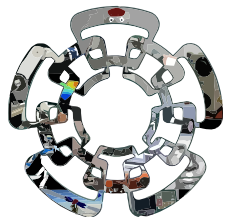
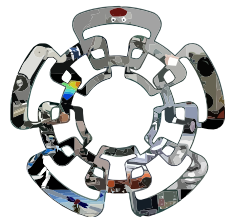
# Uso de un Algoritmo de Cúmulo de Partículas Basado en Hipervolumen para Resolver Problemas Multi-Objetivo

**Iván Christhofer Chaman García**

**Centro de Investigación y de Estudios Avanzados**
**del Instituto Politécnico Nacional**
**Departamiento de Computación.**

5 de Octubre de 2012





# Motivation

## Main goals of this work :

- Implement a robust MOAIS based on simple features
- Respect the immune system metaphor
- Use hypervolume to select/discard solutions
- Investigate the performance of this class of algorithm
- Use a well-adapted language for optimization (C)
- The complexity should be competitive with state-of-the-art algorithms
- Test the algorithm on a large set of MOPs
- Compare results being as fair as possible
- Provide an algorithm that compares results between two algorithms

# Modelisation of a MOP

$$\begin{bmatrix} opt & f_i(\vec{x}) & \forall i \in \{1, ..., m\} \\ s.t & & \\ & g_j(\vec{x}) \leq 0 & \forall j \in \{1, ..., p\} \\ & h_k(\vec{x}) = 0 & \forall k \in \{1, ..., q\} \\ & x_l \in [lb, ub] & \forall l \in \{1, ..., r\} \end{bmatrix}$$

## where :

- $opt \in \{min, max\}$,
- $m$ is the number of objective functions,
- $p$ is the number of inequality constraints,
- $q$ is the number of equality constraints,
- $r$ is the number of decision variables of the problem,
- $lb, ub$ are the lower and upper bound of each variable $x_l$, respectively.

In the following, we assume that all objectives are to be minimized

# Definitions & Notations

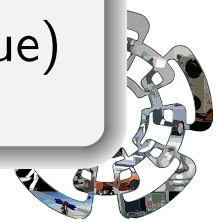
## Pareto dominance

One vector $\vec{x}$ dominates a vector $\vec{y}$ iff $f_i(\vec{x}) \leq f_i(\vec{y})\ \forall i \in \{1, ..., m\}$ and there exists at least one $i$ such that $f_i(\vec{x}) < f_i(\vec{y})$. We also say that $\vec{y}$ is dominated by $\vec{x}$. A binary operator "$\prec$" is defined as :

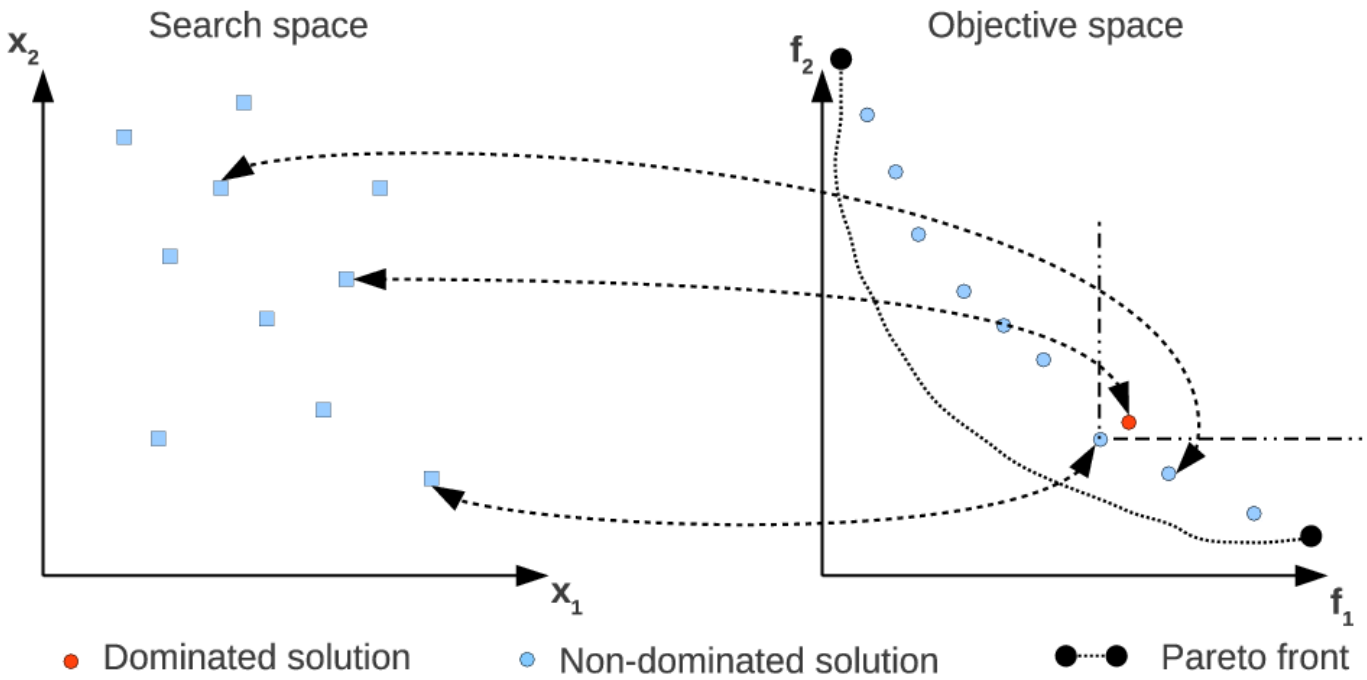
$$\vec{x}\ \texttt{dominates}\ \vec{y} \Longleftrightarrow \vec{x} \prec \vec{y}$$

