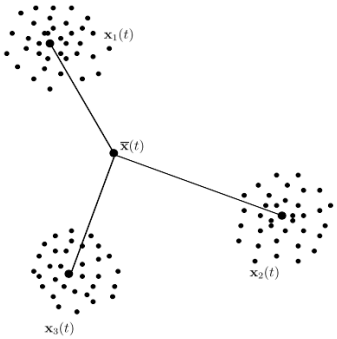
# Variants of multiparent crossover operators




Unimodal distributed (UNDX)




Simplex crossover (SPX)




Parent-centric crossover (PCX)

[Engelbrech, Computational Intelligence An Introduction 2007]

# Variations on Variable-Length Structures

Certain representations employ data structures of variable length

- evolving a neural network that can adapt not only its weights and bias values, but also
  - the number of nodes it uses and
  - the feedback loops that it employs (if any)

  the data structure used might well be of variable length
- evolving a collection of fuzzy membership functions that are used in a fuzzy controller
  - the number of fuzzy functions,
  - as well as their shape and location,

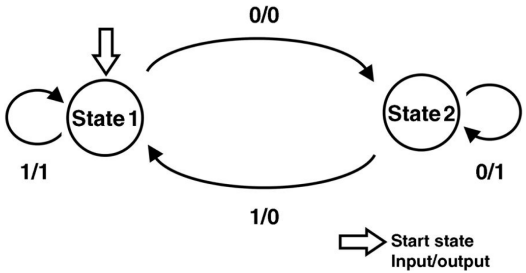
  could be represented by variable length data structure subject to evolutionary adaptation
- other common examples involve
  - finite-state machines,
  - symbolic expressions, and
  - difference equations

In case of these and similar presentations with variable-length data structures, the goal of variation operators is to ensure the possibility of a thorough search of the solution space.

Finite-State Machines have been used in evolutionary algorithms for predicting sequences of symbols (time series prediction).

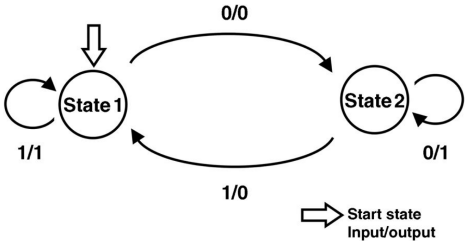
For example, Mealy/Moore finite-state automata [Fogel et al. 1966] with FSM defined by its number of states, its starting state, the input/ output function for each state, and the input/state transition function for each state. For the available symbols were $\{0, 1\}$, one finite-state machine could have the following characteristics:

- Two states.
- State 1 is the start state.
- In state 1, input of 0 yields 0, input of 1 yields 1; in state 2, input of 0 yields 1, input of 1 yields 0,
- In state 1, input of 0 transitions to state 2, input of 1 remains in state 1; in state 2, input of 0 remains in state 2, input of 1 transitions to state 1.



The following modes of mutation naturally follow from the FSM description:

1. Add a state.
2. Delete a state.
3. Change the start state.
4. Change an input–output relationship.
5. Change an input–state transition relationship.



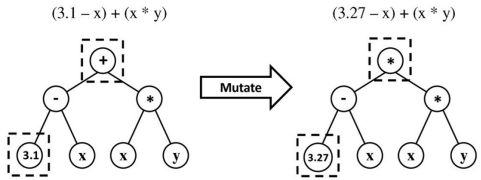

Through probabilistic application of these mutation operators, a parent finite-state machine can create

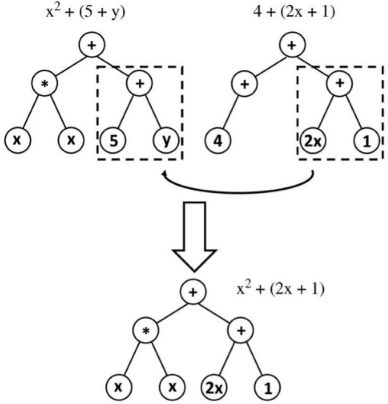
- a very different offspring (in terms of its input–output behavior)
- or one that is very similar.

The effects on the sequences that a finite-state machine may generate (i.e., the "behavior" of the machine) vary across these mutation operators. The effect of deleting a state may be much greater than merely adjusting the output a state generates for some particular input.

Symbolic Expressions. Consider evolving a solution to a problem using a symbolic expression (s–expression). This can be represented as a tree - a data structure of variable length.

