Genetic Algorithms and Genetic Programming

Lecture 15: (17/11/09)

Particle Swarm Optimization II

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 (14-15): Particle swarm optimization and differential evolution
V. NC (16): Overview on DNA computing, Membrane computing, Molecular computing, Amorphous computing, Organic computing, ….
VI. Wrapping up: Metaheuristic search (17)

Not included:

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

The canonical PSO algorithm

For each particle $1 \le i \le n, \quad x_i \in \mathbf{R}^m, \ v_i \in \mathbf{R}^m$

- create random vectors

  r_1, r_2 with components drawn from $U[0,1]$

- update velocities

  inertia — personal best — global best

  $$v_i \leftarrow \omega v_i + \alpha_1 r_1 \circ (\hat{x}_i - x_i) + \alpha_2 r_2 \circ (\hat{g} - x_i)$$

- update positions

  $$x_i \leftarrow x_i + v_i$$

  $\circ$ componentwise multiplication

- update local bests

  $$\hat{x}_i \leftarrow x_i \quad \text{if} \quad f(x_i) < f(\hat{x}_i)$$

- update global best

  $$\hat{g} \leftarrow x_i \quad \text{if} \quad f(x_i) < f(\hat{g})$$

minimization problem!

Analysis of PSO: Simplified algorithm

- Consider a single particle only ("the view from inside")
- Ignore randomness (use a homogeneous mean value)
- Ignore the global best (assume it equals personal best)
- Keep the personal best constant (changes are rare)
- Set inertia to unity (for the moment only)
- i.e. what we had (vector equation of i-th particle)

$$v_i(t+1) = \omega v_i(t) + \alpha_1 r_1 \circ \left(\hat{x}_i - x_i(t)\right) + \alpha_2 r_2 \circ \left(\hat{g} - x_i(t)\right)$$

- becomes now (in component form: $d=1,\ldots,m$ with $p_i = \hat{x}_i$)

$$v_{id}(t+1) = v_{id}(t) + \varphi\left(p_{id} - x_{id}(t)\right)$$

Algebraic point of view

- Introduce $y_t = p - x_t$

$$x_{t+1} = x_t + v_{t+1} \Longrightarrow \begin{aligned} v_{t+1} &= v_t + \varphi\, y_t \\ y_{t+1} &= -v_t + (1-\varphi)\, y_t \end{aligned}$$

- Introduce state vector $P_t = (v_t, y_t)^{\mathrm{T}}$ and

$$M = \begin{pmatrix} 1 & \varphi \\ -1 & 1-\varphi \end{pmatrix} \qquad P_{t+1} = MP_t$$

- Starting from the initial state P_0 we have $P_t = M^t P_0$

M. Clerc & J. Kennedy (2002) The particle swarm – Explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on *Evolutionary Computation* 7:1, 58-73.

Algebraic point of view

- Eigenvalues of $M = \begin{pmatrix} 1 & \varphi \\ -1 & 1-\varphi \end{pmatrix}$ i.e. $AMA^{-1} = L = \begin{bmatrix} e_1 & 0 \\ 0 & e_2 \end{bmatrix}$

$$e_{1/2} = 1 - \frac{\varphi}{2} \pm \frac{\sqrt{\varphi^2 - 4\varphi}}{2}$$

- Transformation matrix: $A = \begin{bmatrix} \varphi + \sqrt{\varphi^2 - 4\varphi} & 2\varphi \\ \varphi - \sqrt{\varphi^2 - 4\varphi} & 2\varphi \end{bmatrix}$

$$P_{t+1} = MP_t \Longrightarrow \begin{aligned} P_{t+1} &= A^{-1}LAP_t \\ \underbrace{AP_{t+1}}_{Q_{t+1}} &= L\underbrace{AP_t}_{Q_t} \\ Q_{t+1} &= LQ_t \end{aligned}$$

