A bestial optimization!

[Un'ottimizzazione bestiale!]

Marco CORAZZA
(corazza@unive.it)
Department of Economics, Ca' Foscari University of Venice

INCONTRI MATHESIS
Venice, Italy – March 15, 2019

Outline

- Introduction
- Heuristics and Metaheuristics
- Particle Swarm Optimization (PSO)
- Conclusion

Introduction

- **Economics** study how a given community manages the **scarce resources** available to it.

- So, the use of such resources must be **optimized**.

Introduction

- Typical microeconomic problem: **Consumer utility maximization**:

$$\min_{x_1,\dots,x_N} u(x_1, \dots, x_N)$$
$$\text{s. t.} \begin{cases} x_1 p_1 + \cdots + x_N p_N = w \\ x_1, \dots, x_N \geq 0 \end{cases}$$

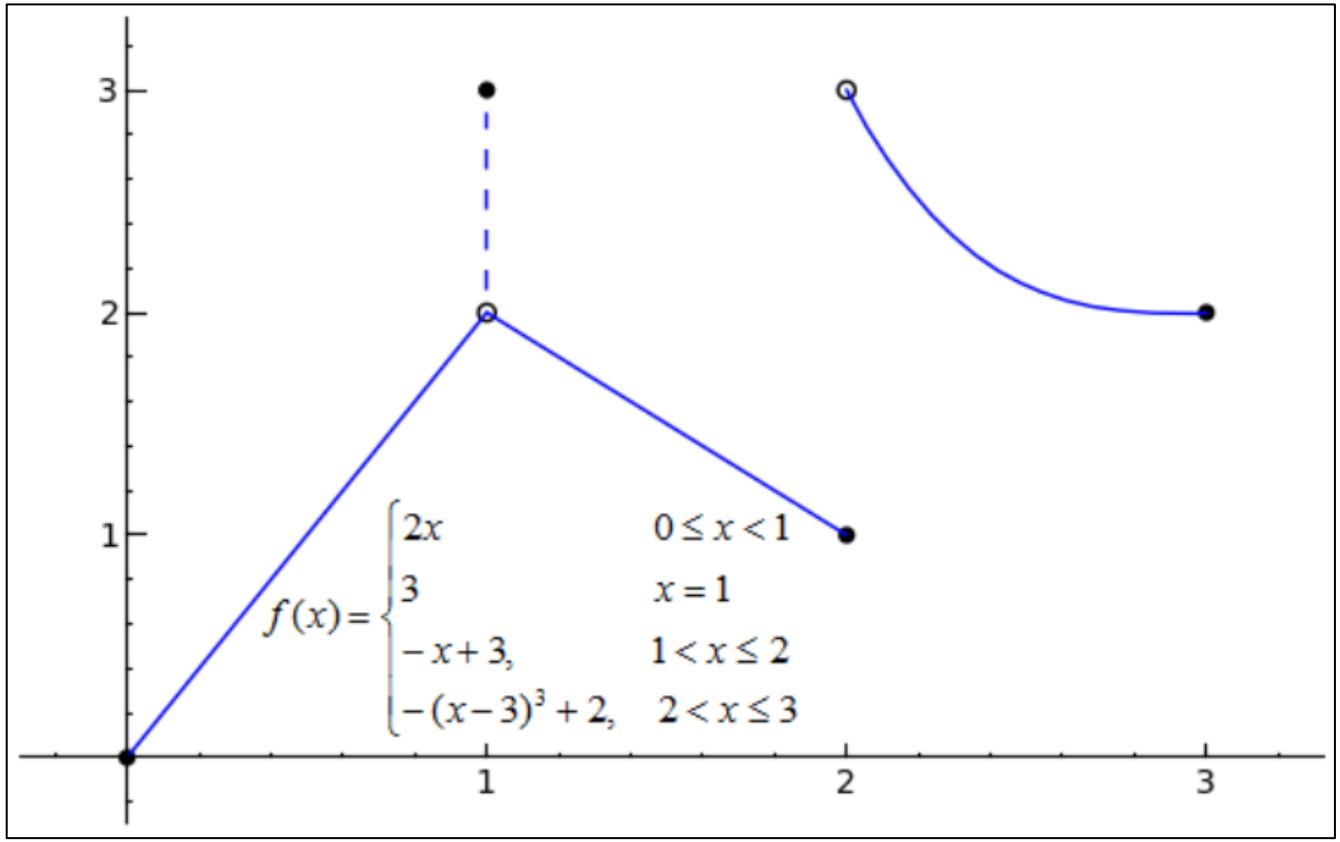
where
x_i is the quantity to buy of the i-th good/service (decisional variable);
p_i is the price of the i-th good/service (parameter);
w is the wage (parameter).

- Generally, such a constrained optimization problem is **easy to solve**.

Introduction

Introduction

- Global/Relative points of minimum/maximum **difficult to find**:



Introduction

How to solve complex optimization problems?

- Generally, the solution procedures we need for solving **complex optimization problem**s strongly depends on the peculiar features of the problem itself.

- So, **ad hoc** solution procedures have to be developed for any considered problem.

- Furthermore, it is strongly believed that they cannot be solved to optimality within **polynomially** bounded computation time.

Introduction

How to solve complex optimization problems?

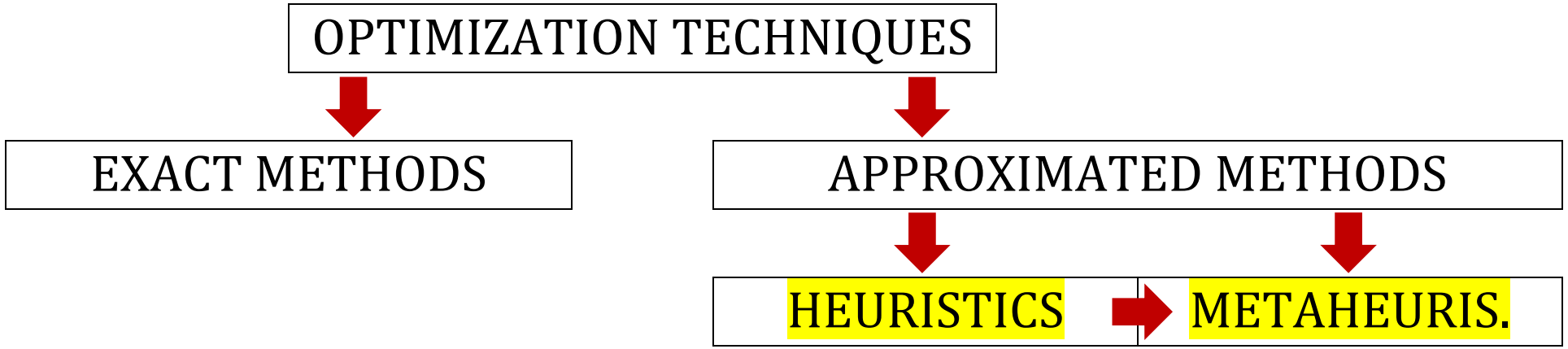
- When optimal solutions cannot be efficiently obtained, the only possibility is **to trade optimality for efficiency**.

