Genetic Algorithms and Genetic Programming

Lecture 16: (20/11/09)

Differential Evolution II and Metaheuristics in General

School of Informatics

Michael Herrmann

michael.herrmann@ed.ac.uk, phone: 0131 6 517177, Informatics Forum 1.42

Overview

I. GA (1-7)

II. GP (8-10)

III. ACO (11-13): Ant colony optimization

IV. PSO (14-16): Particle swarm optimization

V. Differential evolution, Metaheuristic search (16)

VI. NC (17): Overview on DNA computing, Membrane computing, Molecular computing, Amorphous computing, Organic computing, ….

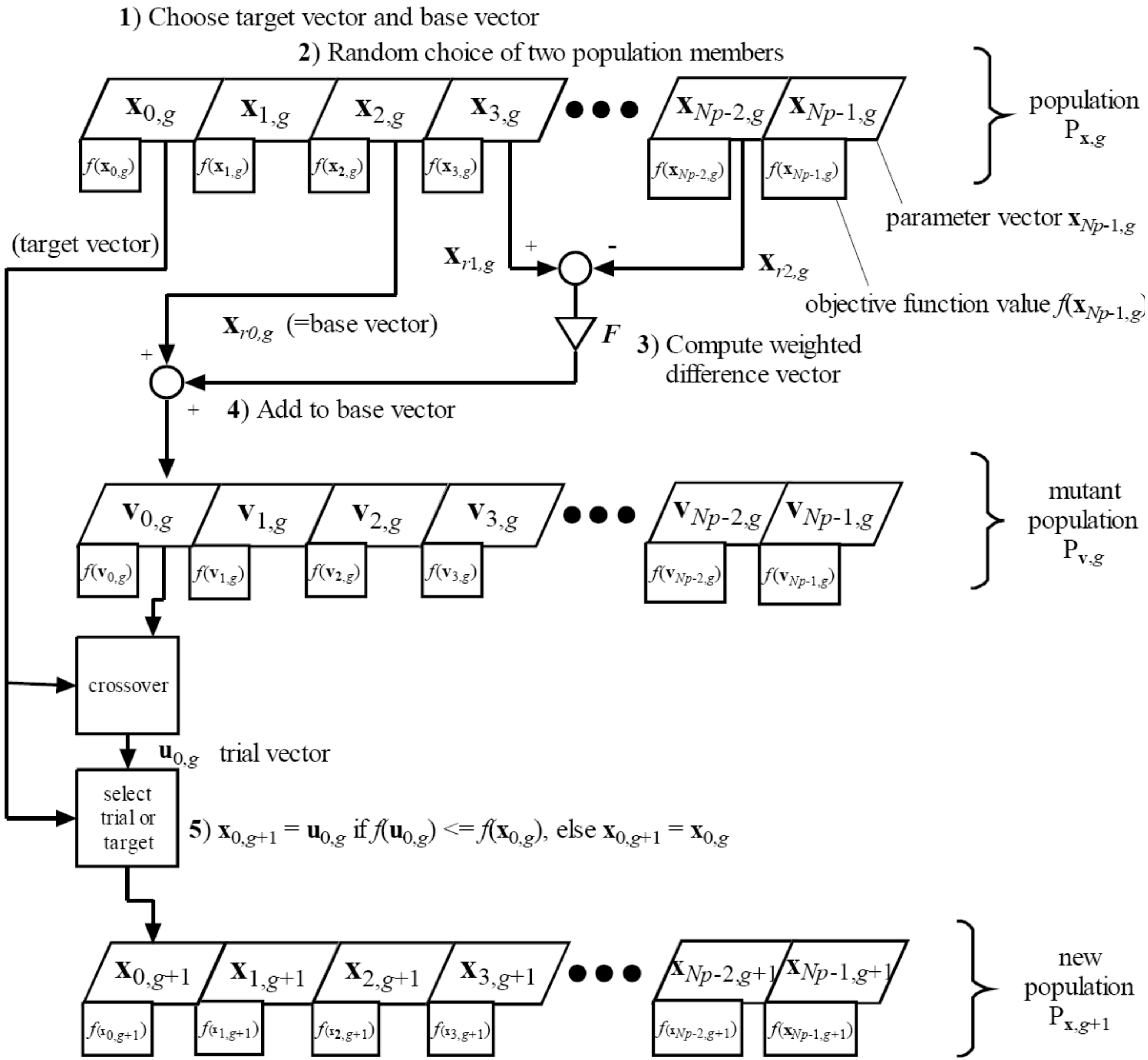
Not included:

artificial neural networks, quantum computing, cellular automata, artificial immune systems

Differential Evolution

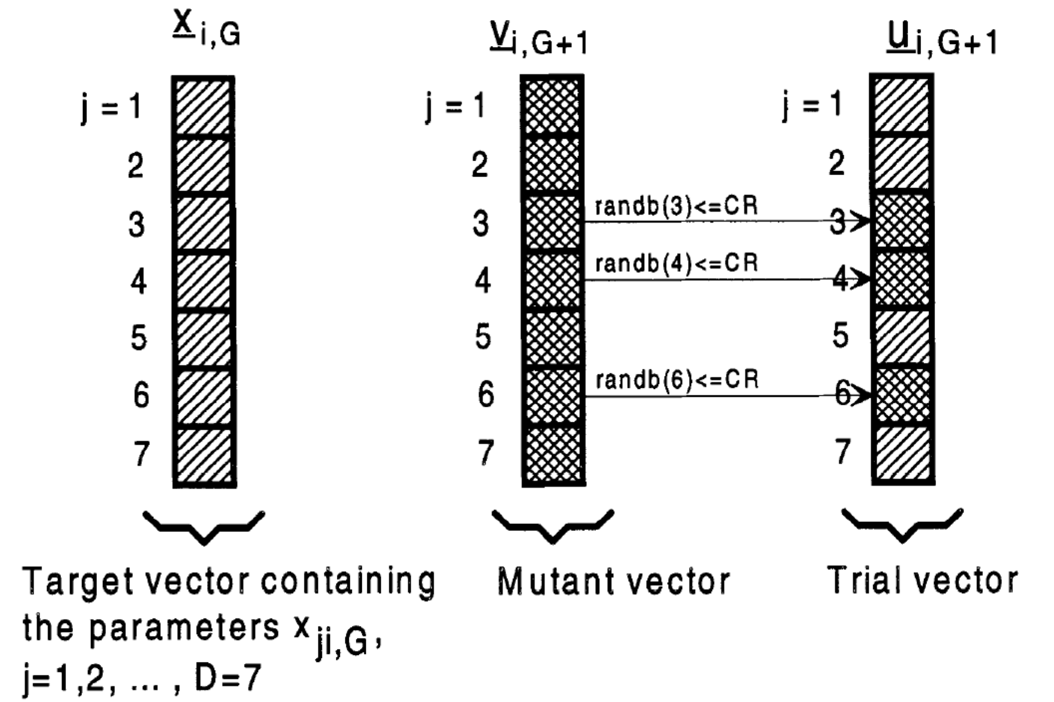
- NP D-dimensional parameter vectors
  x_{iG}; $i = 1, 2, \ldots, NP$; G: generation counter
- **Mutation:** $v_{iG+1} = x_{r_1G} + F * (x_{r_2G} - x_{r_3G})$; ($F$ is just a real number)
- F in [0,2] amplification of the differential variation
- r_i random indexes different from I ("*rnbr*")
- **Crossover:** $u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \ldots, u_{Di,G+1})$
- $$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if } (randb(j) \leq CR) \text{ or } j = rnbr(i) \\ x_{ji,G} & \text{if } (randb(j) > CR) \text{ and } j \neq rnbr(i) \end{cases}$$
  $$j = 1, 2, \ldots, D.$$
- $randb$ in $[0,1]$
- **Selection:** $x_{iG+1} = u_{iG+1}$ if u_{iG+1} is better, otherwise $x_{iG+1} = x_{iG}$