- Thus $Q_t = L^t Q_0$ where $L^t = \begin{bmatrix} e_1^t & 0 \\ 0 & e_2^t \end{bmatrix}$

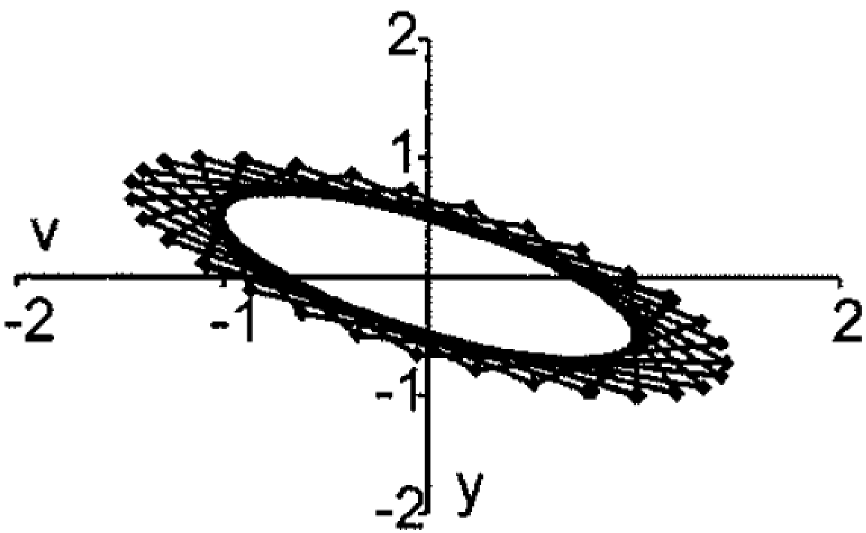
3 Cases: $0 < \varphi < 4$ $\varphi = 4$ $\varphi > 4$

EV complex

$$e_1 = \cos(\theta) + i\sin(\theta)$$
$$e_2 = \cos(\theta) - i\sin(\theta)$$

$$e_1^t = \cos(t\theta) + i\sin(t\theta)$$
$$e_2^t = \cos(t\theta) - i\sin(t\theta)$$

Oscillatory with period k: $\theta = (2k\pi)/t$
(or quasiperiodic)



