When the normal distribution is not enough...

An individual consists of a vector and a scalar: \( v = (x, \sigma) \).

Uncorrelated mutation with one \( \sigma \):
1. \( \sigma' \leftarrow \sigma \cdot \exp(\tau \cdot N(0, 1)) \) (self-adaptation step)
2. \( x_{t+1}^i = x_t^i + N(0, \sigma') \) (mutation step)

Typically:
- the "learning rate": \( \tau \propto \frac{1}{\sqrt{n}} \)
- a boundary rule: \( \sigma' < \varepsilon_0 \Rightarrow \sigma' = \varepsilon_0 \)

Figure: Mutation #1: distribution of 10000 offsprings for the parent \([0,0]\) in 2-dimensional search space.

An alternative distribution:
- Cauchy distribution \( C(0, \sigma) \):
  - The one dimensional probability density function centered at the origin:
    \[
    f_{Cauchy}(x) = \frac{1}{\pi \left( t^2 + x^2 \right)}
    \]
  - where \( t > 0 \) is a scale parameter.

As a result the variance of the Cauchy distribution is infinite.
Therefore, we say Cauchy distribution has no mean, however, from the distribution density function we could try to identify a mean. But the point \((0)\) is just a median and a mode.

Figure: Probability density functions for normal (Gauss) and Cauchy distributions with the scale parameter \( t = 1 \). Advantage over Gauss p.d.f.: so called heavy tails.
Evolutionary Computation

Mutate for real valued representation:

Both Cauchy and Gaussian distribution are the members of a family of \( \alpha \)-stable distributions (Lévy distributions).

\begin{itemize}
  \item \( \alpha \)-stable distributions
  \item \( \alpha = 2 \): Gaussian distribution
  \item \( \alpha < 2 \): heavy-tailed distribution
  \item \( \alpha = 1 \): Cauchy distribution
\end{itemize}

Cauchy random number generator:

```c
double cauchy()
{
  double u, x;
  x = U(0,1)*2-1;
  if ((u+0.27324)*(1+x*x) > 1.27324)
    u = U(0,1);
  return x;
}
```

\[ \text{Cauchy random number generator:} \]

\[ \text{double cauchy()} \]

\[ \{ \]

\[ \text{double } u, x; \]

\[ x = U(0,1)*2-1; \]

\[ \text{if } ((u+0.27324)*(1+x*x) > 1.27324) \]

\[ u = U(0,1); \]

\[ \text{return } x; \]

\[ \} \]

\[ // \text{random variable uniformly distributed on } [0,1] \]

\[ \text{Wieczorkowski, Zieliński, \textit{Komputerowe generatory liczb losowych}, WNT, Warszawa, 1997 str. 91.} \]

\[ \text{N. Hansen, F. Gemperle, A. Auger, and P. Koumoutsakos, When Do Heavy-Tail Distributions Help?, PPSN IX, Springer LNCS 4193, 2006.} \]

\[ \text{Citation:} \]

The effect of anisotropy of a search distribution on the search performance can be tremendous, in particular in higher dimensions (\( \alpha \geq 10 \)). The Cauchy distribution, where coordinates are sampled independently, is highly anisotropic in that large steps occur most often close to the coordinate axes.

Hence, it can perform exceptionally well on separable functions, like any algorithm performing coordinate-wise search.

\[ \ldots \]

If the coordinate system is rotated or the distribution is modified to become isotropic keeping the distribution of the vector norm unchanged the performance becomes indistinguishable from the Gaussian distribution in our experiments.

\[ \text{W. Hansen, T. Stützle, A. Auger, and P. Koumoutsakos, Where Do Heavy-Tail Distributions Help?, PPSN IX, Springer LNCS 4193, 2006.} \]

\[ \text{Citation:} \]
Evolutionary Computation

α-Stable distributions random number generator

- The Chambers-Mallows-Stuck (1976) method for simulating α-stable random variables is presented in eq. (1), (2) and (3) for \( \alpha \geq 1 \) and \( \mu = 0 \).

\[
y_j = \begin{cases} 
\frac{\sin(\pi \alpha)\alpha^\alpha}{\pi^{\frac{\alpha-1}{2}}\Gamma(\frac{\alpha}{2})} (\cos(\theta) - \mu) + \mu & \text{if } \alpha \geq 1 \text{ and } \mu = 0, \\
\frac{\sin(\pi \alpha)}{\pi^{\frac{\alpha}{2}}} (\cos(\theta) - \mu) + \mu & \text{if } \alpha = 1.
\end{cases}
\]

where:

\[
\begin{align*}
R_{\alpha,\beta} & = \alpha^{-1} \arcsin \left( \frac{\tan \frac{\pi \alpha}{2}}{\beta} \right), \\
S_{\alpha,\beta} & = \left( 1 + \beta^2 \sin^2 \left( \frac{\alpha \pi}{2} \right) \right)^{-\frac{1}{\alpha}}.
\end{align*}
\]

\( u \) — a random variate uniformly distributed on \([-\pi/2, \pi/2]\), and \( w \) — an exponential random variate.

