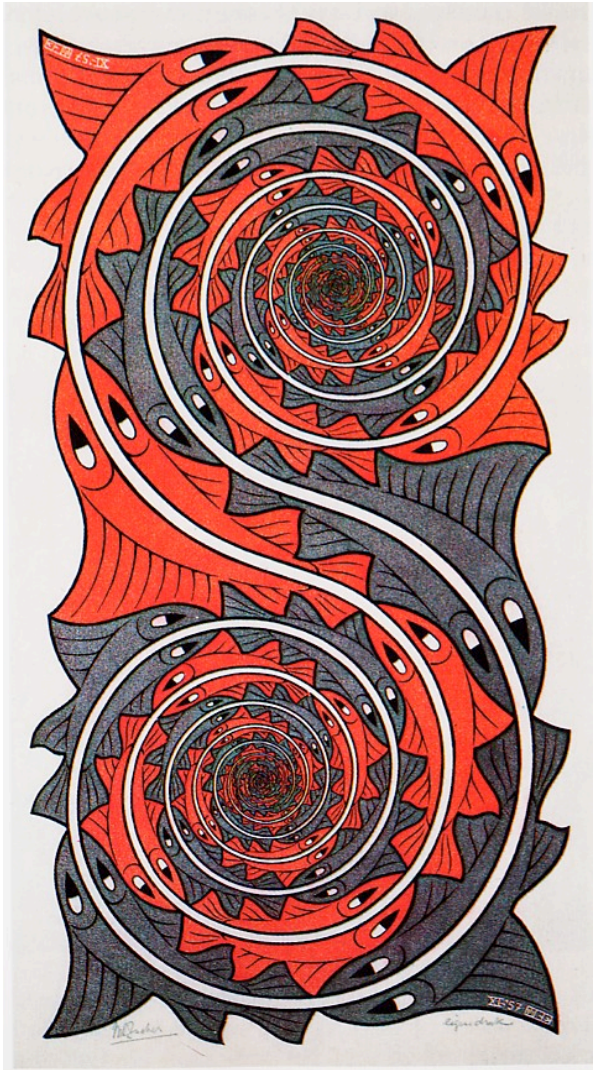
PARTICLE SWARM OPTIMIZATION

PSO

Outline

- Hystorical note
- Recapitulation and terminology
- PSO
  - Origin
  - Basics
  - Terminology
  - Flowchart and code
  - Numerical experiments
- The missing details
- Conclusions

