

Particle swarm optimization (PSO): a potentially useful tool for chemometrics?

Federico Marini¹, Beata Walczak²

¹Sapienza University of Rome, Rome, Italy

²Silesian University, Katowice, Poland

Rome, oct. 2009



Particle Swarm Optimization

- Particle Swarm Optimization (PSO) applies concept of social interaction to problem solving.
- It was developed in 1995 by James Kennedy and Russ Eberhart [Kennedy, J. and Eberhart, R. (1995). "Particle Swarm Optimization", *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942-1948, IEEE Press.] (<http://dsp.jpl.nasa.gov/members/payman/swarm/kennedy95-ijcnn.pdf>)
- It has been applied successfully to a wide variety of search and optimization problems.
- In PSO, a swarm of n individuals communicate either directly or indirectly with one another search directions (gradients).
- PSO is a simple but powerful search technique.

Particle Swarm Optimization - 2

- Evolutionary computation
- Similar to GA (starts with a population of randomly generated solution)
- Unlike GA, each potential solution, called *particle*, is assigned a random velocity and is “flown” through the solution space
- Each particle keeps track of the coordinates of associated to the best solution it has found so far (*pbest*).
- In addition, it keeps record also of the best solution achieved so far by any of the particles in its neighborhood (*lbest* or *gbest*)

Particle Swarm Optimization: Swarm Search

- In PSO, particles never die!
- Particles can be seen as simple agents that fly through the search space and record (and possibly communicate) the best solution that they have discovered.
- So the question now is, “How does a particle move from one location in the search space to another?”
- This is done by simply adding the velocity to the x-vector to get an updated version

$$\mathbf{x}_i(t) = \mathbf{x}_i(t-1) + \mathbf{v}_i(t)$$

PSO: the algorithm

- Initialize a population of particles with random position and velocities in d dimensions
- For each particle, evaluate the fitness
- Compare particle fitness with its pbest. If larger $pbest_i = fit_i(t)$
- Compare particle fitness with global best. If larger $gbest = fit_i(t)$
- Change velocity of the particle according to:

$$v_{id}(t) = v_{id}(t-1) + c1 * rand * (p_{id} - x_{id}) + c2 * rand * (g_d - x_{id})$$

- Update velocity according to:

$$\mathbf{x}_i(t) = \mathbf{x}_i(t-1) + \mathbf{v}_i(t)$$

- Repeat the procedure until convergence or maximum iterations

Particle Swarm Optimization: Swarm Types

- In his paper, [Kennedy, J. (1997), “The Particle Swarm: Social Adaptation of Knowledge”, Proceedings of the 1997 International Conference on Evolutionary Computation, pp. 303-308, IEEE Press.]
- Kennedy identifies 4 types of PSO based on $c1$ and $c2$.
- Given:

$$v_{id}(t) = v_{id}(t-1) + c1 * rand * (p_{id} - x_{id}) + c2 * rand * (g_d - x_{id})$$

- Full Model ($\varphi1, \varphi2 > 0$)
- Cognition Only ($\varphi1 > 0$ and $\varphi2 = 0$),
- Social Only ($\varphi1 = 0$ and $\varphi2 > 0$)
- Selfless ($\varphi1 = 0, \varphi2 > 0$, and $g \neq i$)

Particle Swarm Optimization: Related Issues

- There are a number of related issues concerning PSO:
 - Controlling velocities (determining the best value for V_{\max}),
 - Swarm Size,
 - Neighborhood Size,
 - Updating X and Velocity Vectors,
 - Robust Settings for ($c1$ and $c2$)

Particle Swarm: Controlling Velocities

- When using PSO, it is possible for the magnitude of the velocities to become very large.
- For this reason, velocities are bounded and a maximum absolute value is set.
- V_{\max} determines the resolution with which regions between the present position and the best solution are searched.
- V_{\max} too high \rightarrow particles may fly past good solutions
- V_{\max} too low \rightarrow particles may not explore sufficiently beyond local optima
- Two methods were developed for controlling the growth of velocities:
 - A dynamically adjusted inertia factor, and
 - A constriction coefficient.

Particle Swarm Optimization: The Inertia Factor

- Larger V_{\max} facilitates global exploration, smaller V_{\max} local exploitation
- The concept of inertia weight was introduced to control the balance between exploration and exploitation
- Theoretically, no need for V_{\max}
- When the inertia factor is used, the equation for updating velocities is changed to:

$$v_{id}(t) = \omega(t) * v_{id}(t-1) + c1 * rand * (p_{id} - x_{id}) + c2 * rand * (g_d - x_{id})$$

- Where ω is initialized to 1.0 and is gradually reduced over time (measured by cycles through the algorithm).
- For practical reasons, even if V_{\max} can't be completely eliminated it is enough if it is set to the dynamical range of the var.

PSO: acceleration constants

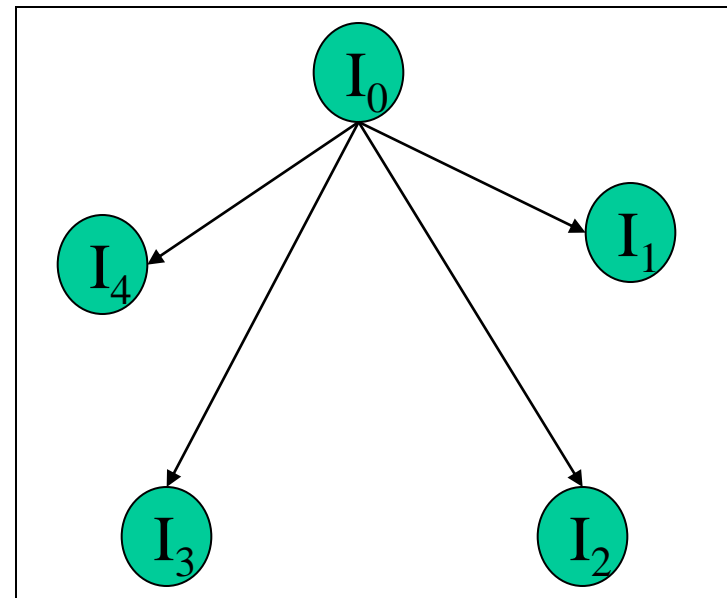
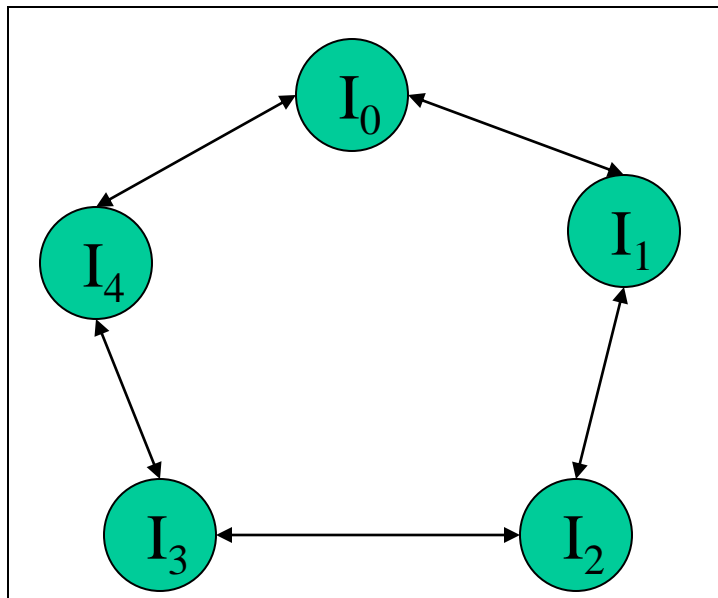
- Acceleration constants (c_1 and c_2) represent the weights of the stochastic acceleration terms
- Low values allow particles to roam far from best solutions before being pushed back
- High values result in abrupt movement towards or past target regions.
- For many applications, a value of 2 for both is usually good

Particle Swarm Optimization: Swarm and Neighborhood Size

- Two possible implementations of PSO: local and global
- In local, particles have information only on their own and neighbors' best rather than of the entire population
- Instead of moving towards a stochastic average of pbest and gbest, they move towards pbest and lbest.
- Neighbors are defined topologically not based on the solution space
- If neighborhood size =2, particle i compares its fitness with $(i-1)$ and $(i+1)$
- Concerning the swarm size for PSO, as with other ECs there is a trade-off between solution quality and cost (in terms of function evaluations).
- It is reported that when using local neighborhood, a neighborhood size of approx. 15% or the particles provides good results.

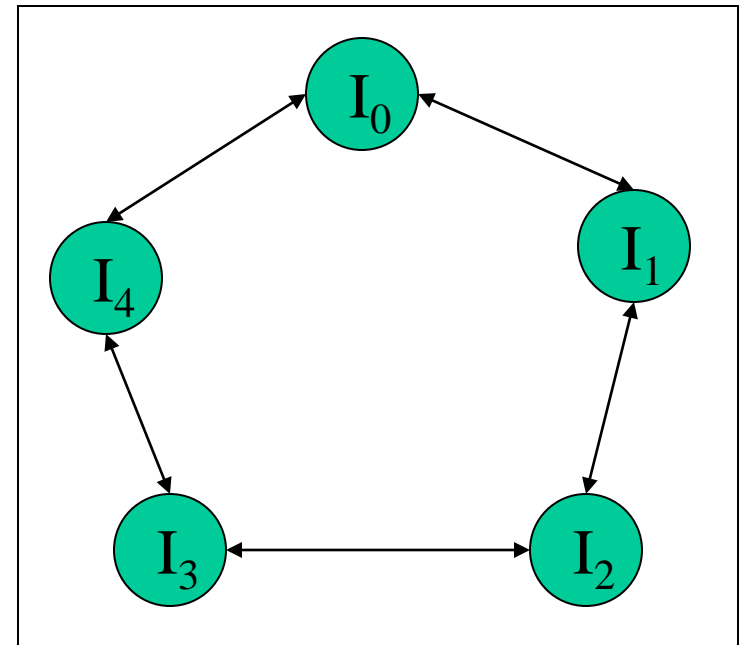
Particle Swarm Optimization: Swarm Topology

- In PSO, there have been two basic topologies used in the literature
 - Ring Topology (neighborhood of 3)
 - Star Topology (global neighborhood)



Particle Swarm Optimization: Particle Update Methods

- There are two ways that particles can be updated:
 - Synchronously
 - Asynchronously
- Asynchronous update allows for newly discovered solutions to be used more quickly



Binary PSO

- As was presented now, PSO works for optimization of real-valued parameters
- It can be slightly modified to cope with binary based vectors

$$v_{id}(t) = \omega(t) * v_{id}(t-1) + c1 * rand * (p_{id} - x_{id}) + c2 * rand * (g_d - x_{id})$$

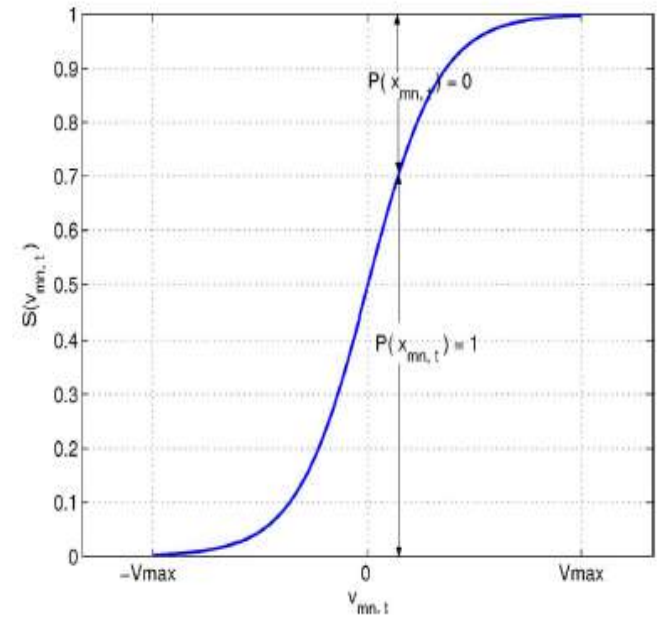
- Distance of each bit to the corresponding bit in personal or global best can take 3 values only:
 - -1 if $p_d = 1$ and $x_d=0$
 - 0 if p_d and x_d are either both 1 or both 0
 - 1 if $p_d=0$ and $x_d=1$
- When multiplied by constants and after considering inertia velocity is still a real valued vector

Binary PSO

- Probability threshold

$$S(v_{id}) = \frac{1}{1 + e^{-v_{id}}}$$

- Generate random value r in $[0,1]$
- If $S(v) > r$ flip the bit, else keep the bit as it is.



Example I:
Real valued-PSO for warping of
chromatographic data

Warping

- Signals can be stretched or shrunken along the horizontal (time) scale with respect to a reference signal
- It's a problem when data are to be used as input for multivariate statistical analysis
- **Time warping** → Transforming time so that the peaks in different chromatograms are aligned precisely
- Majority of algorithms based on dynamic programming (**DTW** or **COW**).
- These algorithms can lead to bad results as sudden jumps or plateaus.
- In this case study, the starting point was the Parametric Time Warping algorithm (Eilers, 2004).