The procedure can easily be applied to the hypersphere located anywhere in the space, simply by addition of each of its coordinates to the respective coordinates calculated with the algorithm.

Evolutionary Computation

α-Stable distributions random number generator

... or you can take source code of the symmetric α-stable random number generator from the internet, e.g.:

http://www.ipipan.waw.pl/~trojanow/alpha/

Evolutionary Computation

Figure: Mutation #1 with symmetric α-stable distribution. Distributions of 10000 offsprings for the parent [0,0] in 2-dimensional search space for different values of \( \alpha \): 2, 1.7, 1.4, 1.1, 0.8, 0.5.

Evolutionary Computation

Mutation #4 — real valued representation:

A two-phase mechanism of mutation:

- First phase — a random direction \( \vec{\theta} \) with \( \| \vec{\theta} \| = 1 \) is proposed, where \( \| \cdot \| \) denotes the euclidean length of a vector. Clearly, this is a random point on the hypersphere surface surrounding the origin.

- Second phase — the distance \( d \) from the origin is evaluated. Then, a new location is determined based on the direction \( \vec{\theta} \) and the distance \( d \).

Evolutionary Computation

The only problem is ...

- Mutation #1 with symmetric α-stable distribution: \( x_t^{i+1} = x_t^i + S_{\alpha}(0, \sigma) \) generates anisotropic distribution of mutated points for all values of \( \alpha \) except from 2 (Gauss).

- Solution: for isotropic distribution of mutated points Mutation #4 is proposed.
Evolutionary Computation

Mutation #4: 2nd phase

- The distance $d$ from the original location is calculated as follows:
  \[ d = S_\alpha(0, \sigma) \]
  \[ \sigma = r S_\alpha(D_w / 2) \]
  $S_\alpha(\cdot, \cdot)$ — a symmetric $\alpha$-stable distribution variate
  $D_w$ — the width of the feasible part of the search space, i.e. the distance between the lower and upper boundaries.

- The new location is determined on the found direction $\theta$ and the distance $d$ from the original location.

Mutation #4 — surrounding effect

- The most probable distance of mutated points is not in a close neighborhood of the origin, but at a certain distance from it.
- The distance increases with the search space dimension $n$.
- In the case of a Gaussian mutation it is proportional to the norm of the standard deviation vector and to $\sqrt{n - 1}$.

- The dead surrounding effect decreases the exploitation properties of an EA while increasing the dimension of the search space.
- The effect can be reduced by the selfadaptation mechanism of $\sigma$. This results in a rapid decrease of $\sigma$ to very small values, especially in the case of low $\alpha$ and high dimensions.

Constraint handling

Constraint

1. something that limits or restricts someone or something
2. control that limits or restricts someone’s actions or behavior

http://www.merriam-webster.com/dictionary/constraint
Constraint Handling in Evolutionary Computation

Introduction

A constrained optimization problem with \( n \) parameters to be optimized is usually written as a nonlinear programming problem of the following form:

\[
\begin{align*}
\text{Minimize :} & \quad f(x), \quad x = (x_1, x_2, \ldots, x_n) \quad \text{and} \quad x \in D \\
\text{subject to :} & \quad g_i(x) \leq 0, \quad i = 1, \ldots, p \\
& \quad h_j(x) = 0, \quad j = p + 1, \ldots, m.
\end{align*}
\]

where:
- \( f(x) \) — does not need to be continuous but it must be bounded,
- \( D \) — is the whole search space,
- \( p \) — the number of inequality constraints,
- \( m - p \) — the number of equality constraints.