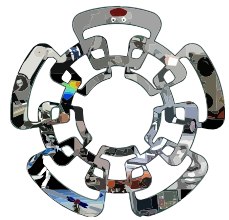
## Pareto optimal Set & Pareto front

All the solutions whose vectors are not dominated by any other vector in $\mathbb{R}^m$ are said to be **non-dominated**.
The Pareto optimal set is composed of all vectors in the search space $\mathbb{R}^n$ that are non-dominated.
The image of the Pareto optimal set (i.e., their objective function value) form the Pareto front.



# Bi-objective example



# Goals in MOO (Multi-Objective Optimization)

The main goal in multi-objective optimization is to generate as many elements of the Pareto optimal set as possible.

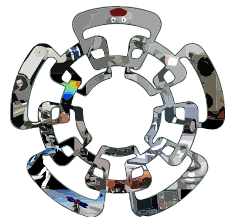
## Convergence

Allows to measure how far the solutions found are, from the true Pareto front.

## Spread

Indicates how well-distributed are the solutions on the true Pareto front (or its approximation).

Usually, solving a MOP consists of converging as fast as possible to the Pareto front (convergence) while keeping a well-distributed set of solutions along the Pareto front (spread).



# Why EAs ?

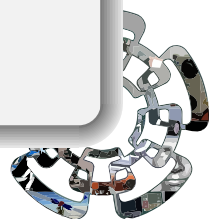
An EA uses some mechanisms inspired by biological evolution, which have been shown to be efficient on a large set of difficult problems. EAs are said to be population-based algorithms because they use a set (population) of solutions that is updated at each iteration (generation).

## Exact algorithms :

- Exploring the search space is time consuming
- Solutions are often given sequentially

## Population-based algorithms ($\supseteq$ EAs) :

- Give a set of solutions in one generation (one run of the algorithm)
- Have a low complexity
- Drawback : convergence cannot be guaranteed



# The immune system

The main goal of the human immune system is to protect the internal components of the human body by fighting against foreign elements.

## Main element of an IS :

- B cells
  B-cells get activated when an infection occurs. B cell starts to divide to produce clones of itself. During this process, two new cell types are created, plasma cells and B memory cells. Plasma cells produce molecules called antibodies that attach to the surface of the infectious agent.
- Antibodies
  Play a key role by attaching the current type of invaders. When an antibody fits to an antigen, "eaters" cells use this recognition and "eats" them.
- Antigens
  Represents the pathogens agents (invaders).

# The immune system

## Some features of an IS :

- Positive and negative selection
  Process of discrimination of self/non-self cells that prevents autoimmuno diseases.
- Clonal selection and expansion
  Process of selection of useful cells that recognize the antigen and reproduce those cells. The clones suffer hypermutation that alters the shape of the receptor, increasing the **affinity** between the clone and the specific antigen
- Immune memory
  B memory cells remember the shape of the antigen that they have fought and recollect when they see the same antigen again. This process helps the system to produce cells with higher affinities
- Jerne's idiotropic network
  Deals with the interaction of antibodies. B cells communicates the shape of the antigen amongst them.

# AIS algorithms

AIS typically exploit the immune system's characteristics of learning and memory to solve a problem.

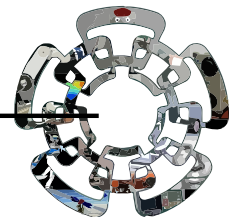
## Main elements of an AIS

- A set of immune agents (antibodies) that try to find the best binding to fit to pathogen agents (antigens).
- A set of cells that record characteristics of antigens previously encountered.
- Communication between these entities.
- The capability of some cells to clone (asexual reproduction) and mutate.

# Canonical MOAIS algorithm

**Algorithm 1:** Outline of the canonical MOAIS

1 Define the search space $\mathbb{S}$, objectives functions $f_i$, constraints $g_j, h_k$;
2 $A(t=0) \longleftarrow$ Initialize offline population;
3 $B(t=0) \longleftarrow$ Initialize online population with random individuals;
4 **while** $\neg$ *stop criterion* **do**
5 Evaluate population $B(t)$ using $f_i, g_j, h_k$;
6 $B_1(t) \longleftarrow$ Define affinities($B(t)$, $[A(t)]$);
7 $B_2(t) \longleftarrow$ Selection for cloning($B_1(t)$, $[A(t)]$);
8 $B_3(t) \longleftarrow$ Proliferation and mutation($B_2(t)$);
9 $B_4(t) \longleftarrow$ Diversification & Suppression;
10 $B(t+1) \longleftarrow B_3(t) \cup B_4(t)$;
11 $A(t+1) \longleftarrow$ Update($A(t), B(t+1)$);
12 $t \longleftarrow t+1$;
13 **end**