Parametric Time Warping

- Models the warping function directly
- $t'=w(t)$ so that $y(t)$ and $x(t')$ are as “close” as possible.
- In the original formulation:
 - The warping function was assumed to be a polynomial (usually quadratic):
$$w(t) = \sum_{k=0}^K a_k t^k$$
 - A sum of squares loss function was assumed
- This leads to a regression problem whose solution needs to be found iteratively until convergence.

Parametric Time Warping (this study)

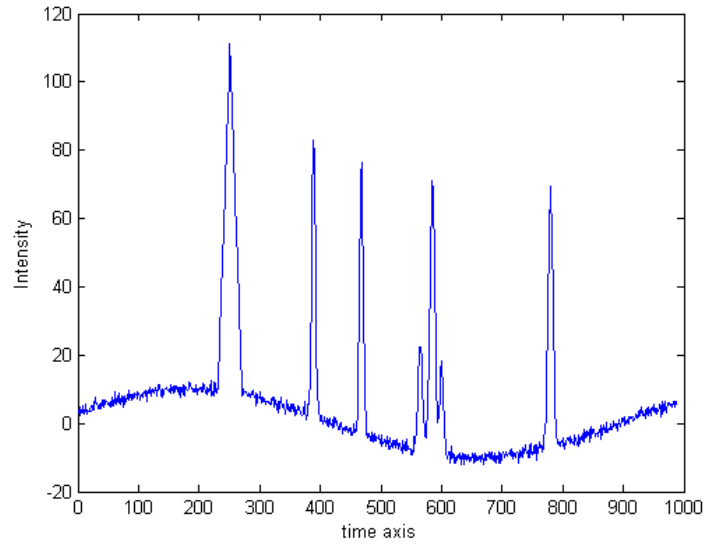
- The warping function is modeled using an RBF approach

$$t' = \sum_{i=1}^N w_i e^{-\frac{(t-t_i)^2}{2\sigma^2}}$$

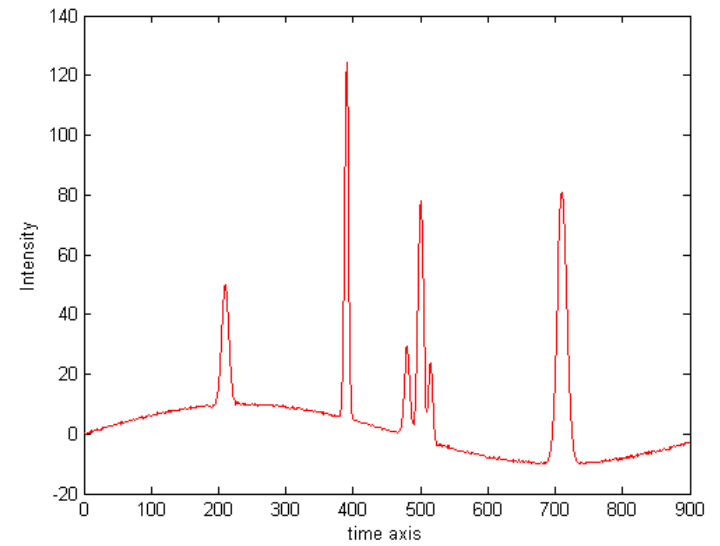
- Centers are evenly spaced along the time scale
- PSO is used to find the optimal value of the RBF parameters (weights).
- A value of 100 was fixed for the parameter σ .
- Correlation between the warped and the target spectrum is chosen as the fitness function.

The data

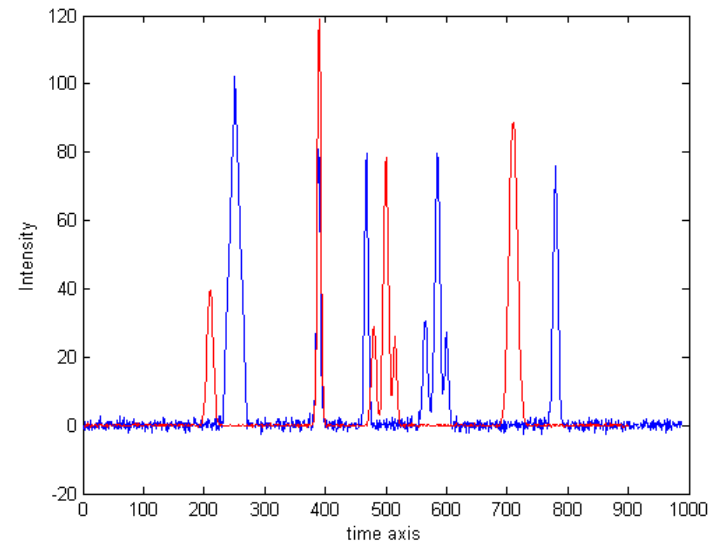
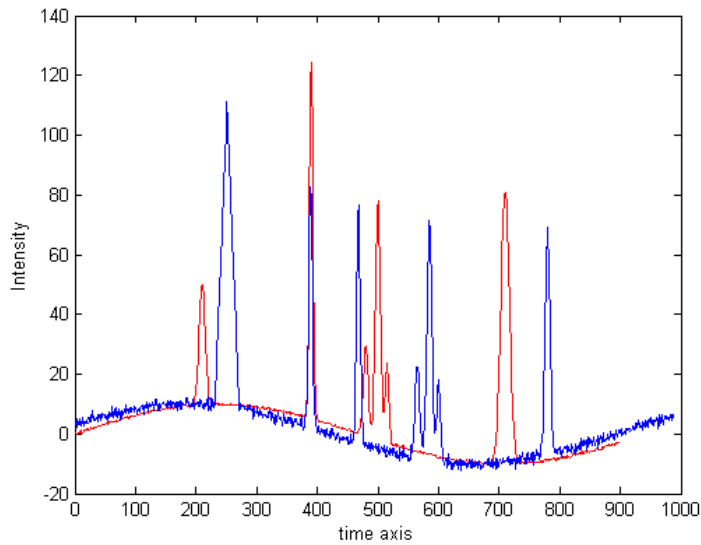
Target profile



Profile to be warped

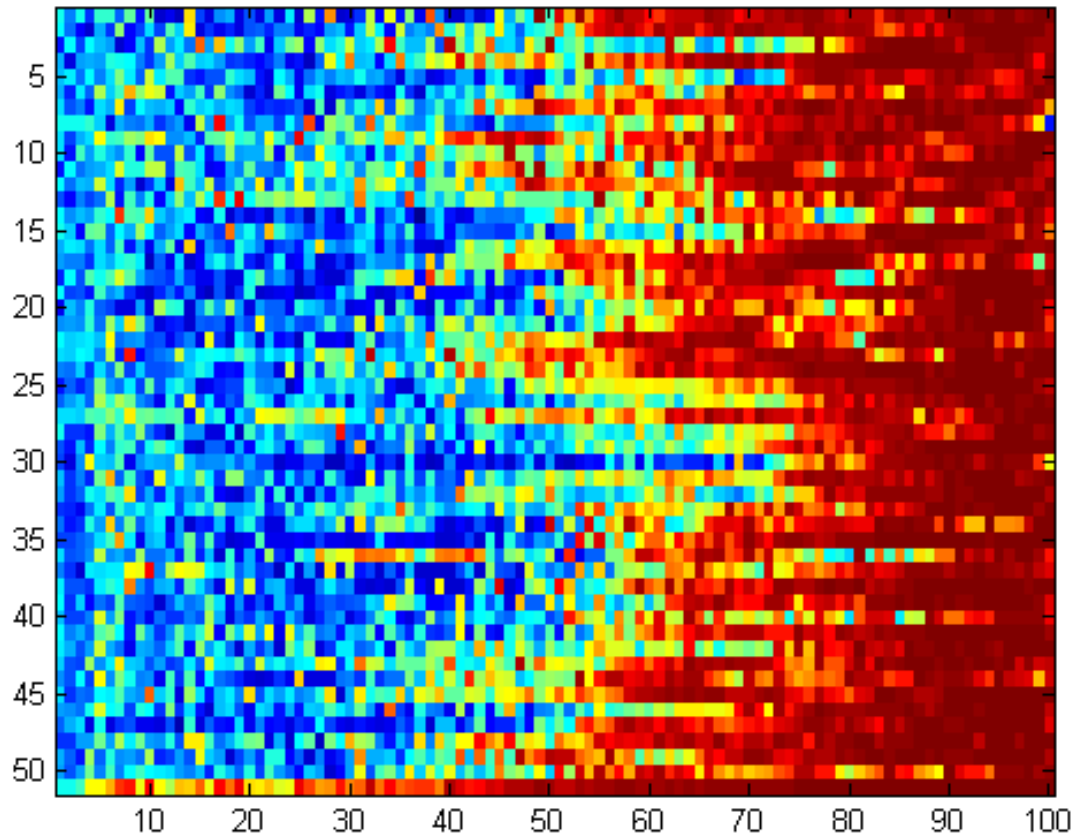


After baseline correction

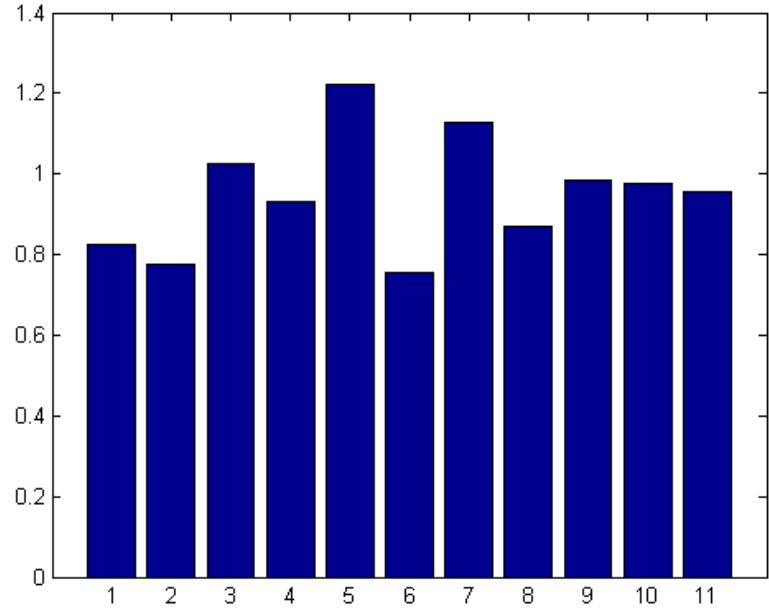
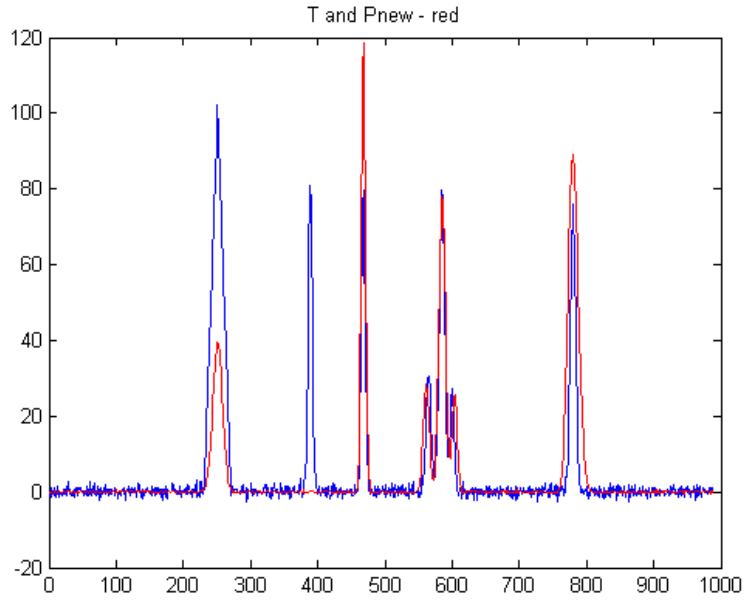


After PSO (11 parameters)