$$M = \begin{bmatrix} 1 & 4 \\ -1 & -3 \end{bmatrix}$$

$$V = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$

Both EV are -1

$$MV = -V$$

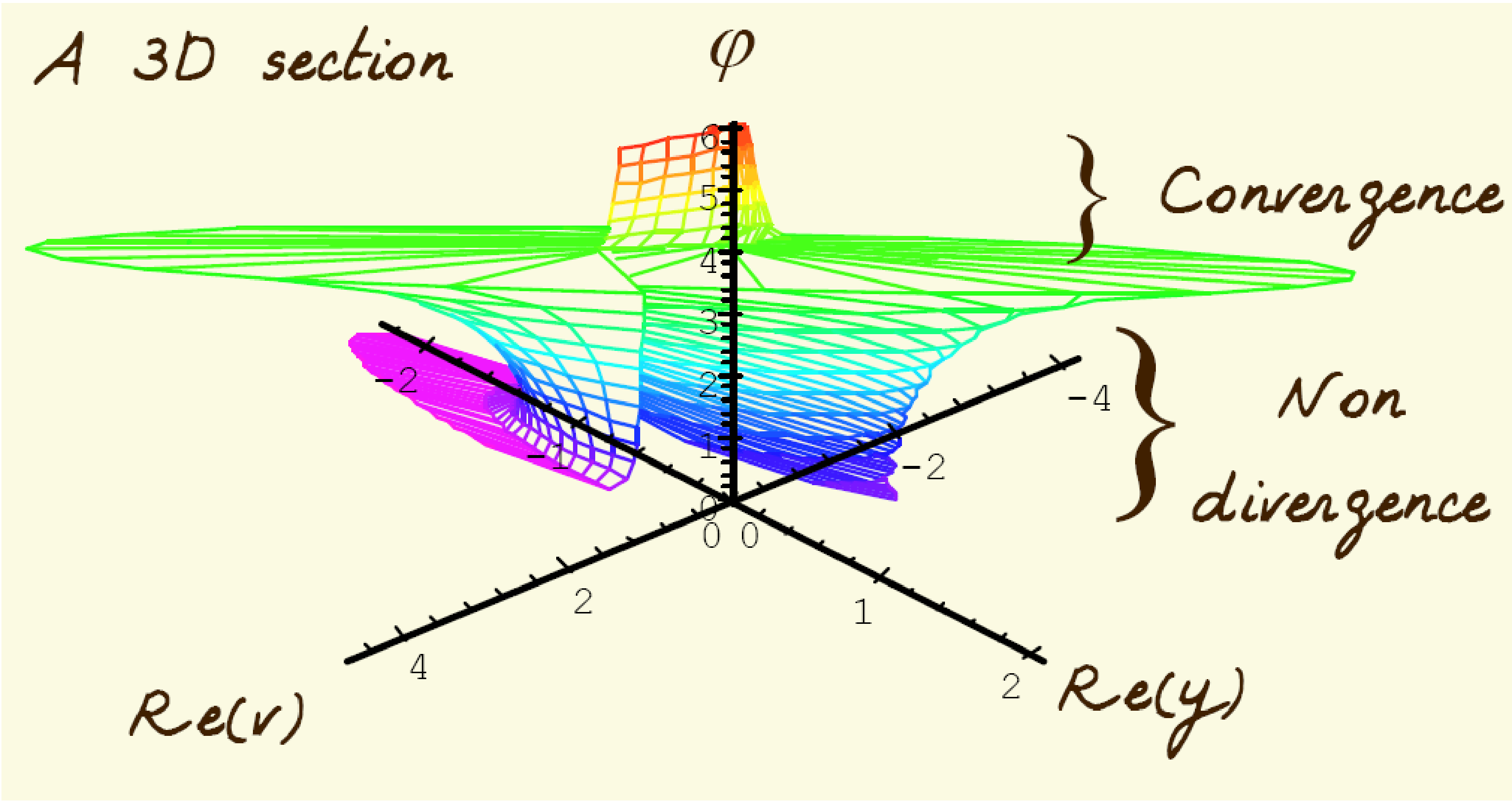
$$P_{t+1} = \pm \begin{bmatrix} 2y_0 \\ -y_0 \end{bmatrix} = -P_t$$

Some algebra implies:
Linear divergence
(unless starting
with the eigenvector *V*)

EV real

Exponent. divergent

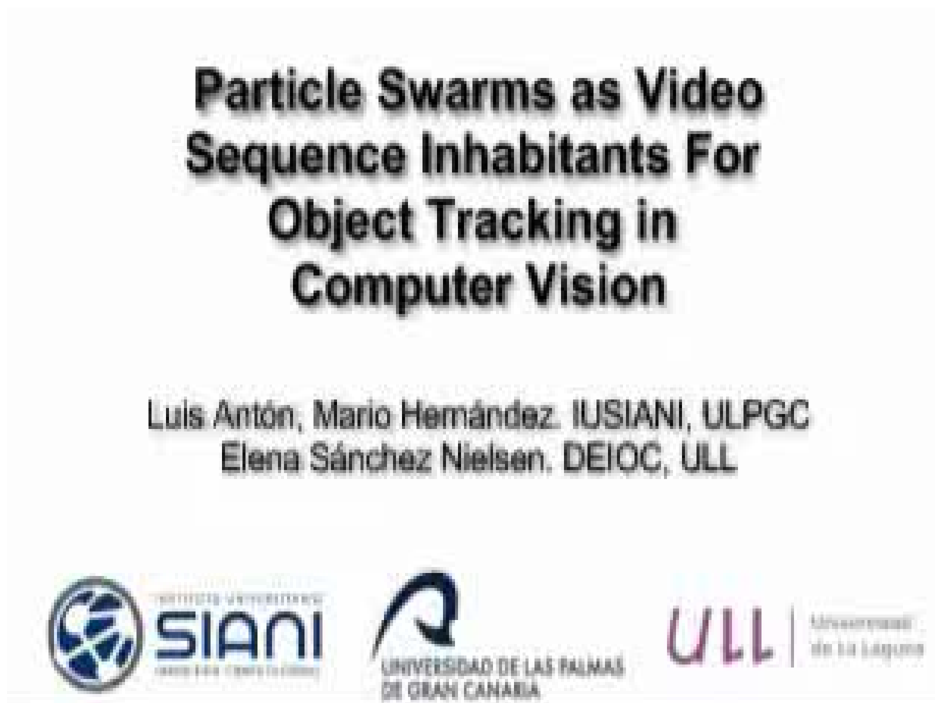
(convergent with constriction)


