technische universität dortmund

Computational Intelligence

Winter Term 2020/21

Prof. Dr. Günter Rudolph

Lehrstuhl für Algorithm Engineering (LS 11)

Fakultät für Informatik

TU Dortmund

- Design of Evolutionary Algorithms
  - Case Study: Integer Search Space
  - Towards CMA-ES

ad 2) **design guidelines for variation operators in practice**

integer search space $X = \mathbb{Z}^n$

a) reachability

b) unbiasedness

c) control

- every recombination results in some $z \in \mathbb{Z}^n$
- mutation of z may then lead to any $z^* \in \mathbb{Z}^n$ with positive probability in one step

ad a) support of mutation should be $\mathbb{Z}^n$

ad b) need maximum entropy distribution over support $\mathbb{Z}^n$

ad c) control variability by parameter

→ formulate as constraint of maximum entropy distribution

ad 2) **design guidelines for variation operators in practice** $X = \mathbb{Z}^n$

task: find (symmetric) maximum entropy distribution over $\mathbb{Z}$ with $E[\,|Z|\,] = \theta > 0$

⇒ need *analytic* solution of an ∞-dimensional, nonlinear optimization problem with constraints!

$$H(p) = -\sum_{k=-\infty}^{\infty} p_k \log p_k \longrightarrow \text{max!}$$

s.t.

$$p_k = p_{-k} \quad \forall k \in \mathbb{Z}\,, \qquad \text{(symmetry w.r.t. 0)}$$

$$\sum_{k=-\infty}^{\infty} p_k = 1\,, \qquad \text{(normalization)}$$

$$\sum_{k=-\infty}^{\infty} |k|\, p_k = \theta \qquad \text{(control “spread“)}$$

$$p_k \geq 0 \quad \forall k \in \mathbb{Z}\,. \qquad \text{(nonnegativity)}$$

result:

a random variable Z with support $\mathbb{Z}$ and probability distribution

$$p_k := P\{Z = k\} = \frac{q}{2-q}(1-q)^{|k|} \ , \ k \in \mathbb{Z}, \ q \in (0,1)$$

symmetric w.r.t. 0, unimodal, spread manageable by *q* and has max. entropy ■

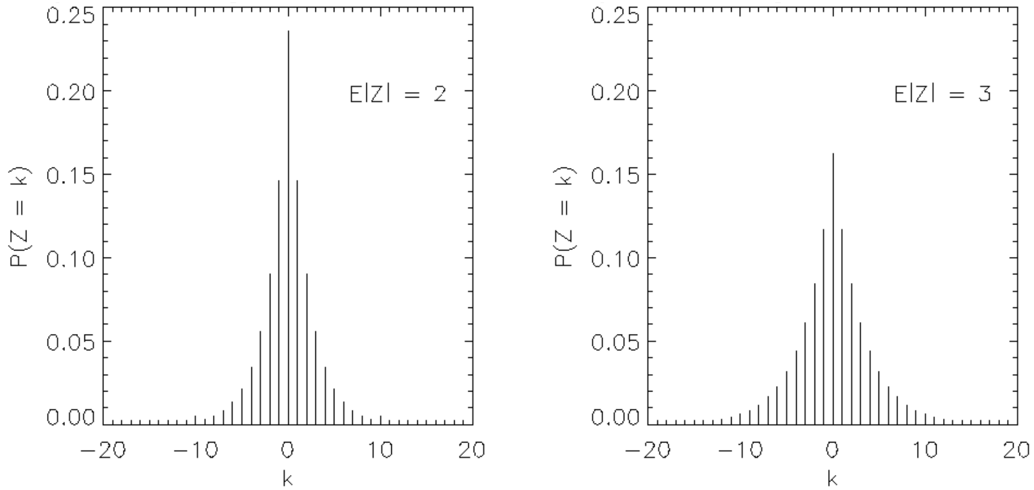
generation of pseudo random numbers: $Z = G_1 - G_2$