- **Approximate algorithms**, also called **heuristics** or **metaheuristics**, seek to obtain **near-optimal solutions** at relatively low **computational cost**.

Introduction

How to solve complex optimization problems?

- A **metaheuristic** is a general algorithmic framework which can be applied to **different optimization problems** with relatively few modifications.



Heuristics and Metaheuristics

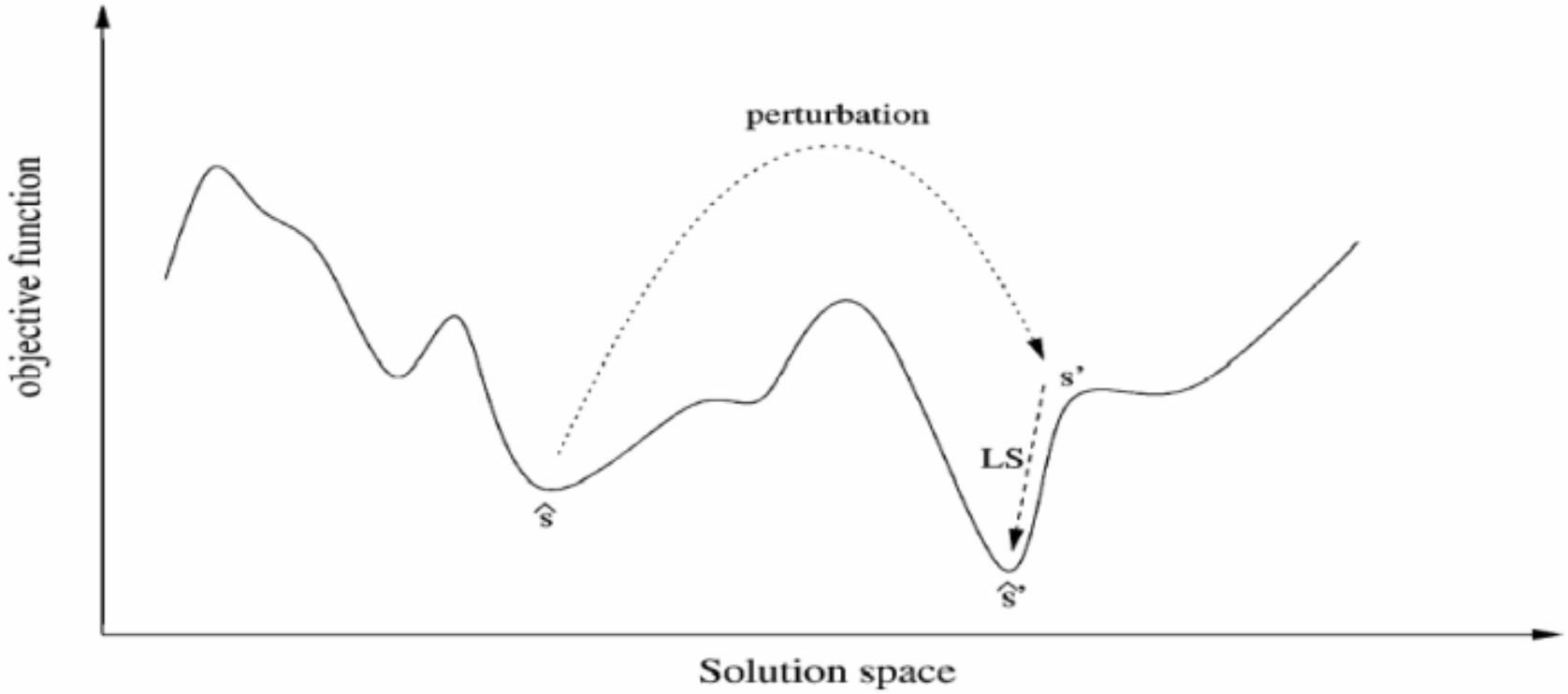
Heuristics

- **Heuristics** are **simple** and **efficient rules** giving a **good solution** to **tough optimization problems** in a **reasonable time**.

- There is no **guarantee** that optimal solutions are reached. This is good when we do not necessarily want the best solutions but rather **good solutions**.

Heuristics and Metaheuristics

Heuristics



Local search method

Heuristics and Metaheuristics

Heuristics

- **Disadvantage**s of heuristics are:
  - Usually, they are problem-specif.
  - Sometimes, they generate a limited number of different solutions;
  - Sometimes, they stop at poor quality local optima.

- **Metaheuristic**s have been proposed since 1980 to bypass these problems.