A 3D section
φ
Convergence
Non divergence
Re(v)
Re(y)

Implications from the algebra

- Oscillation for $\varphi<4$: Exploration near current best
- Divergence for $\varphi>4$: Exploration of the wider environment
- $\varphi=\alpha_1+\alpha_2$ is a combination of the attractiveness of the personal and global best. Since these might be not the same, a slightly larger φ might be needed.
- φ slightly above 4 (e.g. 4.1): particle stays somewhere in between or near personal and global best. If these two coincide the algorithm tends to diverge, i.e. the particle moves on searching elsewhere.
- Divergence can be counteracted by $V_{\max}$ or by constriction.
- Remember that we were considering an averaged version of the algorithm.

Object Tracking in Computer Vision



Applications

- Evolving structure and weights of neural networks
- Complex control involving complex and continuous variables (power systems)
- Industrial mixer in combination with classical optimization
- Image analysis
- Medical diagnosis
- Job scheduling
- Robot path planning, localization
- Electrical generator
- Electrical vehicle

Repulsive PSO algorithm

For each particle $1 \le i \le n$

- create m-dimensional random vectors r_1, r_2, r_3 with components drawn from $U[0,1]$
- update velocities

$$v_i \leftarrow \omega v_i + \alpha_1 r_1 \circ \left(\hat{x}_i - x_i\right) + \alpha_2 r_2 \circ \left(\hat{y} - x_i\right) + \alpha_3 \omega r_3 \circ z$$

- update positions etc.

$\circ$ componentwise multiplication

z random velocity

$\hat{y}$ best of a random neighbor or global best, $\alpha_2 < 0$

- Usually applied alternating with canonical PSO if diversity becomes too small
- Properties: sometimes slower, more robust and efficient

Fully Informed Particle Swarm (FIPS)

- Rui Mendes (2004): Simpler, maybe better
- Distributes total φ across n terms
- All neighbors contribute to the velocity adjustment
- Best neighbor is not selected, but included with a
- Individual not included in neighborhood

$$\vec{\varphi}_k = \vec{\mathrm{U}}\left[0, \frac{\varphi_{max}}{|\mathcal{N}|}\right] \;\; \forall k \in \mathcal{N}$$

$$\vec{P}_m = \frac{\sum_{k \in \mathcal{N}} \mathcal{W}(k)\, \vec{\varphi}_k \otimes \vec{P}_k}{\sum_{k \in \mathcal{N}} \mathcal{W}(k)\, \vec{\varphi}_k}$$

- Fails often, but, if successful, results are good, (stongly dependent on good topology)

Parameters, Conditions, & Tweaks

- Initialization methods
- Population size
- Population diameter
- Absolute vs. signed velocities
- Population topology
- Births, deaths, migration
- Limiting domain (X_{MAX}, V_{MAX})
- Multiobjective optimization
- “Subvector” techniques
- Comparison over problem spaces
- Hybrids