Example problem: find a polynomial expression that returned the value of $x^2 + 2x + 1$ for an input of $x$.
A possible variation is subtree recombination that would take two suboptimal expressions and combine them to create the correct formula (illustrated on the right).
Another example is mutating specific values associated with nodes in the expression (illustrated below).



Difference Equations In mathematical modeling / system identification, input–output examples are given for a particular system and the objective is to construct a mathematical model that represents the input–output relationship. For time series problems, it is common to use autoregressive (AR) moving-average (MA) models (or ARMA models), which are particular forms of difference equations of the form

$$x[t+1] = a_0 x[t] + a_1 x[t-1] + \cdots + a_j x[t-j] + e[t] + b_1 e[t-1] + \cdots + b_k e[t-k]$$

where

- $x[t]$ is the observed variable at time $t$ (this is the AR part),
- $e[t]$ is the random noise occurring at time $t$ (this is the MA part), and
- $a_0, \ldots, a_j,\ b_1, \ldots, b_k$ are coefficients.

A standard approach employs gradient methods to estimate the coefficients for a model that has a predefined number $i$ and $j$ of lag terms for the AR and MA parts of the model.

This is problematic because the best choice of $j$ and $k$ are not known a priori and thus multiple searches must be conducted over different choices to provide confidence in a final result.

An evolutionary approach to this problem encodes both the number of lag terms and the coefficients as a single solution, such as

$$[2, 1.5, 0.7, 0]$$

where the first integer is the number of AR lag terms, the following two are the coefficients of those lag terms, and next integer is the number of lag terms in the MA process, i.e.

$$x[t+1] = 1.5x[t] + 0.7x[t-1] + e[t]$$

The following are the variation operators that follow naturally from this representation:

- vary the number of AR terms
- vary the number of MA terms
- vary the coefficients of any/all terms
- combine two or more models

In cases where the degrees of freedom of a model are allowed to increase as part of the search process, a trade-off should be employed in evaluating fitness (incorporating the number of DoF).

# Constraint Handling

Most real-world problems are constrained problems. For example, optimal bus scheduling

- specific number of existing buses,
- limited budget to purchase additional buses,
- limited number of qualified drivers for each different type of bus,
- each bus has limited capacity,
- required maintenance,
- roads that can and cannot be used,
- the available time for each driver to work each week,
- and so forth.

When applying evolutionary algorithms it is important to consider how to treat the constraints

- some are part of the objective/ some are part of the parameters of a solution
- each may pose hard or soft constraints.

If violated, a hard constraint makes the entire proposed solution worthless.

- For example, when designing a golf club, it must be be at least 18" and not more than 48" long - anything outside of these limits cannot be used in regulation play (according to United States Golf Association, USGA). Therefore individuals corresponding to club lengths outside of this range are invalid.

A soft constraint can be violated, but there is some imposed penalty for violating it (perhaps increasing with the degree of violation).

- For example, when scheduling fuel trucks to refuel gas stations, a constraint would be to design a schedule in which no station is ever empty (which would mean the oil company would have customers waiting at the gas pumps for a refueling truck). But this is not likely to be a hard constraint: a schedule that had a gas station out of fuel for 1min might lead to some bad publicity, but still be acceptable. The more wait, the more bad publicity described using a penalty function.

It is often helpful to craft an objective function that comes in two parts:

1. the primary criteria of interest and
2. a penalty function that treats constraint violations.

Sometimes these two parts can be simply added together, e.g. Akaike's information criteria

$$\text{AIC}(x) = -2\ln(L) + 2p$$

where $x$ is the parameter vector of the model, $L$ is the likelihood function of the model (measure of its fit to the available data), and $p$ is the number of model parameters.

Better models generate lower AIC scores. For each new parameter that is added, the likelihood function will be higher (and thus $-2\ln(L)$ will be lower), but it has to be sufficiently better to "pay for" the addition two points that the extra parameter will cost (i.e. there is no hard constraint on the number of parameters, but each additional parameter comes at a cost).

When a hard constraint is involved, the objective function can be set to an infinitely bad score if the constraint is violated.