50 particles
100 iterations

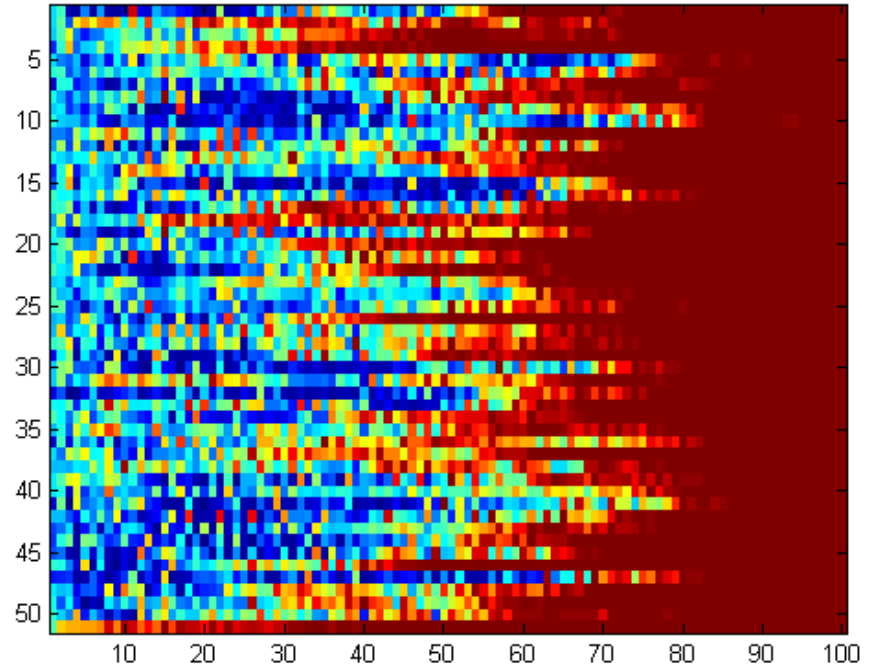
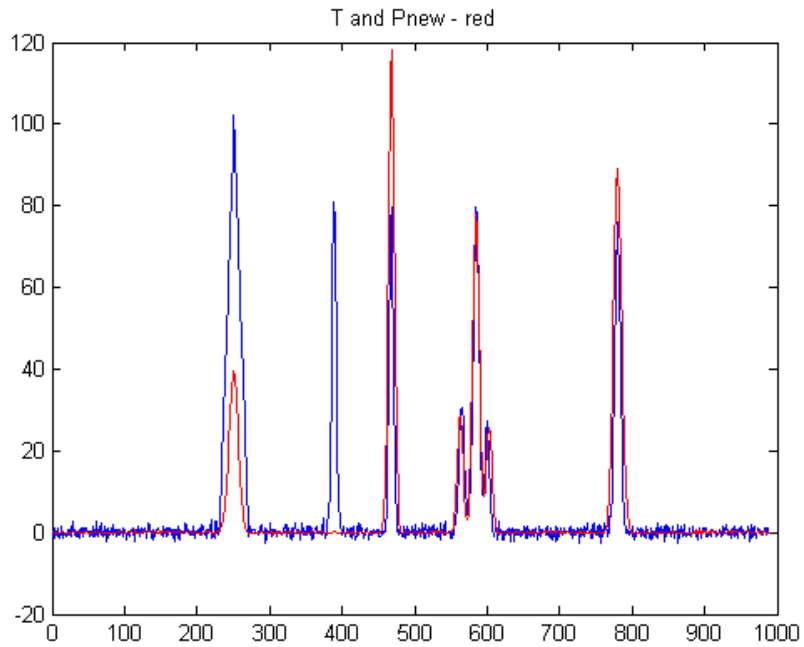


After PSO (11 parameters) - 2



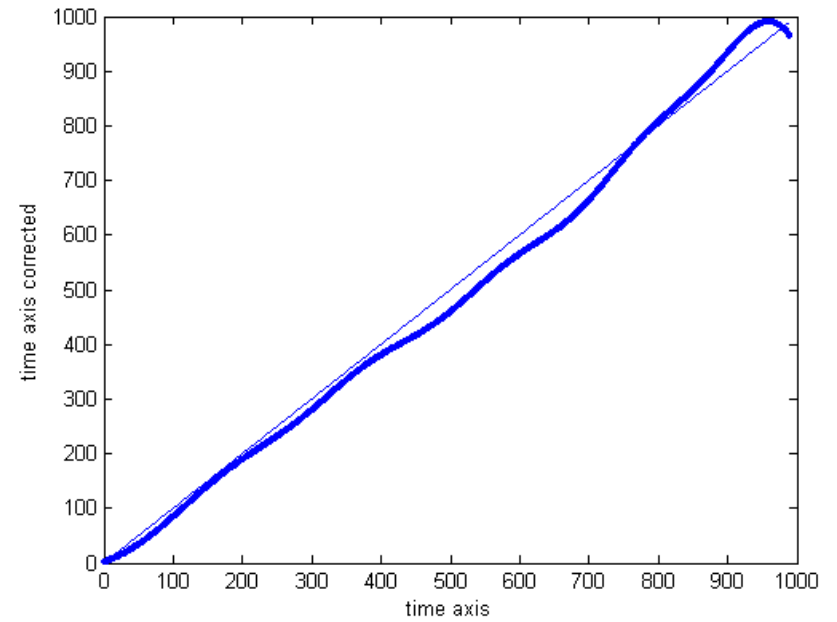
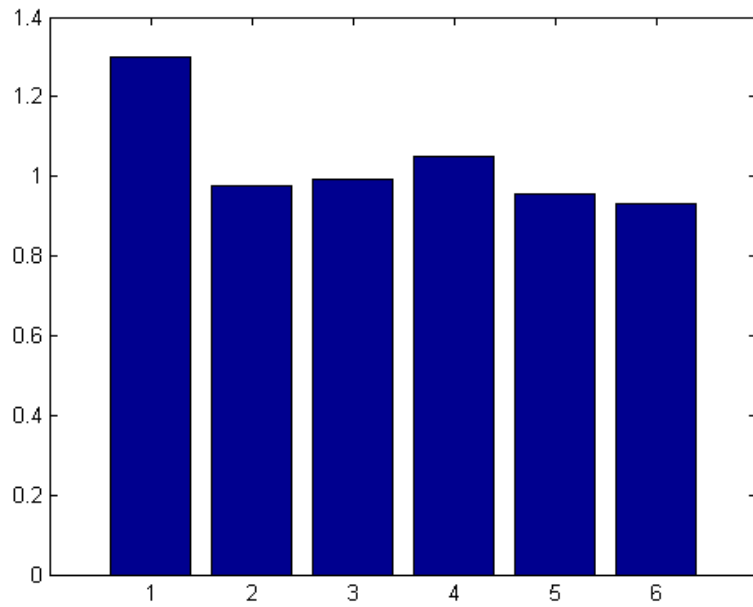
$r = 0.7453$

After PSO (6 parameters)



$$r = 0.7497$$

After PSO (6 parameters) - 2



Example II:

Finding relevant clustering directions in high dimensional data
(binary PSO)

The problem

- Variable/feature selection is often an important step in model building:
 - Enhancement of model interpretability
 - Improvement of model performances
- Approaches to variable selection
 - Univariate
 - Stepwise
 - Multivariate
- In this study variable selection for the identification of variables with high clustering tendency was investigated

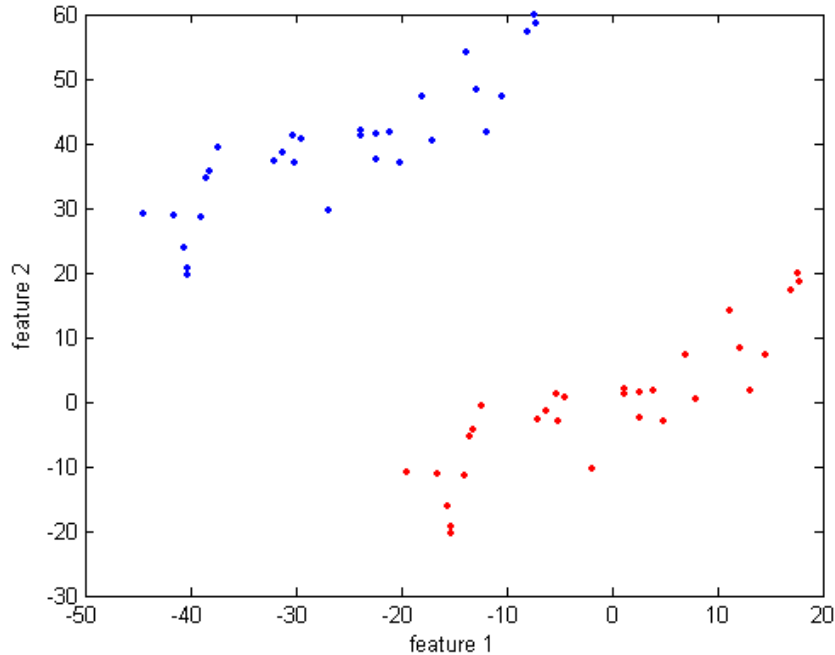
The problem - 2

- Starting inspiration from the analysis of (gen)-omic data:
 - Finding co-expressed genes to build metabolic pathways
 - Biological relevance of individual genes for clinical diagnosis
- In high dimensional space with a large number of uninformative variables, information about cluster structure and variables can be masked.
- PSO was used to identify the variables with an high clustering tendency
- In these first study a two cluster approach was followed.
- The selected variables were used to build a clustering model and the fitness was defined as:

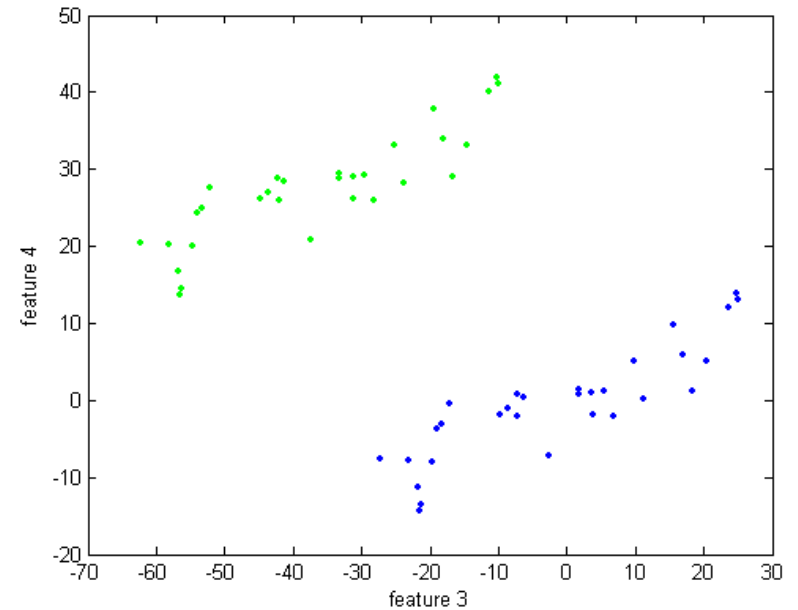
$$fitness = \frac{d_{end}}{d_{end-1}}$$

Case study 1: simulated data set

Var.1, 2



Var.3,4

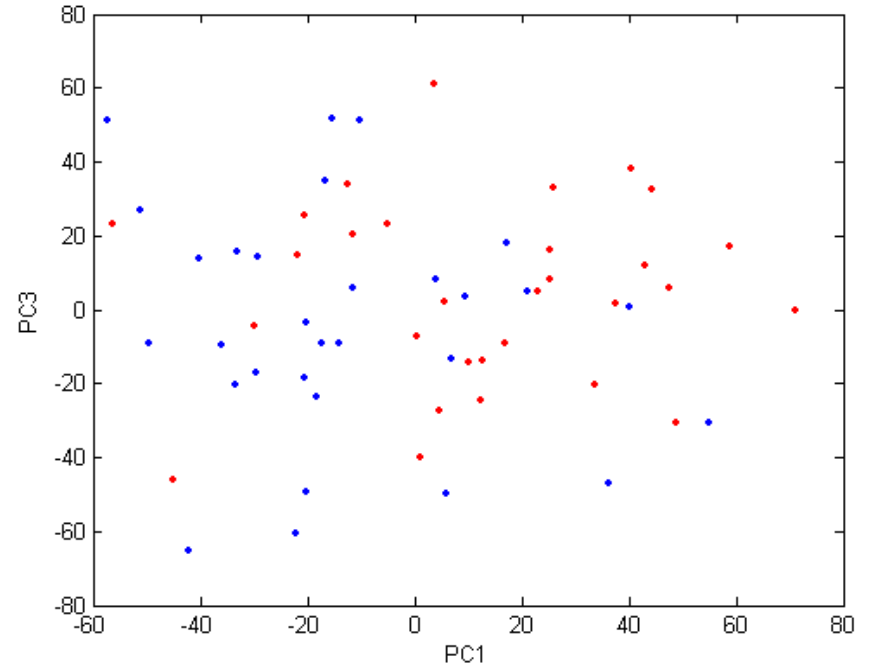
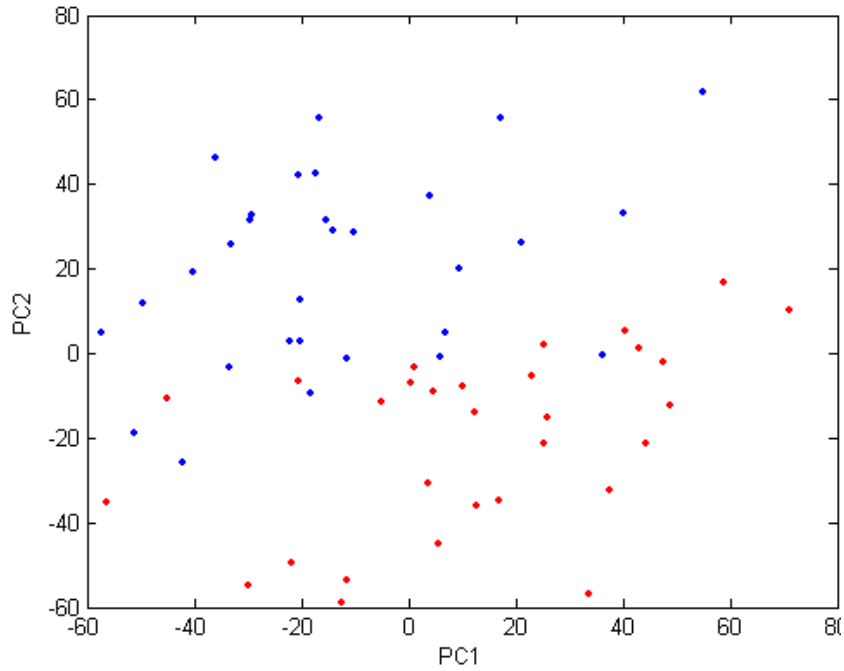