Heuristics and Metaheuristics

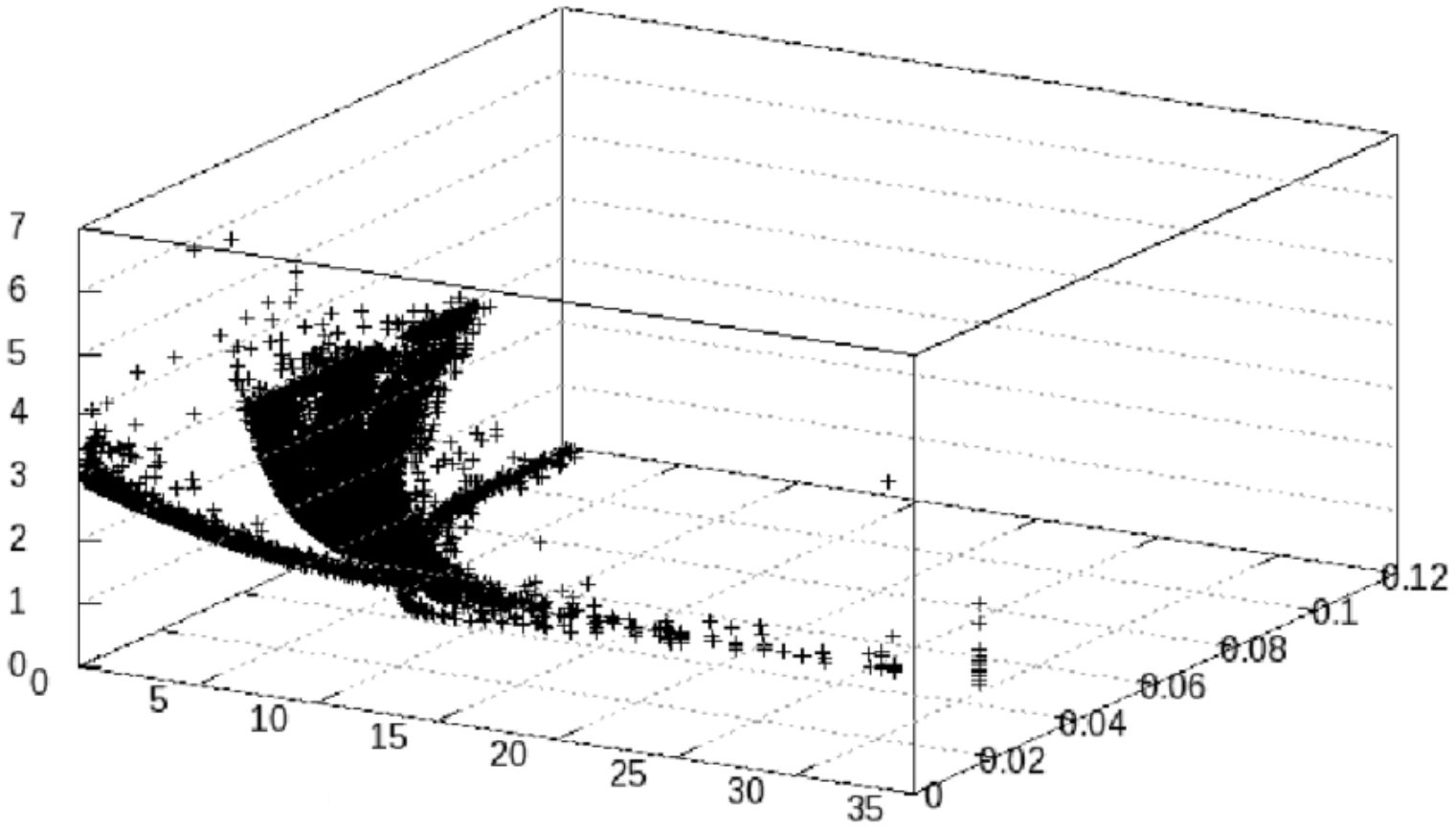
Metaheuristics

- **Metaheuristic**s are development of heuristics. They are **high-level problem-independent algorithmic framework**s that provides a set of guidelines or strategies to develop heuristic optimization algorithms.

- There are different ways to **classify** metaheuristics. The two main components that determine their behavior are:
  - (local) **exploration**;
  - (global) **exploitation**.

Heuristics and Metaheuristics

Metaheuristics



Heuristics and Metaheuristics

Metaheuristics

- **Exploration** means to focus the search in a **local region** of the search space knowing that a current good solution is found in this region.

- **Exploitation** means to generate **different solution**s so as to explore the search space on a **global scale**.

Heuristics and Metaheuristics

Metaheuristics

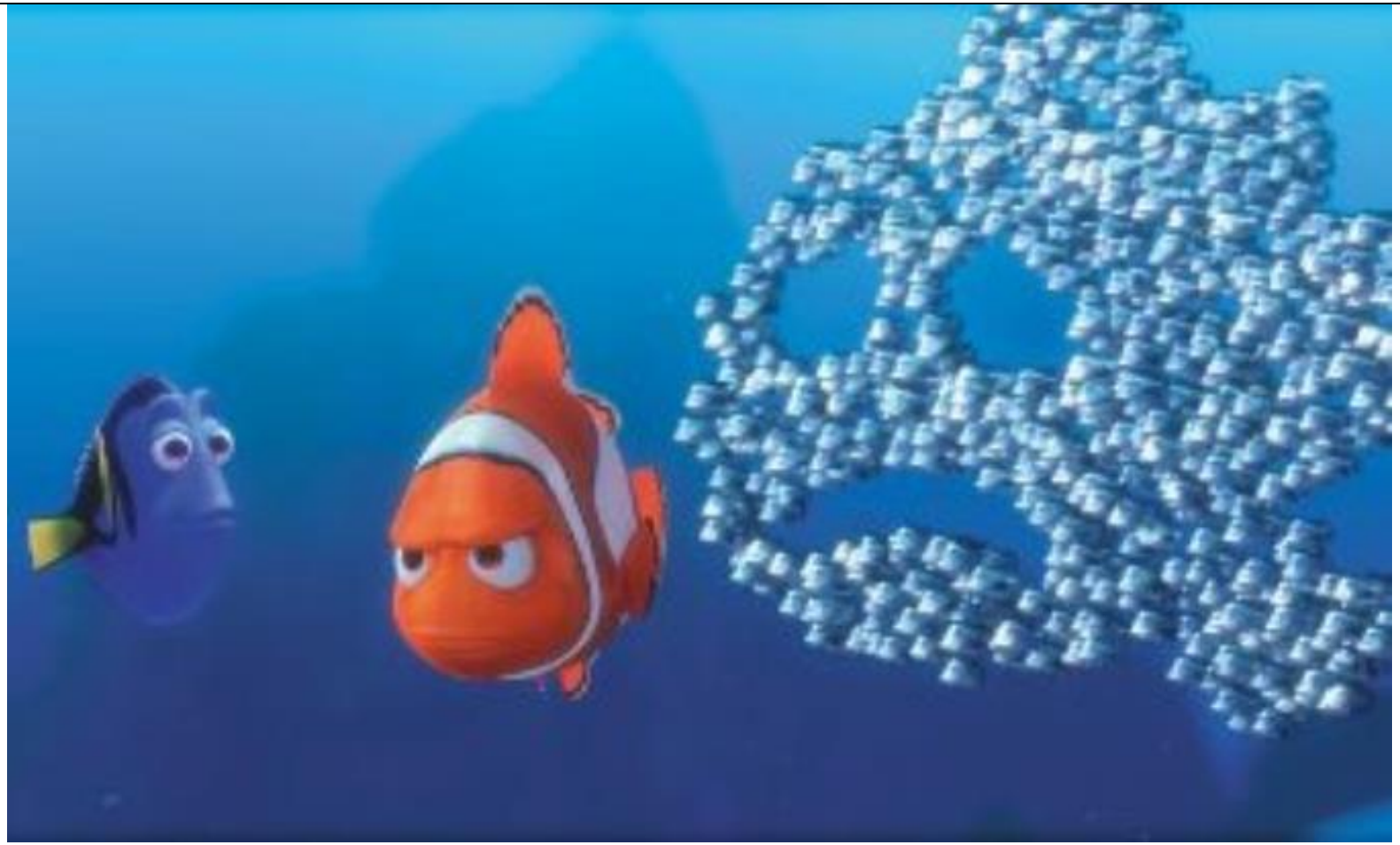
- Another way to classify metaheuristics is:
  - **Trajectory-based metaheuristics**: techniques that start with a **single solution** and, at each step of the search, **replace** the current solution by another one, often **best**;
  - **Population-based metaheuristics**: techniques that make use of a **population of solutions**. The initial population is randomly generated, and then enhanced through an **iterative process**. At each generation of the process, the whole population, or a part of it, is **replaced** by newly generated individuals, often the best ones.