Questions:

1. How should two (feasible and/or unfeasible) solutions be compared?
2. How are the functions \( F_{\text{feasible}} \) and \( F_{\text{infeasible}} \) related to each other?
3. Should we consider infeasible individuals harmful and eliminate them from the population?
4. Should we “repair” infeasible solutions? How?
5. If we repair infeasible individuals, should we replace an infeasible individual by its repaired version in the population or rather should we use a repair procedure for evaluation purposes only?
6. Since our aim is to find a feasible optimum solution, should we choose to penalize infeasible individuals?
7. …

Basics: Boundary (inequality) constraints

Defined by the inequality constraints: \( l_i \leq x_i \leq u_i \) for \( i \in \{1, 2, \ldots, n\} \) where \([l, u]\) is the feasible area of the search space \( D \).

The domain is the interior and the boundary of a hypercube.

Handling boundary constraints

- Conservatism — if the perturbation operator resulted in an infeasible solution, it is rejected.
- Resampling — the perturbation operator is repeated until a feasible solution is obtained.
- Reinitialization — the infeasible solution is replaced by the one generated with the uniform distribution in the feasible area.
- Reflection — the solution is repaired by reflecting coordinate values from the exceeded boundary values (for real valued search space).
- Projection — all coordinate values that exceed bounds are trimmed to the boundary values.
Constraint Handling in Evolutionary Computation

Observation:
Generally, boundary constraints are not regarded as “true” constraints but rather form the definition domain $D$.

Constraint handling strategies

Main classification:
1. Reject strategies — only feasible solutions are kept and then infeasible solutions are automatically discarded.
2. Repairing strategies — heuristics transform an infeasible solution into feasible one (problem specific).
4. Penalizing strategies — infeasible solutions are considered, however, the objective function is extended by a penalty function that will penalize infeasible solutions.
5. Decoding strategies — there exist a mapping $\mathcal{R} \rightarrow D$ that associates with each representation $x \in \mathcal{R}$ a feasible solution $x' \in D$.

Constraint handling strategy #5: Decoding

Application of a homomorphous mapping function $\mathcal{R} \rightarrow D$ such that:

- For each $r \in \mathcal{R}$ there exists a feasible solution $x \in D$
- For each $x \in D$ there exists $r \in \mathcal{R}$.

Requirements:
1. The computation complexity of the decoder must be reduced.
2. Feasible solutions in $D$ must have the same number of corresponding solutions in $\mathcal{R}$.
3. The locality property — the distance between solutions in $\mathcal{R}$ must be positively correlated with the distance between solutions in $D$.

Constraint handling strategy #4: Penalizing

Linear penalty function depends on:
1. The number of violated constraints:

$$ F(x) = F(x) + \sum_{i=1}^{n} (w_i \cdot a_i) $$

- $w_i$ — cost of the constraint violation,
- $a_i = 1$ if $i$-th constraint is violated and $a_i = 0$ otherwise.
2. Amount of infeasibility or repairing cost — how close a solution is to a feasible region:

$$ F(x) = F(x) + \sum_{i=1}^{m} (w_i \cdot d_i) $$

- $d_i$ — a distance metric for the constraint $i$, such that:
  - for $g_i(x) \leq 0$ (the inequality constraints) — $d_i = h_i(x)$
  - for $h_i(x) = 0$ (the equality constraints) — $d_i = |h_i(x)|$.

Constraint handling strategy #4: Penalizing

Tuning cost of the constraint violation $w_i$:
1. Static — $w_i$ is the same for the whole search.
2. Dynamic — $w_i$ changes during the search, that is, the severity of violating constraints may be increased in time.
3. Adaptive — knowledge on the search process is included; for instance, $w_i$ is decreased, when many feasible solutions is generated and increased otherwise.

Constraint handling strategy #5: Decoding

A homomorphous mapping function example:
- We have two spaces: $\mathcal{R}$ is a cube of $n$-dimensional space $[-1, 1]^n$ and the search space $D$ with a feasible region $D_F$.
- A bijective mapping between points in the cube and points in the feasible region $D_F$ is defined.
Constraint Handling in Evolutionary Computation

Constraint handling strategy #5: Decoding

A homomorphous mapping function — example:

▶ In both spaces origins $r^*$ (reference points) are selected.
▶ For point $A'$ we can determine a radial line starting from the origin $r^*$ and going toward $A'$ and denote this radial line: $y = y^* - x$ where $y^*$ is the coordinates of point $A'$ and $t$ is a parameter.
▶ $l_0$ — distance between point $A$ and the origin.
▶ $T'$ — the maximum value of $t$ (intersection point with the boundary of the region).
▶ $l_2$ — the distance between $T$ and point $A$.

