Evolution strategies

- Proposed in 60’s as experimental optimization methods
- Based on real coding
- Very fast with good support of theory
- First usage of *self-adaptation* (change of itself) as a standard accessory

The most simple ES

$t = 0$

Create P, evaluate P

while (**not** stopping_criterion) {

$t = t+1$

$n_i = p_i + G(0,\sigma)$ (mutation)

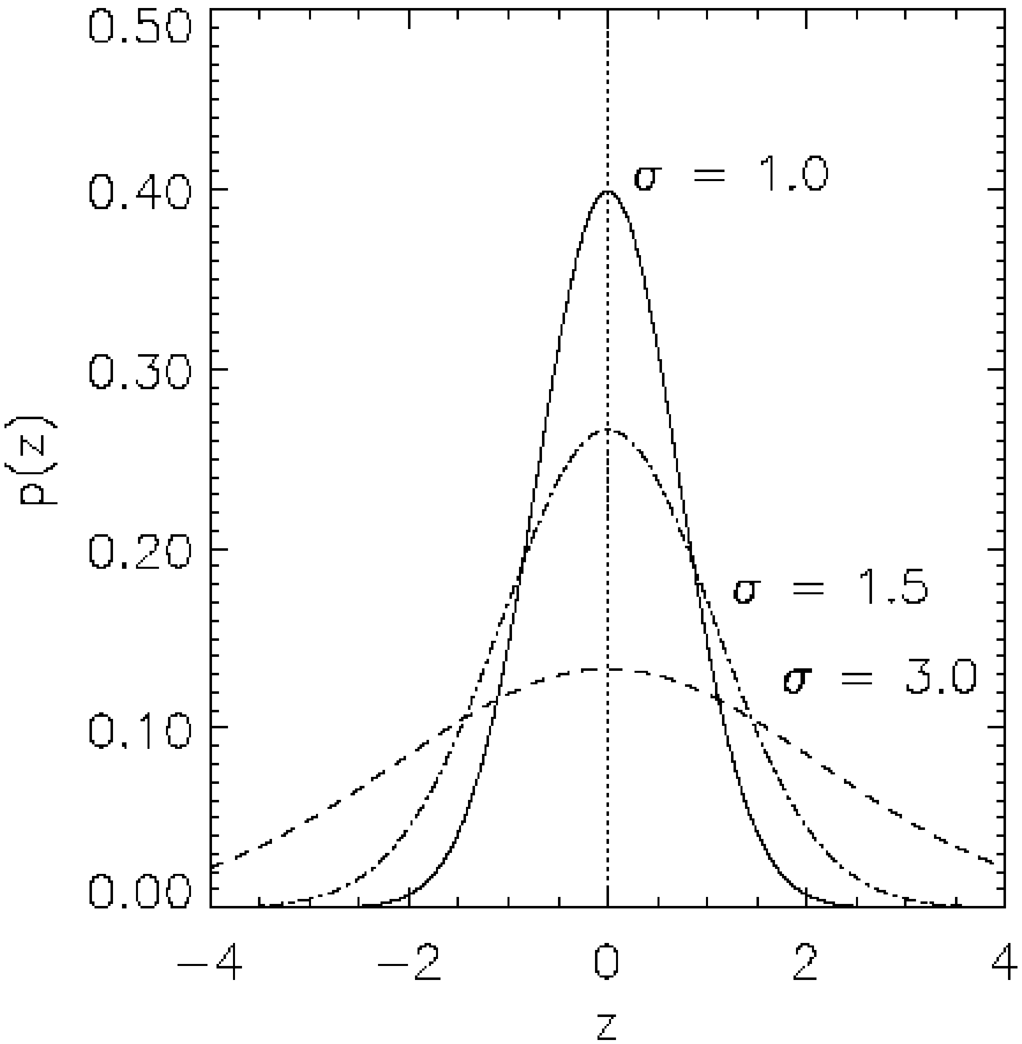
If N is better than P, then P=N

}

Mutation

- Standard deviation σ is called „degree of mutation“
- σ can be changed by “1/5 successful rule”:
- σ is changed every *k* iterations
  - $\sigma = \sigma / c$ iff $p_s > 1/5$
  - $\sigma = \sigma \cdot c$ iff $p_s < 1/5$
  - $\sigma = \sigma$ iff $p_s = 1/5$
- where p_s is % of successful mutations, $0.8 \leq c \leq 1$

Illustration of Gaussian distribution



Self-adaptation

- Included as degree of mutation into the solution
- Several possibilities:
  - One σ for all solutions
  - One independent σ for every solution
  - Correlated standard deviations