Particle Swarm Optimization (PSO)

Flocks of bird

Shoal of fishes

Particle Swarm Optimization (PSO)



Finding Nemo – The Walt Disney Co.

Particle Swarm Optimization (PSO)

- **PSO** is an intelligent bio-inspired iterative metaheuristics for the **solution** of **global optimization problems**.

- The idea of PSO is to replicate the behavior of **flocks of birds**/**shoals of fishes** when they **cooperate** in order to **optimize the search for food**.

- On this purpose **every member**, i.e. **particle** of the swarm, explores the search area keeping memory of its best position reached so far. This information is then **exchanged with the neighbors** in the swarm.

Particle Swarm Optimization (PSO)

- In its mathematical counterpart the paradigm of a flying flock may be formulated as follows.

- Given a **minimization/maximization problem**.
  - Every member of the swarm represents a possible solution of the minimization/maximization problem.
  - Every particle is **initially positioned randomly** in the feasible set of the problem.
  - Every particle is also **initially assigned** a **random velocity**, which is used to determine its initial direction of movement.

Particle Swarm Optimization (PSO)

- For a formal description of the PSO algorithm, let us consider the following global optimization problem:

$$\min_{\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{d}}} f(\boldsymbol{x})$$

where
$\boldsymbol{f}: \mathbb{R}^{\boldsymbol{d}} \rightarrow \mathbb{R}$.

Particle Swarm Optimization (PSO)

- Let us assume that M **particle**s are considered.

$$\left(x_{1,1}, x_{2,1}, \dots, x_{d,1}\right)$$
$$\left(x_{1,2}, x_{2,2}, \dots, x_{d,2}\right)$$
$$\dots$$
$$\left(x_{1,M}, x_{2,M}, \dots, x_{d,M}\right)$$

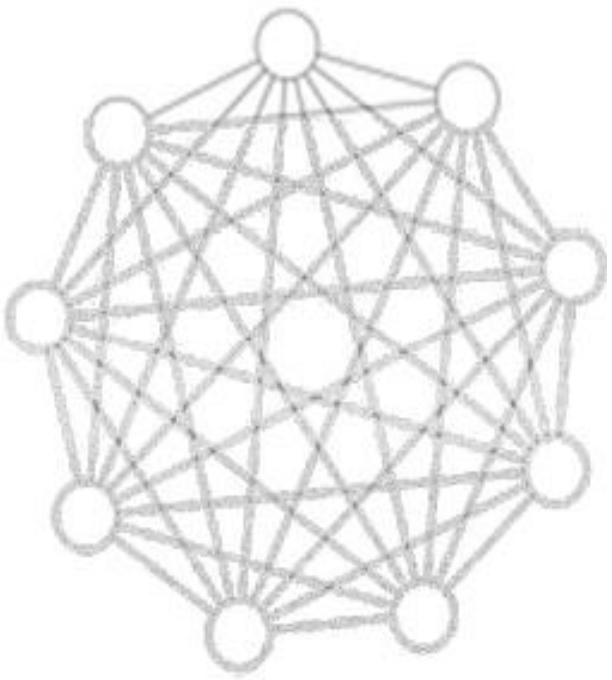
Particle Swarm Optimization (PSO)

- At the k-th **step** of the PSO algorithm three **vectors** and a **function** are associated to the j-th **particle**:
  - $\boldsymbol{x}_j^k \in \mathbb{R}^{\boldsymbol{d}}$: position at **step** k of the j-th **particle**;
  - $\boldsymbol{v}_j^k \in \mathbb{R}^{\boldsymbol{d}}$: velocity at **step** k of the j-th **particle**;
  - $\boldsymbol{p}_j \in \mathbb{R}^{\boldsymbol{d}}$: best position visited so far by j-th **particle** ($pbest_j = f(\boldsymbol{p}_j)$);
  - $\boldsymbol{f}_j^k \in \mathbb{R}$: objective function measuring the "goodness" at **step** k of the j-th **particle**.

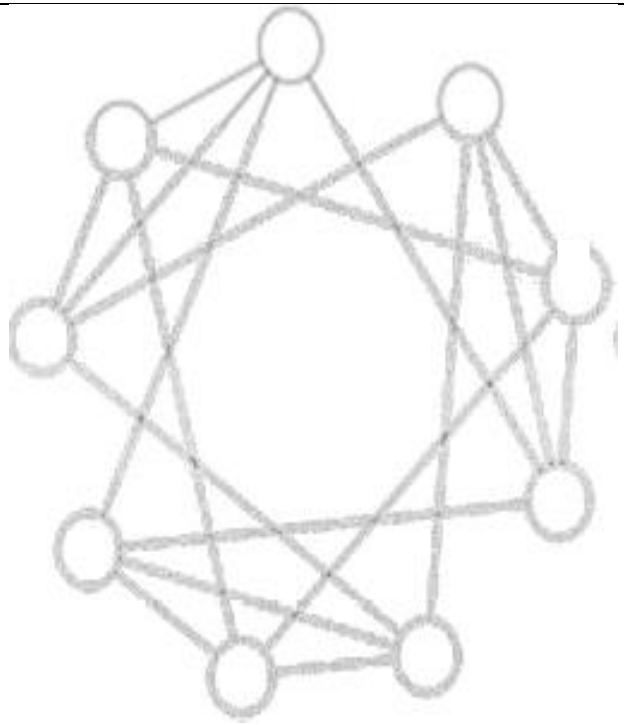
- Furthermore, $\boldsymbol{p}_g \in \mathbb{R}^{\boldsymbol{d}}$ indicates the best position visited so far by the members of the swarm belonging to a **neighborhood** of any particle.