Jim Kennedy
Russ Eberhart:
Tutorial on Particle Swarm Optimization

IEEE Swarm Intelligence Symposium 2005
Pasadena, California
USA, June 8, 2005

Remarks on PSO

- Consider boundaries as physical (e.g. by reflection from walls)
- Try adaptive versions: variable swarm size, variable ratios α_1/α_2
- Try different topologies (e.g. “tribes”)
- For local variants, consider using other norms in high-dimensional spaces (Euclidean unit sphere volume decays)

Relation to probabilistic methods

- Strict probabilistic methods are based on assumtions (Gaussianity, optimal sampling etc.) which often do not hold in practical applications
- There are many examples where meta-heuristic approaches do well
  - Toy examples are often designed ad-hoc for a particular method and are thus unsuitable for a fair comparison.
  - Success in real-world examples depends much on domain knowledge, quality of analysis, iterative re-design etc.
- Meta-heuristic algorithms may use strict algorithms for local search
- Meta-heuristic algorithms can be used to initialize, adapt, optimize or tune the “exact” algorithms

Particle filters

- Given observations Y_k; reconstruct true states X_k

  Initial distribution $p\left(X_0 \middle| Y_0\right) = P\left(X_0\right)$

  Markovian evolution $p\left(X_k \middle| Y_{1,\ldots,k-1}\right) = \int p\left(X_0 \middle| X_{k-1}\right) p\left(X_{k-1} \middle| Y_{1,\ldots,k-1}\right)$

  Bayes' rule $p\left(X_k \middle| Y_{1,\ldots,k}\right) = \dfrac{p\left(Y_k \middle| X_k\right) p\left(X_k \middle| Y_{1,\ldots,k-1}\right)}{p\left(Y_k \middle| Y_{1,\ldots,k-1}\right)}$

- Represent posterior distribution by N weighted samples obtained from $p\left(X_k \middle| Y_{1,\ldots,k-1}\right)$: $p\left(X_k \middle| Y_{1,\ldots,k}\right) \propto \sum_{i=1}^{N} p\left(Y_k \middle| X_k\right) p_i\left(X_k\right)$
- Problems: impoverishment and sample size effects (if the likelihood is concentrated at the tail of the prior

G. Tong, Z. Fang, X. Xu (2006) A PS optimized PF for non-linear system state estimation. Proc. Congress on Evolutionary Computation, 438-442.

PSO PF

- Use PSO for sampling
- Standard PSO with Gaussian randomness in the velocity update ("Gaussian swarm")
- fitness $$f = \exp\left(-\frac{1}{2R_k}\left(y_{\text{new}} - y_{\text{pred}}\right)^2\right)$$
- R_k: observation covariance
- modulate weights: $$w_i^k = w_i^{k-1} p\left(y^k \middle| x_i^k\right)$$
- Now represent posterior by weighted samples
- Avoids divergence and does well with less particles.

Comparison of GA and PSO

- Generally similar:
  1. Random generation of an initial population
  2. Caclulate of a fitness value for each individual.
  3. Reproduction of the population based on fitness values.
  4. If requirements are met, then stop. Otherwise go back to 2.
- Modification of individuals
  - In GA: by genetic operators
  - In PSO: Particles update themselves with the internal velocity. They also have memory.
- Sharing of information
  - Mutual In GA. Whole population moves as a group towards optimal area.
  - One-way in PSO: Source of information is only gBest (or lBest). All particles tend to converge to the best solution quickly.
- Representation
  - GA: discrete
  - PS: continuous