To keep the relative distance to the origin (the reference point), the image of point $A'$ in Fig (b), i.e., point $A$ in Fig (a), should satisfy:

$$\frac{L_1}{l_1 + l_2} = \frac{h_1}{h_1 + h_2}$$

The case when the feasible part of search space consists of a number of disconnected nonconvex regions.

The numerical calculation for finding all the intersections between the radial line and the boundary of the feasible region is hard.

The mapping cannot handle equality constraints.

Sometimes finding the initial feasible solutions is not a trivial task — a bad selection of $r^*$ might yield bad results.

Summary – disadvantages of the example approach

▶ The numerical calculation for finding all the intersections between the radial line and the boundary of the feasible region is hard.
▶ The mapping cannot handle equality constraints.
▶ Sometimes finding the initial feasible solutions is not a trivial task — a bad selection of $r^*$ might yield bad results.

Summary – advantages of the example approach

▶ The mapping will always preserve the feasibility of a population, so there is no requirements for evaluating infeasible solutions.
▶ Although users need to provide a reference point $r^*$ in the feasible region before the map, theoretically any point could be selected and would not affect the map.
▶ No additional parameters need to be provided by users.
**Lecture #6**

**Constraint Handling in Evolutionary Computation**

**Constraint handling strategies — equality constraints**

- Equality constraints \( h_j(x) = 0 \) where \( j = p + 1, \ldots, m \) might be the most difficult part of the problem → make feasible part of the search space extremely small compared to the whole search space.

- In this case we need to relax the equality constraints to inequality constraints as follows:
  \[
  |h_j(x)| \leq \delta, \quad j = p + 1, \ldots, m.
  \]
  where \( \delta \) is the tolerance value predefined by the users.

- \( \delta = 0.001 \) or \( \delta = 0.0001 \) are commonly suggested fixed values.

**Constraint handling strategies – summary**

- Two main groups of approaches:
  1. a special encoding and decoding procedure and variation operators to ensure that the search is only in \( D \).
  2. application of penalty coefficients to combine the objective value and the constraint violation and then change constrained optimization problems into unconstrained problems.

- The third group separates the objective value and the constraint violation.

- In the parental selection or replacement procedure, when comparisons are necessary between every two individuals, special rules are suggested, e.g.:
  1. Between two feasible solutions → the one with the higher quality wins.
  2. Between one feasible solution and one infeasible → the feasible one wins.
  3. Between two infeasible solutions → the one with the smaller overall constraint violation wins.

- Designing a Constrained Optimization Evolutionary Algorithm is the art of trade-off between feasible and infeasible individuals.

**Fitness Sharing and Other Niching Methods**

**Introduction**

- Traditional evolutionary algorithms with elitist selection converge to a single solution of the search space. However, ... real optimization problems — lead to multimodal domains and so require the identification of multiple optima, either global or local.

- Niching methods extend simple EA’s by promoting the formation of stable subpopulations in the neighborhood of optimal solutions.

- developed to reduce the effect of genetic drift resulting from the selection operator in the standard GA.

  Genetic drift means that the GA may quickly lose most of its genetic diversity and the search proceeds in a way that is not beneficial for crossover. This is because the random initial population quickly converges.

  - maintain population diversity and permit the GA to investigate many peaks in parallel.
  - based on the mechanics of natural ecosystems — in nature, animals compete to survive:  
    1. A niche can be viewed as a subspace in the environment that can support different types of life.
    2. For each niche, the physical resources are finite and must be shared among the population of that niche.
Fitness Sharing and Other Niching Methods

Introduction

- By analogy, niching methods tend to achieve a natural emergence of niches and species in the environment (search space).
- A niche is commonly referred to as an optimum of the domain, the fitness representing the resources of that niche.
- Species (a group of individuals with similar biological features capable of interbreeding among themselves) can be defined as similar individuals in terms of similarity metrics.
- Niching approaches:
  1. fitness sharing
  2. crowding
  3. clearing

1. Fitness Sharing

- Sharing must be implemented with the less restricted operators to promote stability of subpopulations (crossovers between individuals of different niches often lead to poor individuals — mating restriction schemes are needed).

1. Fitness Sharing

- Sharing can be done at genotypic or phenotypic levels. For example:
  1. Genotypic level: Hamming distance
  2. Phenotypic level: Euclidean distance

The key is how to define the “distance” (i.e., similarity).

