Intelligent Systems Engineering

EC1 Evolutionary Computation

Petr Musilek

University of Alberta

Outline

These notes are based on [Keller et al. 2017] chapter 10. Slides 30-37 are based on [Weis, 2011]
Weis, T., Global Optimization Algorithms: Theory and Application, 2011 (click the title to download)

Basic Ideas and Fundamentals

- Life has been evolving on Earth for over 3 billion years [Schopf,2006]
- Compared to less 100 years of a typical human life: 3×10^9 may be hard to comprehend
- Fossil records (bones, fossilized foot prints, occasional bug in amber) provide clues about life in ancient times
- A comparison to the world around us now leads to an obvious observation:

Nature's organisms are, and have often been, wonderfully adapted to their environments.

In the grand scale of time, these adaptations are ephemeral: they only last for moments, when compared to the history of the universe - "a blink of the universe's eye".

Over many of these "short periods", nature has solved some very challenging engineering problems through evolution by variation and natural selection, for example

- Flies can walk upside down on a ceiling
- Spiders do not stick to their own webs
- Geckos seemingly defy gravity as they climb walls and hang upside down

See textbook [Keller 2016, chapter 10, pp. 209-210] for more details on these examples.

Evolution is a very complex process. However, its essence – variation and selection can be illustrated using a series of mapping functions that connect genetics with behavior.

Living organisms act as a duality of their

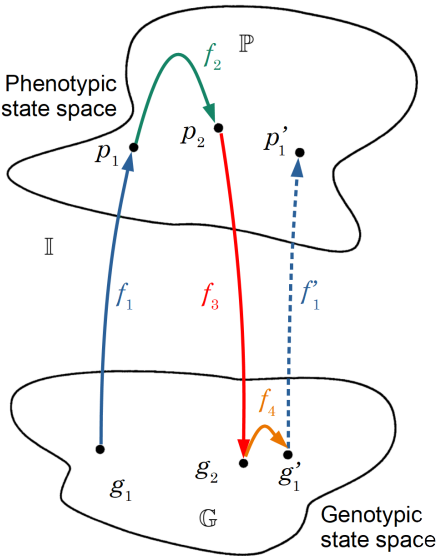
- genotype (the underlying genetic coding) and
- phenotype (the manner of response contained in the behavior, physiology, and morphology of the organism).

This may be viewed as a pairing across two state spaces

- informational (genetic) space $\mathbb{G}$, and
- behavioral (phenotypic) space $\mathbb{P}$

through four functions that map elements in $\mathbb{G}$ and $\mathbb{P}$ to each other.

$$f_1 : \mathbb{I} \times \mathbb{G} \to \mathbb{P}$$
$$f_2 : \mathbb{P} \to \mathbb{P}$$
$$f_3 : \mathbb{P} \to \mathbb{G}$$
$$f_4 : \mathbb{G} \to \mathbb{G}$$



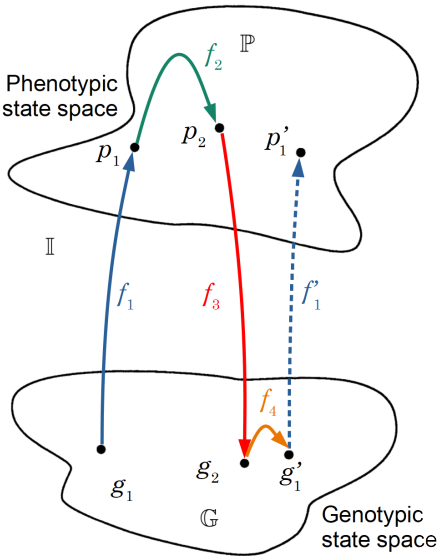
f_1 is called epigenesis; it maps an element $g_1 \in \mathbb{G}$ and indexed set of symbols $(i_1, \ldots, i_k) \in \mathbb{I}$ (the set of all such environmental sequences) into the phenotypic space $\mathbb{P}$ as a particular collection of phenotypes p_1, whose development is modified by its environment; the same set of genetics can correspond to alternative observed behaviors when exposed in different environments.