Particle Swarm Optimization (PSO)

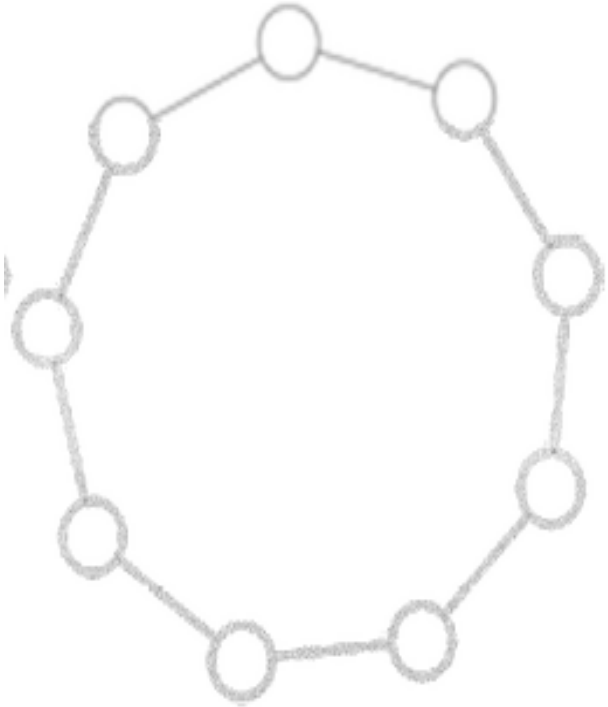
- There exist several typology of neighborhood.



Fully connected neighborhood



von Neumann neighborhood



Ring neighborhood

Particle Swarm Optimization (PSO)

- The PSO algorithm (in the version with **inertia weight**) is:
  1. Set $k = 1$ and evaluate $f\left(\boldsymbol{x}_j^k\right)$ for $j = 1, \dots, M$. Set $pbest_j = +\infty$.
  2. Repeat until a **stopping criterion** is satisfied.
     - If $f\left(\boldsymbol{x}_j^k\right) < pbest_j$ then set $\boldsymbol{p}_j = \boldsymbol{x}_j^k$ and $pbest_j = f\left(\boldsymbol{x}_j^k\right)$.
     - Update position and velocity of the j-th particle as

$$\begin{cases} \boldsymbol{v}_j^{k+1} = w^{k+1}\boldsymbol{v}_j^k + \boldsymbol{U}_{\phi_1} \otimes \left(\boldsymbol{p}_j - \boldsymbol{x}_j^k\right) + \boldsymbol{U}_{\phi_2} \otimes \left(\boldsymbol{p}_g - \boldsymbol{x}_j^k\right) \\ \boldsymbol{x}_j^{k+1} = \boldsymbol{x}_j^k + \boldsymbol{v}_j^{k+1} \end{cases}$$

     where $\boldsymbol{U}_{\phi_1}$ and $\boldsymbol{U}_{\phi_2}$ are uniformly randomly distributed in $[0, \phi_1]$ and $[0, \phi_2]$, respectively, and $\otimes$ indicates the component-wise product.
  3. Update $k = k + 1$ and go to 2.

Particle Swarm Optimization (PSO)

- The values of ϕ_1 and ϕ_2 can affect the performance.
- In order to yield the convergence of the swarm, ϕ_1 and ϕ_2 have to be set in accordance with the value of the **inertia weight** w^k.
- w^k is generally linearly decreasing with the number of steps, i.e.

$$w^k = w_{max} + \frac{w_{min} - w_{max}}{K} k$$

where
w_{min} and w_{max} are usually 0.4 and 0.9, respectively;
K indicates the maximum number of steps allowed.

Particle Swarm Optimization (PSO)

- Velocity:

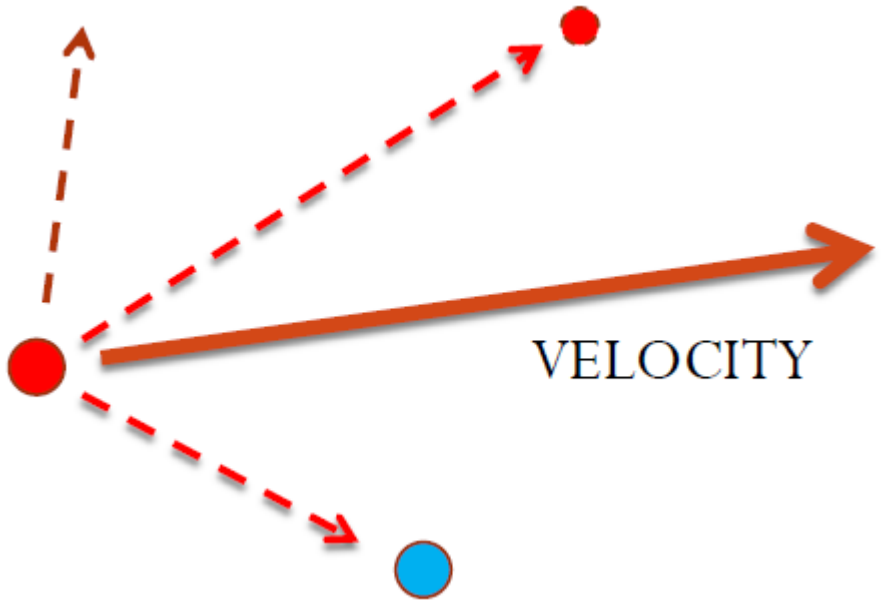
$$\boldsymbol{v}_j^{k+1} = w^{k+1}\boldsymbol{v}_j^k + \boldsymbol{U}_{\phi_1} \otimes \left(\boldsymbol{p}_j - \boldsymbol{x}_j^k\right) + \boldsymbol{U}_{\phi_2} \otimes \left(\boldsymbol{p}_g - \boldsymbol{x}_j^k\right)$$