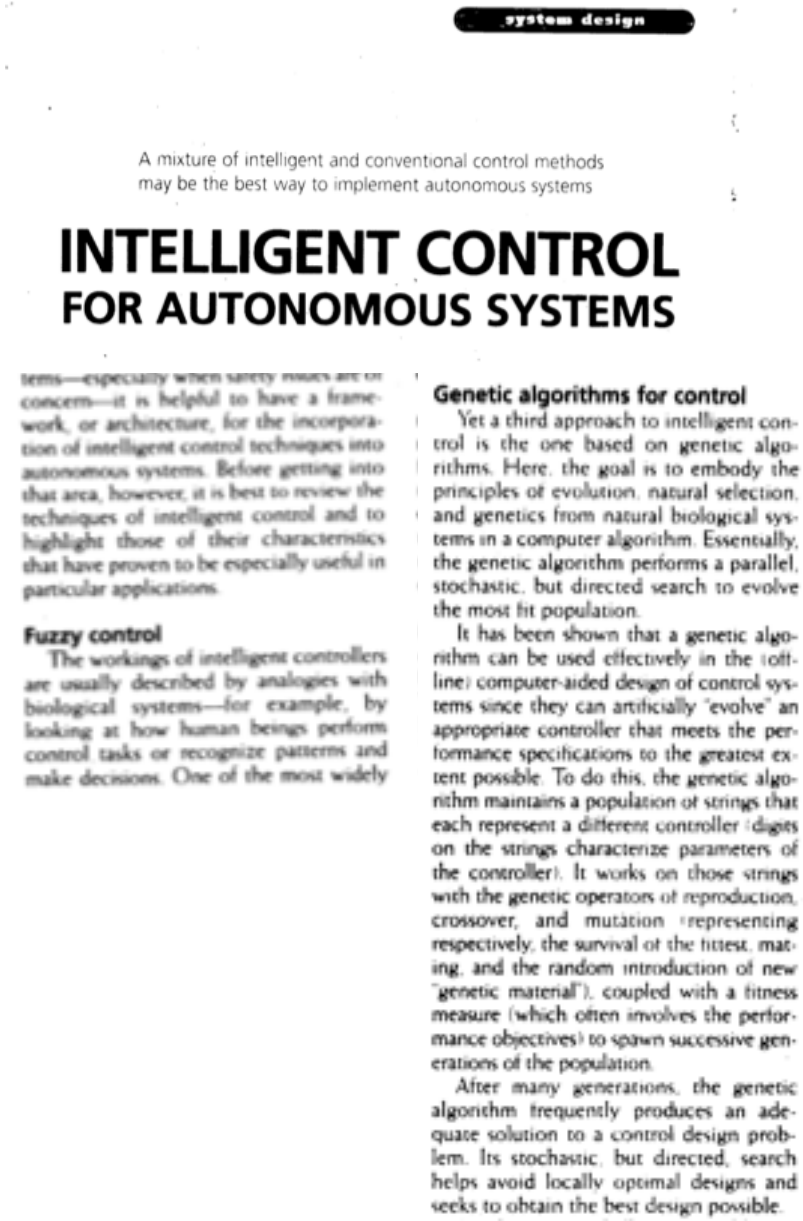


Whirlpools. November 1957. Wood engraving and woodcut, second state in red, gray, and black, printed from two blocks, 438 × 235 mm (cat. 423).

Hystorical note

- 1995, IEEE Spectrum

Codenamed **P6**, the **Intel Pentium Pro** was released by Intel in November 1995 and contained 5.5 million transistors, was available in clock speeds of 150MHz to 200MHz, and was the first Intel processor that utilized the i686 architecture.



system design

A mixture of intelligent and conventional control methods may be the best way to implement autonomous systems

INTELLIGENT CONTROL FOR AUTONOMOUS SYSTEMS

tems—especially when safety issues are of concern—it is helpful to have a framework, or architecture, for the incorporation of intelligent control techniques into autonomous systems. Before getting into that area, however, it is best to review the techniques of intelligent control and to highlight those of their characteristics that have proven to be especially useful in particular applications.

Fuzzy control

The workings of intelligent controllers are usually described by analogies with biological systems—for example, by looking at how human beings perform control tasks or recognize patterns and make decisions. One of the most widely

Genetic algorithms for control

Yet a third approach to intelligent control is the one based on genetic algorithms. Here, the goal is to embody the principles of evolution, natural selection, and genetics from natural biological systems in a computer algorithm. Essentially, the genetic algorithm performs a parallel, stochastic, but directed search to evolve the most fit population.

It has been shown that a genetic algorithm can be used effectively in the (off-line) computer-aided design of control systems since they can artificially "evolve" an appropriate controller that meets the performance specifications to the greatest extent possible. To do this, the genetic algorithm maintains a population of strings that each represent a different controller (digits on the strings characterize parameters of the controller). It works on those strings with the genetic operators of reproduction, crossover, and mutation (representing respectively, the survival of the fittest, mating, and the random introduction of new "genetic material"), coupled with a fitness measure (which often involves the performance objectives) to spawn successive generations of the population.

After many generations, the genetic algorithm frequently produces an adequate solution to a control design problem. Its stochastic, but directed, search helps avoid locally optimal designs and seeks to obtain the best design possible.