50 variables

2 different groupings (v1 & 2v; v3 & v4)

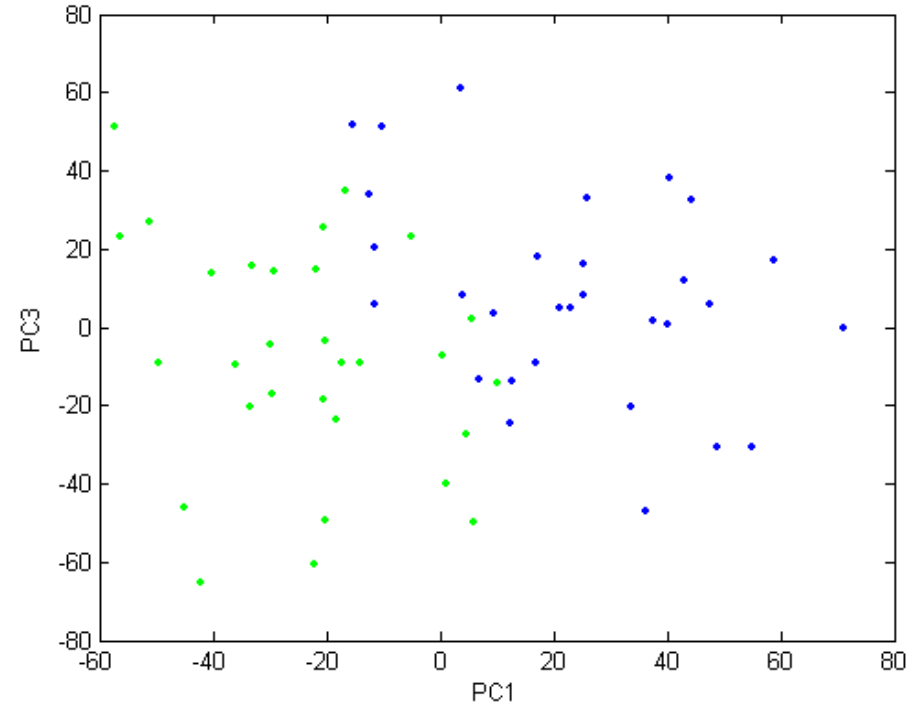
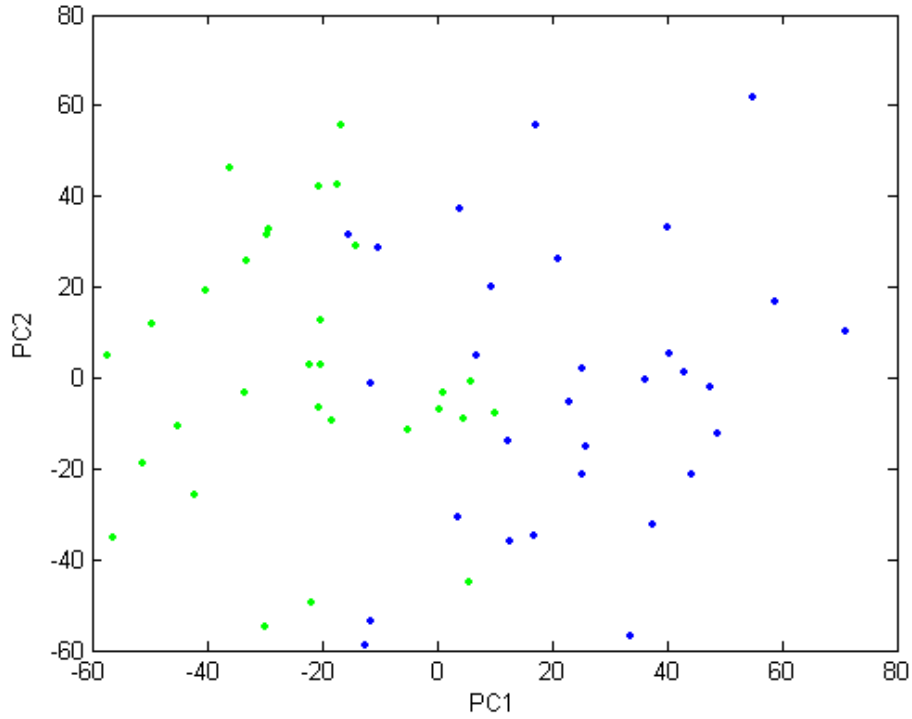
46 uninformative vars

Case study 1: simulated data set - 2



First grouping (acc. v1&v2)

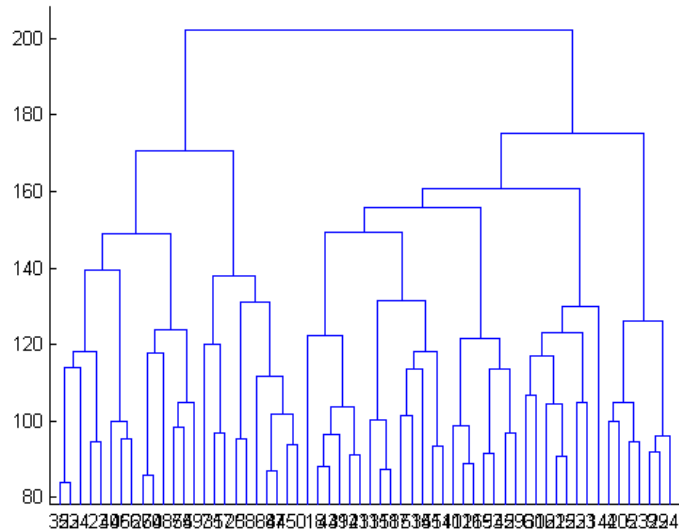
Case study 1: simulated data set - 3



Second grouping (acc. v3&v4)

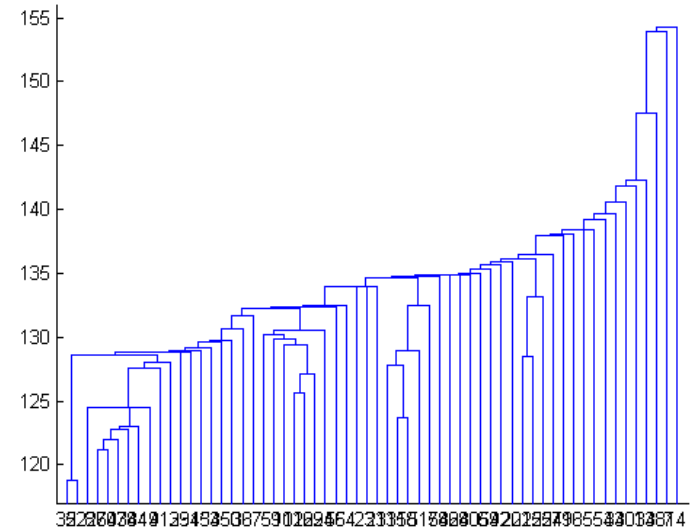
Case study 1: Clustering (all vars)

ward



Fitness = 1.1531

Single linkage

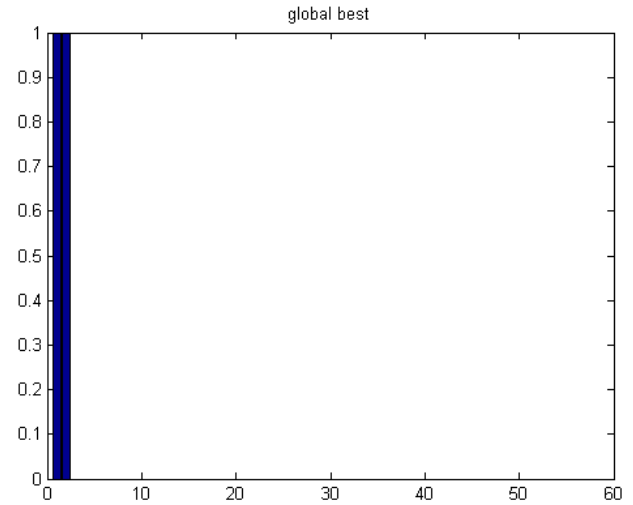
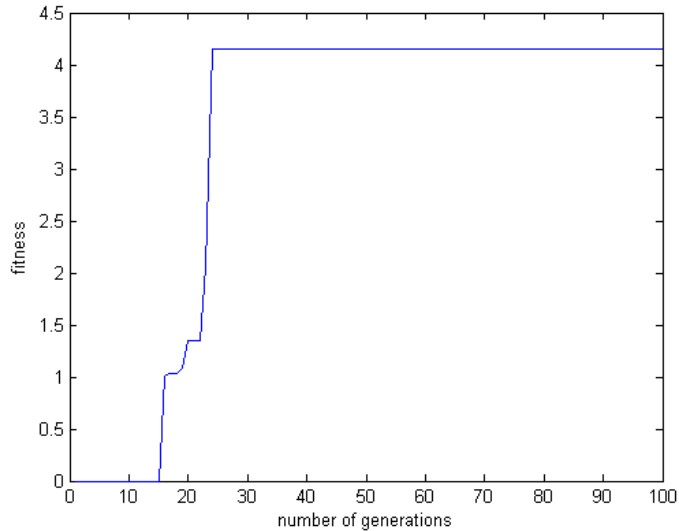


Fitness = 0

If one of the two last clusters had $<k$ (usually 5) objects fitness was set =0

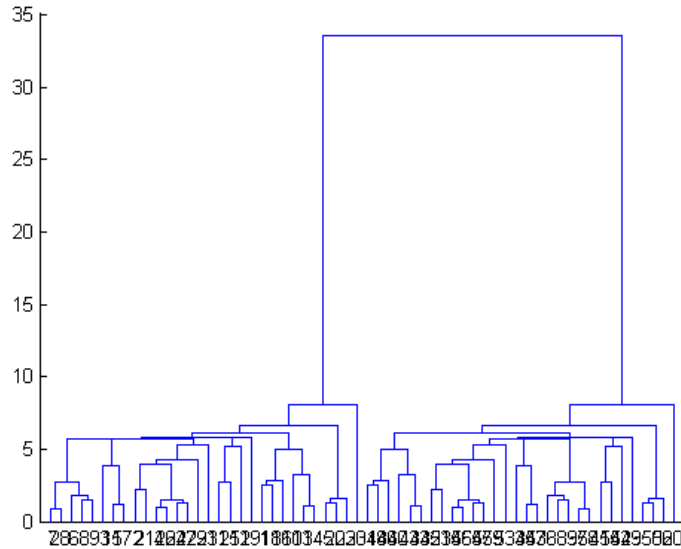
Case study 1: PSO single linkage

- 50 particles, 100 generations



Variables 1 & 2 were selected in the fittest model

Case study 1: PSO single linkage

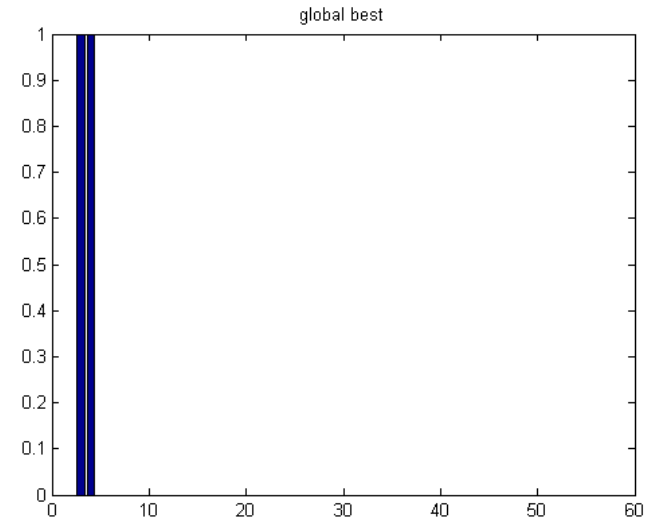
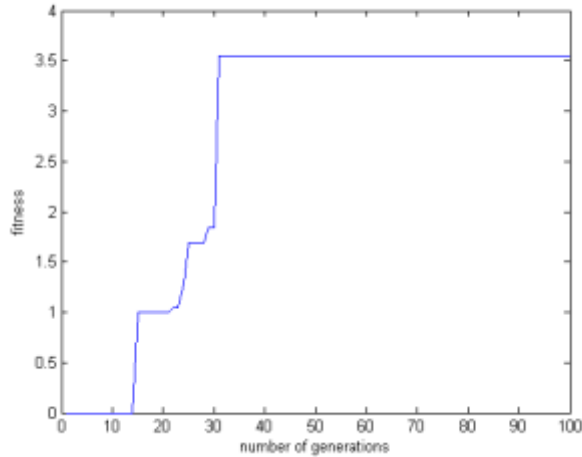