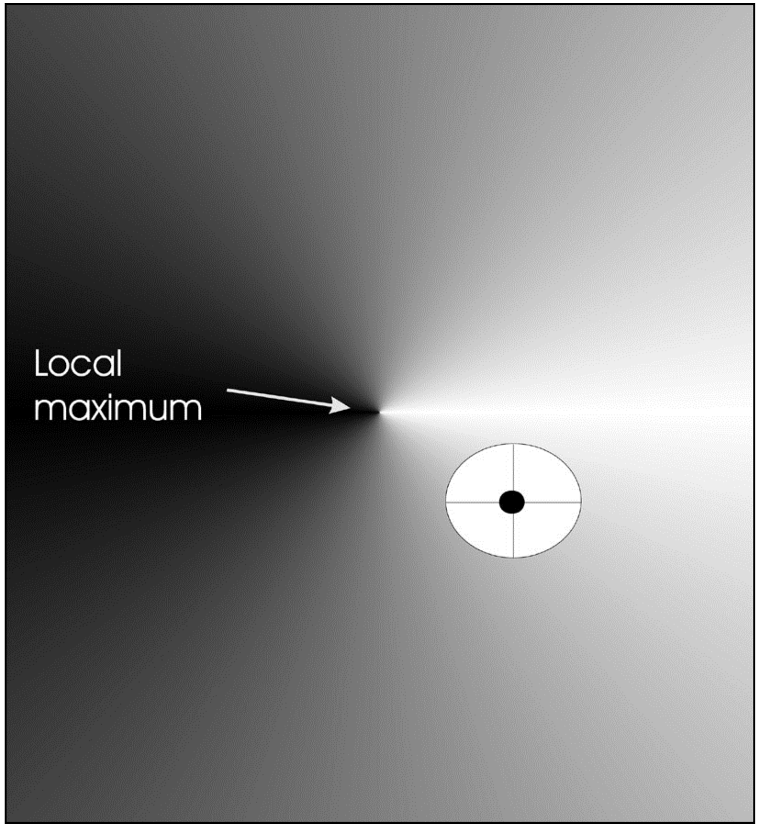
Mutation ①

- Solution: $\langle x_1,\ldots,x_n, \sigma \rangle$
- $\sigma' = \sigma \cdot \exp(\tau \cdot N(0,1))$
- $x'_i = x_i + \sigma' \cdot N(0,1)$
- Typical "learning rate"

$$\tau \propto 1/\, n^{½}$$

- Bound for minimal value $\sigma' < \varepsilon_0 \Rightarrow \sigma' = \varepsilon_0$

Mutation ①

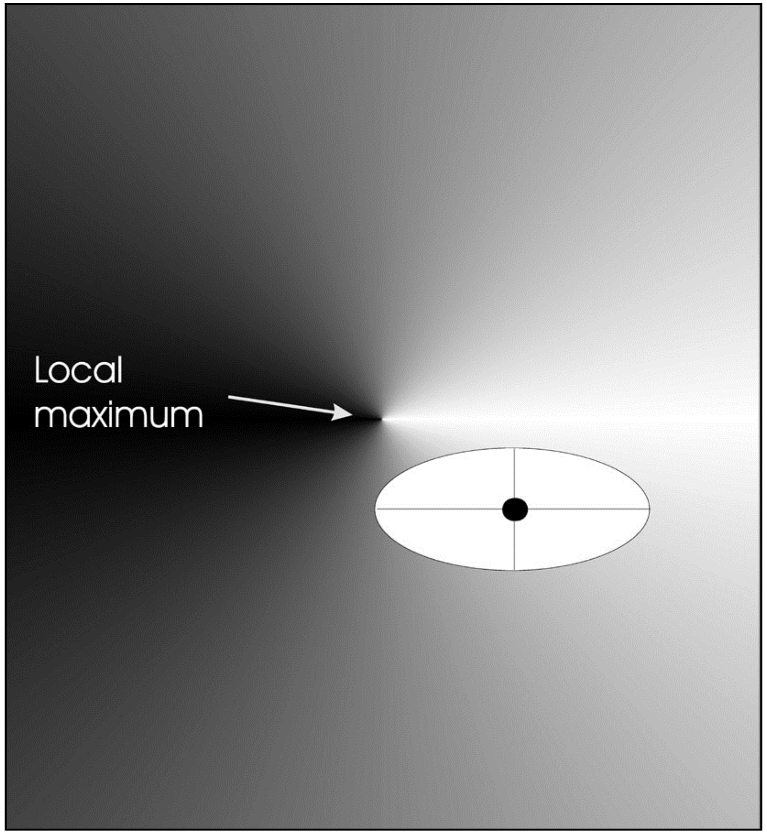


The circle represents points with the same probability of mutation

Mutation ②

- Solution: $\langle x_1,\ldots,x_n, \sigma_1,\ldots, \sigma_n \rangle$
- $\sigma'_i = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1))$
- $x'_i = x_i + \sigma'_i \cdot N_i(0,1)$
- Two learning parameters:
  - τ' overall
  - τ for individual variables
- $\tau' \propto 1/(2n)^{½}$ and $\tau \propto 1/(2n^{½})^{½}$
- And also $\sigma_i' < \varepsilon_0 \Rightarrow \sigma_i' = \varepsilon_0$