where

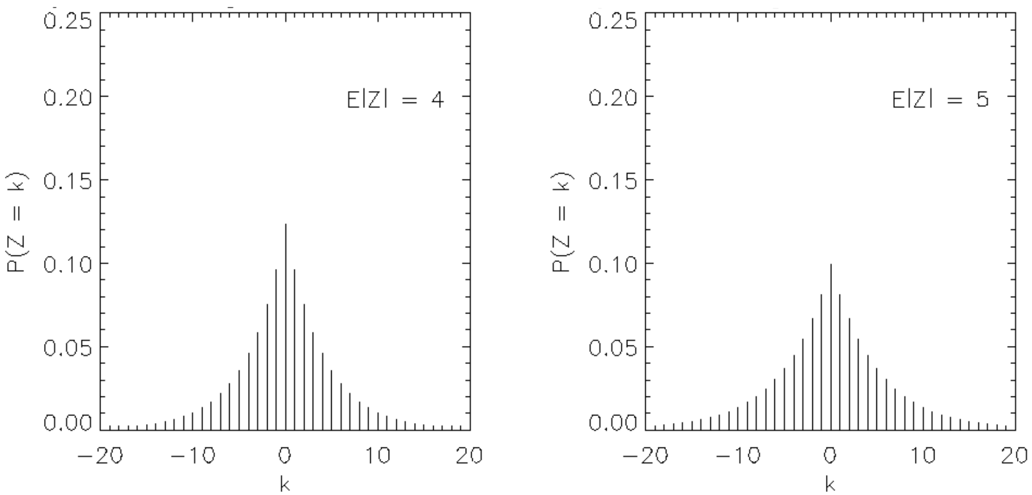
$$U_i \sim U(0,1) \ \Rightarrow \ G_i = \left\lfloor \frac{\log(1-U_i)}{\log(1-q)} \right\rfloor \ , \ i = 1, \ 2.$$

stochastic independent!

probability distributions for different mean step sizes E|Z| = θ



probability distributions for different mean step sizes E|Z| = θ



How to control the spread?

We must be able to adapt q ∊ (0,1) for generating Z with variable E|Z| = θ !

self-adaptation of q in open interval (0,1) ?

⟶ make mean step size $E[\,|Z|\,]$ adjustable!

$$E[\,|Z|\,] = \sum_{k=-\infty}^{\infty} |k|\, p_k = \theta = \frac{2\,(1-q)}{q\,(2-q)} \quad \Leftrightarrow \quad q = 1 - \frac{\theta}{(1+\theta^2)^{1/2}+1}$$

$\theta \in \mathbb{R}_+$ → θ adjustable by $\underbrace{\text{mutative self adaptation}}$

$q \in (0,1)$ → get q from θ

like mutative step size size control of σ in EA with search space $\mathbb{R}^n$!

Mutative Step Size Control

Individual (x, θ) ∈ $\mathbb{Z}^n \times \mathbb{R}_+$

First, mutate step size $\theta_{t+1} = \theta_t \cdot L$ where L = exp(N) with N ~ N(0, 1/n)

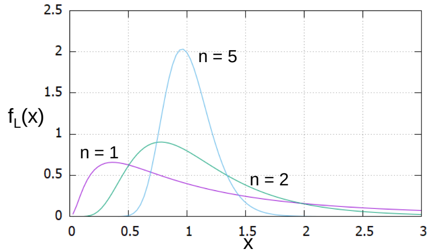
Second, mutate parent $Y = x + \theta_{t+1} \cdot Z$

log-normal distributed

P{ L > c } = P {L < 1/c } for c ≥ 1

Often: assure minimal step size ≥ 1

$\theta_{t+1} = \max\{ 1, \theta_t \cdot L \}$



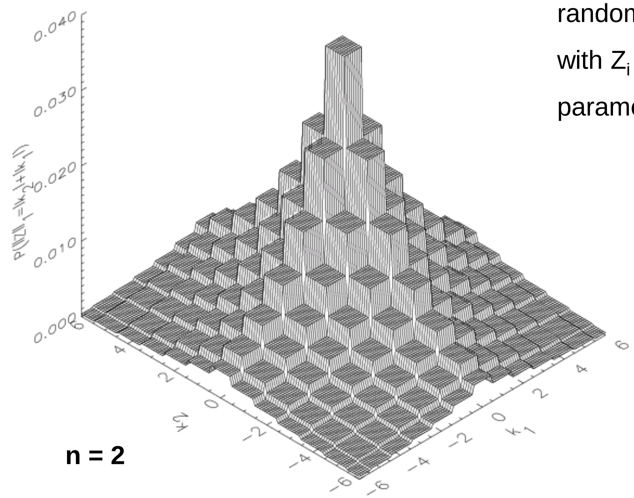
→ invented: Schwefel (1977) for real variables
→ transferred: Rudolph (1994) for integer variables