Fitness = 4.1461

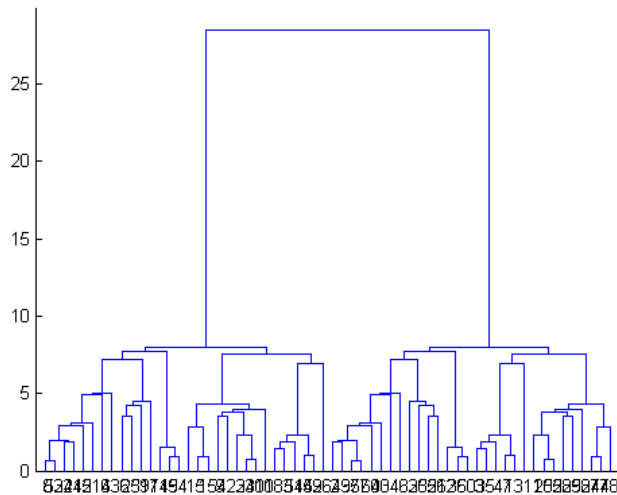
Found clusters corresponded to the simulated ones

Case study 1: PSO single linkage

If PSO is applied again to the remaining 48 variables, variables 3 & 4 are selected

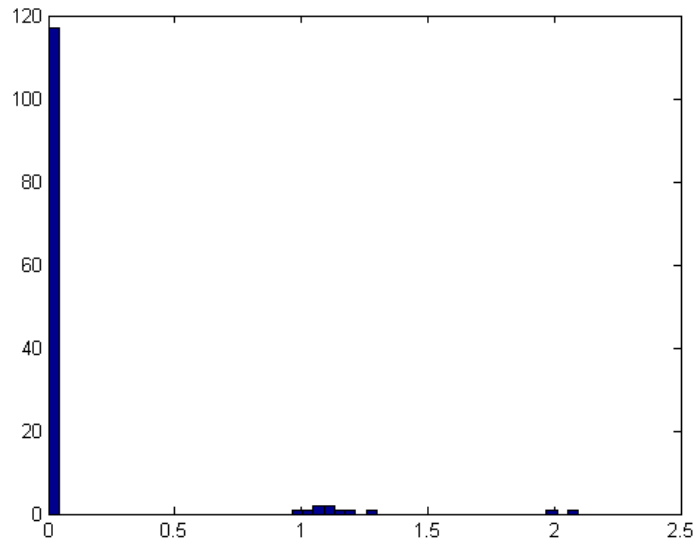


Fitness = 3.5500

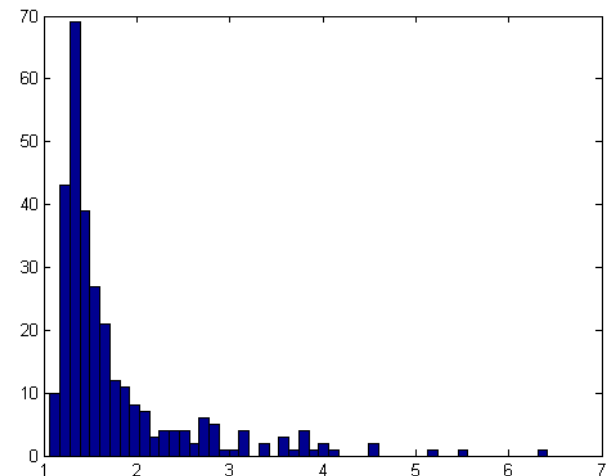


PSO: significance of clustering

- In order to estimate whether the cluster structure could be significant, a null distribution was simulated.
- Repetitions (Matrices of random number having the same mean and variance of the original variables)

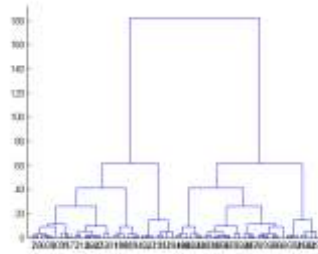
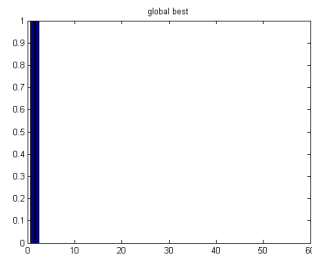
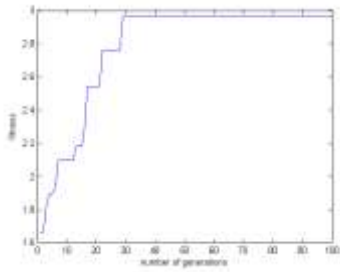


Min 6 objects per cluster



Min 2 objects per cluster

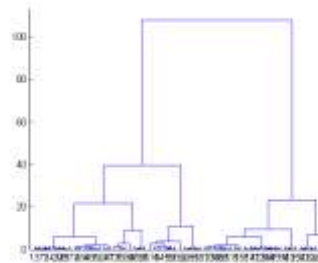
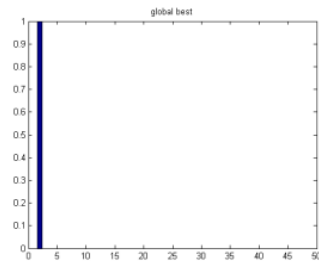
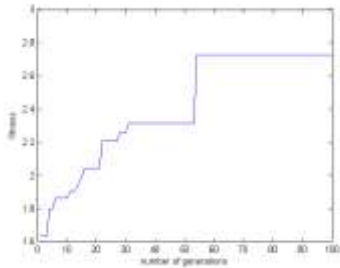
Case study 1: PSO Ward



Fitness = 2.9658

Var.1 ,2

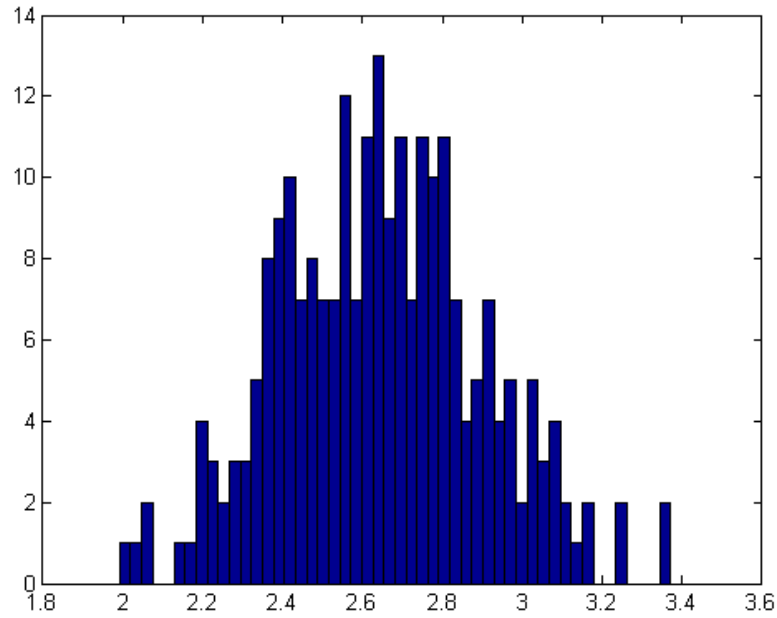
Globalbest==0



Fitness = 2.7275

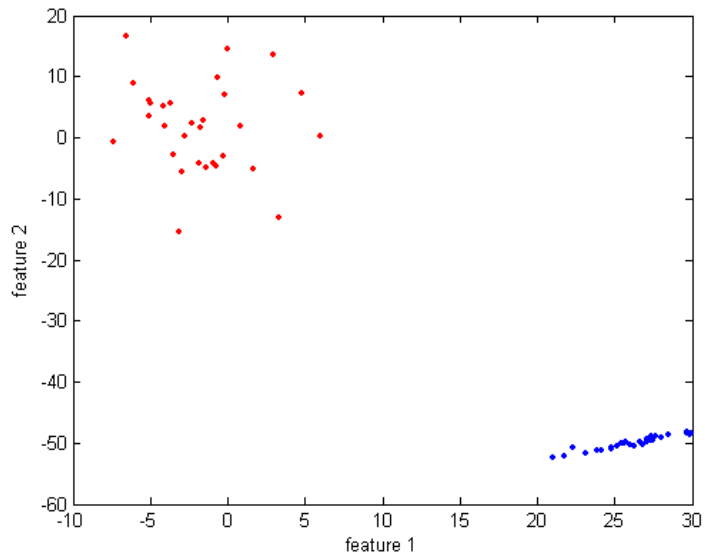
var4

Case study 1: PSO Ward threshold

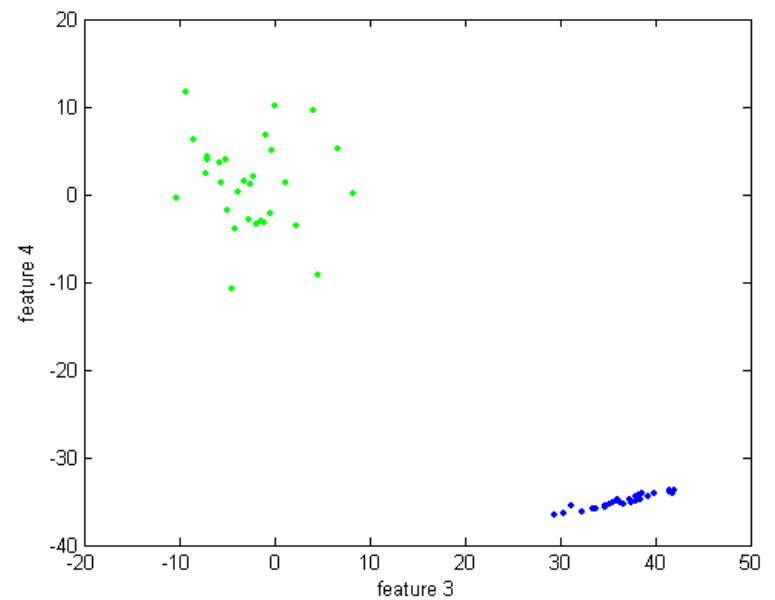


Case study 2: simulated data set

Var.1, 2



Var.3,4

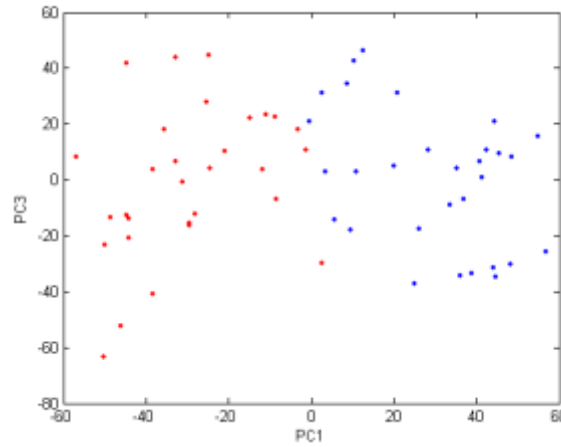
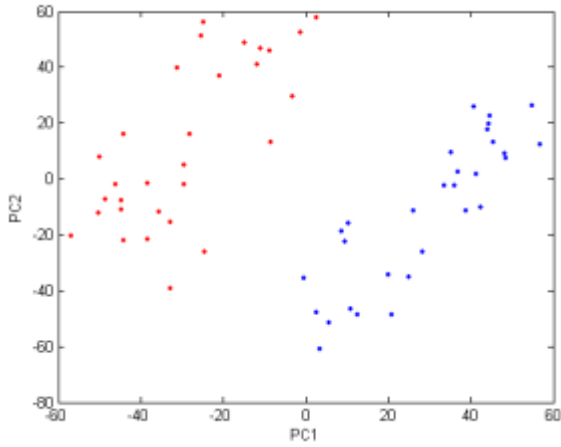


50 variables

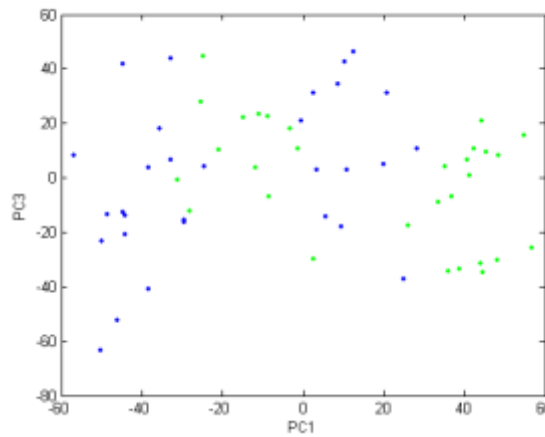
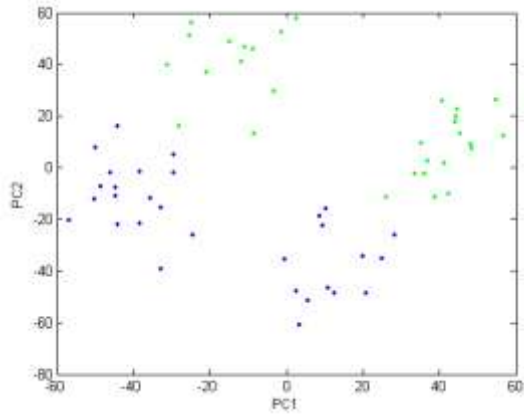
2 different groupings (v1 & v2; v3 & v4)