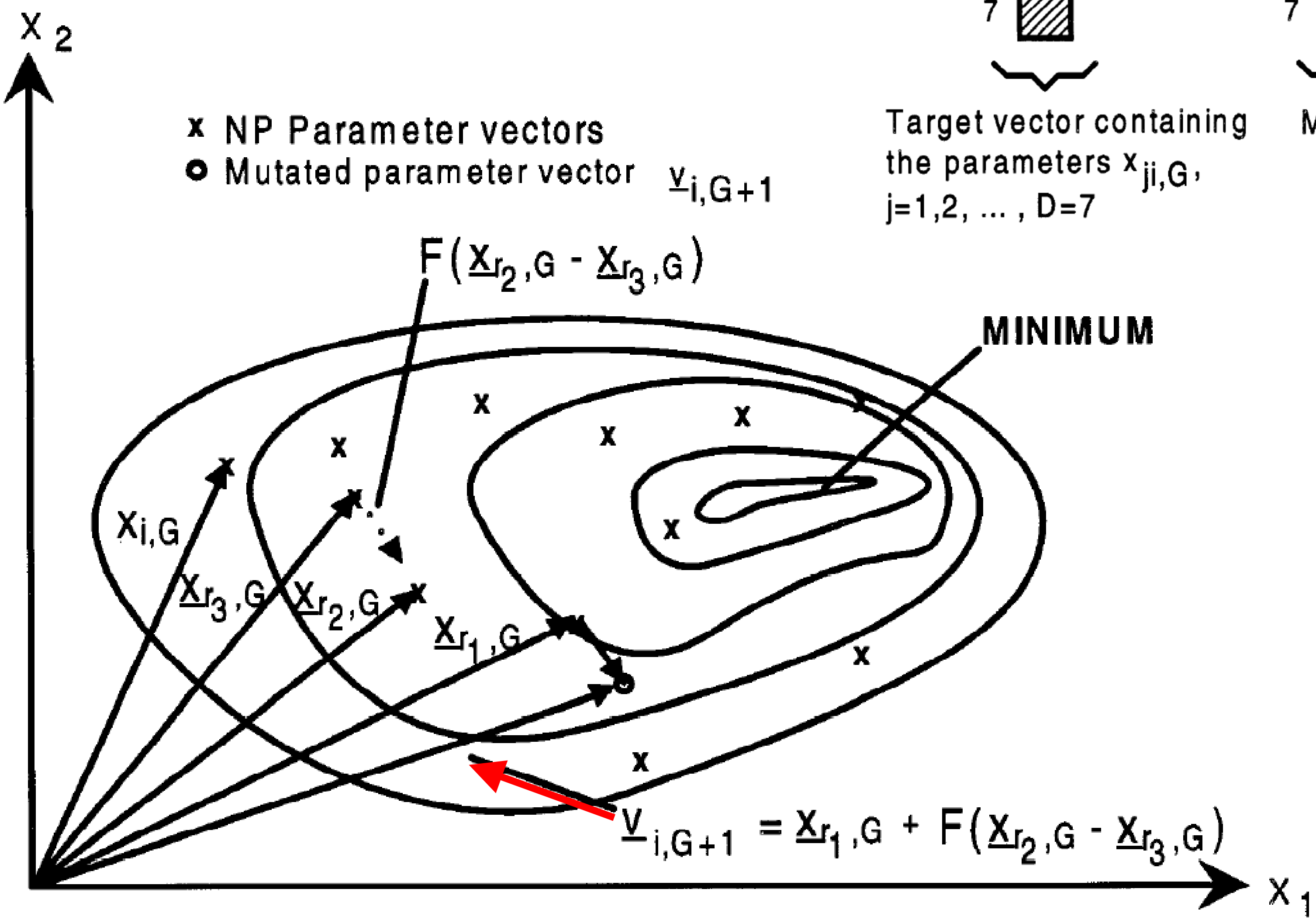
Rainer Storn & Kenneth Price (1997) Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization* **11:** 341–359,



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

Differential Evolution





DE: Details

- Properties
  - Simple, very fast
  - Reasonably good results
  - Diversity increases in flat regions (divergence property)
- Parameters
  - NP=5D (4 … 10D)
  - CR=0.1 (0 … 1.0)
  - F=0.5 (0.4 …. 1.0)
  - a proof exist that effectiveness requires $F \geq F_{\text{crit}} = \sqrt{\frac{1-\frac{CR}{2}}{NP}}$

Search in Differential Evolution

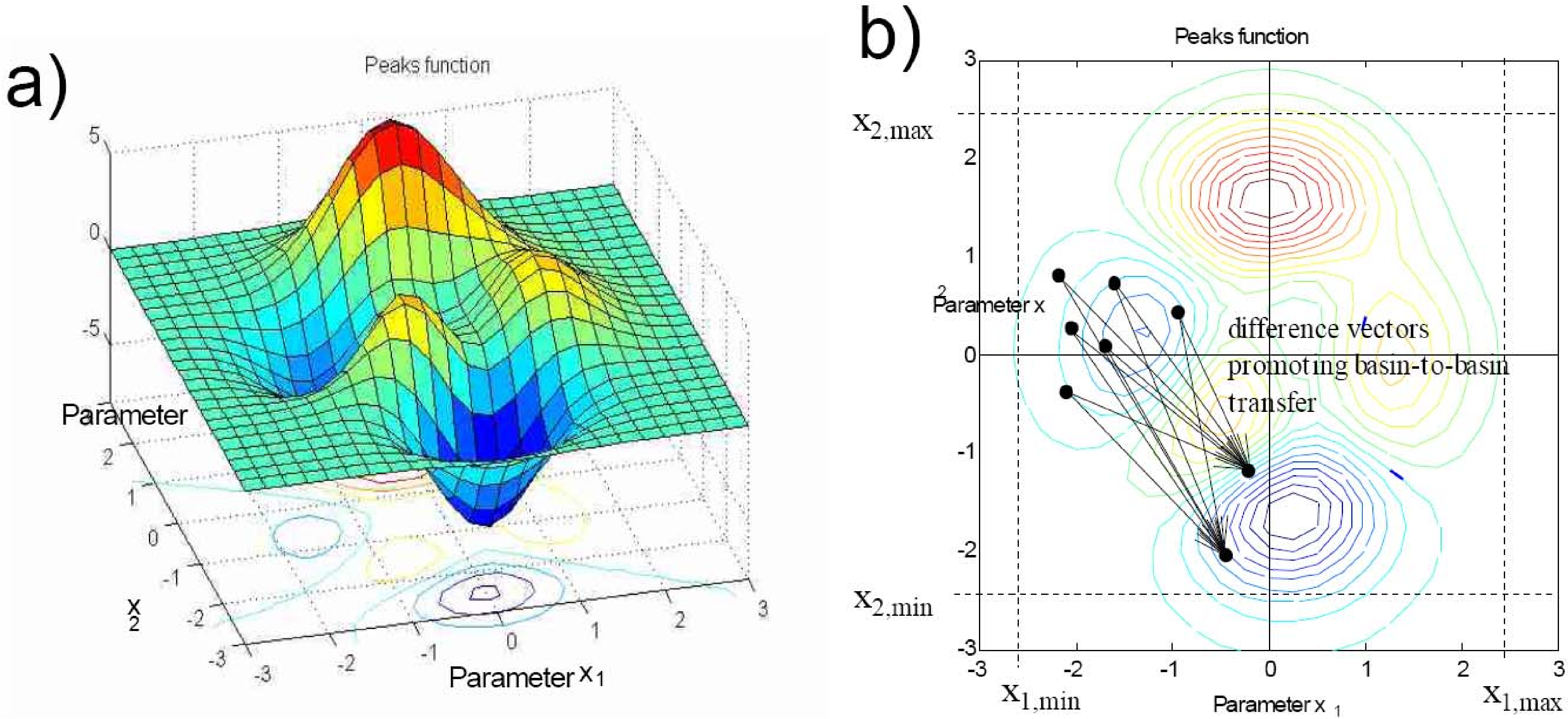


Figure 1.2: *Peaks function a) and illustration of difference vectors b) that promote transfer of points between two basins of attraction of the objective function surface.*

Rainer Storn (2008) Differential Evolution Research – Trends and Open Questions. Chapter 1 of *Uday K. Chakraborty: Advances in Differential Evolution*