Inertia | **Personal influence** | **Social influence**

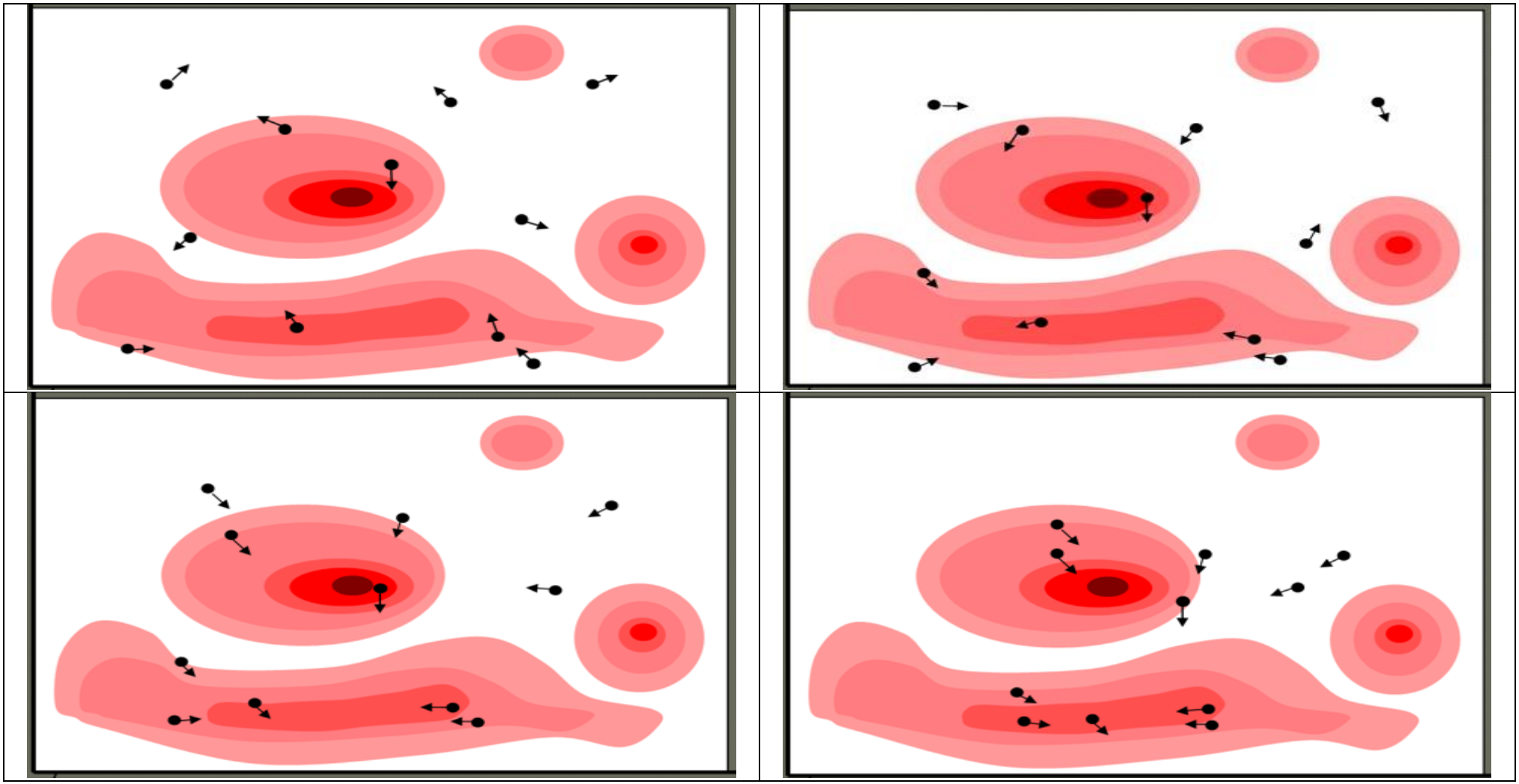
- **I** moves the particle in the same direction.
  PI moves the particle toward the best past position it has visited.
  SI moves the particle toward the best past position the swarm has visited.

Particle Swarm Optimization (PSO)

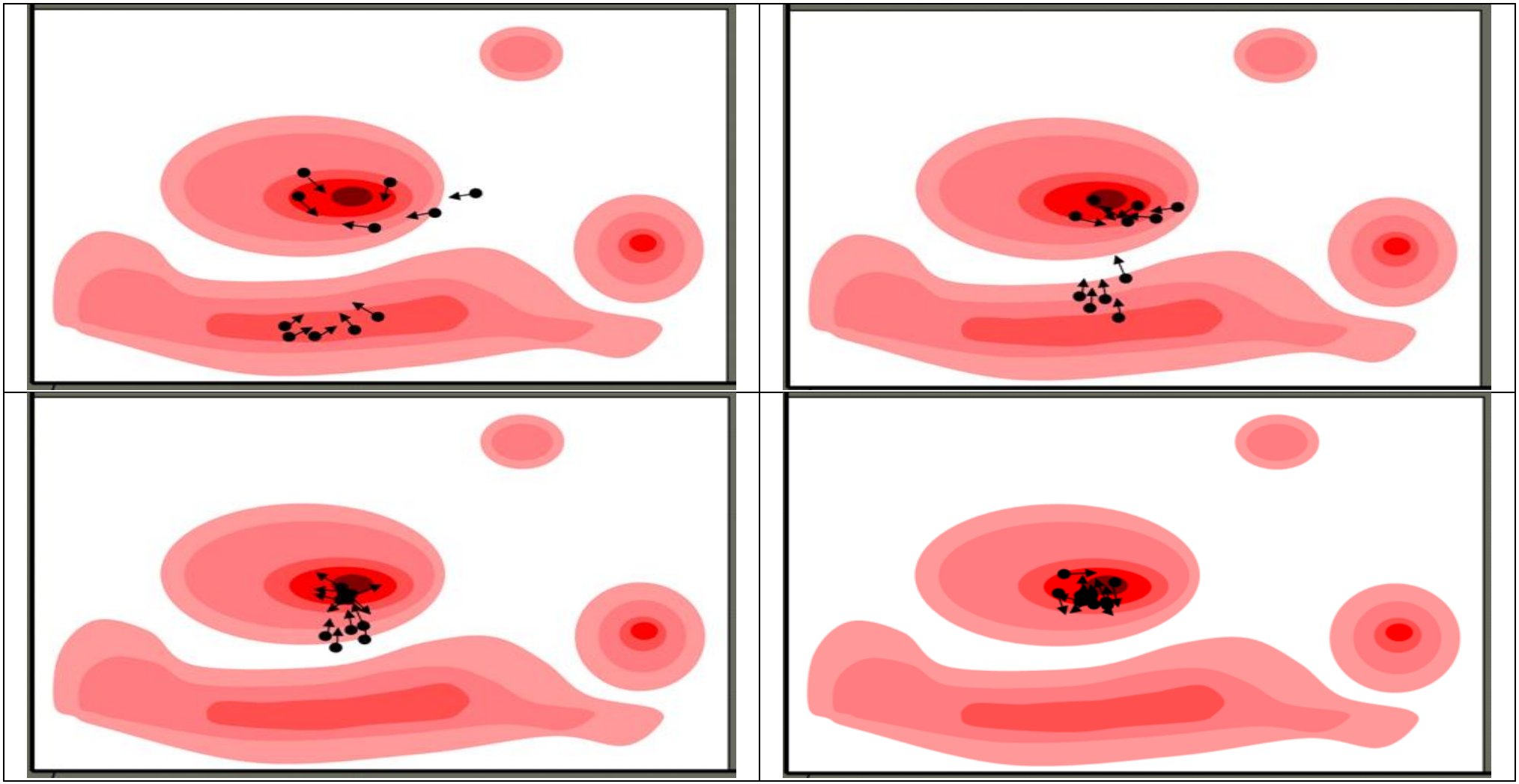
- Velocity



Particle Swarm Optimization (PSO) – Example



Particle Swarm Optimization (PSO) – Exmaple



Particle Swarm Optimization (PSO) – Applications

Basic portfolio selection problem

Harry M. Markowitz – 1990 Nobel Prize Laureate in Economics

[Constructed a micro theory of portfolio management
for individual wealth holders.]