n - dimensional generalization

random vector $Z = (Z_1, Z_2, \ldots Z_n)$

with $Z_i = G_{1,i} - G_{2,i}$ (stoch. indep.);

parameter q for all G_{1i}, G_{2i} equal



n = 2

n - dimensional generalization

$$\boxed{P\{Z_i = k\} = \frac{q}{2-q}(1-q)^{|k|}}$$

$$P\{Z_1 = k_1, Z_2 = k_2, \ldots, Z_n = k_n\} = \prod_{i=1}^{n} P\{Z_i = k_i\} =$$

$$\left(\frac{q}{2-q}\right)^n \prod_{i=1}^{n} (1-q)^{|k_i|} = \left(\frac{q}{2-q}\right)^n (1-q)^{\sum_{i=1}^{n} |k_i|}$$

$$= \left(\frac{q}{2-q}\right)^n (1-q)^{\|k\|_1} .$$

⇒ n-dimensional distribution is symmetric w.r.t. ℓ_1 norm!

⇒ all random vectors with same step length have same probability!

How to control E[|| Z ||₁] ?

$$E[\,\|Z\|_1\,] = E\left[\sum_{i=1}^{n} |Z_i|\right] = \sum_{i=1}^{n} E[|Z_i|] = n \cdot E[|Z_1|]$$

by def. linearity of E[·] identical distributions for Z_i

$$\underbrace{n \cdot E[\,|Z_1|\,]}_{=\theta} = n \cdot \frac{2\,(1-q)}{q\,(2-q)} \Leftrightarrow q = 1 - \frac{\theta/n}{(1+(\theta/n)^2)^{1/2}+1}$$

self-adaptation

calculate from θ

Algorithm:

individual : $(x, \theta) \in \mathbb{Z}^n \times \mathbb{R}_+$

mutation : $\theta^{(t+1)} = \theta^{(t)} \cdot \exp(N), \quad N \sim N(0, 1/n).$

if $\theta^{(t+1)} < 1$ then $\theta_{t+1} = 1$

calculate new q for G_i from θ_{t+1}

$\forall j = 1, \ldots, n : X_j^{(t+1)} = X_j^{(t)} + (G_{1,j} - G_{2,j})$

recombination : discrete (uniform crossover)

selection : (μ, λ)-selection

(Rudolph, PPSN 1994)

Example: (1, λ)-EA with λ = 10; f(x) = x‘x → min! ; n = 10

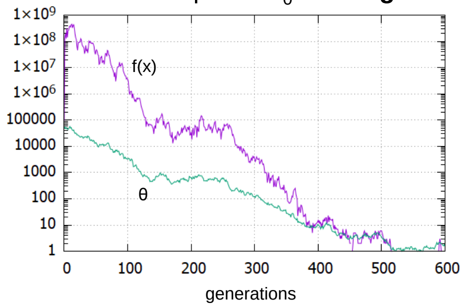
$X^{(0)} \in [100, 101]^n \cap \mathbb{Z}^n$

θ_0 = 50 000

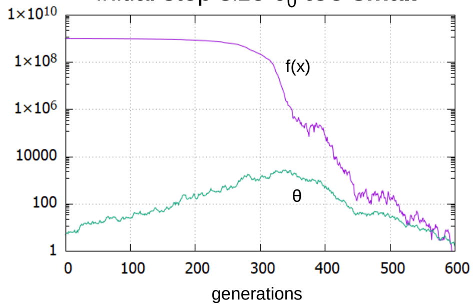
$X^{(0)} \in [10000, 10100]^n \cap \mathbb{Z}^n$

θ_0 = 5

initial step size θ_0 **too large**



initial step size θ_0 **too small**



ad 2) **design guidelines for variation operators in practice**

continuous search space X = $\mathbb{R}^n$

a) reachability → mutation distribution with unbounded support

b) unbiasedness → mutation distribution with maximum entropy

c) control → mutation distribution with parameters

⇒ leads to CMA-ES !