46 uninformative vars

Case study 2: simulated data set - 2

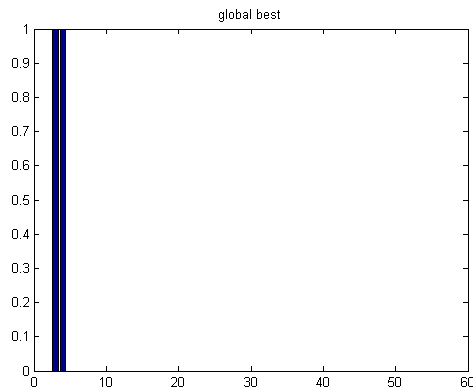


First grouping
(v1&v2)

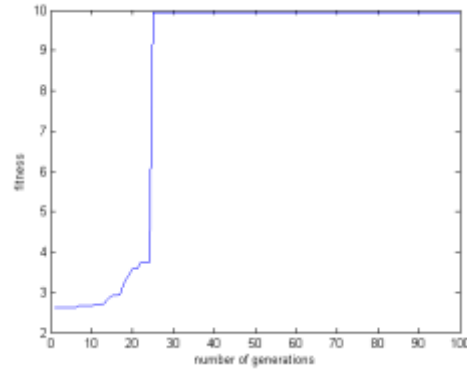


Second grouping
(v3&v4)

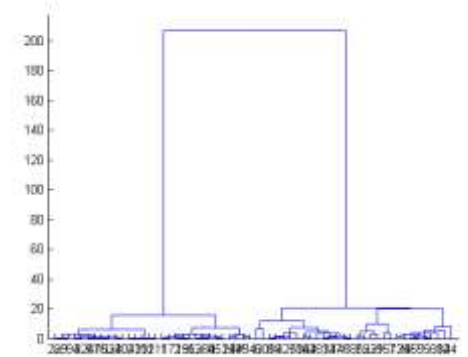
Case study 2: PSO ward



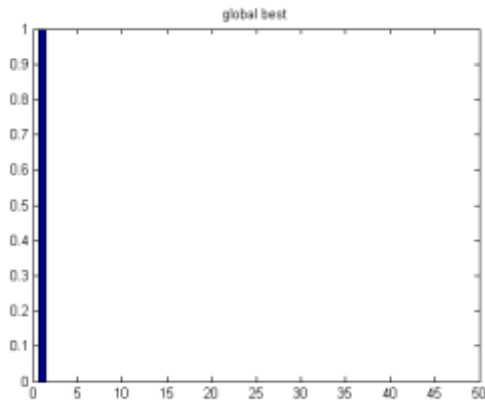
Var. 3, 4



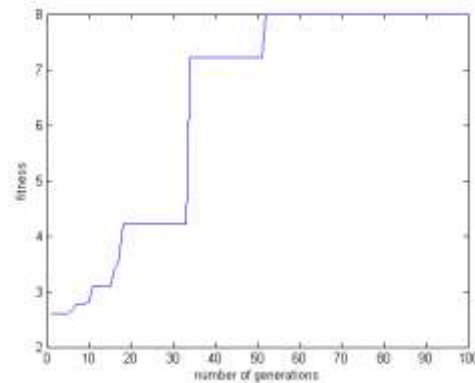
fitness 9.9315



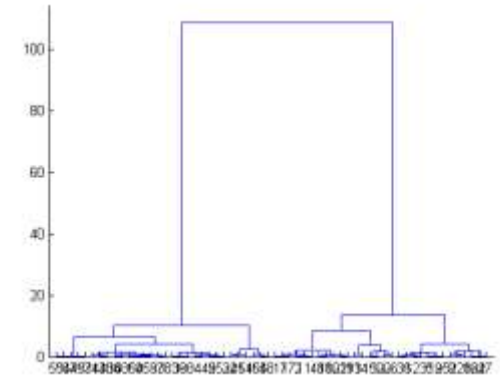
And when performing PSO on the remaining variables



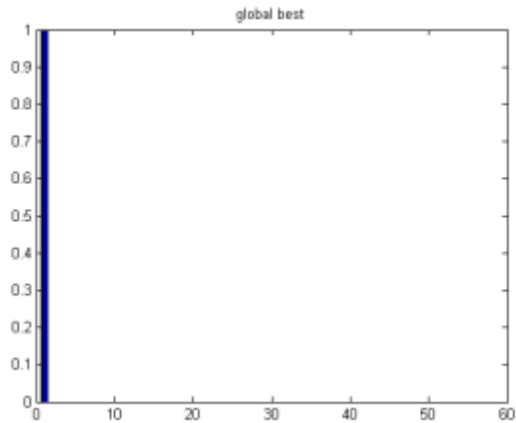
Var. 1,2



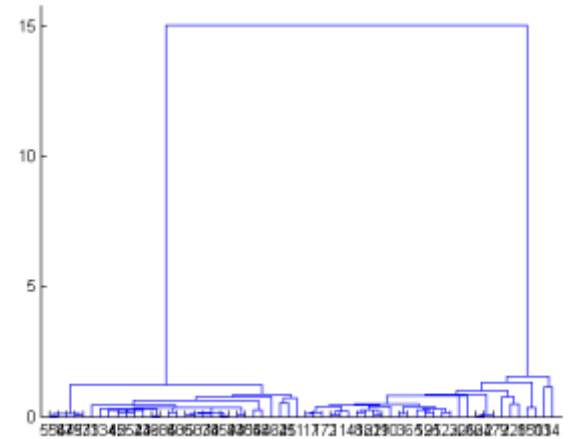
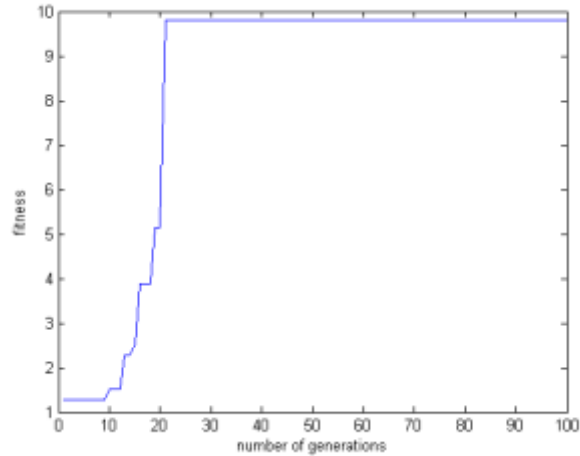
fitness 7.9840



Case study 2: PSO single linkage

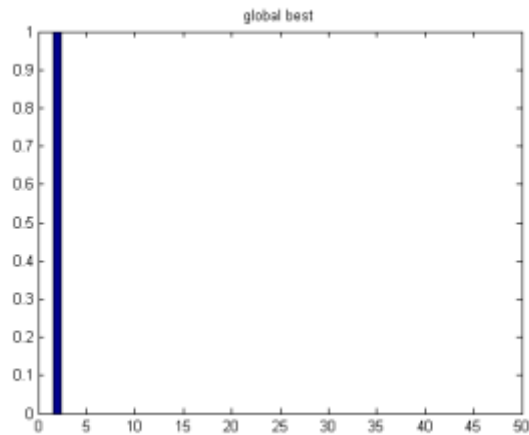


Var. 1

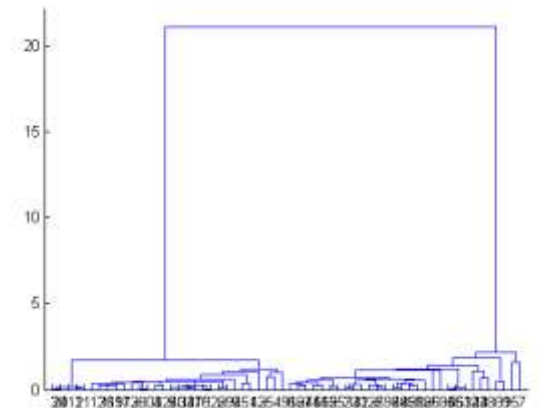
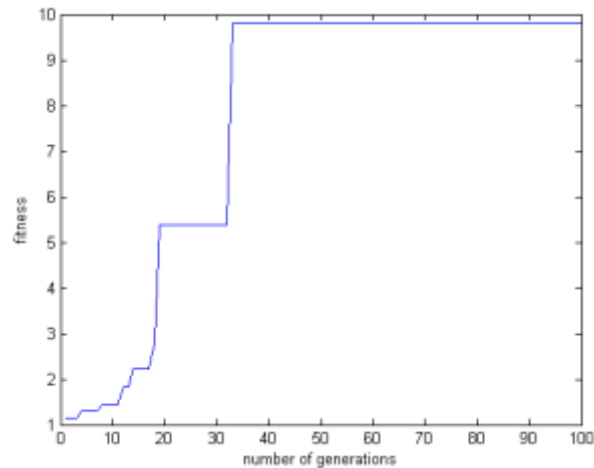


9.8037

And performing PSO on the remaining 49 variables:



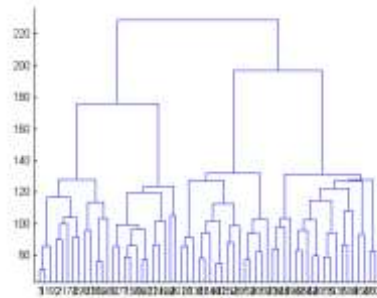
Var. 3



9.8037

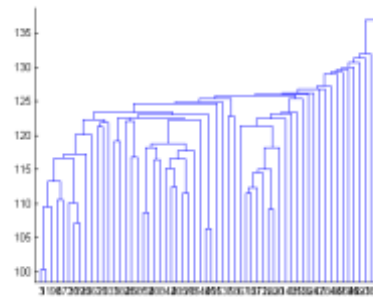
Case study 2: comparison of results

Ward



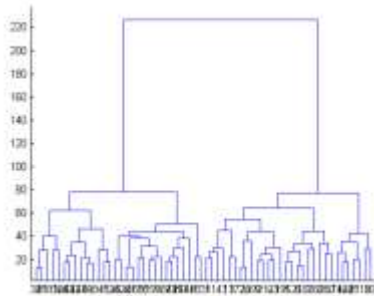
1.1635

Single linkage

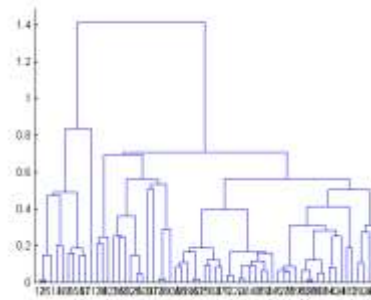


0

All variables

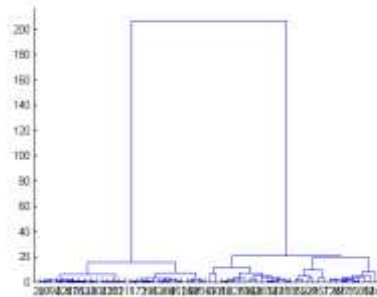


2.9106; th=3.4

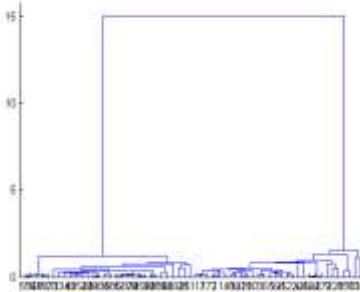


1.7024

Swarm(PCs)