Mutation ②



The ellipse represents points with the same probability of mutation

Mutation ③

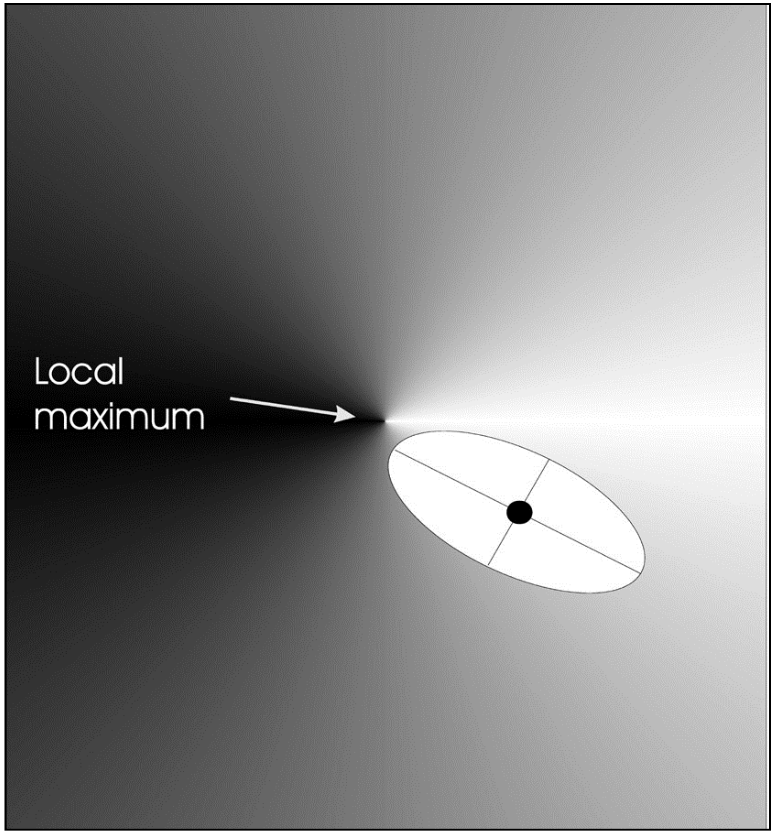
- Solution: $\langle x_1,\ldots,x_n, \sigma_1,\ldots, \sigma_n ,\alpha_1,\ldots, \alpha_k \rangle$
- where $k = n \cdot (n-1)/2$
- Covariant matrix given by:
  - $c_{ii} = \sigma_i^2$
  - $c_{ij} = 0$ if i and j are not correlated
  - $c_{ij} = ½ \cdot (\sigma_i^2 - \sigma_j^2) \cdot \tan(2 \alpha_{ij})$ if i and j are correlated

Mutation ③

Then the mutation is given by:

- $\sigma'_i = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1))$
- $\alpha'_j = \alpha_j + \beta \cdot N(0,1)$
- $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{N}(\boldsymbol{0},\boldsymbol{C}')$
  - $\boldsymbol{C}'$ is covariant matrix $\boldsymbol{C}$ after mutation of α values
- $\tau' \propto 1/(2n)^{1/2}$ and $\tau \propto 1/(2n^{1/2})^{1/2}$ and $\beta \approx 5°$
- $\sigma_i' < \varepsilon_0 \Rightarrow \sigma_i' = \varepsilon_0$ and
- $|\alpha'_j| > \pi \Rightarrow \alpha'_j = \alpha'_j - 2\pi\,\mathrm{sign}(\alpha'_j)$