↓

Covariance
Matrix
Adaptation

mutation: Y = X + Z   Z ~ N(0, C) multinormal distribution

↓

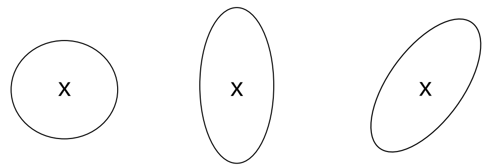
maximum entropy distribution for support $\mathbb{R}^n$, given expectation vector and covariance matrix

how should we choose covariance matrix C?

unless we have not learned something about the problem during search

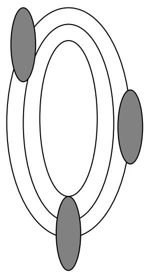
⇒ don‘t prefer any direction!

⇒ covariance matrix C = I_n (unit matrix)



$C = I_n$ $C = \mathrm{diag}(s_1, \ldots, s_n)$ C orthogonal

claim: mutations should be aligned to isolines of problem (Schwefel 1981)



if true then covariance matrix should be inverse of Hessian matrix!

$\Rightarrow$ assume f(x) $\approx$ ½ x'Ax + b'x + c $\Rightarrow$ H = A

Z ~ N(0, C) with density

$$f_Z(x) = \frac{1}{(2\pi)^{n/2}\,|C|^{1/2}} \exp\left(-\frac{1}{2}x'C^{-1}x\right)$$

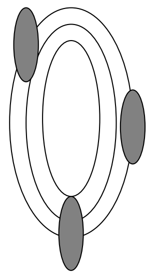
since then many proposals how to adapt the covariance matrix

$\Rightarrow$ extreme case: use n+1 pairs (x, f(x)),

apply multiple linear regression to obtain estimators for A, b, c

invert estimated matrix A! OK, **but:** $O(n^6)$! (Rudolph 1992)

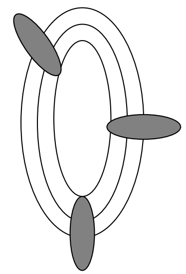
doubts: are equi-aligned isolines really optimal?



principal axis

should point into negative gradient direction!

(proof next slide)



most (effective) algorithms behave like this:

run roughly into negative gradient direction,
sooner or later we approach longest main principal axis of Hessian,

now negative gradient direction coincidences with direction to optimum,
which is parallel to longest main principal axis of Hessian,
which is parallel to the longest main principal axis of the inverse covariance matrix

(Schwefel OK in this situation)

Z = rQu, A = B'B, B = Q^{-1}

$$\begin{aligned}
f(x + r\boxed{Qu}) &= \tfrac{1}{2}(x + rQu)'A\,(x + rQu) + b'(x + rQu) + c \\
&= \tfrac{1}{2}(x'Ax + 2\,rx'AQu + r^2u'Q'AQu) + b'x + rb'Qu + c \\
&= f(x) + rx'AQu + rb'Qu + \tfrac{1}{2}r^2u'Q'AQu \\
&= f(x) + r(Ax + b + \tfrac{r}{2}AQu)'Qu \\
&= f(x) + r(\nabla f(x) + \tfrac{r}{2}AQu)'Qu \\
&= f(x) + r\,\nabla f(x)'Qu + \frac{r^2}{2}\,u'Q'AQu \\
&= f(x) + r\,\nabla f(x)'\boxed{Qu} + \frac{r^2}{2} \qquad \to \text{min!}
\end{aligned}$$

if $\boxed{\text{Qu}}$ were deterministic ...

$\Rightarrow$ set Qu = $-\nabla f(x)$ (direction of steepest descent)

Apart from (inefficient) regression, how can we get matrix elements of Q?