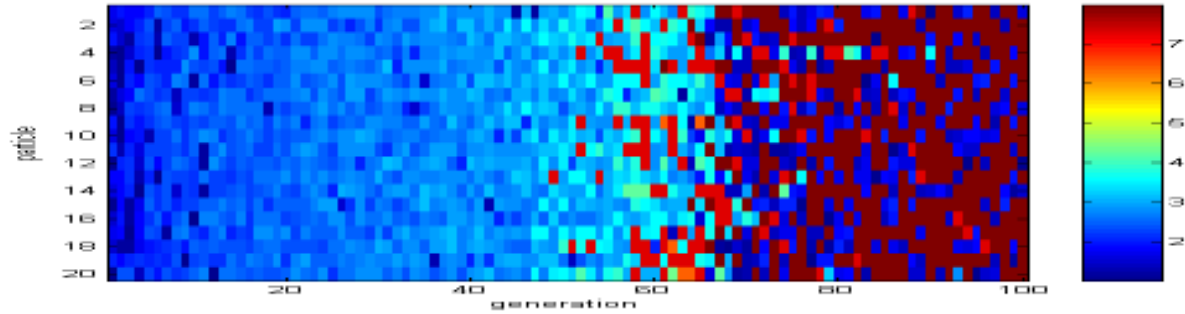
9.9315; th=3.2



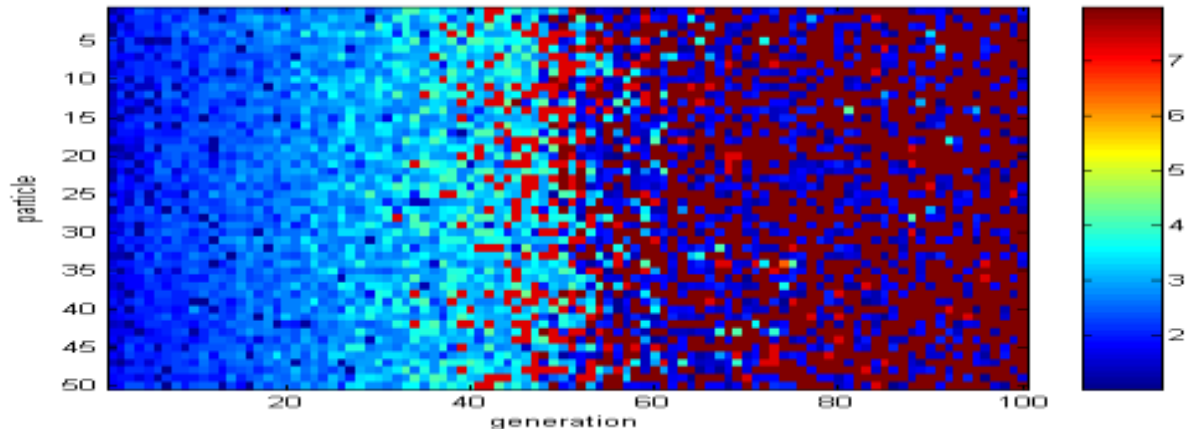
9.8037

Swarm(x)

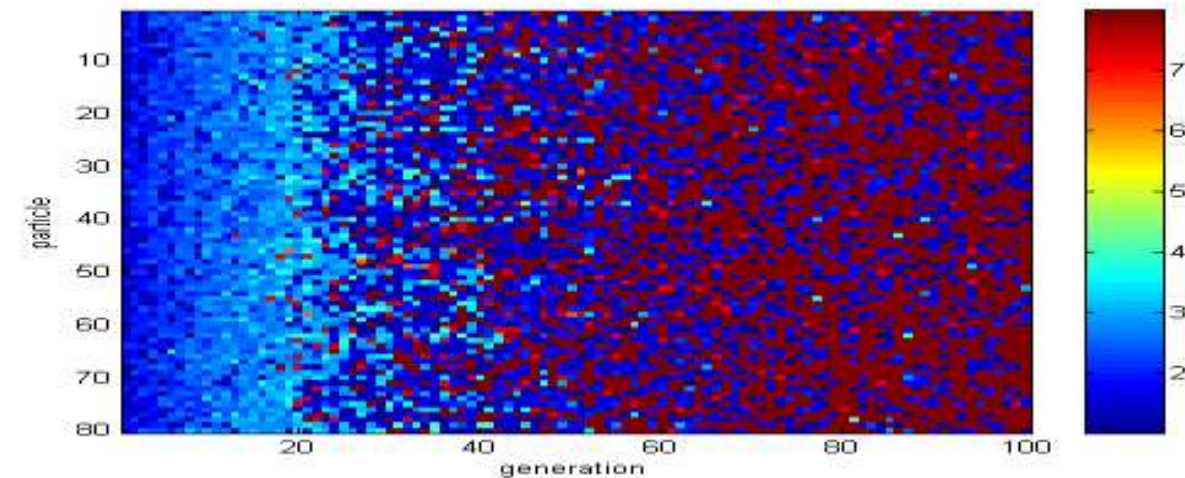
Effect of the number of particles



20 particles

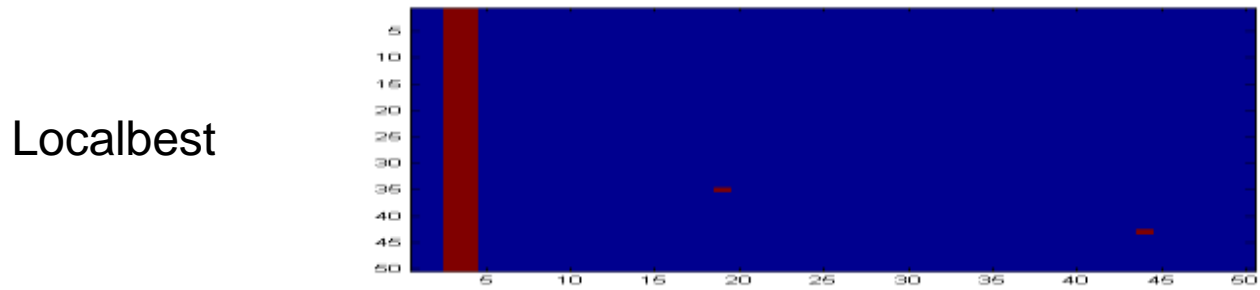
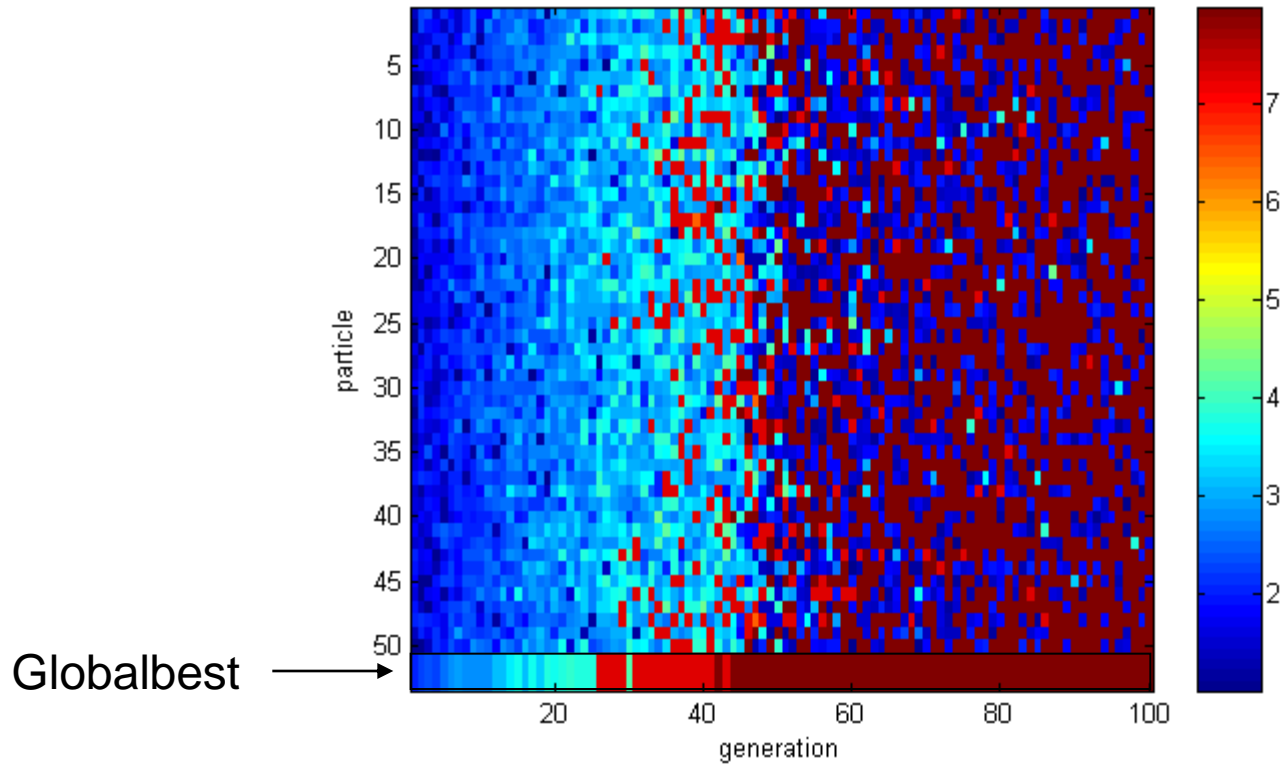


50 particles

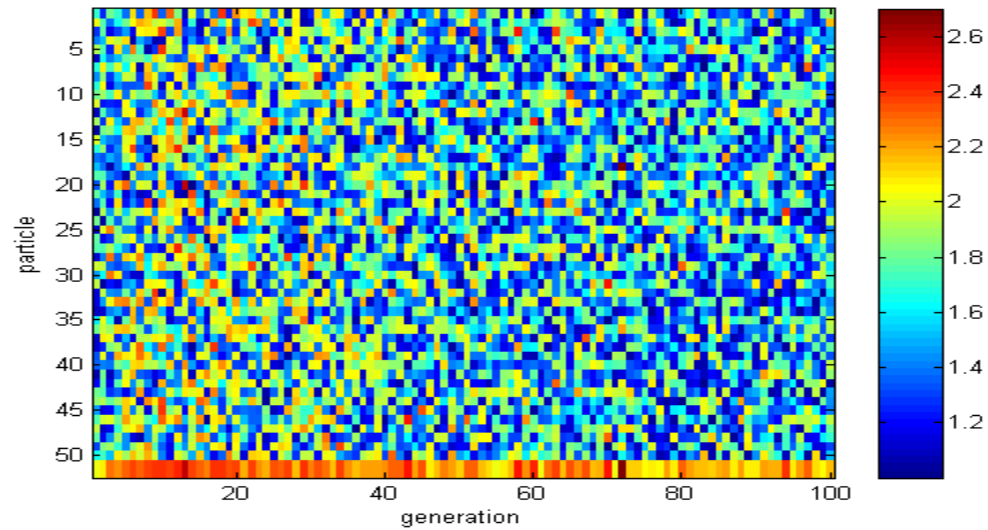
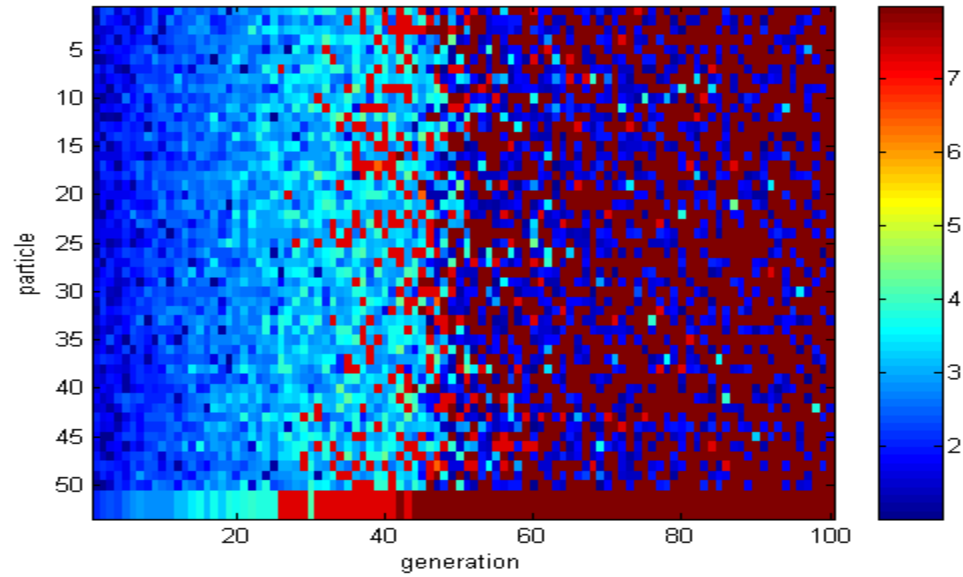