Fitness Sharing and Other Niching Methods

1. Fitness Sharing

- Sharing function $sh(d_{ij})$:

$$sh(d_{ij}) = \begin{cases} 
1 - \frac{(d_{ij})^\alpha}{\alpha} & \text{if } d_{ij} < s_s \\
0 & \text{otherwise}
\end{cases}$$

where:

- $s_s$ — the threshold of dissimilarity (also the distance cutoff or the niche radius),
- $\alpha$ — regulates the shape of the sharing function (commonly set to one).

Fitness Sharing and Other Niching Methods

1. Fitness Sharing

- Sharing modifies the search landscape by reducing the payoff in densely populated regions, briefly, transforms the raw fitness of an individual into the shared one (usually lower).
- decreases individual’s fitness by an amount nearly equal to the number of similar individuals in the population.

The idea is that there is only limited and fixed amount of “resources” (i.e., fitness value) available at each niche. Individuals occupying the same niche will have to share the resources.

Fitness Sharing and Other Niching Methods

1. Fitness Sharing

- Sharing function $sh(d_{ij})$:

$$sh(d_{ij}) = \begin{cases} 
1 - \frac{(d_{ij})^\alpha}{\alpha} & \text{if } d_{ij} < s_s \\
0 & \text{otherwise}
\end{cases}$$

where:

- $s_s$ — the threshold of dissimilarity (also the distance cutoff or the niche radius),
- $\alpha$ — regulates the shape of the sharing function (commonly set to one).
1. **Fitness Sharing**

   **Limitations**
   - Sharing radius, $\sigma_s$, can be difficult to set:
     1. Setting the dissimilarity threshold requires a priori knowledge of how far apart the optima are.
     2. $\sigma_s$ is the same for all individuals — all peaks must be nearly equidistant in the domain.
   - The sharing scheme is very expensive as a result of the computation of niche counts of complexity $O(N^2)$ per generation.
   - Fitness sharing as described above may not work well.

2. **Crowding**

   - **Standard Crowding**
     1. Only a fraction of the global population specified by a percentage $G$ (generation gap) reproduces and dies each generation.
     2. An offspring replaces the most similar individual taken from a randomly drawn subpopulation of size $CF$ (crowding factor) from the global population.
     3. Drawback: great number of replacement errors.
   - **Deterministic Crowding**
     1. Each child replaces the nearest parent if it has a higher fitness.
     2. Two sets of tournaments: (parent 1 against child 1, and parent 2 against child 2) or (parent 1 against child 2, and parent 2 against child 1).
     3. The set of tournament that yields the closest competitions is held.

   **Fitness Scaling – Dilemma**
   - With a low scaling factor: individuals won’t converge to the real optima because they are not attractive.
   - With a high scaling factor: “super individuals” in initial populations may dominate the population quickly. The evolution may not be able to locate all peaks.
   - Solutions:
     1. Large population,
     2. Soft selection,
     3. Anneal $\beta$, i.e., starting from $\beta = 1$ and increasing it gradually.

   **Fitness Scaling**
   
   \[
   F_{\text{share}}(x) = \frac{F(x)^\beta}{m_1}
   \]
   where $\beta > 1$ is a scaling factor.

   **Deterministic Crowding**
   - Initially select two elements from the population to undergo crossover and mutation.
   - After recombination, a random sample of $CF$ individuals is taken from the population as in standard crowding.
   - Each offspring competes with the closest sample element.
   - The winners are inserted in the population.
   - This procedure is repeated $N/2$ times.
3. Clearing

Instead of sharing the resources between all individuals of a single subpopulation, only the best members of the subpopulation are attributed to the resources of a niche — the capacity $k$ of a niche specifies the maximum number of elements that this niche can accept.

How?

- the fitness of the dominant individuals is preserved,
- the fitness of all the other individuals of the same subpopulation is reset to zero.

**Summary:**

- Niching techniques enable us to find multiple peaks simultaneously in evolution.
- Every niching technique has its own “niche.” There is no single best niching method for all problems.
- All fitness sharing techniques transform fitness values.
- Population size becomes an important parameter when fitness sharing is used.

Niching techniques are still a subject of interest:

2017 IEEE Congress on Evolutionary Computation in Donostia-San Sebastian, Spain

Competition on:

Niching Methods for Multimodal Optimization

organized in association with the 2017 IEEE CEC Special Session on Niching Methods for Multimodal Optimization.

- 20 benchmark multimodal functions with different characteristics and levels of difficulty
- Test suite for the competition as well as the performance measure are implemented in Matlab, Java and C++.

http://www.epitropakis.co.uk/cec17-niching/competition/index.html

Thank you.