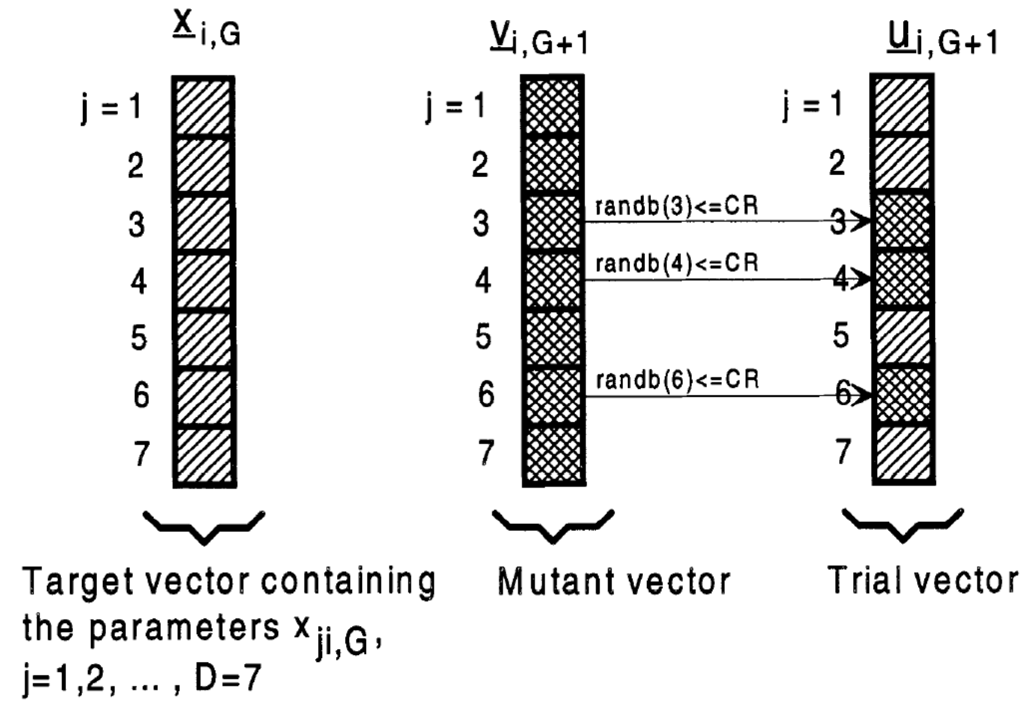
www.swarmintelligence.org/tutorials.php

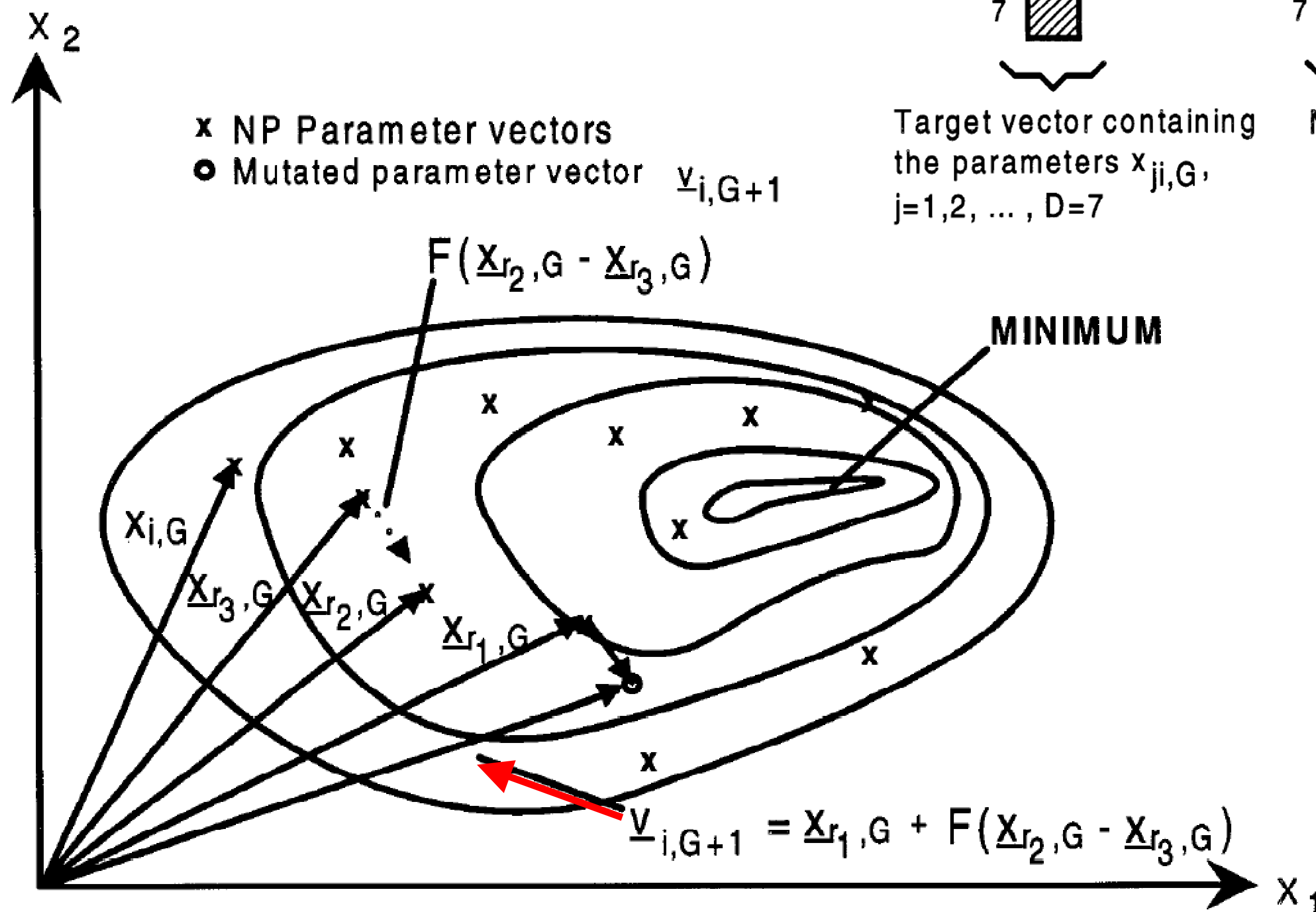
Differential Evolution

- *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 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,

Differential Evolution





DE: Details

- Properties
  - Simple, very fast
  - Reasonably good results
  - Diversity increases in flat regions (divergence property)
- Parameters
  - NP=5D (4 … 10D)
  - F=0.5 (0.4 …. 1.0)
  - CR=0.1 (0 … 1.0)