Particle Swarm Optimization (PSO) – Introduction

- **Portfolio selection** consists in sharing a **starting capital** among various **stock**s whose future **performance**s are unknown, this in order to optimize some risk-return profile of the portfolio itself.

- Given the uncertainty of the future stock returns, a crucial role is played by the measurement of the risk associated to such stock returns. The classical approach represents the future stock returns in terms of random variables.

Particle Swarm Optimization (PSO) – Applications

Basic portfolio selection problem

$$\min \textbf{\textit{Portfolio risk}}$$
$$\text{s. t.} \begin{cases} \textit{Obtain a predetermined } \textbf{\textit{portfolio return}} \\ \textit{Invest all the } \textbf{\textit{starting capital}} \end{cases}$$

Particle Swarm Optimization (PSO) – Applications

Basic portfolio selection problem

- Given N tradable risky assets, the basic (and simplest) portfolio selection problem is:

$$\min_{\boldsymbol{x}} \boldsymbol{x}'\boldsymbol{V}\boldsymbol{x}$$
$$\text{s. t.} \begin{cases} \boldsymbol{x}'\boldsymbol{r} = \pi \\ \boldsymbol{x}'\boldsymbol{e} = 1 \end{cases}$$

where

✓ $\boldsymbol{x}' = (x_1, \dots, x_N)$ is the vector of percentages to invest in the risky assets;

Particle Swarm Optimization (PSO) – Applications

Basic portfolio selection problem

- ✓ $\boldsymbol{V} = \begin{pmatrix} \sigma_1^2 & \cdots & \sigma_{1,N} \\ \vdots & \ddots & \vdots \\ \sigma_{N,1} & \cdots & \sigma_N^2 \end{pmatrix}$ is the variance-covariance matrix of the returns of the risky assets;
- ✓ $\boldsymbol{r}' = (r_1, \ldots, r_N)$ is the vector of the expected returns of the risky assets;
- ✓ π is the expected returns the investor wishes from the portfolio;
- ✓ $\boldsymbol{e}' = (1, \ldots, 1)$.

- This portfolio selection problem is quadratic-linear.

Particle Swarm Optimization (PSO) – Applications

Basic portfolio selection problem

$$\min_{\boldsymbol{x}} \boldsymbol{x}'\boldsymbol{V}\boldsymbol{x}$$

$$\text{s. t.} \begin{cases} \boldsymbol{x}'\boldsymbol{r} = \pi \\ \boldsymbol{x}'\boldsymbol{e} = 1 \end{cases}$$

where

$$\boldsymbol{V} = \begin{pmatrix} 0.000748044 & -0.000238393 & -0.000234123 \\ -0.000238393 & 0.000841507 & 0.000053341 \\ -0.000234123 & 0.000053341 & 0.000707590 \end{pmatrix};$$

$$\boldsymbol{r}' = (0.046492, 0.050035, 0.047376).$$

Particle Swarm Optimization (PSO) – Applications

Basic portfolio selection problem

- $\pi = 0.049149 \sim 0.05 = 5\%$.
- Number of individuals/particles $= 30$.
- Number of iterations for GA $= 32{,}000$.
- Number of iterations for PSO $= 8{,}000$.

Method	x_1	x_2	x_3	$x'Vx$	$x'r$	$x'e$
Exact	*16.1%*	*72.1%*	*11.8%*	*0.000411*	*0.049149*	*1.000000*
PSO	15.6%	71.6%	12.8%	0.000409	0.049142	1.000000

Particle Swarm Optimization (PSO) – Applications

Complex portfolio selection problem

$$\min_{\boldsymbol{x},\boldsymbol{z}} a\|\max\{0, \boldsymbol{x}'\boldsymbol{r} - E(\boldsymbol{x}'\boldsymbol{r})\}\|_1 + (1-a)\|\max\{0, E(\boldsymbol{x}'\boldsymbol{r}) - \boldsymbol{x}'\boldsymbol{r}\}\|_p - E(\boldsymbol{x}'\boldsymbol{r})$$

$$\text{s. t.} \begin{cases} \boldsymbol{x}'\boldsymbol{r} \geq \pi \\ \boldsymbol{x}'\boldsymbol{e} = 1 \\ k_d \leq \boldsymbol{z}'\boldsymbol{e} \leq k_u \\ z_i d \leq x_i \leq z_i u \ \ \forall \ \ i \\ z_i \in (0,1) \ \ \forall \ \ i \end{cases}$$

where

- ✓ $\boldsymbol{z}' = (z_1, \ldots, z_N)$ is a vector of binary variables; $z_i = 0$ means that the i-th asse t is not selected, $z_i = 1$ means that the i-th asset is selected;