80 particles

Consistency of the results

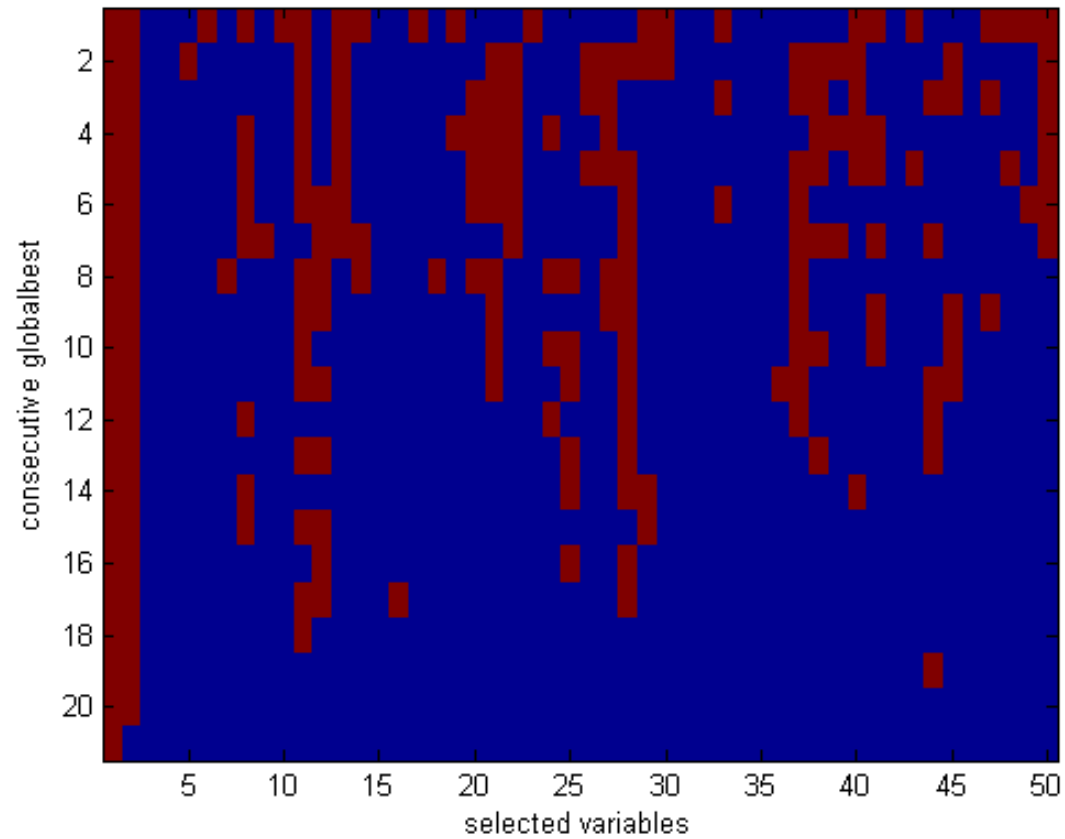


Effect of inertia



With inertia

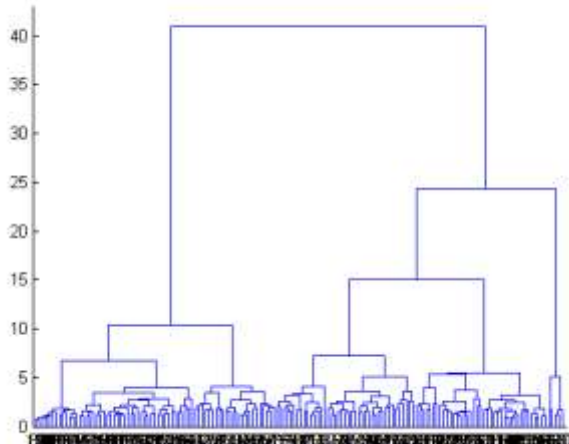
Training evolution



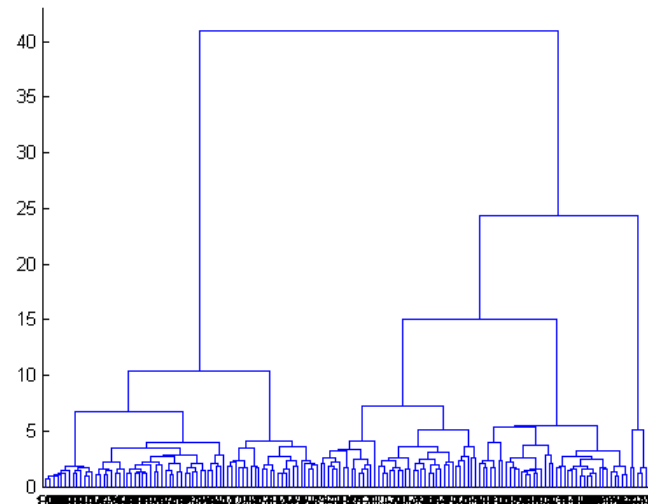
Case study 3 – microarray data

- Gene expression from 156 samples
- Original data 156x22413 reduced to 156x144 using SOMs
- Diseased and healthy cells

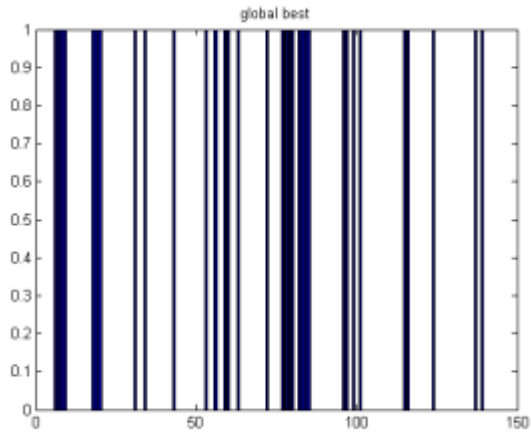
All variables
fit= 1.6885



144 vars
fit = 1.6885

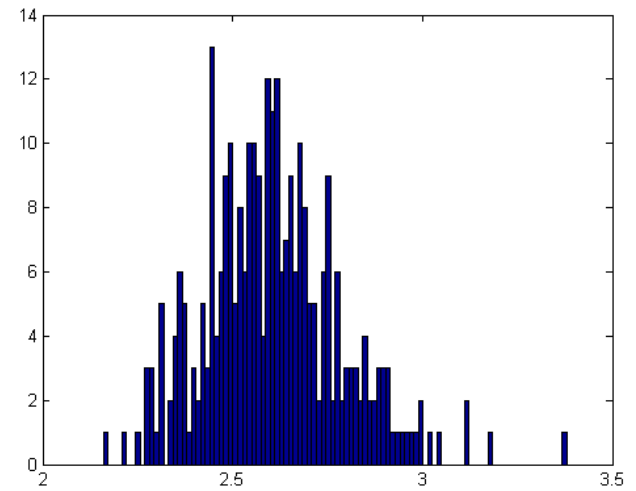
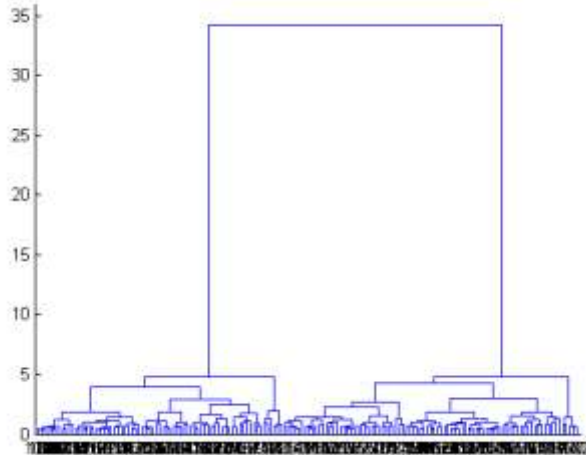
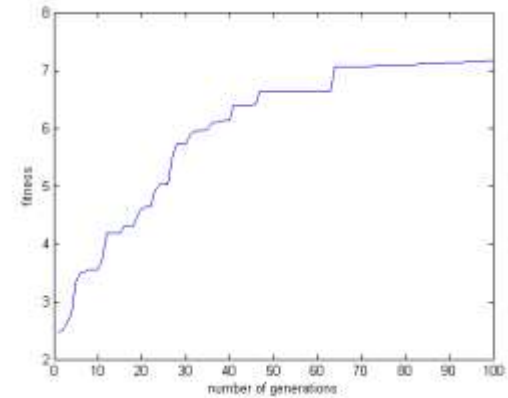


Case study 3 – microarray data

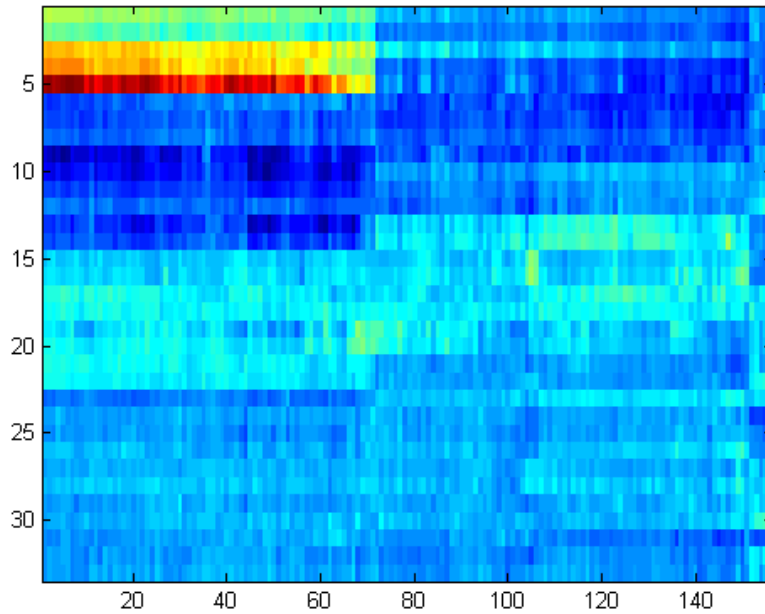
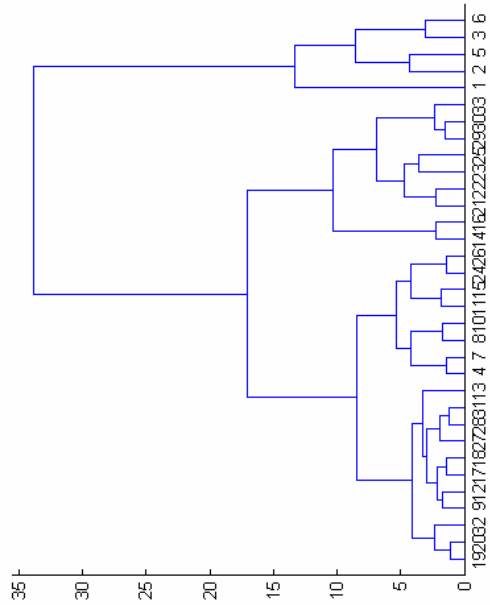
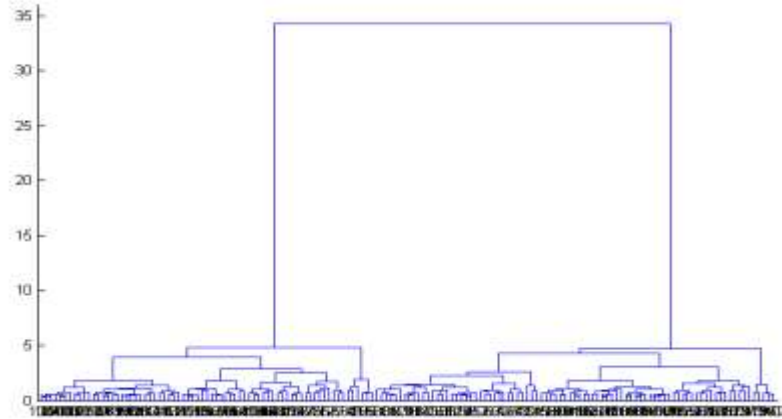


50 particles; 100 gen

7.1693



Case study 3 – microarray data



Example III:
Real-valued PSO for the calculation of
Robust PCA

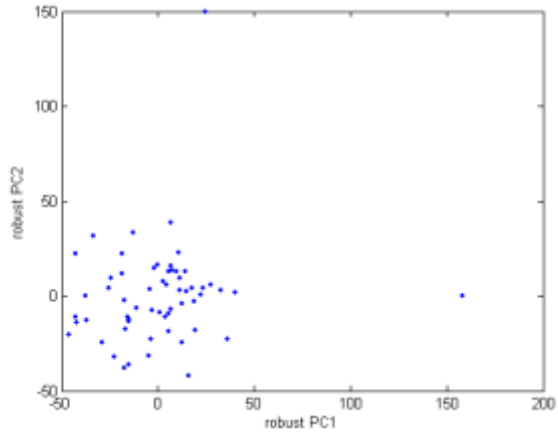
Robust PCA

- Different algorithms have been proposed for the computation of Robust PCA models
- Here Projection pursuit + PSO
- Robust scale Q_n is used as a fitness criterion
- Component are extracted sequentially after deflation.
- 60 particles were used

Robust PCA

PP

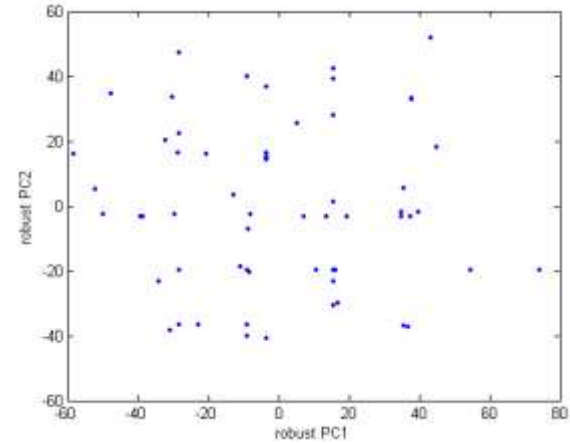
qn=19.3130



qn=22.3255

swarmPP

qn=34.7391



qn=40.1268

Huber's Robust PCA

T1 qn=36.0317

T2 qn=35.0934

Conclusions

- PSO has proved to give interesting results on many different examples
- Works well both on real-valued and binary-coded problems
- Relatively simple and stable

СПАСИБО!