f_2 is called selection; it maps phenotypes p_1 into p_2. As natural selection operates only on the phenotypic expressions of the genotype, the underlying coding g_1 is not involved in function f_2 (natural selection acts on the behaviors generated by organisms and thus only indirectly on their underlying genes)

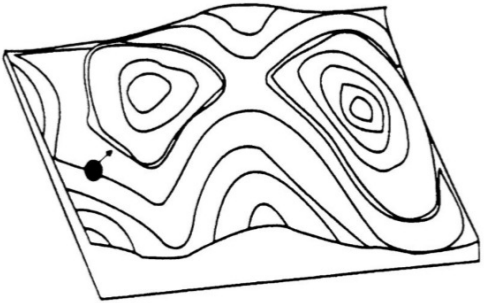
f_3 is genotypic survival; it describes the effects of selection and migration processes on $\mathbb{G}$. For the phenotypes that survive the selection function f_2, there is a collection of genotypes g_2 that correspond to those surviving phenotypes (it differs from g_1 because selection and migration have removed individuals from g_1).

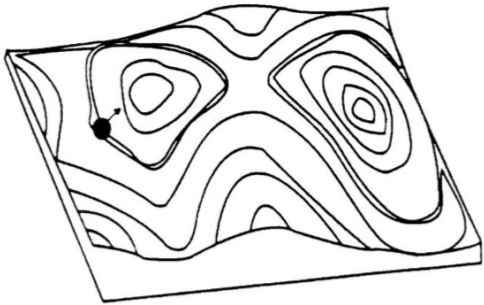
These new collection of genotypes are now the focus of mutation, function f_4. It maps the representative codings $g_2 \in G$ to the point $g_1' \in G$. Thus, f_4 represents the "rules" of genetic variation that occur in the creation of offspring from parents: individual gene mutations, recombination, inversion, and so forth.

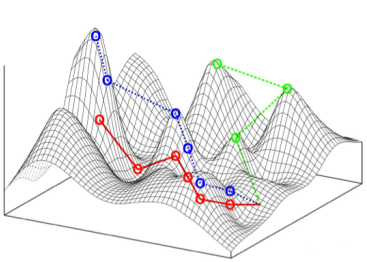
With the creation of a new population of genotypes g_1', one generation is complete. Evolutionary adaptation occurs over successive iterations of these mappings.



The ability to survive and reproduce in a specific environment is evaluated by fitness. In essence, evolution occurs on a landscape (topography) resulting from a function describing the fitness to the environment, so called fitness landscape.







An adaptive landscape invokes the concept of function optimization in engineering, math, or computing science. However, rather than searching for maximum, minimum, or saddle point on the adaptive landscape, evolution moves without purpose.

[Image by Randy Olson CC BY-SA 3.0]

Still, its movements appear purposeful in that

- the results of evolution reflect opportunities to improve fitness, or
- inversely to reduce predictive error.

Variation and selection, coupled together, provide a mechanism for searching over a landscape, resulting in outcomes that often appear highly engineered or "designed" for survival:

- variation provides an unending source of searching for new possibilities, while
- selection eliminates organisms that are less fit relative to others.

Engineering optimization problems usually take the form

$$\text{find } x | f(x) \text{ is minimized}$$

where x is a scalar value and $f(x)$ is a real-valued function of x.

Problems of **optimization in evolution** involve numerous variables acting simultaneously; there is

- interaction between behavioral traits, with environment and with some genetic basis (epistasis – gene at one locus alters the phenotypic expression of a gene at another locus).
- relationship between genes and behaviors is not one-to-one, but often one-to-many (pleiotropy – one gene has many effects) and many-to-one (polygeny – one effect coming from many genes), simultaneously

About fitness

Fitness may be difficult to measure instantaneously; it is a function of the environment that includes all the other organisms present. For example, this fly looks like a bee:

- It gets the benefit of looking like a bee, which may be a big benefit if there are other bees around.
- In the absence of bees and other insects with similar warning patterns, this particular pattern would be of much less value or no value at all.