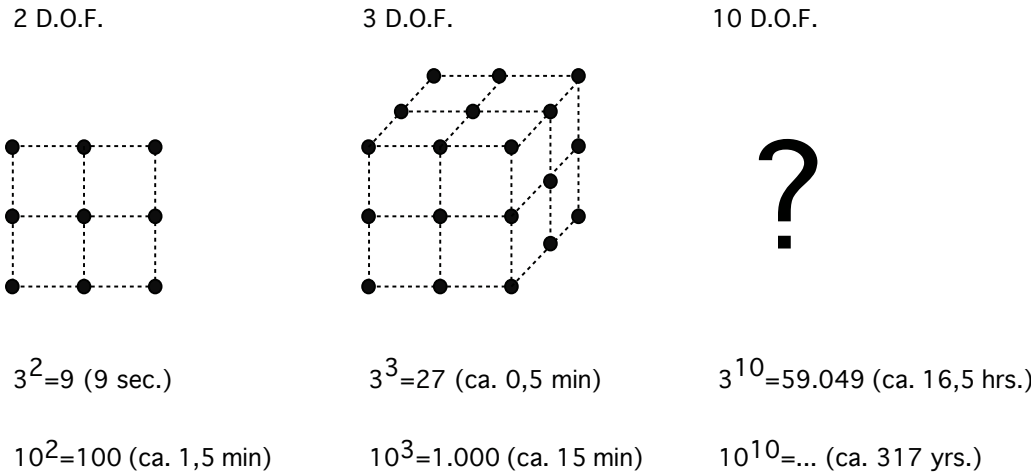
Recapitulation and terminology

- Optimization (minimization, optimal design, inverse pb.,…)
- Variables (degrees of freedom, parameters, (derivatives?)…)
- Objectives (objective functions, quality, fitness, …)
- Constraints (equality, inequality, feasible, unfeasible, …)
- Minimum (local, global, sensitivity, robustness, …)
- Analysis (forward problem, cheap/expensive)

- Did I forget something ?

Recapitulation and terminology

- Curse of dimensionality

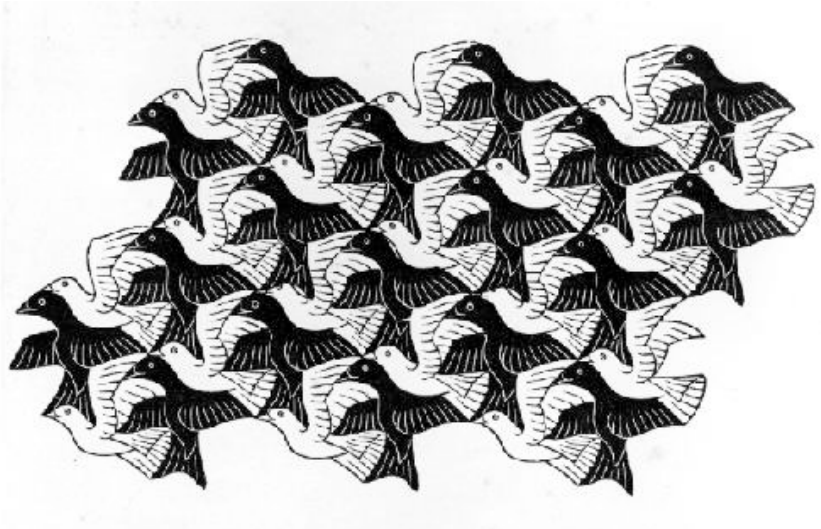


- No-free-lunch theorem
  - D. H. Wolpert and W.G. Macready, "No Free Lunch Theorems for Optimization," IEEE Transactions on Evolutionary Computation 1, pp. 67, 1997.
  - “any two optimization algorithms are equivalent when their performance is averaged across all possible problems”
  - BUT: we do not try to solve ALL possible problems, only a domain-specific subset !

Origins

- J. Kennedy and R. Eberhart, "Particle Swarm Optimization", *Proc. of IEEE International Conf. on Neural Networks* **IV**, pp. 1942–1948, 1995.

- Russel Ebenhart (Electrical Engineer) and James Kennedy (Social Psychologist) in 1995 (both U. Indiana, Purdue).
- Categories: Swarm Intelligence techniques and Evolutionary Algorithms for optimization.
- Inspired by the **social behavior** of birds, studied by Craig Reynolds (a biologist) in late 80s and early 90s. He derived a formula for representation of the flocking behavior of birds.
- This was later used in computer simulations of virtual birds, known as Boids.
- Ebenhart and Kennedy recognized the suitability of this technique for optimization



In ES population members derive from each other but do not exchange information, in PSO they do!