Mutation ③



The ellipse represents points with the same probability of mutation

Comparison of mutation ② and ③

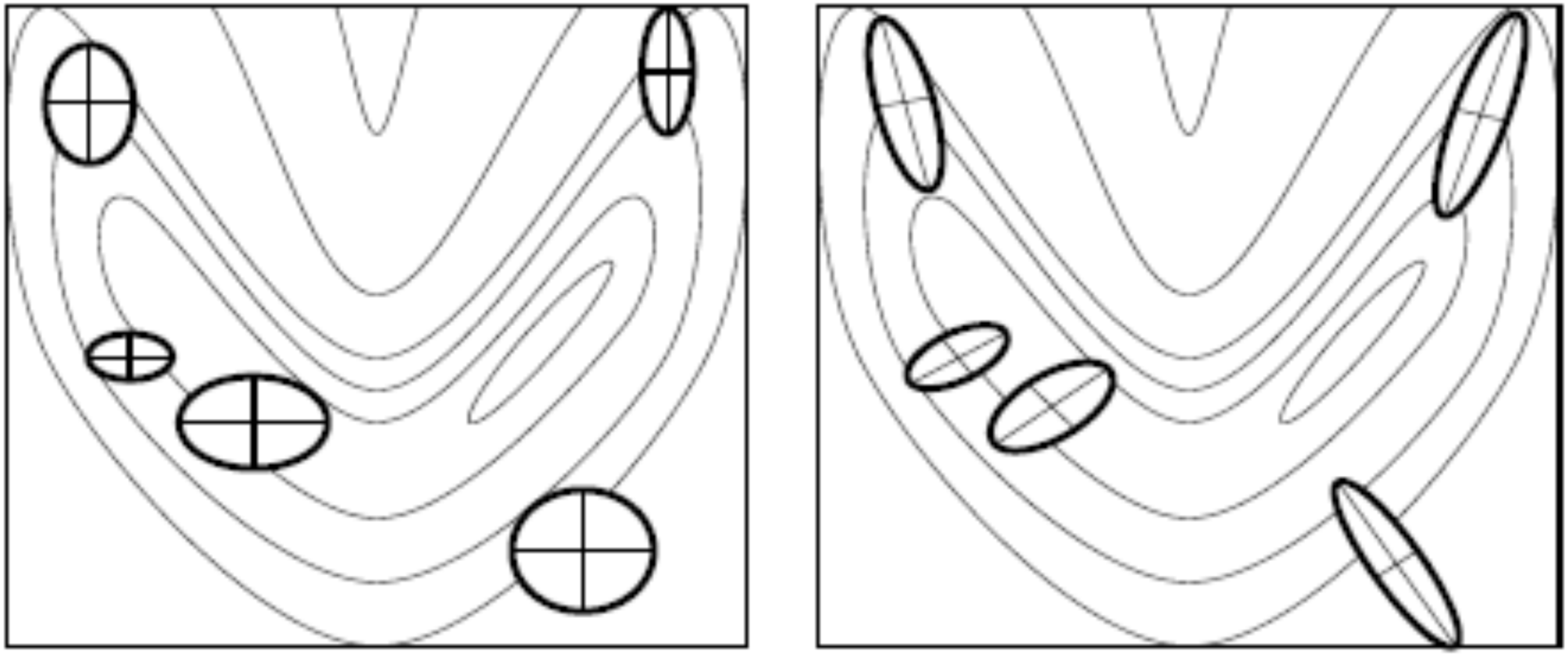
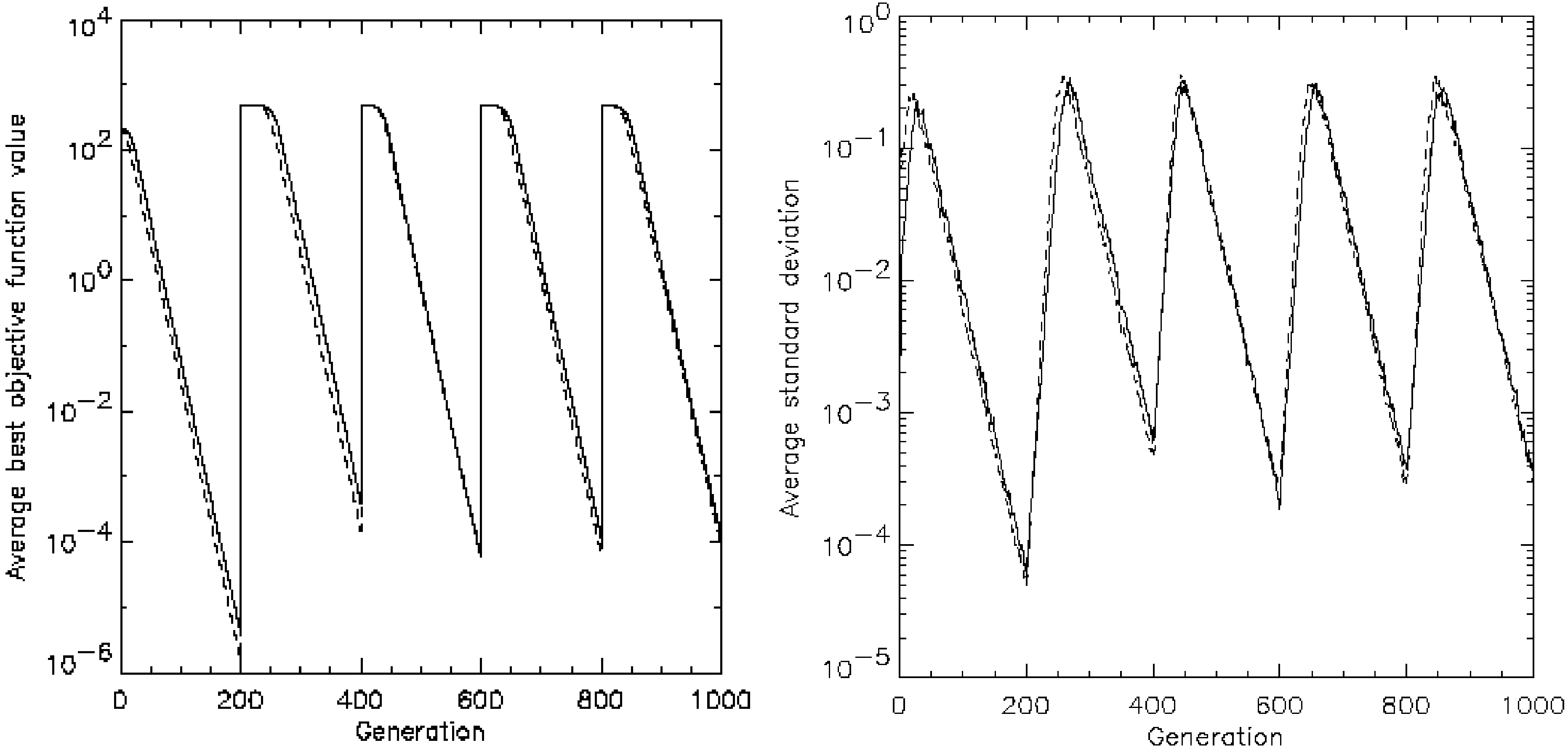
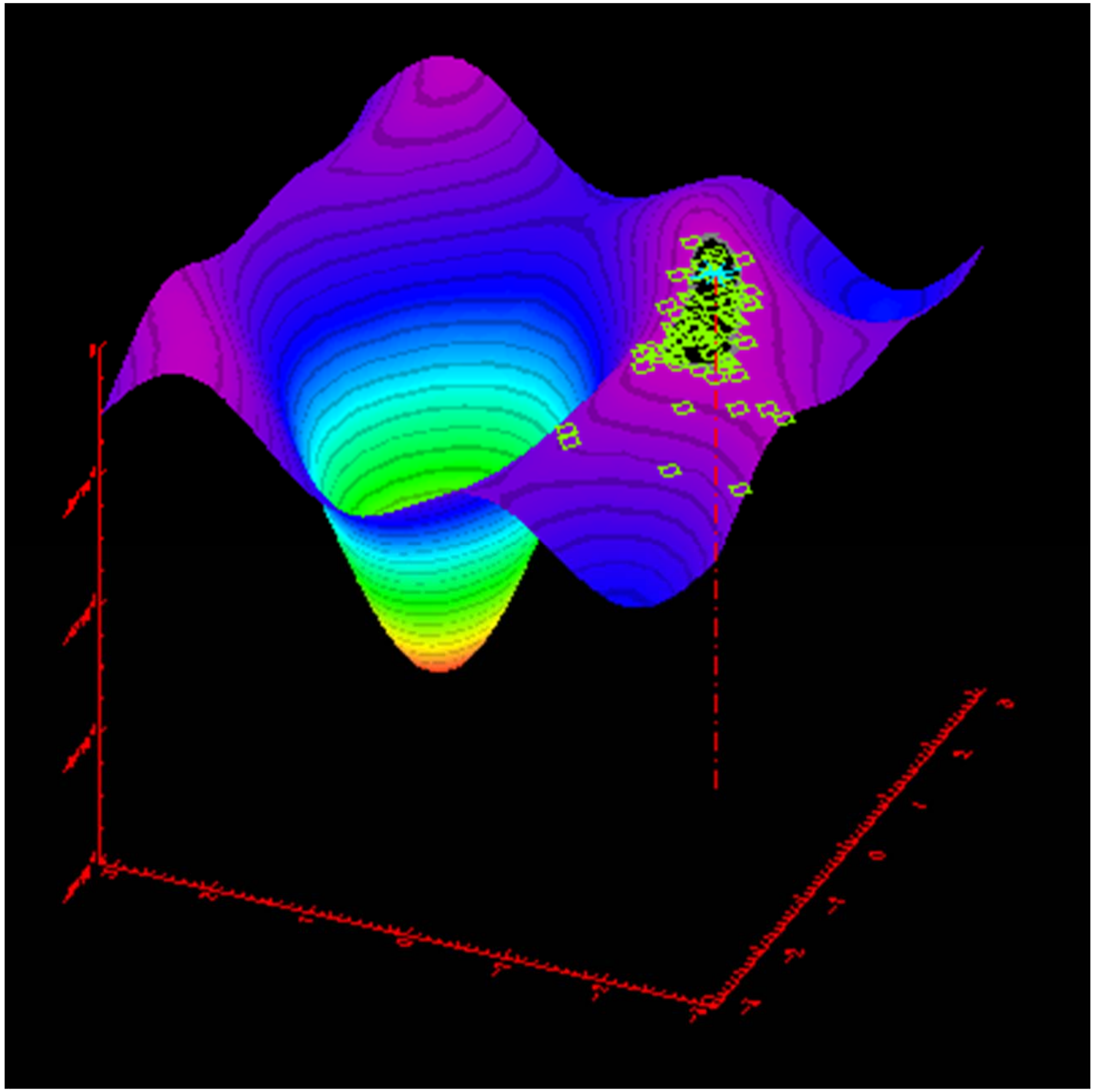