Thus, fitness is a temporal function of the world in which the organism lives. Adaptive landscapes look simple in three dimensions, but they are in fact highly complex, span numerous dimensions, and vary in time.

Evolutionary Computation

Evolutionary computation simplifies evolutionary processes in software (evolutionary algorithms) used for engineering purposes

- optimization,
- design, and
- learning.

The simplest evolutionary algorithm is a search procedure that

- generates potential solutions to a problem,
- tests each for suitability,
- and then generates new solutions.

How this process differs from exhaustive search or blind random search?

Exhaustive Search

Consider a sample space $\mathcal{S}$ of size $k = |\mathcal{S}|$, from which individual potential solutions $s_1, ..., s_k$ can be selected.

Suppose there is a real-valued function $f(s)$ for which it's desired to find a maximum.

An exhaustive search of S can be described in pseudo code by this procedure

$i = 1;$
best $= i;$
bestscore $= f(s(i));$
repeat
  $i = i + 1;$
  if $f(s(i)) >$ bestscore **then**
    best $= i;$
    bestscore $= f(s(i));$
  end if
until $(i == k)$

This will find the best solution s^* with corresponding greatest score $f(s^*)$. While exhaustive search is guaranteed to find the best solution, the time required may be prohibitive for many large, practical problems (cf. textbook demonstrating how solutions to so called transcomputationally large problems might take billions years to find the solution)

Blind Random Search

Random search blindly chooses a series of possible solutions $s_1, ..., s_n$ from $\mathcal{S}$, where the sequence of length $n \leq k$ is chosen uniformly without replacement. This means that for the first sample s_1, each solution in $\mathcal{S}$ has a $1/k$ probability of being selected; then for s_2, this probability changes to $1/(k-1)$, and so forth.

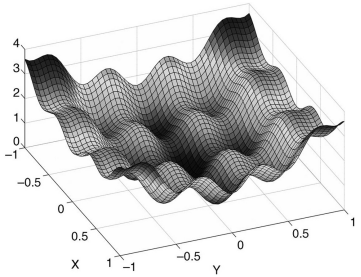
$i = 1$;
selected $= U(1, k)$; //chosen uniformly at random over the integers 1,...,k
best = selected; bestscore $= f(s(\text{selected}))$;
repeat
  $i = i + 1$;
  remove s(selected) from $\mathcal{S}$;
  selected $= u(1, k - i + 1)$; //chosen uniformly over the remaining solutions
  if $f(s(\text{selected})) >$ estscore **then**
    best = selected; bestscore $= f(s(\text{selected}))$;
  end if
until $(i == n)$ // completes in a reasonable amount of time for appropriate value of n
// unfortunately, this procedure often performs poorly on real-world problems

Both the exhaustive search and the blind random sampling choose each next solution without regard to what has been chosen previously. This "blindness" is often a big handicap.

Traditional search procedures described in mathematics are not blind

- they explore for a maximum/minimum using information from the function being searched
- this often involves the gradient or higher order statistics to move rapidly in a direction that is presumed to be beneficial (i.e., leading to higher or lower values)

  - these procedures can converge quickly on a maximum or minimum, but
  - there is the risks of stalling at a saddle point (zero gradient), or
  - becoming trapped in maxima or minima that are only optimal locally



Evolutionary algorithms (EA)

Evolutionary algorithms operate differently from traditional gradient methods, in two ways:

- rather than executing a point-to-point search, they incorporate a population of solutions, each individual solution competing for survival
- instead of utilizing gradient information from the response surface being searched, they utilize random variation to explore for new solutions

$g = 0$; // g is for generation number
choose initial population; // often this is selected at random from $\mathcal{S}$
repeat
    evaluate population; // assign a score to each individual in the population
    select parents; // based on the scores, choose a subset of individuals
    generate offspring; // based on the parents, create offspring
    $g = g + 1$;
until $(g == \max)$