- For example, suppose that in the above AIC minimization problem there was a constraint not to use more than 10 parameters. Then the objective function could be written as

$$f(x) = \begin{cases} \text{AIC}(x), & p \leq 10 \\ \infty, & p > 10 \end{cases}$$

The difficulty? This sets up areas of the search space that have no information to direct the evolutionary search to improve performance (anything violating constraints is penalized equaly). An alternative

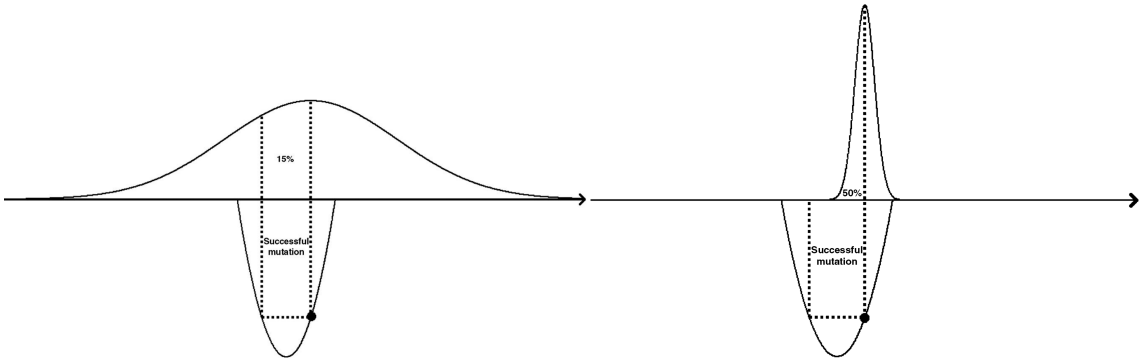
- impose a penalty for violating the hard constraint that increases in effect gradually over successive generations
- this way, the constraint is treated as a soft constraint at first but over time solidifies as a hard constraint.

A drawback: the requirement for tuning a schedule for the transition.

# Self Adaptation

Consider a simple evolutionary algorithm, e.g. one with only Gaussian mutation to components of potential solutions, $x \in \mathbb{R}$

- the setting of the standard deviation of the mutation will affect the degree of progress that can be made toward an optimum.



# The 1/5 Rule

Rechenberg studied two minimization problems that are linear and quadratic functions of $x$

- using a $(1 + 1)$ evolutionary algorithm that relied on zero-mean Gaussian mutation
- what is the trade-off between
  - the rate of improvement
    (i.e., how quickly the best solution's score decreases with generations)
  - and the probability of improvement
    (i.e., the likelihood that an offspring will be better than its parent).

As the number of dimensions $n \to \infty$, the maximum rate of improvement occurred

- for the linear function when the probability of improvement was approximately 0.184,
- and for the quadratic function it was approximately 0.27.

Value of 0.2 was selected as a compromise and the basis of a new heuristic [Rechenberg 1973]

**The 1/5 rule:** The ratio of successful mutations to all mutations should be about 1/5.

There is a simple method for computing this empirically [Schwefel 1995]

- after every $m$ mutations, determine the success rate over the previous $10m$ mutations
  - if the success rate is less than 0.2, multiply the step size ($\sigma$) by 0.85
    (the step sizes will become smaller)
  - if the success rate is greater than 0.2, then divide the step size ($\sigma$) by 0.85
    (the step sizes will increase)

Recall that 1/5 rule is a heuristic that was derived for specific conditions

- two simple types of functions,
- as the number of dimensions tends to infinity,
- and for a $(1 + 1)$ evolutionary algorithm.

There is no general case to make for the utility of the 1/5 rule (there are counter examples).

Still, the 1/5 rule illustrates that static parameters for variation operators are very unlikely to lead to the best rates of progress toward optimum solutions.

# Meta-Evolution on Real-Valued Parameters

More general methods for controlling the degree to which variation operators act coarsely or finely are desired (given the limited utility of the 1/5 rule)

- A common approach views the parameters that control the evolutionary search as part of the evolutionary process to be adapted while searching for an optimum.
- This poses a form of meta-evolution in that adaptation takes place on two levels within the evolutionary algorithms:
  - the parameters that are used to evaluate the objective function (objective parameters)
  - and the parameters that are used to control variation (strategy parameters).

With the problem of minimizing (maximizing) a real-valued function $f(x)$,

- encode a possible solution as $(x, \sigma)$,
  - where $x$ is the vector of real-valued objective parameters
  - and $\sigma$ is the vector of strategy parameters, which designate the positive standard deviation to apply a Gaussian mutation in the given dimension.