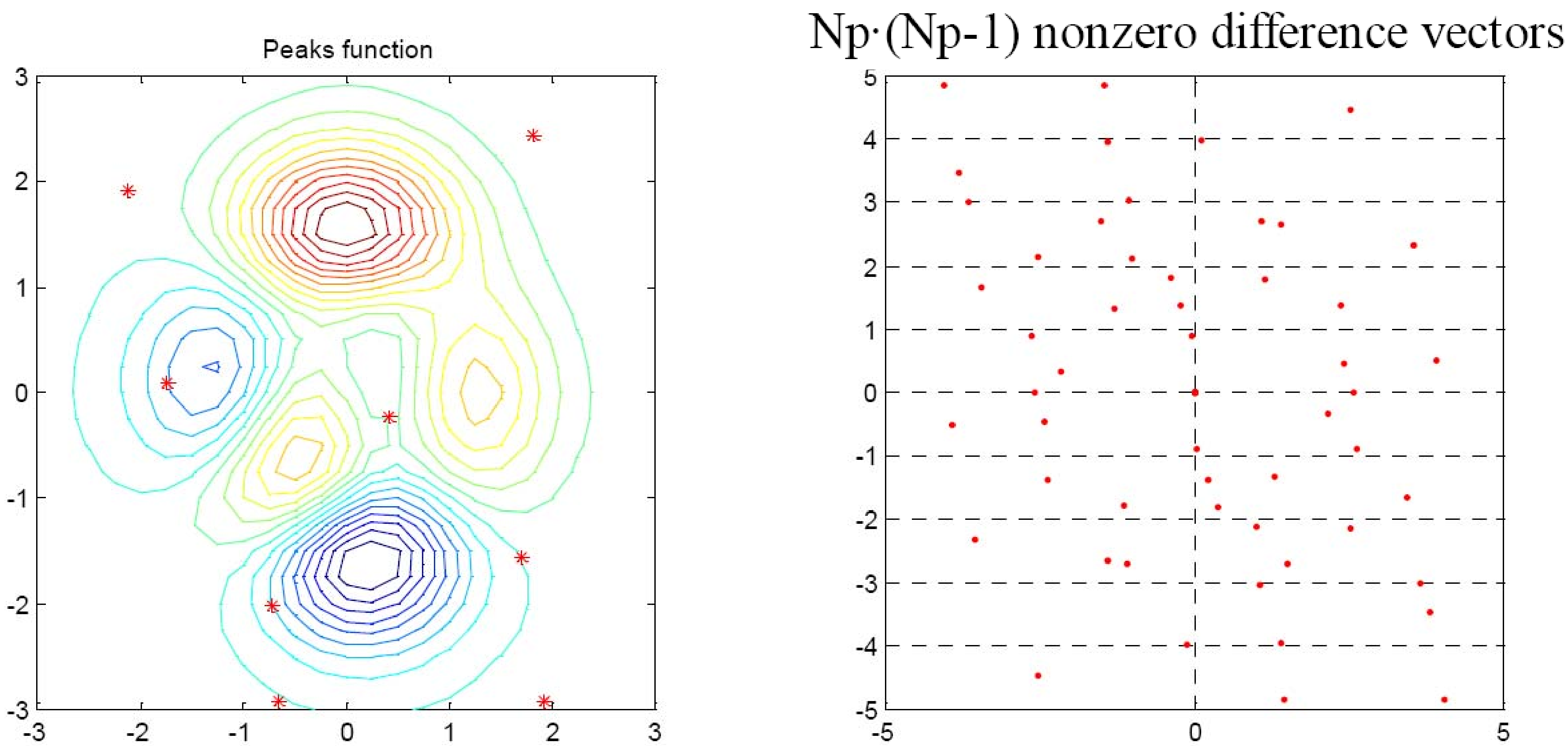


Figure 1.3: *Generation g=1 using Np = 8.*

Objective function used here:

$$f(x_1, x_2) = 3(1-x_1)^2 \cdot \exp\left(x_1^2 + (x_2+1)^2\right) - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right)$$
$$\cdot \exp\left(x_1^2 + x_2^2\right) - \frac{1}{3} \cdot \exp\left((x_1+1)^2 + x_2^2\right)$$

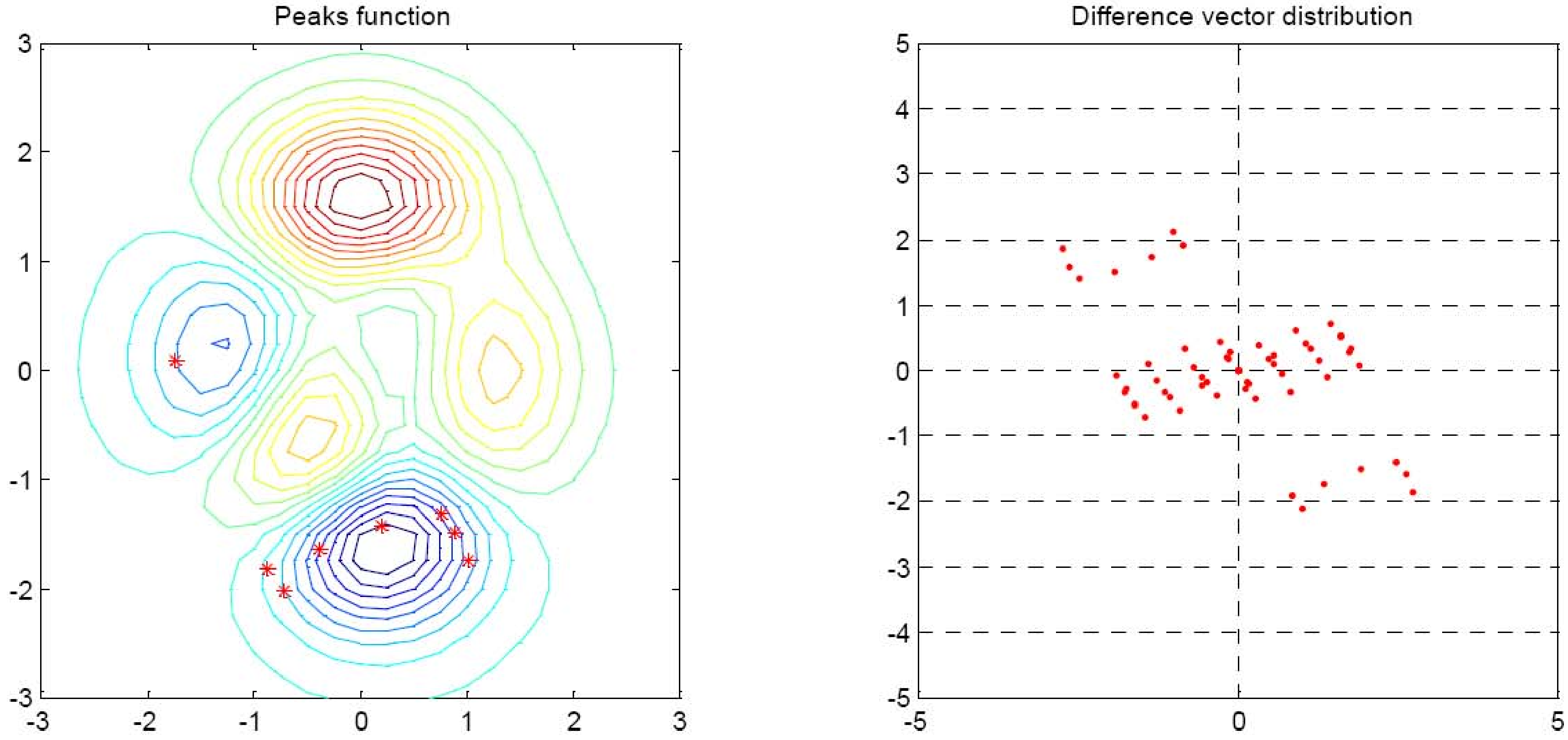


Figure 1.4: *Generation g=10 using Np = 8 . The difference vector distribution (only endpoints shown) exhibits three main clouds where the outer ones promote the transfer between two basins of attraction.*