Particle Swarm Optimization (PSO) – Applications

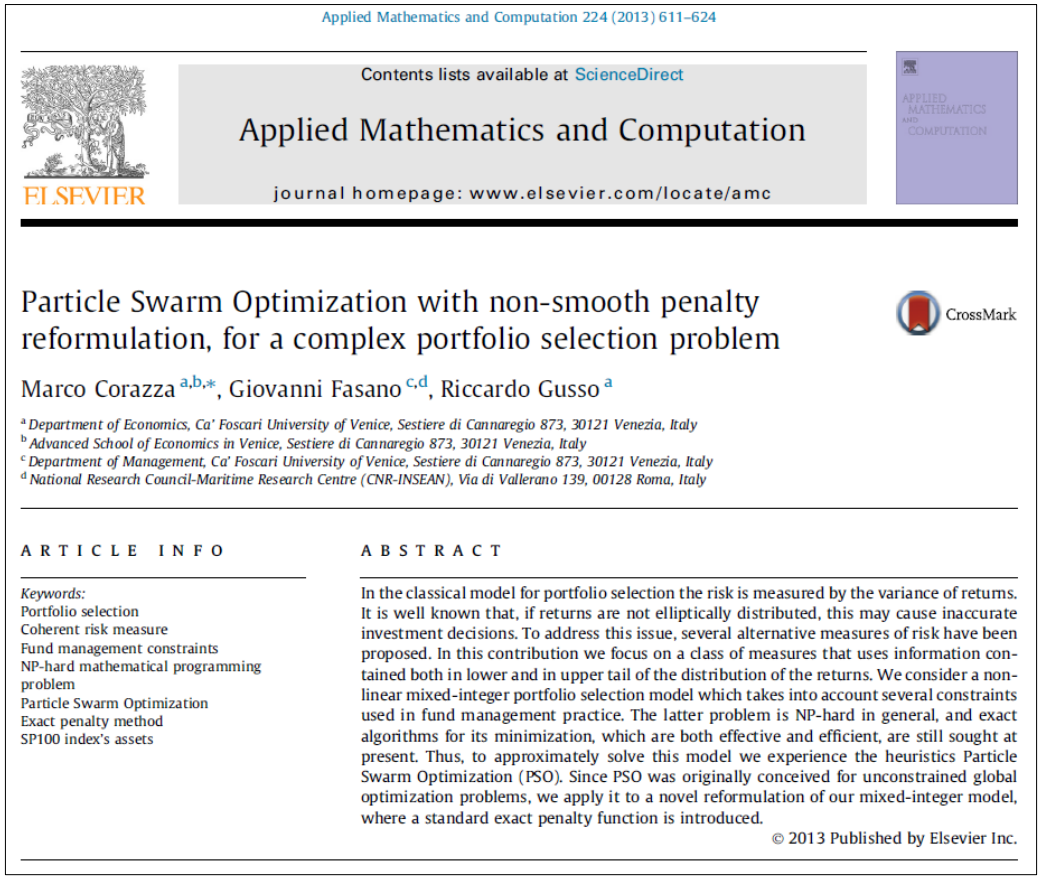
Complex portfolio selection problem

- ✓ k_d and k_u are respectively the minimum and the maximum numbers of assets in which to invest;
- ✓ d and u are respectively the minimum and the maximum percentages of starting capital to invest in the i-th asset, if selected;
- ✓ $\|\boldsymbol{x}\|_q = \left(\sum_{i=1}^{n} |x_i|^q\right)^{1/q}$

- This portfolio selection problem is **highly nonlinear**, **nondifferentiable** and **mixed-integer**.

Particle Swarm Optimization (PSO) – Applications

Complex portfolio selection problem



Applied Mathematics and Computation 224 (2013) 611–624

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Particle Swarm Optimization with non-smooth penalty reformulation, for a complex portfolio selection problem

Marco Corazza [a,b,*], Giovanni Fasano [c,d], Riccardo Gusso [a]

[a] Department of Economics, Ca' Foscari University of Venice, Sestiere di Cannaregio 873, 30121 Venezia, Italy
[b] Advanced School of Economics in Venice, Sestiere di Cannaregio 873, 30121 Venezia, Italy
[c] Department of Management, Ca' Foscari University of Venice, Sestiere di Cannaregio 873, 30121 Venezia, Italy
[d] National Research Council-Maritime Research Centre (CNR-INSEAN), Via di Vallerano 139, 00128 Roma, Italy

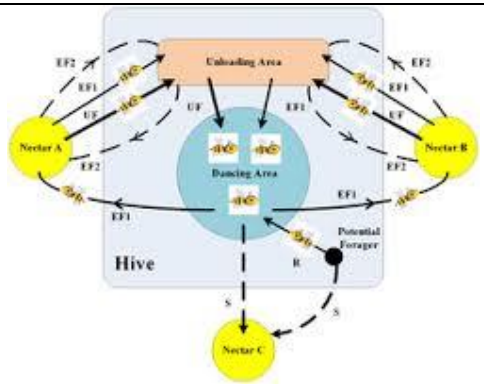
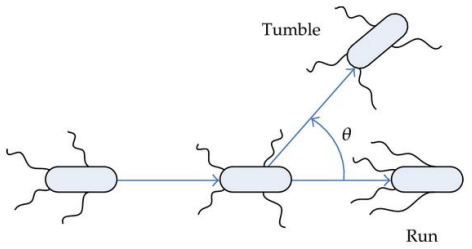
ARTICLE INFO

Keywords:
Portfolio selection
Coherent risk measure
Fund management constraints
NP-hard mathematical programming problem
Particle Swarm Optimization
Exact penalty method
SP100 index's assets

ABSTRACT

In the classical model for portfolio selection the risk is measured by the variance of returns. It is well known that, if returns are not elliptically distributed, this may cause inaccurate investment decisions. To address this issue, several alternative measures of risk have been proposed. In this contribution we focus on a class of measures that uses information contained both in lower and in upper tail of the distribution of the returns. We consider a non-linear mixed-integer portfolio selection model which takes into account several constraints used in fund management practice. The latter problem is NP-hard in general, and exact algorithms for its minimization, which are both effective and efficient, are still sought at present. Thus, to approximately solve this model we experience the heuristics Particle Swarm Optimization (PSO). Since PSO was originally conceived for unconstrained global optimization problems, we apply it to a novel reformulation of our mixed-integer model, where a standard exact penalty function is introduced.

Conclusion

Genetic Algorithms	
Artificial Bee Colony	
Bacterial Foranging Optimiz.	

References

- Corazza M., Fasano G. and Gusso R. (2013) "Particle Swarm Optimization with non-smooth penalty reformulation, for a complex portfolio selection problem", *Applied Mathematics and Computation*, 224, 611-624.
- Poli R., Kennedy J. and Blackwell T. (2007) "Particle Swarm Optimization: An overview", *Swarm Intelligence*, 1, 33-57.

Appendix

Fitness function

- Let us consider the following global **constrained** optimization problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})$$
$$s.t. \quad \begin{cases} h_j(\boldsymbol{x}) = 0, & j = 1, \dots, m \\ g_i(\boldsymbol{x}) \geq 0, & i = 1, \dots, p \end{cases}.$$

Appendix

Fitness function

- It is possible to prove that the previous global **constrained** optimization problem admits the **same solution**(s) than the following global **unconstrained** optimization problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) + \frac{1}{\epsilon}\left(\sum_{j=1}^{m}\left|h_j(\boldsymbol{x})\right| + \sum_{i=1}^{p}\max\{0, -g_i(\boldsymbol{x})\}\right)$$

where

ϵ is the so-called **penalty parameter**.

Appendix

Fitness function

- Note that
  - $\left|h_j(\boldsymbol{x})\right|$ measures the **violation** of the j-th equality constraints;
  - $\max\{0, g_i(\boldsymbol{x})\}$ measures the **violation** of the j-th inequality constraints.

Appendix

Fitness function

- In our case, the fitness function related to the basic version of the portfolio selection problem could be

$$\boldsymbol{x}'\boldsymbol{V}\boldsymbol{x} + \frac{1}{\varepsilon}(|\boldsymbol{x}'\boldsymbol{r} - \pi| + |\boldsymbol{x}'\boldsymbol{e} - 1|)$$

where
ε is a suitable positive penalty.