Illustration of the EA procedure

Find the minimum of the function $f(x) = x^2$ (this is a convex continuous function, thus gradient methods would be more appropriate; but it is used here for illustration)

1. the value g (number of generations completed) starts at $g = 0$
2. the initial population of size $(\mu + \lambda)$ is chosen at random from a broad range of $\mathcal{S}$
3. each of the $(\mu + \lambda)$ solutions is evaluated in light of $f(x)$
4. the μ individuals with the best scores are selected to become parents
5. these parents then become the basis for generating λ offspring
6. the generation counter is incremented $g = g + 1$
7. steps 3–6 continue until a maximum number (max) of generations is completed

The relationship between parent and offspring defines **inheritance**. Without inheritance, the procedure would be like a blind random search conducted multiple samples at a time.

Suppose instead that the value of an offspring x' is related to its parent x as follows

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

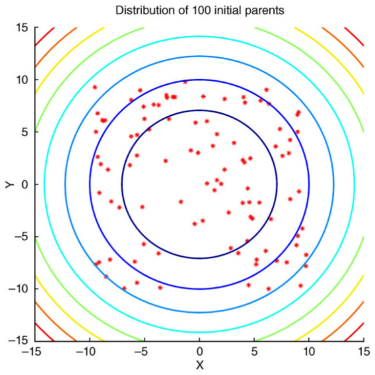
where $N(0, \sigma)$ is a Gaussian random variable with zero mean and standard deviation σ.

- If $\sigma \to \infty$, the search is like a blind random search
- If $\sigma \to 0$, there is no search at all
  (each offspring merely replicates its parent; complete inheritance from parent to offspring)
- With $0 < \sigma < \infty$, the procedure becomes a stochastic parallel search
  (with many interesting mathematical properties to be discussed later)

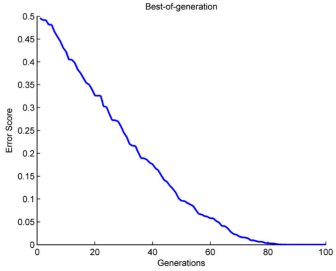
Simple evolutionary algorithm at work

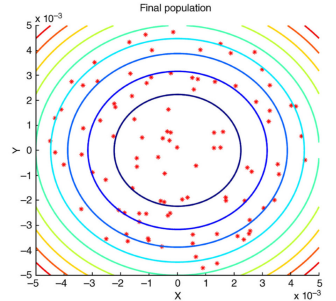
Use an evolutionary approach to approximate the answer that minimizes $f(x, y) = x^2 + y^2$.

- initial population: $\mu = 100$ candidate solutions chosen uniformly at random from the range $(x, y) \in (-10, 10)^2$
- a very basic approach: each parent generate one offspring by mutating the (x, y) coordinates
- mutation: adding a Gaussian random variable from $N(0, 0.01)$ to the x– and y–coordinates of the parent
- after the creation of $\lambda = 100$ offspring, 100 best solutions are selected from among the $\mu + \lambda = 200$ parents and offspring to be parents for the next generation



- Best-of-generation: improvement in the score of the best solution at each generation
- Generation 100: distribution of the 100 surviving solutions (distributed from 0.005 to 0.005 in each dimension, confined in a box that's 0.01 units on each side – area that is $2.5 \cdot 10^{-7}$ of the area of the initial box we started with – 20 units on a side)
- Another way to describe the improvement that the evolutionary algorithm offers in this case: The area of the best answer shrunk by a factor of over 1 million in 100 generations.





Representation, Search, and selection Operators

Evolutionary algorithms are often viewed in terms of

- the data structures that are used to represent solutions,
- the search operators that are applied to those data structures, and
- the selection operators that determine which solutions in a population will influence the creation of the next generation.