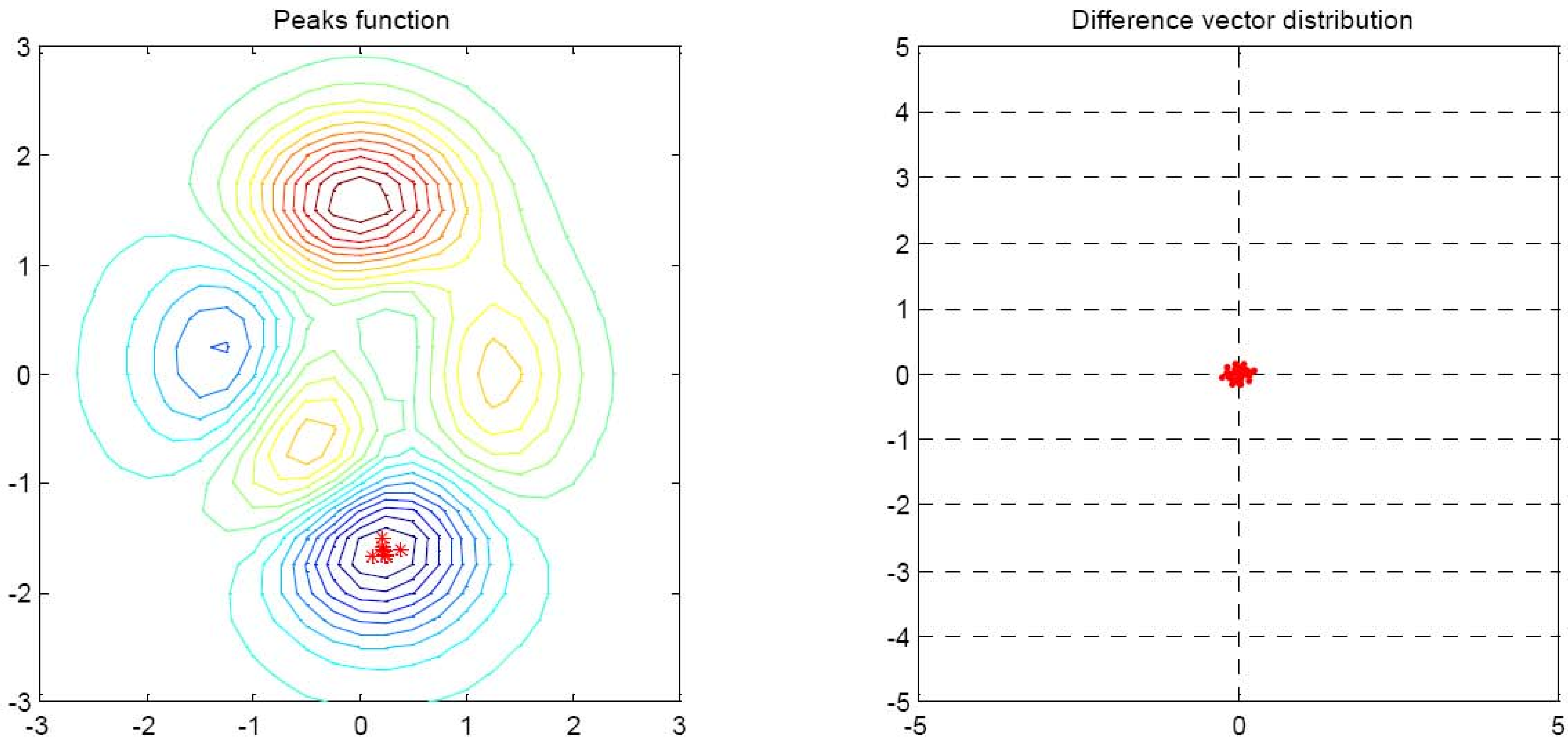


Figure 1.5: *Generation g=20 using Np = 8 . Now the difference vector distribution fosters th e local search of the minimum the vector population is enclosing.*

DE with Crossover

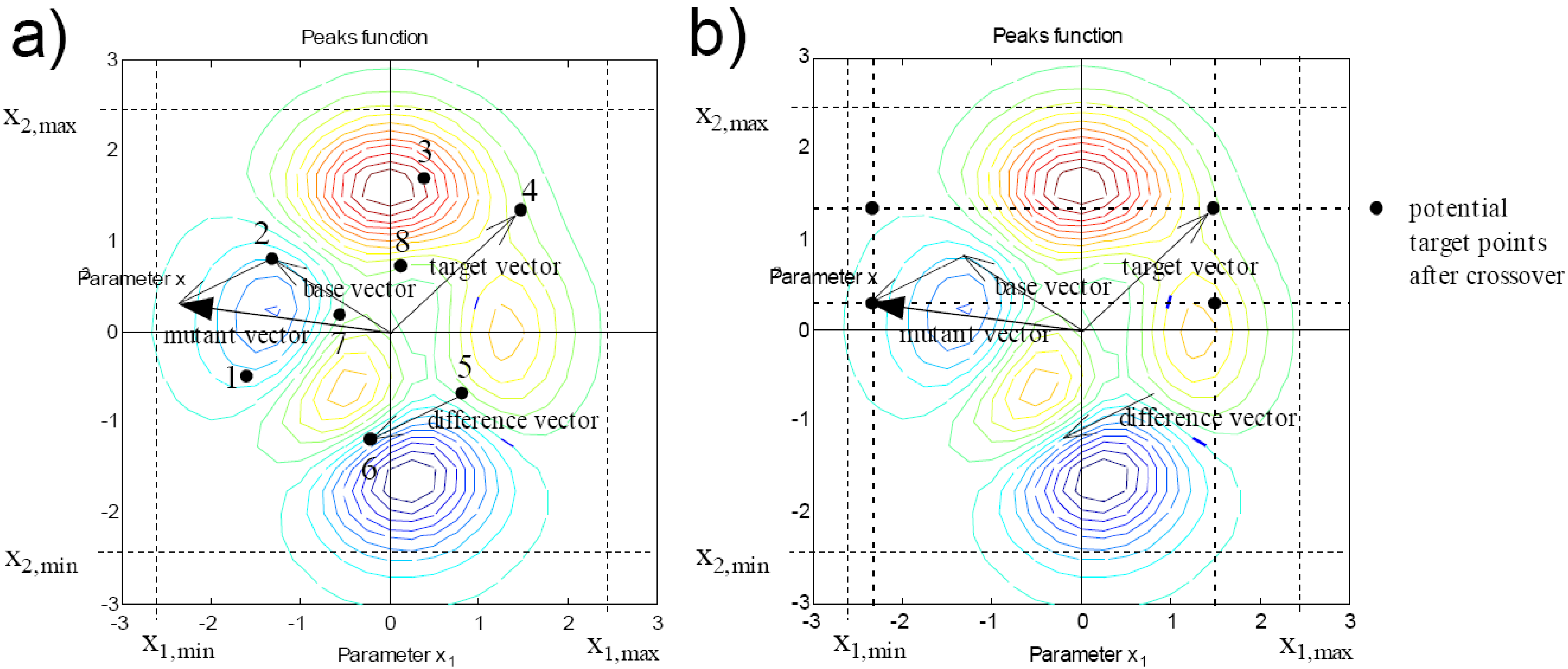


Figure 1.6: *Example for a population of Np=8 points and a mutation step a). The figure on the right b) shows the potential points when using crossover.*

Invariant representations

- Crossover depends on the coordinate directions and is thus not rotationally invariant
- Using randomly rotated coordinate systems the search becomes isotropic

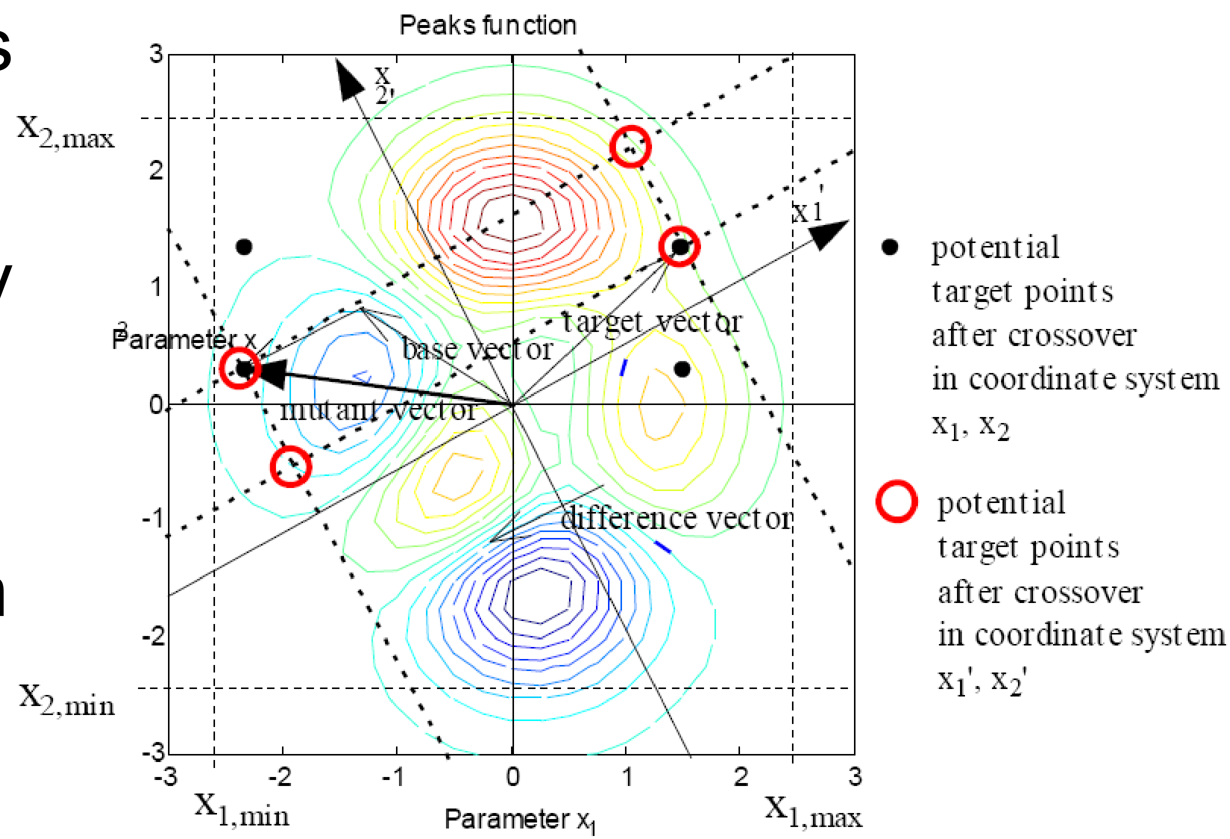


Figure 1.7: *Potential trial points after crossover for coordinate system x_1, x_2 and system x_1', x_2'*