Basics

- Main idea: Mimic bird flocking or fish schooling

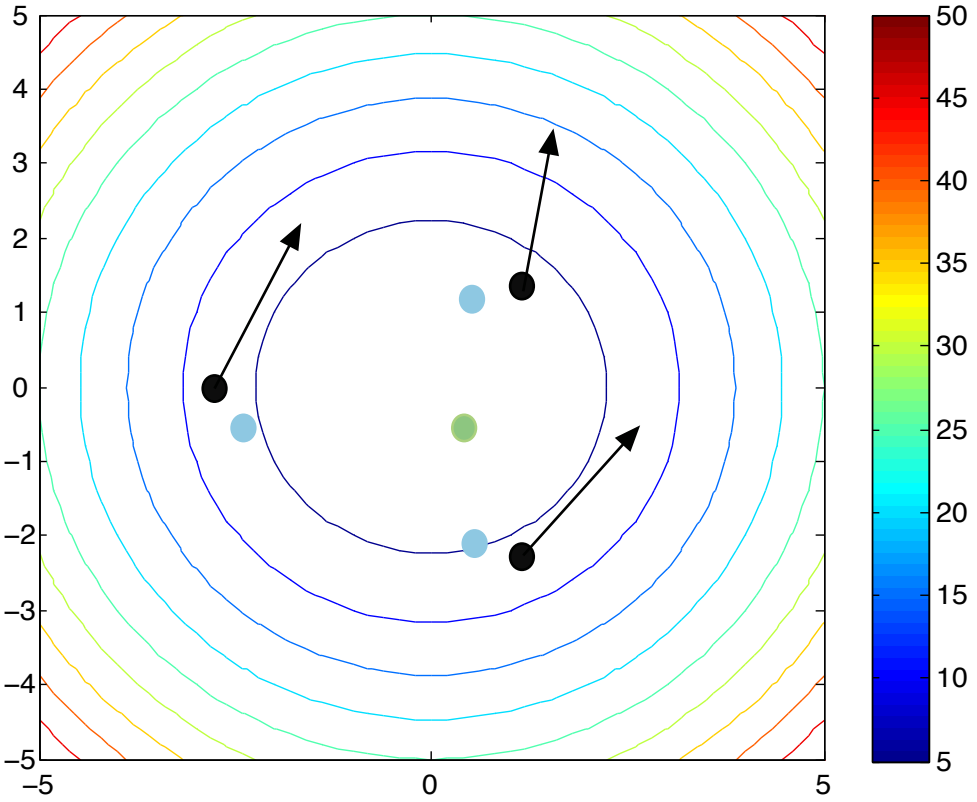
Nature	Algorithm
Birds or fish	Particles
Explore the environment (3d) in search for food	Explore objective space (Nd) in search for good function values
Exchange information by acoustical or optical means	Exchange information by sharing positions of promising locations

Basic models of flocking behavior are controlled by three simple rules:

- Separation - avoid crowding neighbors (short range repulsion)
- Alignment - steer towards average heading of neighbors
- Cohesion - steer towards average position of neighbors (long range attraction)