Solutions can be encoded into representations that are unintuitive and difficult to work with; e.g., for a solution (x, y) to the real-valued function $f(x, y) = x^2 + y^2$,

- the most natural representation is a pair of real values (it allows you to think intuitively about the problem with an image in mind similar to the shape of the function)
- alternatively, it is possible to encode any real value into, say, some number of binary digits to a given degree of precision (e.g. representing integers with four digits, first for sign and remaining three for value in binary representation of the integer; thus, $[0111] \equiv 7$)
- with very long bit strings, real-valued numbers can be represented with lots of precision (like in computers), but it would not be intuitive

Representation for a problem must be done in light of what search operators (i.e., variation operators) will be applied to that representation.

- variation operators applied to a current population modify a probability density function over the sample $\mathcal{S}$
- this provides for a biased random walk in that space, searching for better solutions
- common search operators include
  - mutation functions that apply to a single parent, and
  - recombination functions that apply to two or more parents
  - one special case of recombination is crossover (swapping segments between parents)

The choice of search operators affects performance of algorithms in alternative cases. However, it can be shown that across all problems no single search operator is superior to any other.

Selection Operators

Evolutionary algorithms are often viewed in terms of

- the data structures that are used to represent solutions,
- the search operators that are applied to those data structures, and
- the selection operators that determine which solutions in a population will influence the creation of the next generation.

In addition to their connection to representations, search operators are also closely related to selection operations. Selection can vary in effect from strong to weak

- strong selection ensures that only the very best solutions in a population will serve as parents for the next generation
- weak selection offers nonzero probability that weaker solutions may also serve as parents
- one common form of weak selection is called proportional selection, in which parents are chosen with probabilities that are proportional to their relative fitness.

Strong vs. weak selection

Strong and weak selection may also be used to describe the percentage of a population that is retained as parents; referring to the previous example (cf. slide 19)

- if $\mu \gg \lambda$; then this would be weak selection, while
- if $\mu \ll \lambda$; then this would be strong selection

Selection pressure can also be viewed as continuum between

random (weakest) – and – **deterministic (strongest)**

Search and selection are intertwined in effects

- A very narrow search can be made effectively broader by using weak selection
  - this would allow subpar solutions to become parents,
  - thus expanding the application of the narrow search operation.
- Similarly, a broad search can be made effectively narrower by using strong selection.

Certain heuristics are sometimes applied to ensure that the best solution in a population is not lost under weak selection. These heuristics are generally described as elitist.

Search algorithms can be viewed in the form of an update equation

$$\boldsymbol{r}[t+1] = k\big(v(\boldsymbol{r}[t])\big)$$

where $k(\cdot)$ is the selection operation,
$v(\cdot)$ is the variation operation(s),
$\boldsymbol{r}[t]$ is the population at time t under representation $\boldsymbol{r}$, and
there is an implicit generation of $\boldsymbol{r}[0]$ (e.g. through uniform sampling).

The use of bold notation for $\boldsymbol{r}$ emphasizes that there is a population of multiple solutions forming a vector. Formally, r is a transformation on $s \in \mathcal{S}$, such that $r(s)$ is a solution s being represented as $r(s)$, or r for convenience.

When applied to describe the behavior of evolutionary algorithms, it shows that the designer has certain choices to make about initialization, representation, variation, and selection. Variation is random, and often so is selection. Thus, the equation is a stochastic update equation that describes the probability of sampling from different locations in $\mathcal{S}$ at each generation t. Across all problems, there are no choices that will be uniformly superior.