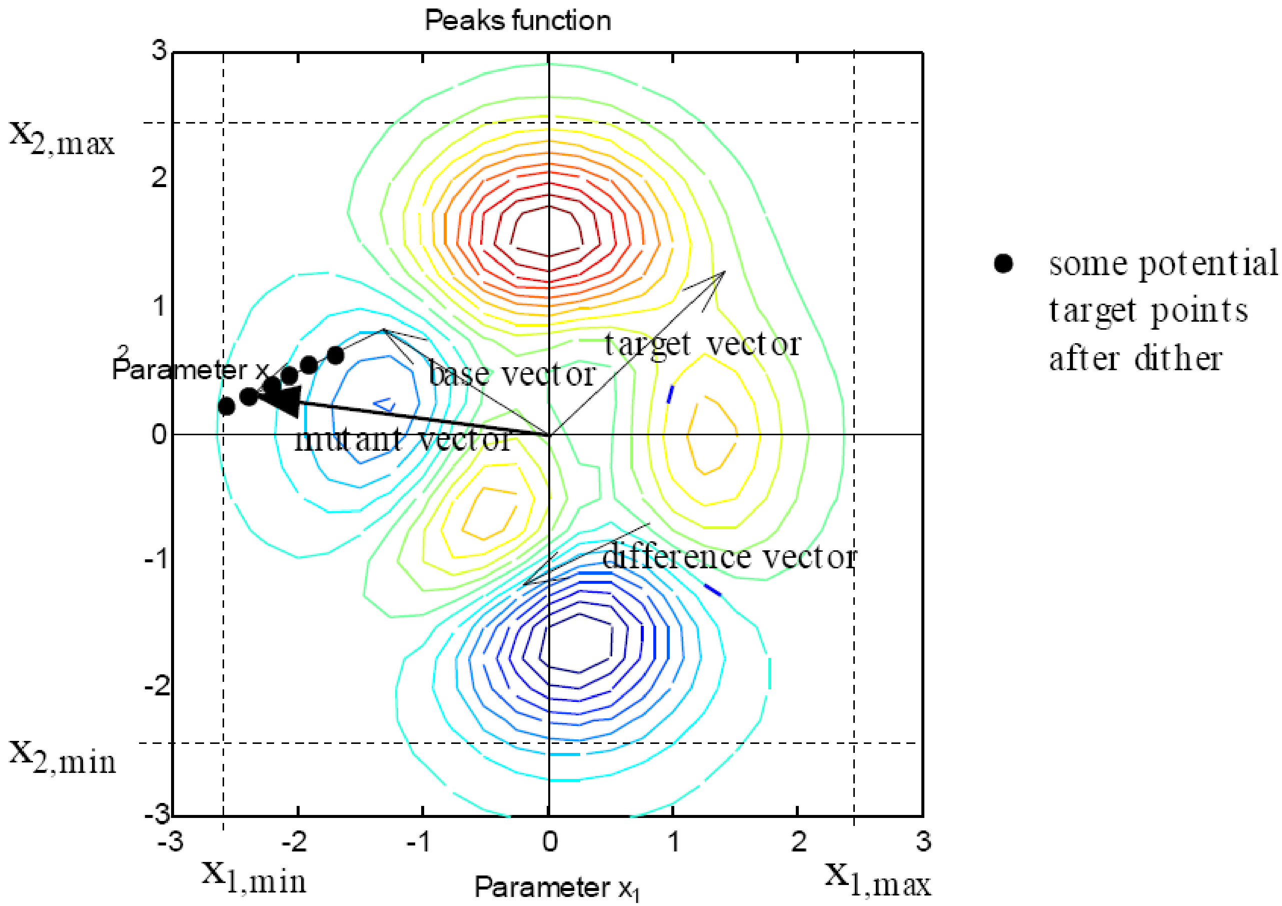
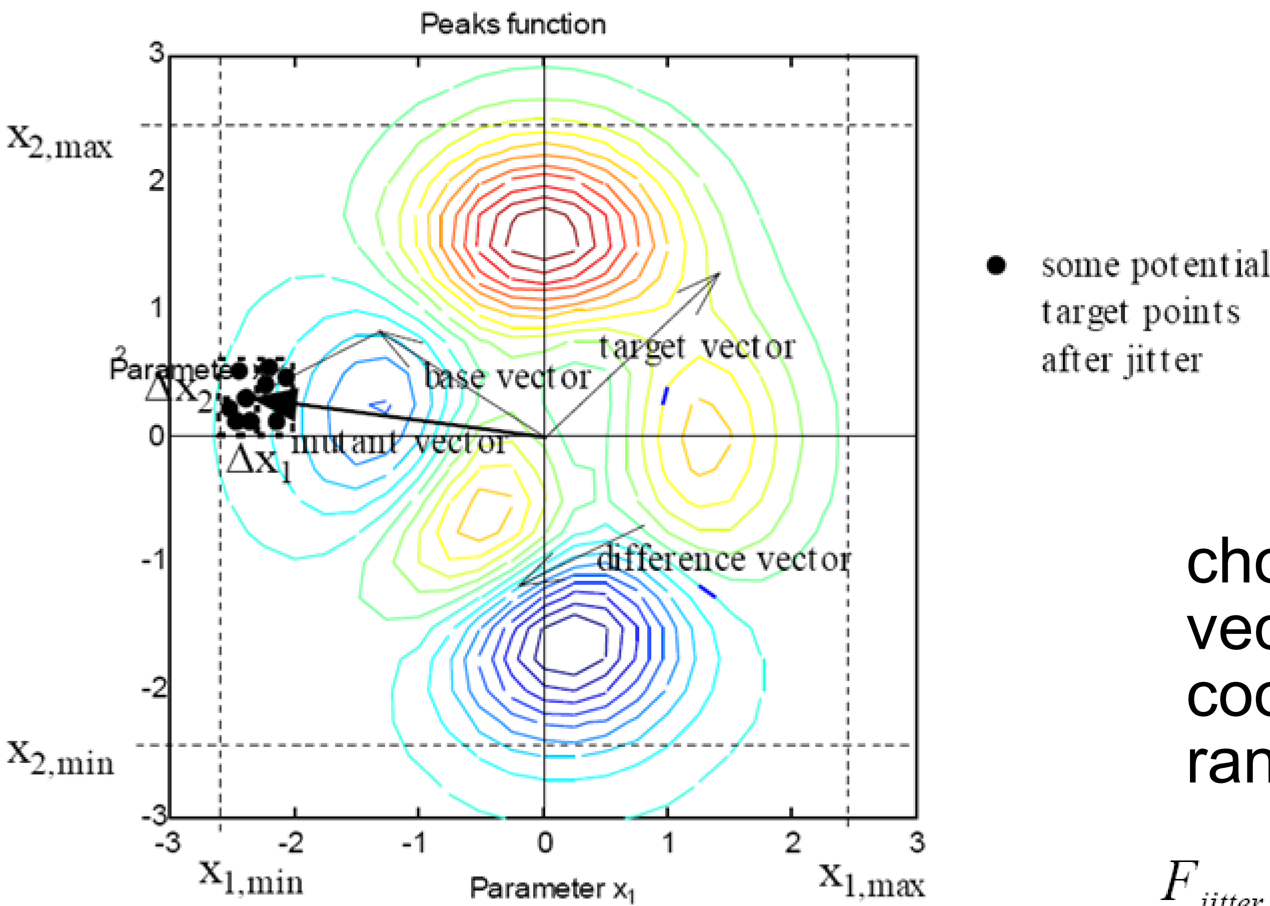


Figure 1.8: *Pictorial representation of d ither which simply randomizes the mutation scale factor F and hence does not compromise DE's contour matching.*

DE with Jitter



choose for each vector *i* and for each coordinate *j* a different random increment, e.g.:

$$F_{jitter,i} = F \cdot \left(1 + \delta \cdot \left(rand_j[0,1) - 0.5\right)\right)$$

Figure 1.9: *Jitter randomizes the difference vector in all parameter directions*