Illustration of self-adaptation



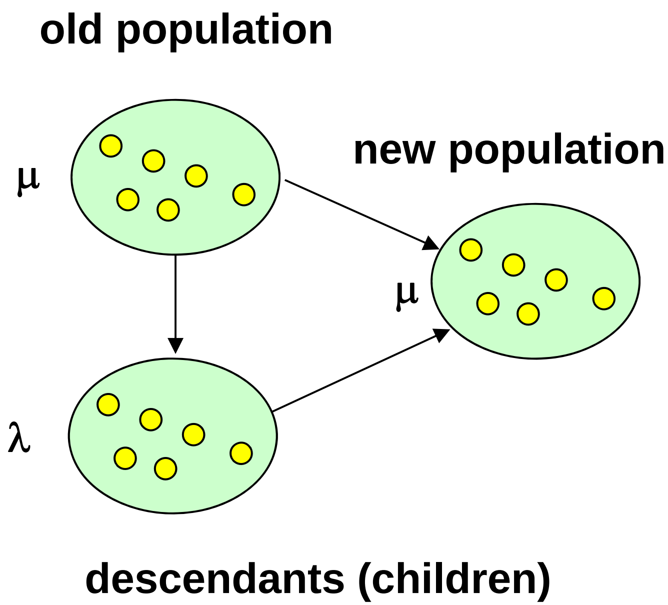
Change of dynamical objective function (left) and corresponding change of standard deviation (right)