Major Research and Application Areas

Optimization is the most common application area of EA

- problems span numerical and combinatorial optimization
- representation and operators tailored to the specific problems
- many real-world problems also pose multiple criteria that must be aggregated and traded-off, and
- are subject to constraints, which must be addressed to generate feasible solutions

Design is often intrinsically connected to optimization

- designed of an electronic circuit to fulfill a particular function described objectively in mathematical terms
- sometimes real-world in situ experimentation can be used to provide a means for evolving solutions (when there is no clear knowledge of an objective function available)
- in other situations, a human in the loop can provide guidance for selection (interactive evolutionary computation)

Learning often refers to the case of adjusting strategies for accomplishing a task based on feedback about the quality of performance being generated, e.g. learning mapping functions to classify patterns such as cases of malignancy in mammograms

- evolutionary algorithms can be used to adjust the weights between nodes in a neural network as well as the topology of the network simultaneously
- when available data in a learning application are static, the task is essentially another form of direct optimization: adjusting available parameters to maximize or minimize some objective function (e.g., minimizing the squared error between the output of a neural network and the desired target value for each pattern).
- this becomes intricate when the task requires learning in the face of dynamic data, such as controlling a nonlinear system (e.g., the famous cart–pole system presents a dynamic learning problem of having to prevent a pole connected to a moving cart from falling over)

Games Learning strategies in control systems can be made even more demanding when the system being controlled is itself purposeful.

- this is the essence of gaming, in which two or more players allocate resources in order to achieve desired objectives
- these objectives may be opposing, such as in checkers or chess: when one player wins, the other necessarily loses
- in other cases, the objectives may seem to offer the possibility for mutual cooperation
- evolutionary algorithms can be used to model each player in games such as these

Theory – analysis and evaluation of evolutionary algorithms in terms of their

- efficiency,
- rates of convergence,
- quality of evolved solution,
- computational requirements, and so forth

to gain insight

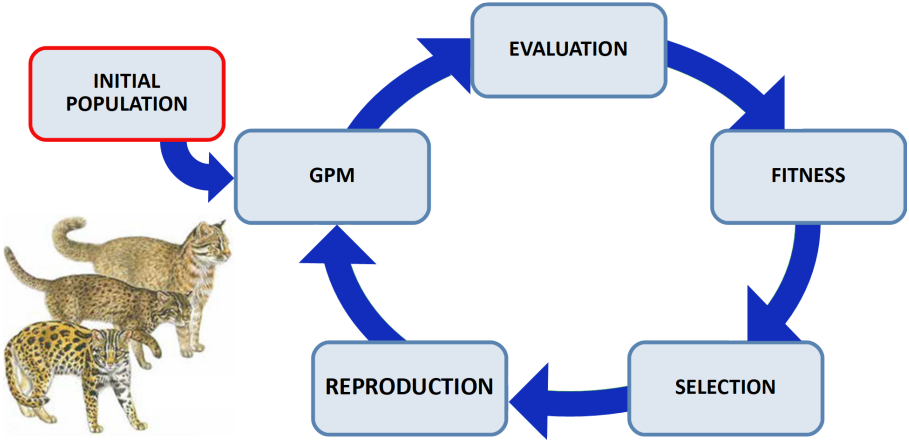
- into some properties of EAs that appear important for designing efficient searches on specific problems, or
- to reject some conventional wisdom that has been proven either incorrect or unhelpful, thus saving wasted time and effort.

This helps engineers gain best intuition about how evolutionary algorithms work and how best to apply them in problem solving.

Darwin's Idea of Evolution (On the Origin of Species, 1859)