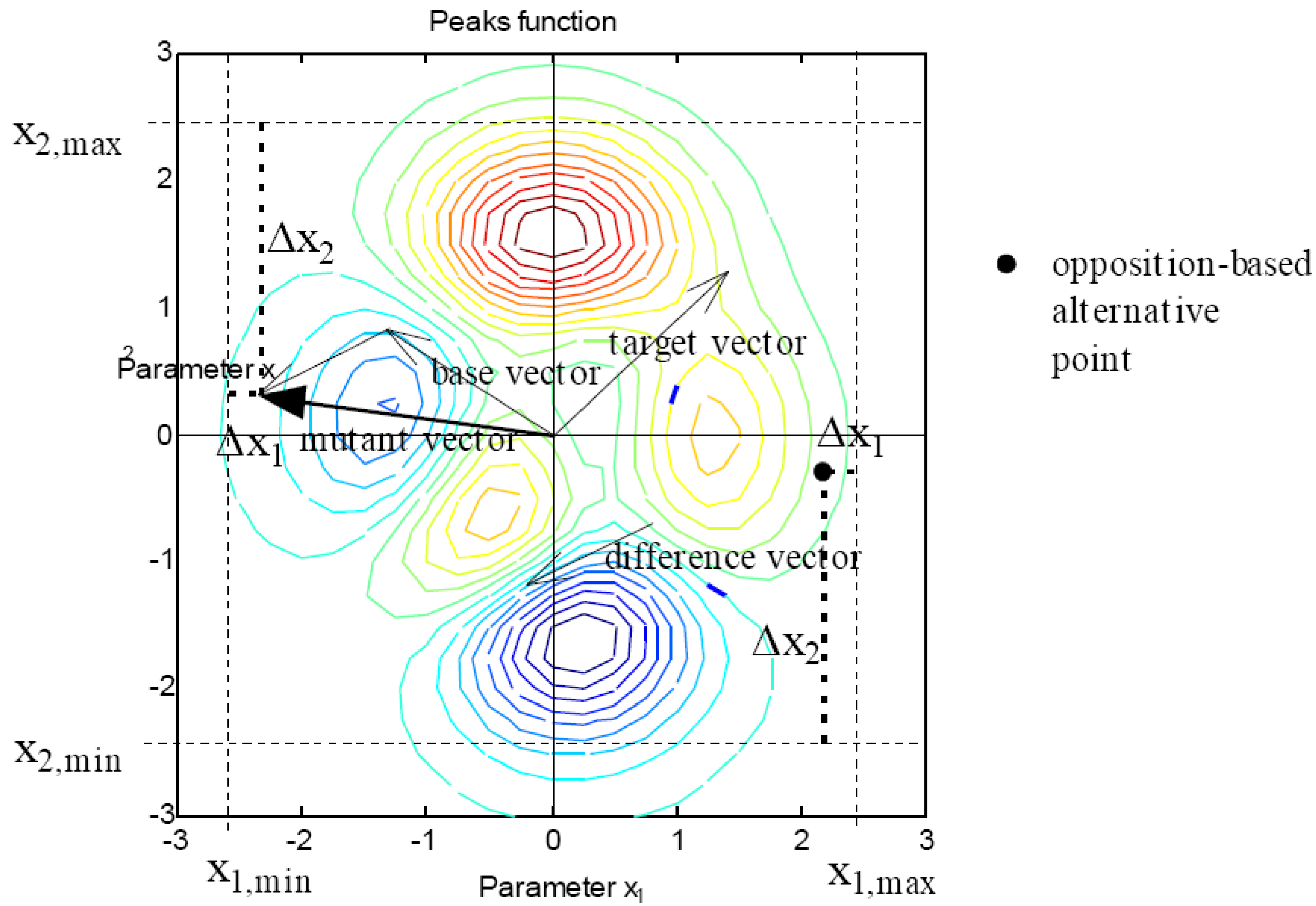


Figure 1.11: *Illustration of the construction of an opposition -based population point. Note that this point generating scheme does neither satisfy rotational invariance nor basin -to-basin transfer.*

DE: Variants

- Mutability and threshold parameters can also be evolved for each individual (as the step sizes in ES), i.e. dimension becomes D+2.
- Scheme for denoting DE variants: $DE/x/y/z$

x specifies the vector to be mutated which currently can be "rand" (a randomly chosen population vector) or "best" (the vector of lowest cost from the current population).

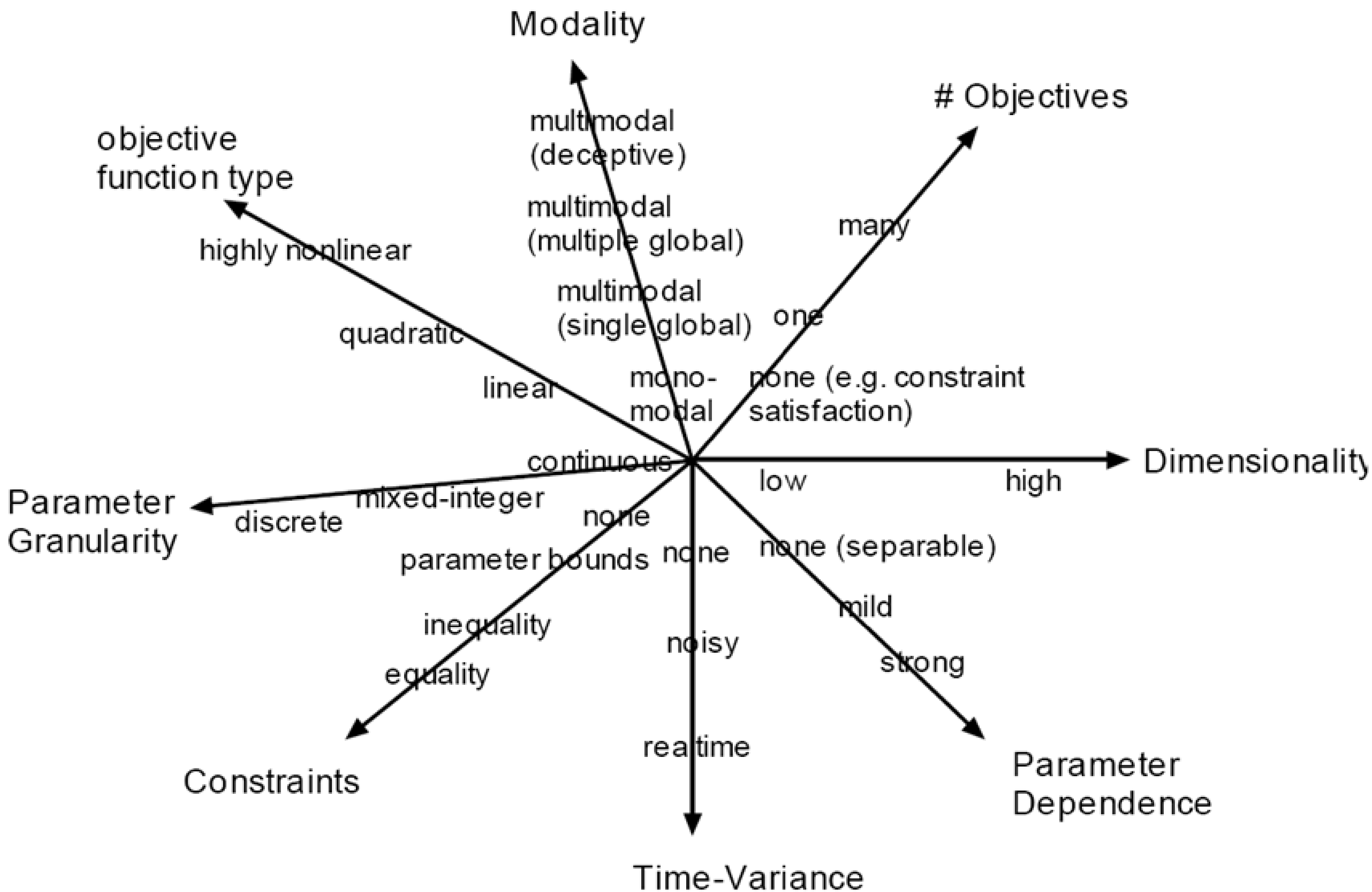
y is the number of difference vectors used.

z denotes the crossover scheme. The current variant is "bin" (Crossover due to independent binomial experiments as explained in Section 2)