Illustration of self-adaptation

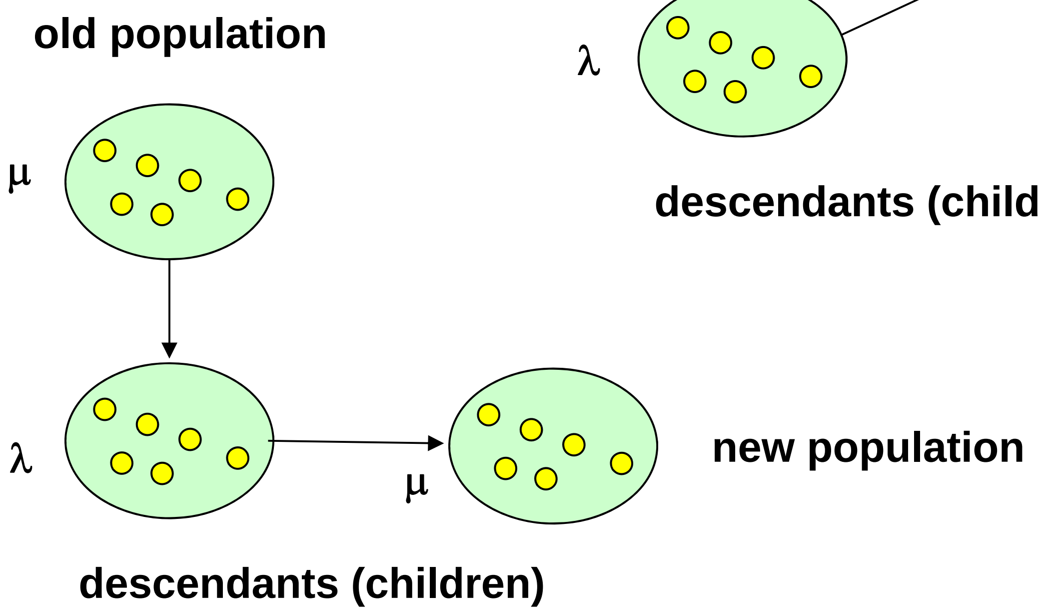


Notation of population-based ES's

- **Two types of algorithms:**



  - **(μ+λ)-ES**



  - **(μ,λ)-ES**

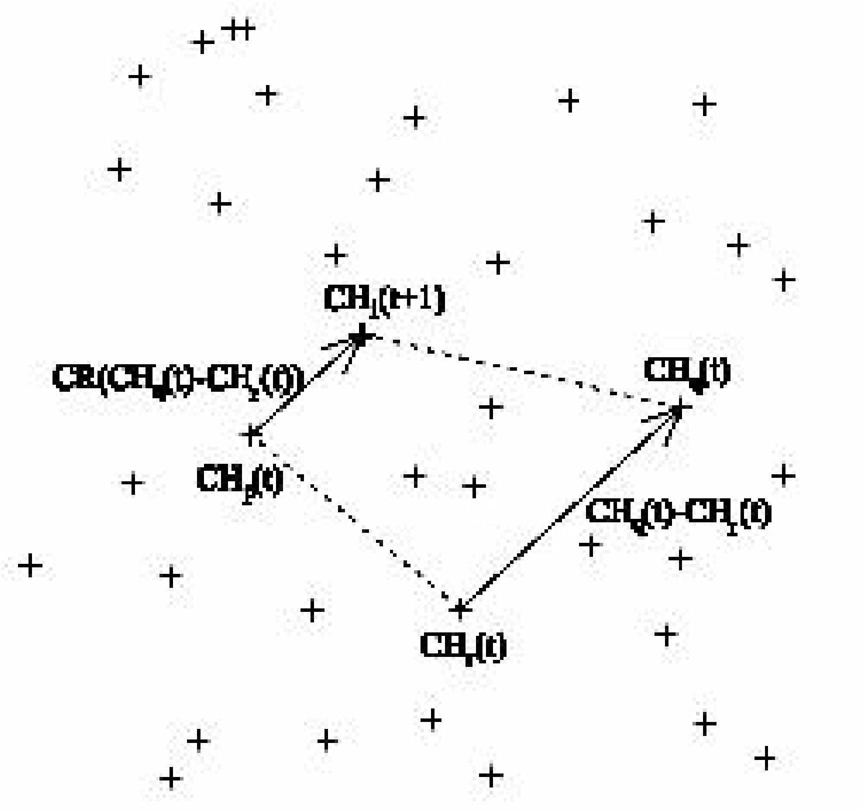
Recommendations

- $\mu > 1$ to ensure different strategies
- $\lambda > \mu$ to create different strategies
- Not so „strong“ selection, e.g., $\lambda \approx 7 \cdot \mu$
- (μ,λ)-selection, to kill solutions with bad σ