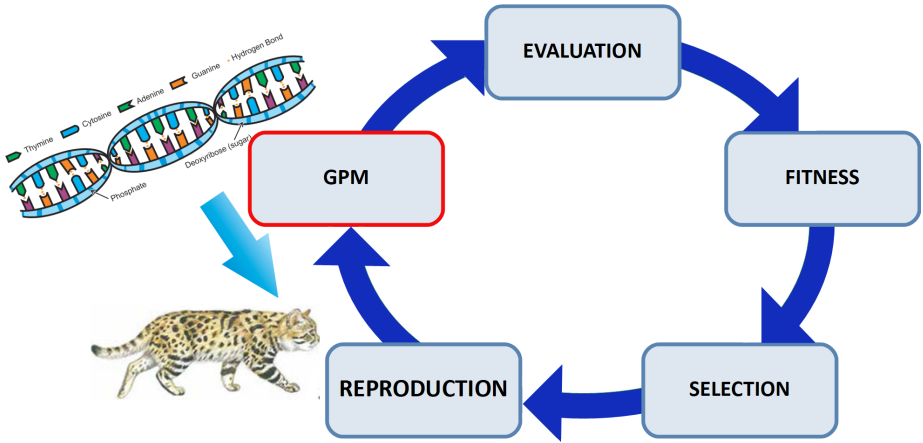
- Individuals are fertile and produce more offspring than can grow into adults
- Without external influences, the population size of a species remains roughly constant, and the food resources are limited but stable
- Individuals compete for these limited resources
- In sexual reproducing species, no two individuals are equal
- Some of the variations between the individuals will affect their fitness and hence, their ability to survive
- A fraction of these variations are inheritable
- Individuals less fit are less likely to reproduce, whereas the fittest individuals will survive and produce offspring more probably
- Individuals that reproduce will likely pass on their traits to offspring
- **Hence, a species will slowly change and adapt to a given environment**

Evolution Cycle: First Generation



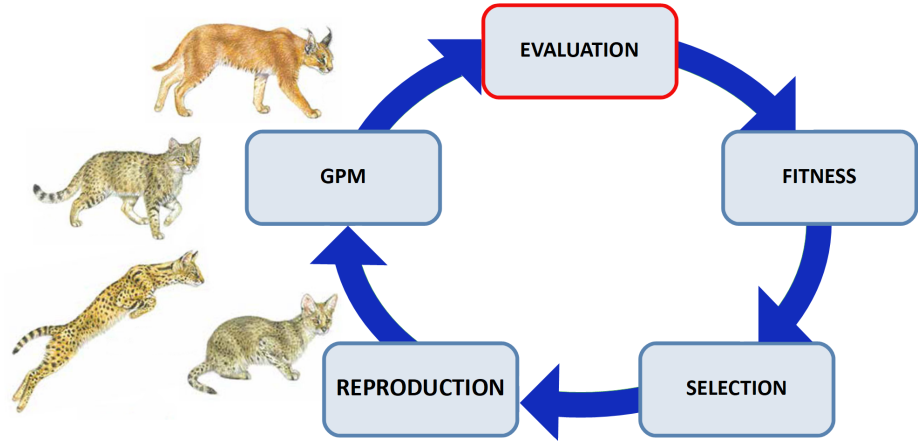
The cycle starts with a random set of individuals (their genetic codes)

Evolution Cycle: Genotype-Phenotype Mapping



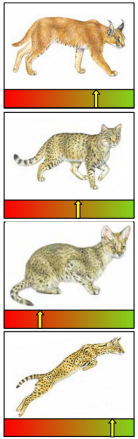
Life begins with the development (mapping) from genotype to phenotype (GPM)