e.g. DE/best/2/bin

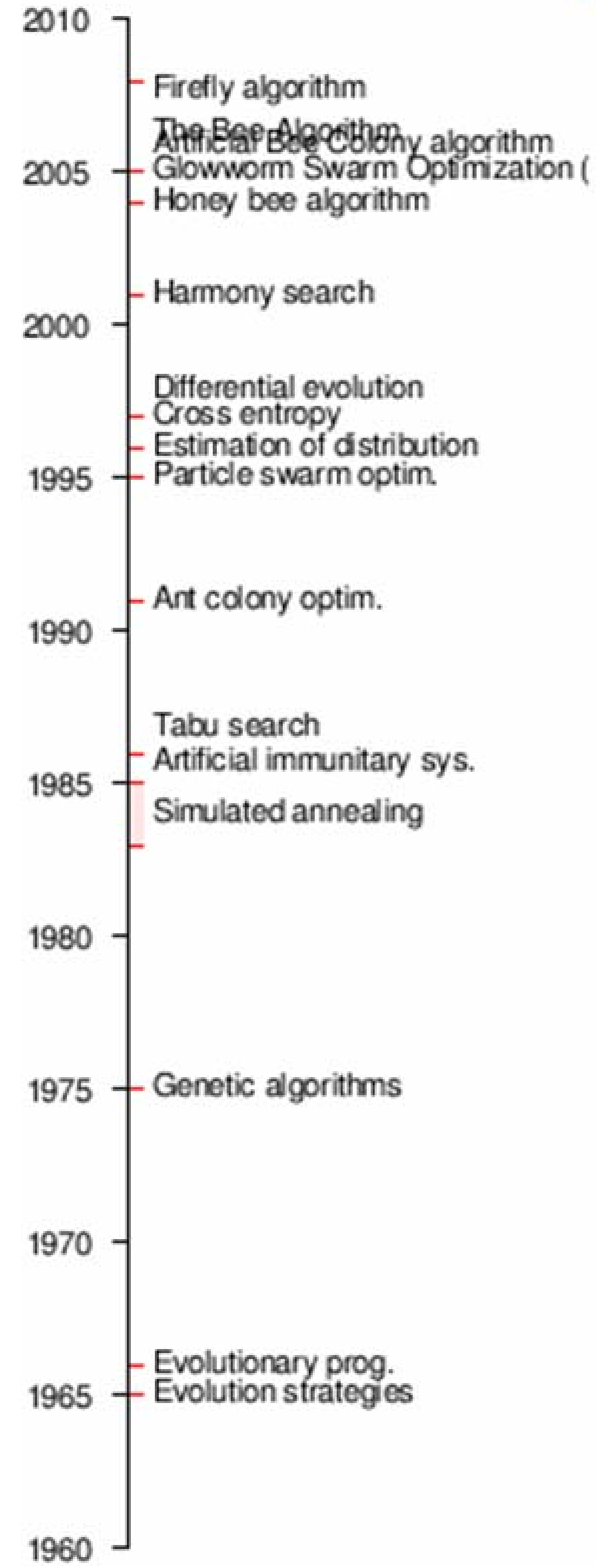
$$v_{i,G+1} = x_{best,G} + F \cdot (x_{r_1,G} + x_{r_2,G} - x_{r_3,G} - x_{r_4,G})$$

- Also a number of self-adapting variants exist cf. [Storn, 08]


Modality
Objectives
objective function type
multimodal (deceptive)
multimodal (multiple global)
multimodal (single global)
mono-modal
many
one
none (e.g. constraint satisfaction)
highly nonlinear
quadratic
linear
continuous
mixed-integer
discrete
Parameter Granularity
Dimensionality
low
high
none
parameter bounds
inequality
equality
Constraints
none
noisy
real time
Time-Variance
none (separable)
mild
strong
Parameter Dependence

Meta-Heuristic Search

- μετα “beyond”, ευρισκειν "to find“
- applied mainly to combinatorial optimization
- The user has to modify the algorithm to a greater or lesser extend in order to adapt it to specific problem
- These algorithms seem to defy the no-free lunch (NFL) theorem due to the combination of
  - biased choice of problems
  - user-generated modifications
- Can often be outperformed by a problem-dependent heuristic



The General Scheme

1. Use populations of solutions/trials/individuals
2. Transfer information in the population from the best individuals to others by selection+crossover/attraction
3. Maintain diversity by adding noise/mutations/ intrinsic dynamics/amplifying differences

- Avoid local minima (leapfrog/crossover/more noise/ subpopulations/border of instability/checking success)

4. Whenever possible, use building blocks/partial solutions/royal road functions
5. Store good solutions in memory as best-so-far/iteration best/individual best/elite/pheromones
6. Use domain knowledge and intuition for encoding, initialization, termination, choice of the algorithm
7. Tweak the parameters, develop your own variants

Contra

- No free lunch theorem implies that there must be some implicit assumptions that single out “good” problems (one such assumption is the correlation between goal function values at nearby candidate solutions)
- If these assumptions were made explicit more specific algorithms could be designed
- Random search is the essential component beyond this
- The quality of a ME algorithm is not well-defined because user-provided domain knowledge enters
- There are many “classical” problems which are fully understood and where ME algorithms perform comparatively poor. (LS is usually not state of the art)
- Dilettantism: A few hours of reading, thinking and programming can easily save months of computer time

en.wikipedia.org/wiki/Metaheuristic

“Banal Metaheuristic”
(*humant colony algorithm*;-)
*** in three easy steps ***

1. Call the user-provided state generator.
2. Print the resulting state.
3. Stop.

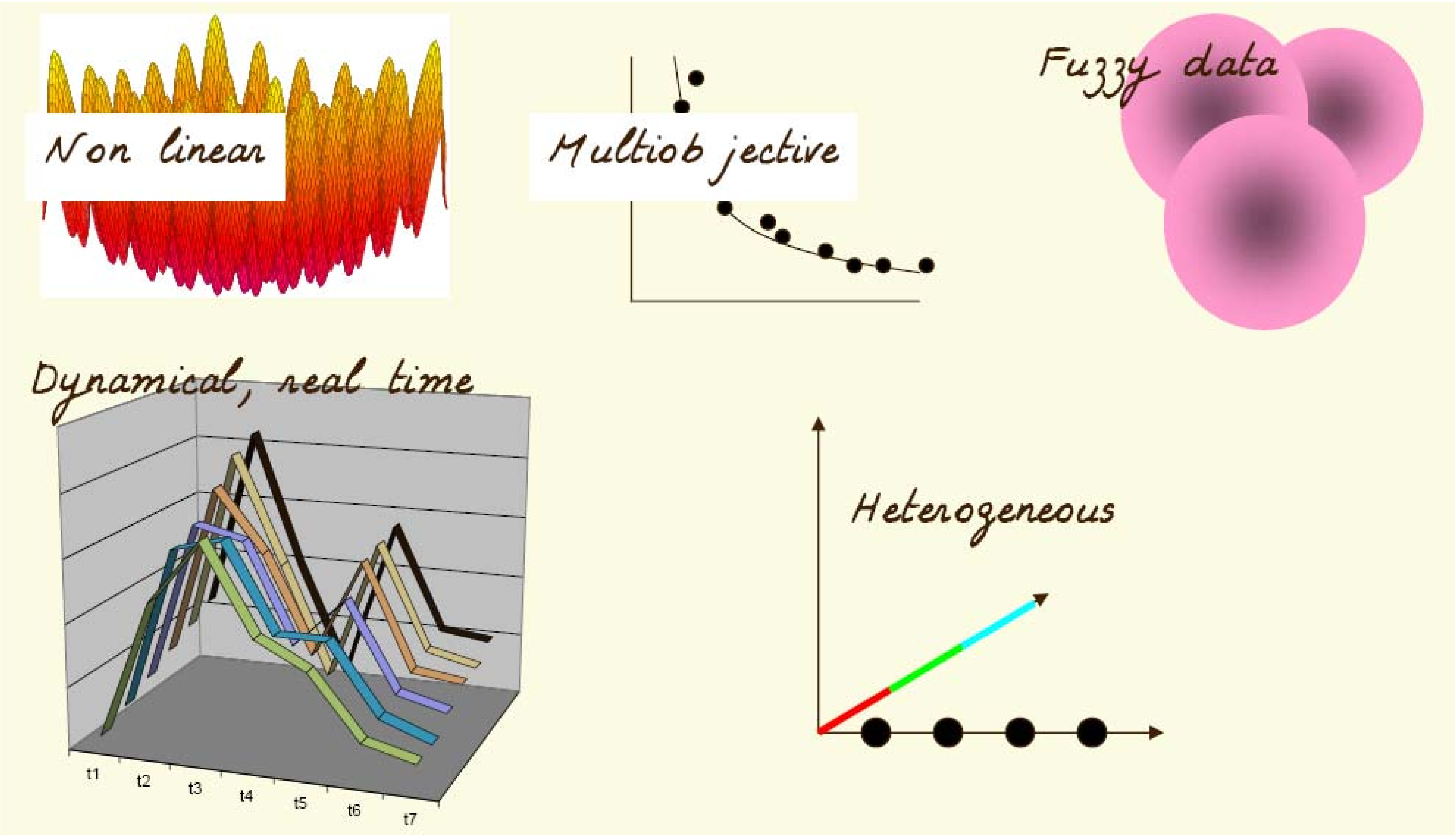
Given any two distinct metaheuristics M and N, and almost any goal function f, it is usually possible to write a set of auxiliary procedures that will make M find the optimum much more efficient than N, by many orders of magnitude; or vice-versa. In fact, since the auxiliary procedures are usually unrestricted, one can submit the basic step of metaheuristic M as the generator or mutator for N.

en.wikipedia.org/wiki/Metaheuristic

Pro

- If you know a better solution then why using ME?
  But if not, then why not?
- Its not just random search
- There are a number of applications where ME are performing reasonably well
- Theoretical expertise, problem analysis, modeling and implementation are cost factors in real-world problems
- There are domains where modeling is questionable, but the combination of existing solutions is possible (*minority games*, e.g. esthetic design, financial markets)
- Nature is an important source of inspiration
- It may help to understand decision making in nature and society