Terminology

- Particles
- Velocities
- Personal best
- Global best



Particles: $\mathbf{x}_i$

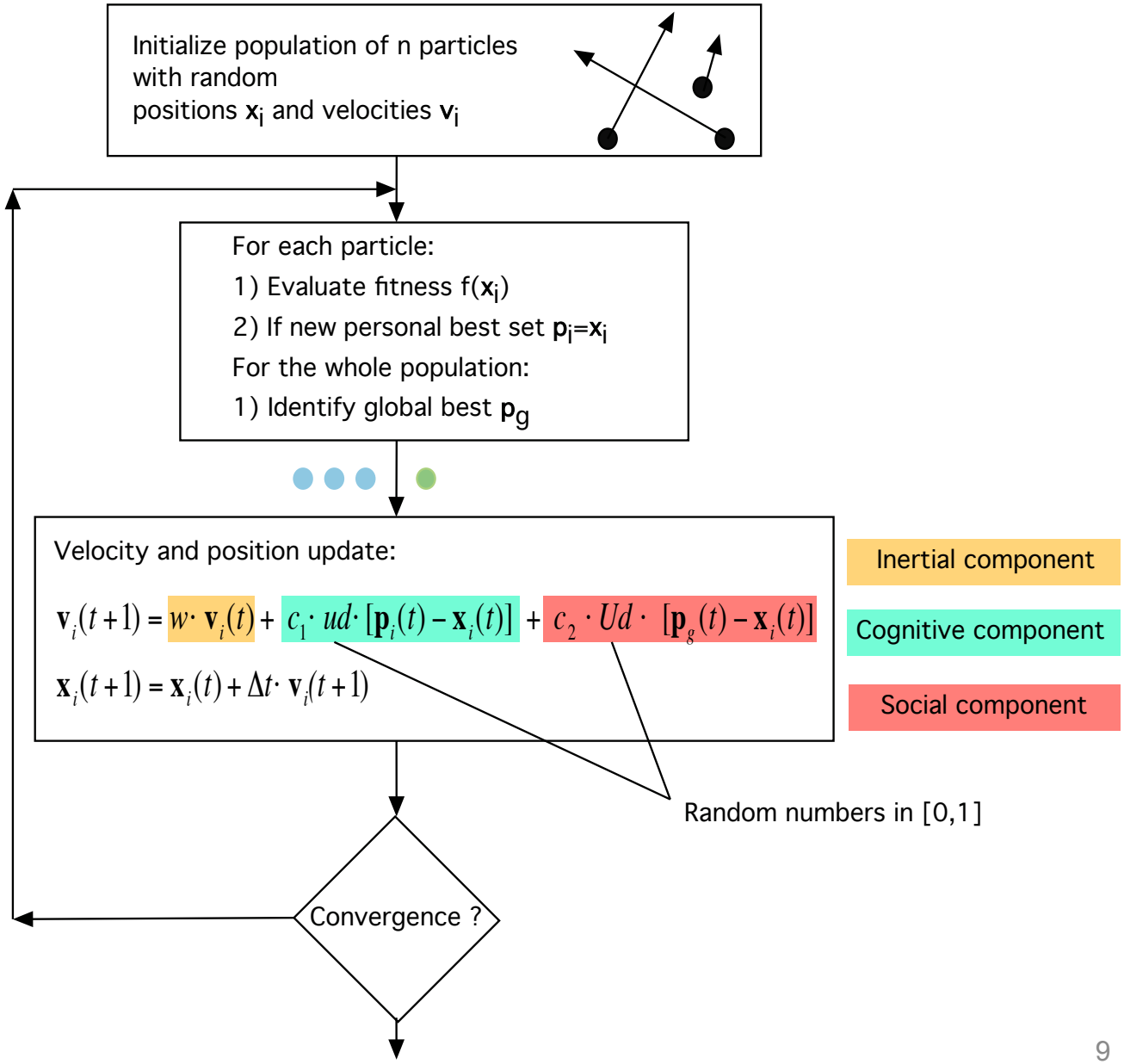
Velocities: $\mathbf{v}_i$

Personal best positions ever: $\mathbf{p}_i$

Global best position ever: $\mathbf{g}_{best}$

Flowchart

- Alg. params.:
  - n
  - w, c_1, c_2



Code

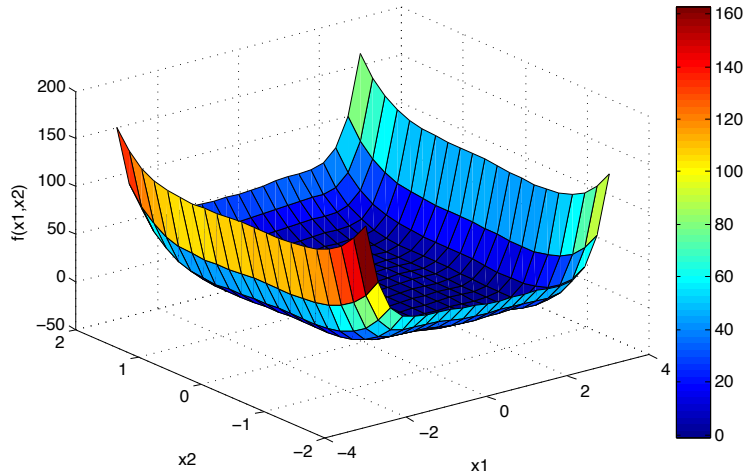
- Matlab

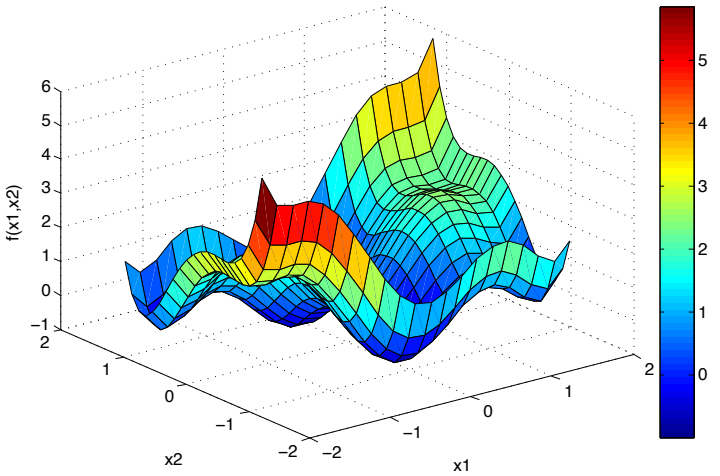
```
clear all; close all; clc
%Function to be minimized
D=2;
objf=inline('4*x1^2-2.1*x1^4+(x1^6)/3+x1*x2-4*x2^2+4*x2^4','x1','x2');
objf=vectorize(objf);
%Initialization of PSO parameters
N=20; %population size (total function evaluations will be itmax*N)
itmax=30;
c1=1.05; c2=1.05;
wmax=1; wmin=0.3; %set to same value for constant w
w=linspace(wmax,wmin,itmax); %linear variation of w
%Problem and velocity bounds
a(1:N,1)=-1.9;  b(1:N,1)=1.9; %bounds on variable x1
a(1:N,2)=-1.1;  b(1:N,2)=1.1; %bounds on variable x2
d=(b-a);
m=a; n=b;
q=(n-m)/4; %initial velocities are 1/4 of parameter space size