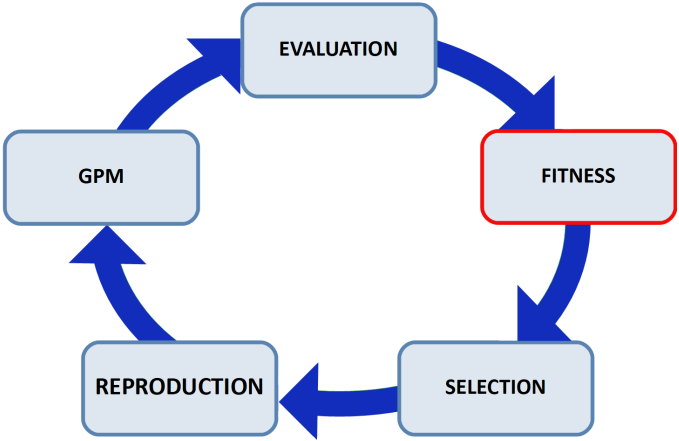
Evolution Cycle: Evaluation



Features of each individual are tested and evaluated

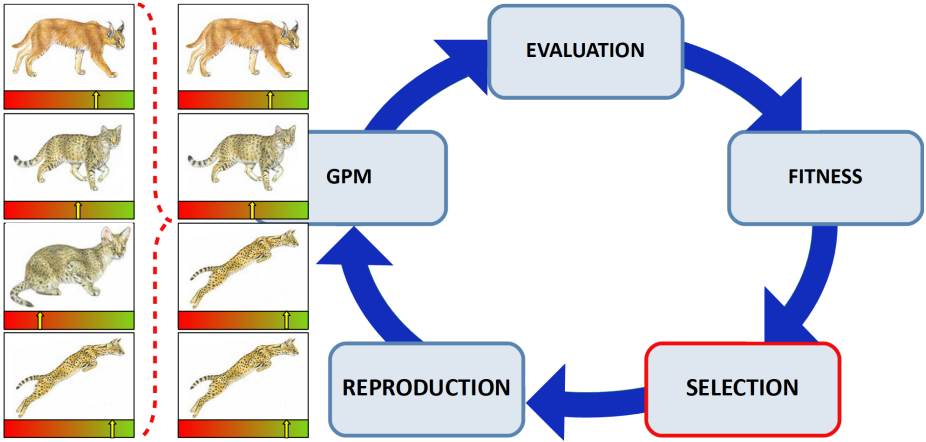
Evolution Cycle: Fitness





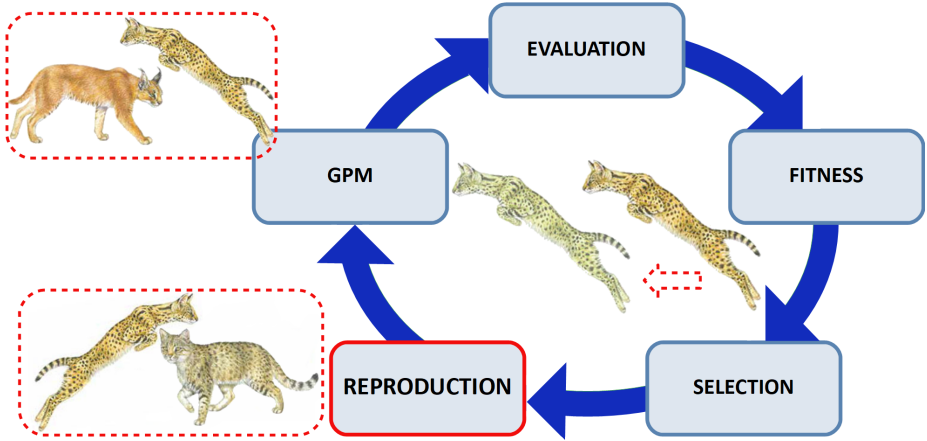
Each individual is assigned a fitness value (relative, determined/defined as number of offspring)

Evolution Cycle: Selection



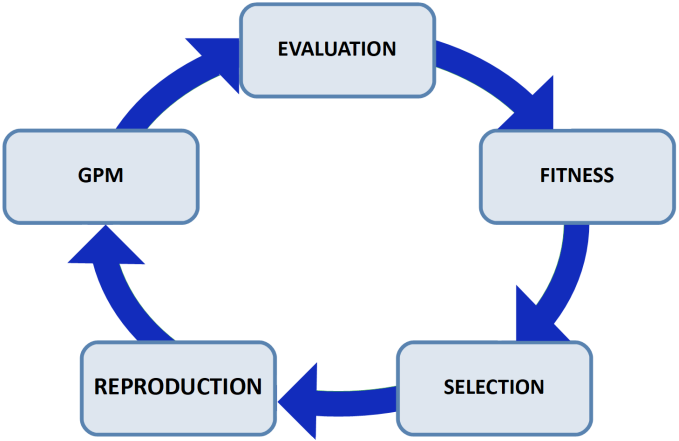
Fitter individuals usually survive selection ...

Evolution Cycle: Reproduction



... and have more offspring through asexual and sexual reproduction

Evolution Cycle: Next Generation



Cycle starts again with the next "generation"