Differential Evolution

- New method, developed in 1995
- Operations on real numbers
- Population – based algorithm
- Only one operator and two tuning parameters
  => simple yet robust method

Differential operator

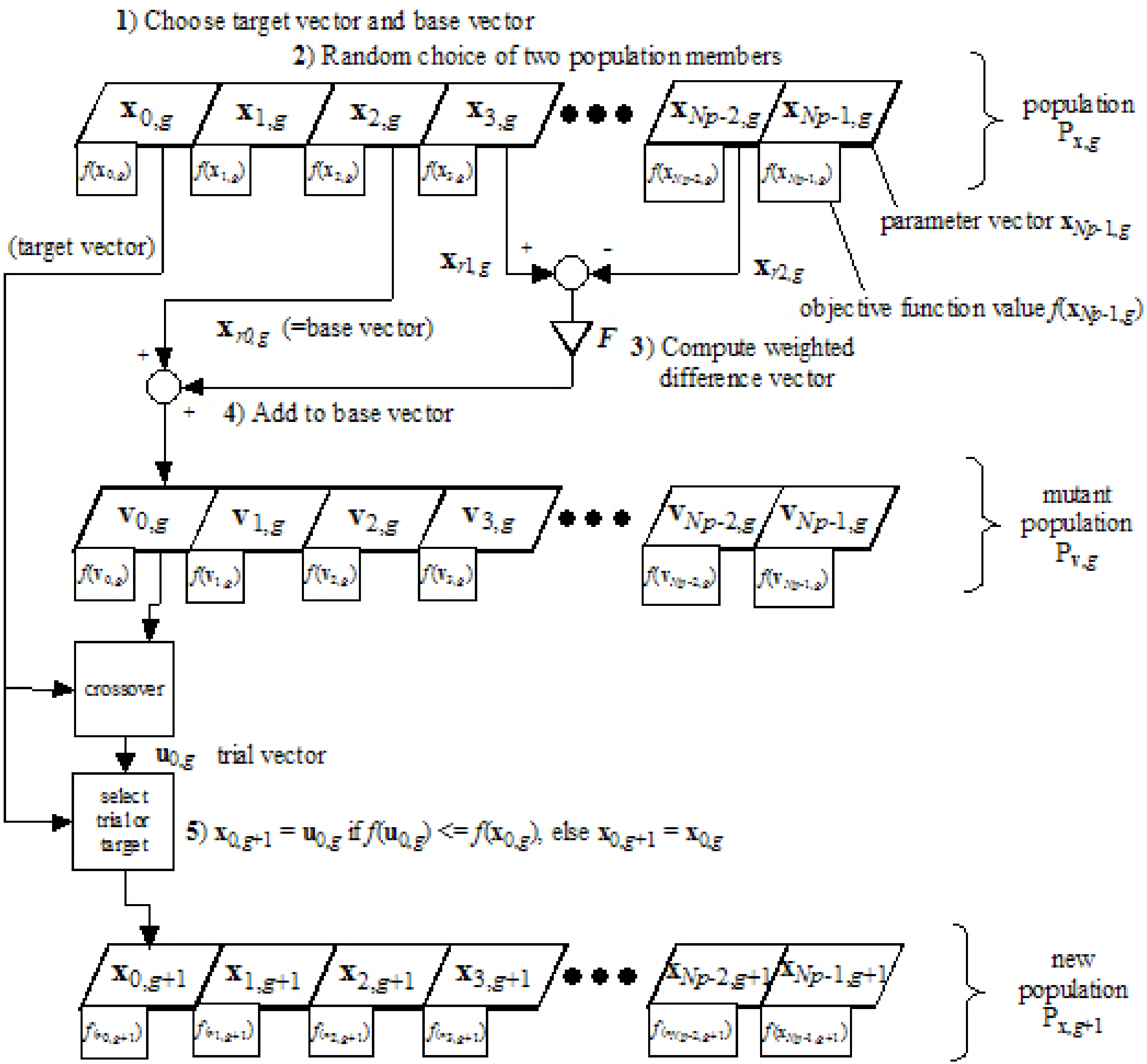


$$ch_{ij} = ch_{ij} + F_1(ch_{qj} - ch_{rj})$$
$$+ F_2(ch_{\text{best } j} - ch_{ij})$$