%Random initialization of positions and velocities
x=a+d.*rand(N,D);
v=q.*rand(N,D);
%Evaluate objective for all particles
f=objf(x(:,1),x(:,2));
%Find gbest and pbest (in this case coincides with x)
[fgbest,igbest]=min(f);
gbest=x(igbest,:);
pbest=x; fpbest=f;
%Iterate
for it=1:itmax;
    %Update velocities and positions
    v(1:N,1:D)=w(it)*v(1:N,1:D)+c1*rand*(pbest(1:N,1:D)-x(1:N,1:D))↵
+c2*rand*(repmat(gbest,N,1)-x(1:N,1:D));
    x(1:N,1:D)=x(1:N,1:D)+v(1:N,1:D);
    %Evaluate objectives
    f=objf(x(:,1),x(:,2));
    %Find gbest and pbest
    [minf,iminf]=min(f);
    if minf<= fgbest
       fgbest=minf; gbest=x(iminf,:);
    end
    inewpb=find(f<=fpbest);
    pbest(inewpb,:)=x(inewpb,:); fpbest(inewpb)=f(inewpb);
end %end loop on iterations
[gbest,fgbest]
```

Please note that this code actually works !

Benchmark

- Six hump camel back





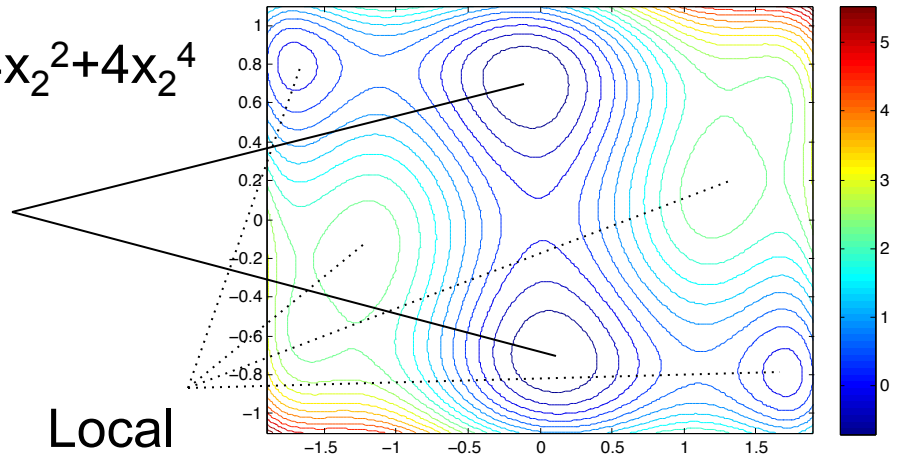
$$f(x_1,x_2)=4x_1^2-2.1x_1^4+x_1^6/3+x_1x_2-4x_2^2+4x_2^4$$