Ecological niches for MH algorithms



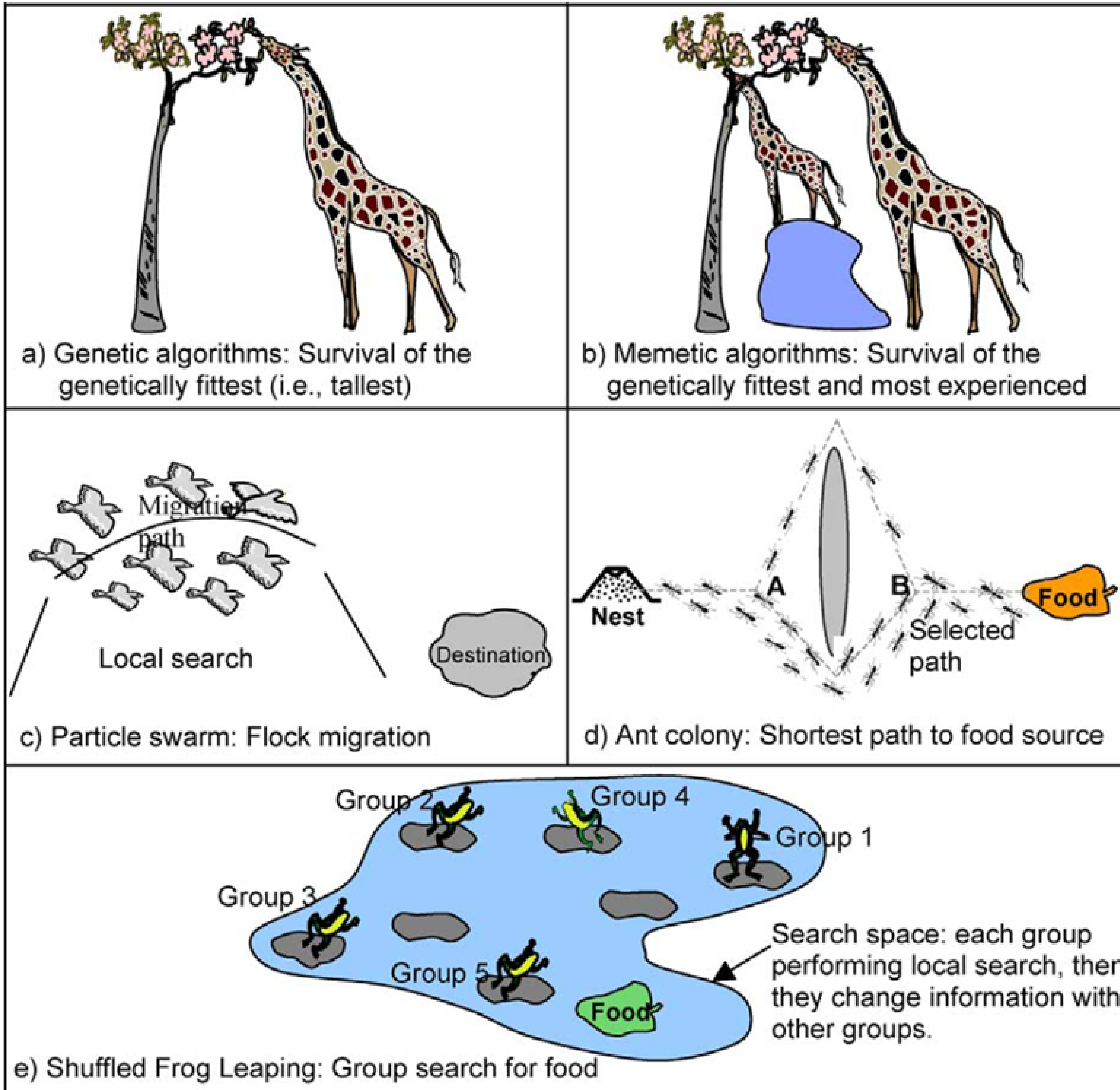
Comparison among ME algorithms

- Comparisons are not always meaningful
- Competitions are an option
- Global bests in a standard set of benchmark problems (testbed) based on a standard solution quality metrics (neither does exist)
- Asymptotic space and time complexity (e.g. runtime growth rate)
- Dimension and sensitivity of the parameter space

Comparisons

- **First experimental principle:** The problems used for assessing the performance of an algorithm cannot be used in the development of the algorithm itself.
- **Second experimental principle:** The designer can take into account any available domain-specific knowledge as well as make use of pilot studies on similar problems.
- **Third experimental principle:** When comparing several algorithms, all the algorithms should make use of the available domain-specific knowledge, and equal computational effort should be invested in all the pilot studies. Similarly, in the test phase, all the algorithms should be compared on an equal computing time basis.

Mauro Birattari, Mark Zlochin and Marco Dorigo: Toward a theory of practice in metaheuristic design: A machine learning perspective. RAIRO-Inf. Theor. Appl. 40 (2006) 353-369.



Emad Elbeltagi, Tarek Hegazy, Donald Grierson (2005) Advanced Comparison among five evolutionary-based optimization algorithms. *Engineering Informatics* **19**, 43–53.

Results of the continuous optimization problems

Comparison criteria	Algorithm	Number of variables						
		F8				*EF10*		
		10	20	50	100	10	20	50
% Success	GAs (Evolver)	50	30	10	0	20	0	0
	MAs	90	100	100	100	100	70	0
	PSO	30	80	100	100	100	80	60
	ACO	–	–	–	–	–	–	–
	SFL	50	70	90	100	80	20	0
Mean solution	GAs (Evolver)	0.06	0.097	0.161	0.432	0.455	1.128	5.951
	MAs	0.014	0.013	0.011	0.009	0.014	0.068	0.552
	PSO	0.093	0.081	0.011	0.011	0.009	0.075	2.895
	ACO	–	–	–	–	–	–	–
	SFL	0.08	0.063	0.049	0.019	0.058	2.252	6.469

Results of the discrete optimization problem

Algorithm	Minimum project duration (days)	Average project duration (days)	Minimum cost ($)	Average cost ($)	% Success rate	Processing time (s)
GAs	113	120	162,270	164,772	0	16
MAs	110	114	161,270	162,495	20	21
PSO	110	112	161,270	161,940	60	15
ACO	110	122	161,270	166,675	20	10
SFL	112	123	162,020	166,045	0	15

Emad Elbeltagi, Tarek Hegazy, Donald Grierson (2005) Advanced Comparison among five evolutionary-based optimization algorithms. *Engineering Informatics* 19, 43–53.

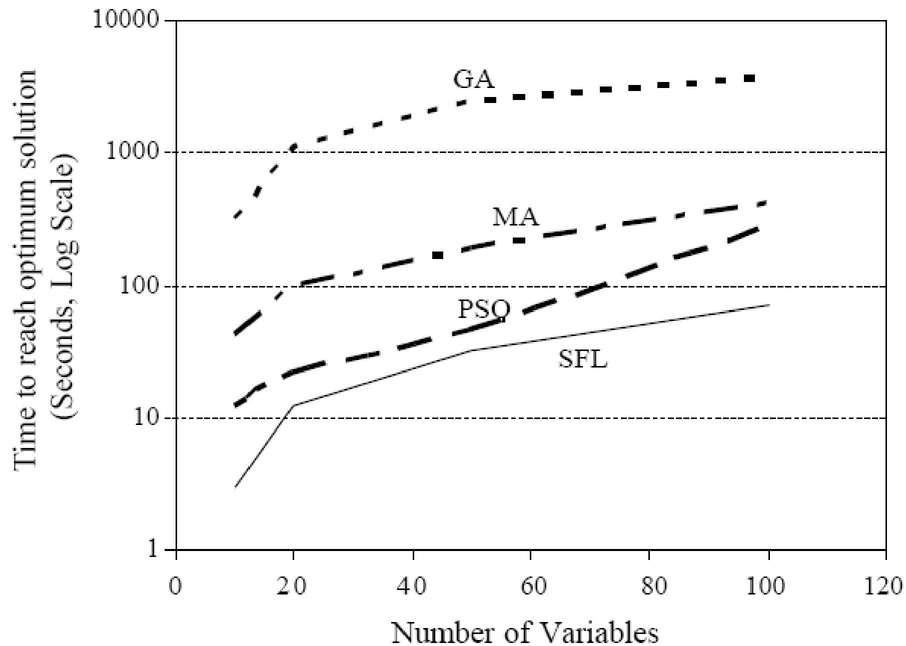


Fig. 5. Processing time to reach the optimum for *F8* function.

$$f(x_{i|i=1,N}) = 1 + \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{N} (\cos(x_i/\sqrt{i}))$$

… and the winner is

PSO

(check back for the next competition)