More variants exist

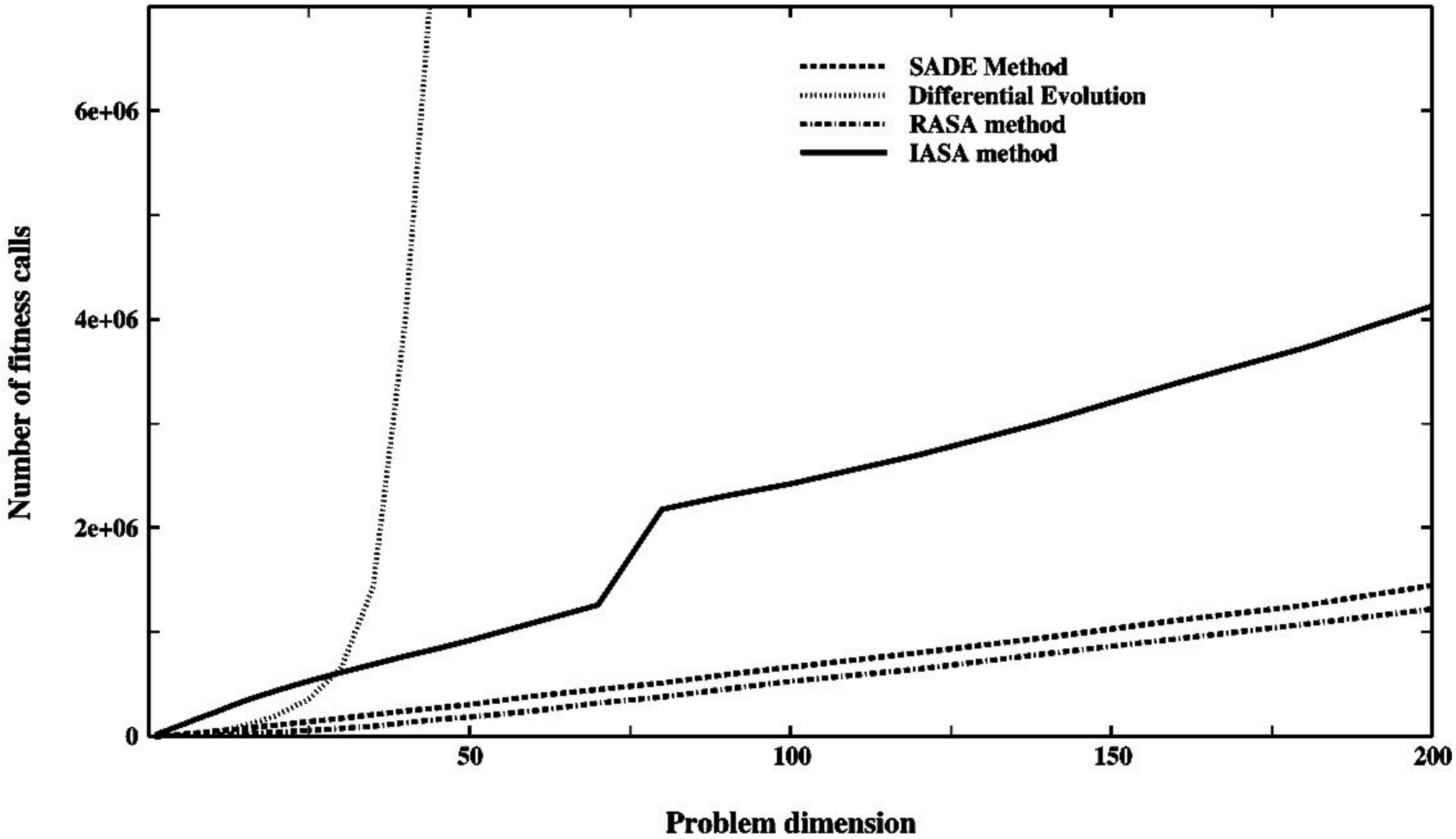
Random selection

New solution replaces given one only in case of improvement


1) Choose target vector and base vector
2) Random choice of two population members
population $P_{x,g}$
parameter vector $x_{Np-1,g}$
objective function value $f(x_{Np-1,g})$
(target vector)
$x_{r1,g}$
$x_{r2,g}$
$x_{r0,g}$ (=base vector)
F
3) Compute weighted difference vector
4) Add to base vector
mutant population $P_{v,g}$
crossover
$u_{0,g}$ trial vector
select trial or target
5) $x_{0,g+1} = u_{0,g}$ if $f(u_{0,g}) <= f(x_{0,g})$, else $x_{0,g+1} = x_{0,g}$
new population $P_{x,g+1}$

- Usually fast convergence, however has problems missing true mutation operator

***Type 0* function**



References

[1] Rechenberg, I. (1973). Evolution strategy: Optimization of technical systems by means of biological evolution. Fromman-Holzboog, Stuttgart.

[2] Schwefel, H.-P. and Bäck, T. (1995). Evolution strategies. In Périaux, J. and Winter, G., editors, Genetic Algorithms in Engineering and Computer Science. John Wiley & Sons Ltd, Chichester.

[3] Storn, R. and Price, K. (1995). Differential Evolution : A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, University of Berkeley.

[4] Storn, R. (1996). On the usage of differential evolution for function optimization. In NAPHIS 1996, pages 519–523. Berkeley.

References

[5] A. E. Eiben, J. E. Smith, Agoston E. Eiben, J. D. Smith (2003). Introduction to Evolutionary Computing. Springer.

[6] Storn, R. (WWW). Homepage of Differential Evolution. http://www.icsi.berkeley.edu/~storn/code.html.

A humble plea. Please feel free to e-mail any suggestions, errors and typos to **`matej.leps@fsv.cvut.cz`**.

Date of the last version: 23.11.2011
Version: 001