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

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

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

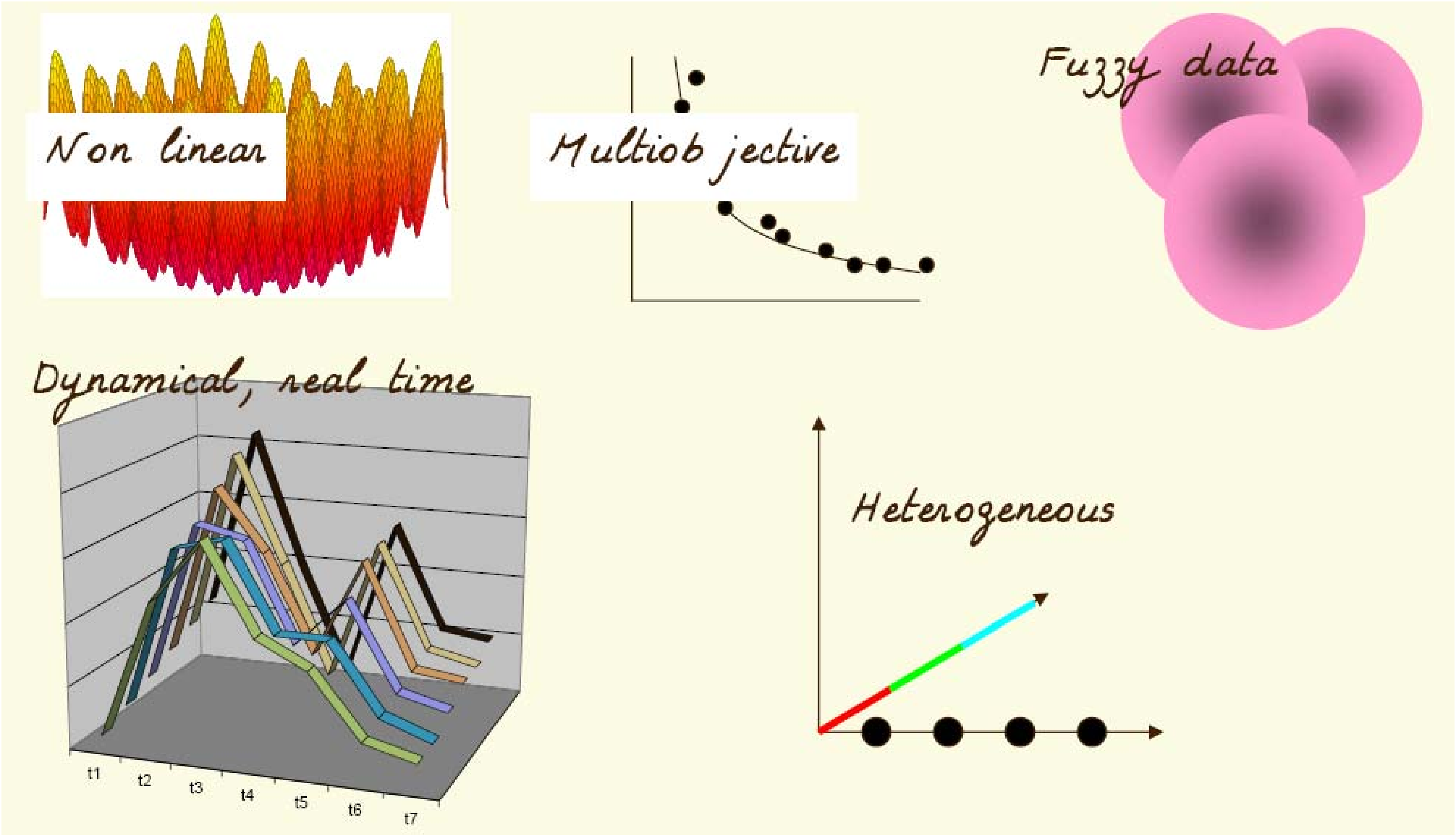
- Avoid local minima (leapfrog/crossover/more noise/ subpopulations/border of instability/checking success)

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

It is thanks to these eccentrics, whose behaviour is not conform to the one of the other bees, that all *fruits sources around the colony are so quickly found.*

Karl von Frisch *1927*

Ecological niche



Literature on swarms

- Eric Bonabeau, Marco Dorigo, Guy Theraulaz: Swarm Intelligence: From Natural to Artificial Systems (Santa Fe Institute Studies on the Sciences of Complexity) (Paperback) OUP USA (1999)
- J. Kennedy, and R. Eberhart, *Particle swarm optimization*, in Proc. of the IEEE Int. Conf. on Neural Networks, Piscataway, NJ, pp. 1942–1948, 1995.
- Y Shi, RC Eberhart (1999) Parameter selection in particle swarm optimization. Springer.
- Eberhart Y. Shi (2001) PSO: Developments, applications ressources. IEEE.
- www.engr.iupui.edu/~eberhart/web/PSObook.html
- Tutorials: www.particleswarm.info/
- Bibliography: icdweb.cc.purdue.edu/~hux/PSO.shtml