$\Rightarrow$ iteratively: $C^{(k+1)}$ = update($C^{(k)}$, Population$^{(k)}$)

basic constraint: $C^{(k)}$ must be positive definite (p.d.) and symmetric for all k ≥ 0,

otherwise Cholesky decomposition impossible: C = Q'Q

Lemma

Let A and B be quadratic matrices and $\alpha, \beta > 0$.

a) A, B symmetric $\Rightarrow \alpha A + \beta B$ symmetric.

b) A positive definite and B positive semidefinite $\Rightarrow \alpha A + \beta B$ positive definite

Proof:

ad a) $C = \alpha A + \beta B$ symmetric, since $c_{ij} = \alpha a_{ij} + \beta b_{ij} = \alpha a_{ji} + \beta b_{ji} = c_{ji}$

ad b) $\forall x \in \mathbb{R}^n \setminus \{0\}$: $x'(\alpha A + \beta B)\,x = \underbrace{\alpha\, x'Ax}_{>0} + \underbrace{\beta\, x'Bx}_{\geq 0} > 0$ ■

Theorem

A quadratic matrix $C^{(k)}$ is symmetric and positive definite for all $k \ge 0$,

if it is built via the iterative formula $\mathbf{C^{(k+1)} = \alpha_k C^{(k)} + \beta_k v_k v'_k}$

where $C^{(0)} = I_n$, $v_k \ne 0$, $\alpha_k > 0$ and $\liminf \beta_k > 0$.

Proof:

If $v \ne 0$, then matrix $V = vv'$ is symmetric and positive semidefinite, since

- as per definition of the dyadic product $v_{ij} = v_i \cdot v_j = v_j \cdot v_i = v_{ji}$ for all i, j and
- for all $x \in \mathbb{R}^n$: $x' (vv') x = (x'v) \cdot (v'x) = (x'v)^2 \ge 0$.

Thus, the sequence of matrices $v_k v'_k$ is symmetric and p.s.d. for $k \ge 0$.

Owing to the previous lemma matrix $C^{(k+1)}$ is symmetric and p.d., if

$C^{(k)}$ is symmetric as well as p.d. and matrix $v_k v'_k$ is symmetric and p.s.d.

Since $C^{(0)} = I_n$ symmetric and p.d. it follows that $C^{(1)}$ is symmetric and p.d.

Repetition of these arguments leads to the statement of the theorem. ■

Idea: Don't estimate matrix **C** in each iteration! Instead, approximate iteratively!

(Hansen, Ostermeier et al. 1996ff.)

→ **C**ovariance **M**atrix **A**daptation **E**volutionary **A**lgorithm (CMA-EA)

Set initial covariance matrix to $C^{(0)} = I_n$

$$C^{(t+1)} = (1-\eta)\, C^{(t)} + \eta \sum_{i=1}^{\mu} w_i\, (x_{i:\lambda} - m^{(t)})\, (x_{i:\lambda} - m^{(t)})'$$

η : "learning rate" $\in (0,1)$

w_i : weights; mostly $1/\mu$

$$m = \frac{1}{\mu} \sum_{i=1}^{\mu} x_{i:\lambda}$$ mean of all selected parents

complexity: $\mathbf{O}(\mu n^2 + n^3)$

sorting: $f(x_{1:\lambda}) \le f(x_{2:\lambda}) \le \ldots \le f(x_{\lambda:\lambda})$

Caution: must use mean $m^{(t)}$ of "old" selected parents; not „new" mean $m^{(t+1)}$!

⇒ Seeking covariance matrix of fictitious distribution pointing in gradient direction!

State-of-the-art: **CMA-EA** (currently many variants)

→ many successful applications in practice

C, C++, Java
Fortran, Python,
Matlab, R, Scilab

available in WWW:

- `http://cma.gforge.inria.fr/cmaes_sourcecode_page.html`
- `http://image.diku.dk/shark/` (EAlib, C++)
- …

advice:

before designing your own new method
or grabbing another method with some fancy name ...
try CMA-ES – it is available in most software libraries and often does the job!