$x_1^*=-0.089842,\ x_2^*=0.712656$

$x_1^*=0.089842,\ x_2^*=-0.712656$

$f^*=-1.031628453$

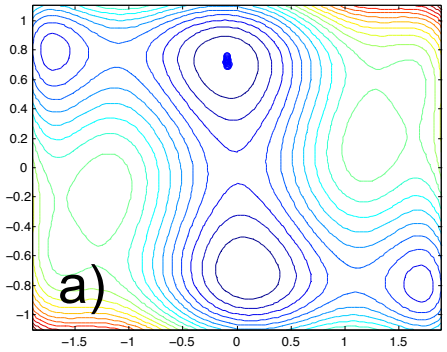
Global

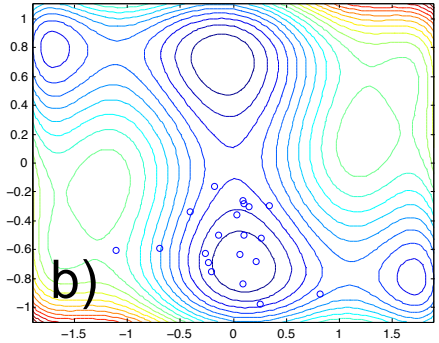


Local

What happens if…

- Inertia (1000 runs, $|f_{opt}-f^*|$)





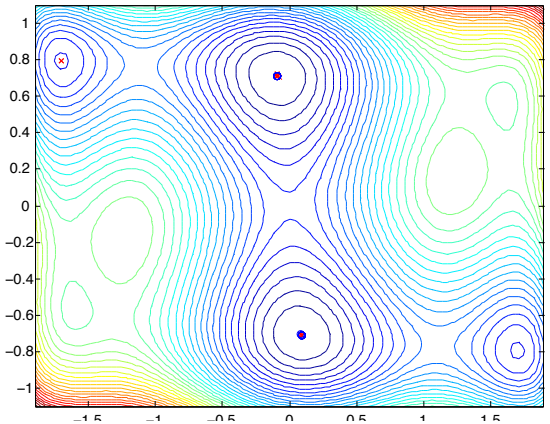
	w	N.	Iter.	Tot.	Max.	Avg.	Min.	Std. Dev.
1)	0,1	20	30	600	8,4 E-01	3,2 E-02	1,4 E-11	8,6 E-02 (b)
2)	0,5	20	30	600	8,2 E-02	2,4 E-03	1,5 E-12	4,5 E-02
3)	1,0	20	30	600	2,4 E-01	5,0 E-03	9,8 E-07	1,2 E-02
4)	2,0	20	30	600	1,0 E-00	1,6 E-01	1,1 E-04	1,6 E-01
5)	1➔0,5	20	30	600	1.5 E-02	1,4 E-04	6,1 E-09	8,7 E-04
6)	1➔0,3	20	30	600	**6,9 E-04**	**4,7 E-06**	4,6 E-12	**3,3 E-05 (a)**
7)	1➔0,1	20	30	600	5,0 E-03	8,1 E-06	2,8 E-12	1,7 E-04

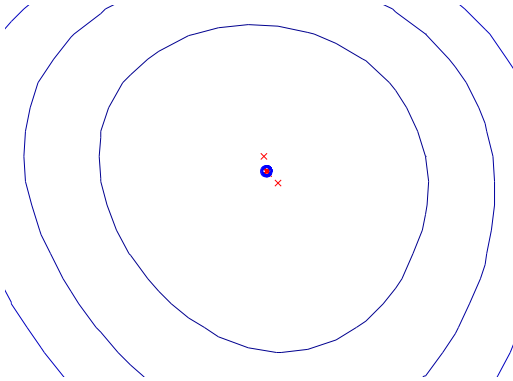
- $c_1 = c_2 = 1{,}05$
- $v_{ini} = \Delta x/4$

Case 6) has best worst case, best results on average, and is the most robust (lowest standard deviation) !

What happens if…

- Population size
  (1000 runs, $|f_{opt}-f^*|$)





	N.	Iter.	Tot.	Max.	Avg.	Min.	Std. Dev.
1)	40	15	600	2,2 E-03	3,2 E-05	3,0 E-10	**1,4 E-04**
2)	30	20	600	1,4 E-02	2,3 E-05	1,3 E-10	4,6 E-04
3)	20	30	600	2,1 E-02	**2,0 E-05**	5,2 E-12	4,7 E-04
4)	10	60	600	8,2 E-01	8,9 E-04	5,0 E-12	2,6 E-02
5)	5	120	600	9,9 E-01	7,5 E-03	1,8 E-11	7,2 E-02

- $c_1 = c_2 = 1{,}05$
- w = lin. from 1,0 to 0,3
- $v_{ini} = \Delta x/4$

Small populations (red cross in figures) don't perform well. Cases 1), 2) and 3) work well. Case 3) will be used in the following.

What happens if…

- Cognitive component (1000 runs, $|f_{opt}-f^*|$)

c_1	N.	Iter.	Tot.	Max.	Avg.	Min.	Std. Dev.
0,0	20	30	600	**6,9 E-03**	**1,9 E-05**	1,5 E-12	**4,4 E-04**
0,5	20	30	600	1,9 E-02	2,5 E-05	6,2 E-13	6,1 E-04
1,05	20	30	600	2,5 E-01	2,6 E-04	2,1 E-11	8,0 E-03
1,5	20	30	600	8,6 E-01	8,3 E-04	8,6 E-13	2,6 E-02
2,0	20	30	600	6,2 E-02	1,6 E-03	1,4 E-11	3,9 E-02

- $c_2 = 1,05$
- w = lin. from 1,0 to 0,3
- $v_{ini} = \Delta x/4$

It seems that it would be better not to have the cognitive component at all (c1=0) but wait for next slide…

What happens if…

- Social component (1000 runs, $|f_{opt}-f^*|$)

c_2	N.	Iter.	Tot.	Max.	Avg.	Min.	Std. Dev.
0,0	20	30	600	3,7 E-01	1,5 E-02	2,2 E-07	3,2 E-02
0,5	20	30	600	1,1 E-01	2,2 E-03	4,5 E-09	3,9 E-03
1,05	20	30	600	1,6 E-02	2,3 E-04	1,1 E-10	5,3 E-04
1,5	20	30	600	**4,1 E-03**	**2,1 E-05**	2,5 E-11	**2,0 E-04**
2,0	20	30	600	1,6 E-01	4,0 E-04	3,4 E-11	5,5 E-03

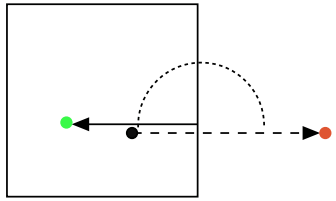
- c_1 = 1,05
- w = lin. from 1,0 to 0,3
- v_{ini}= $\Delta x/4$

The combined analysis of this and the preceeding table shows that c_2 should be bigger than c_1, but not too much…

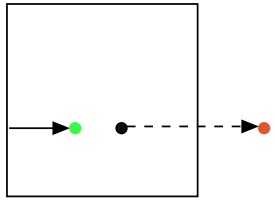
The missing pieces

- What happens if a particle flies out of the parameter space?
  - Mirroring (a1)
  - Loop around (a2)
  - Random (a3)
- What happens if a particle violates constraints an becomes unfeasible?
  - Discard (b1)
  - Keep (b2)
- Integer/discrete variables
- Adaptivity and different random number distributions

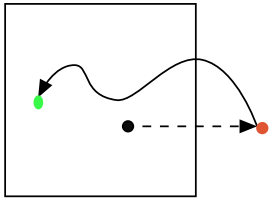
a1



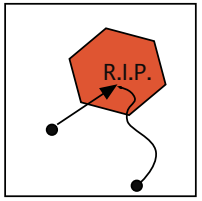
a2



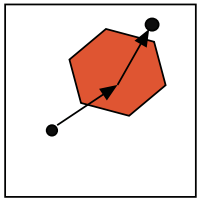
a3



b1



b2



Conclusions

- PSO algorithm basics introduced
- Ease of Matlab implementation shown
- Numerical experiments:
  - Inertia should be linearly decreasing between 1 and 0,3
  - Population size matters (N between 20 and 30)
  - $c_2=c_1+\alpha$,with α between 1 and 1,5

- Thanks for attending
- See you next time !