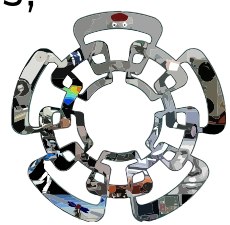


# Hypervolume definition

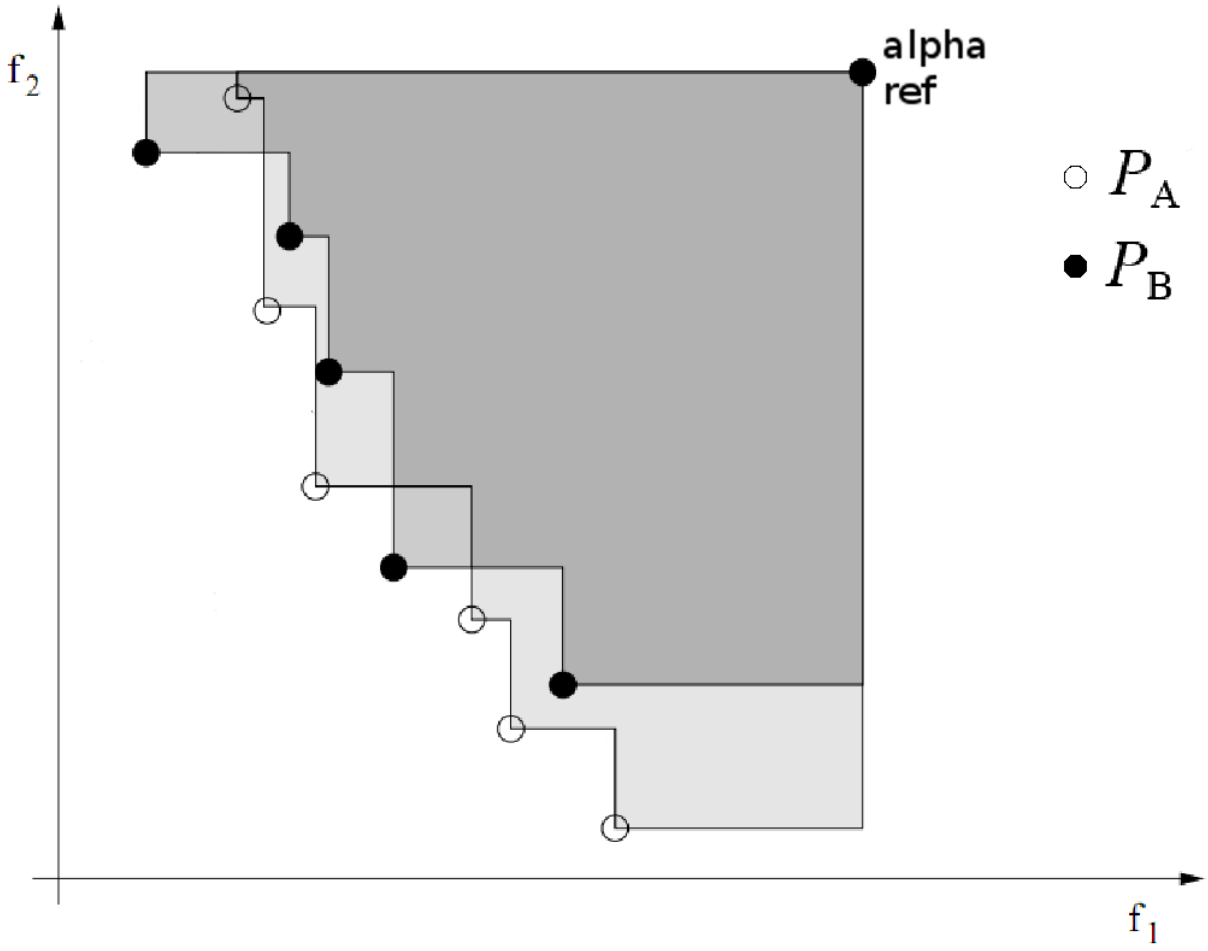
Hypervolume is an indicator used to measure and compare the quality of final solutions in population-based algorithms. This indicator represents the surface (or the volume for more than 2 objectives) of the region dominated by solutions found so far.
Let $\Lambda$ denote the Lebesgue measure, then the $\mathcal{S}$ metric is defined as

$$\mathcal{S}(A, y_{ref}) = \Lambda \left( \bigcup_{y \in A} \{ y' |\ y \prec y' \prec y_{ref} \} \right), A \subseteq \mathbb{R}^m$$

where :
$A$ : subset of the objective space,
$y_{ref}$ : reference point that is dominated by all Pareto-optimal solutions,
" $\prec$ " : the dominance relation.



# Hypervolume



# Hypervolume vs Hypervolume contribution

We wants to select the $p$ solutions that maximize the hypervolume among a population of $n$ individuals.

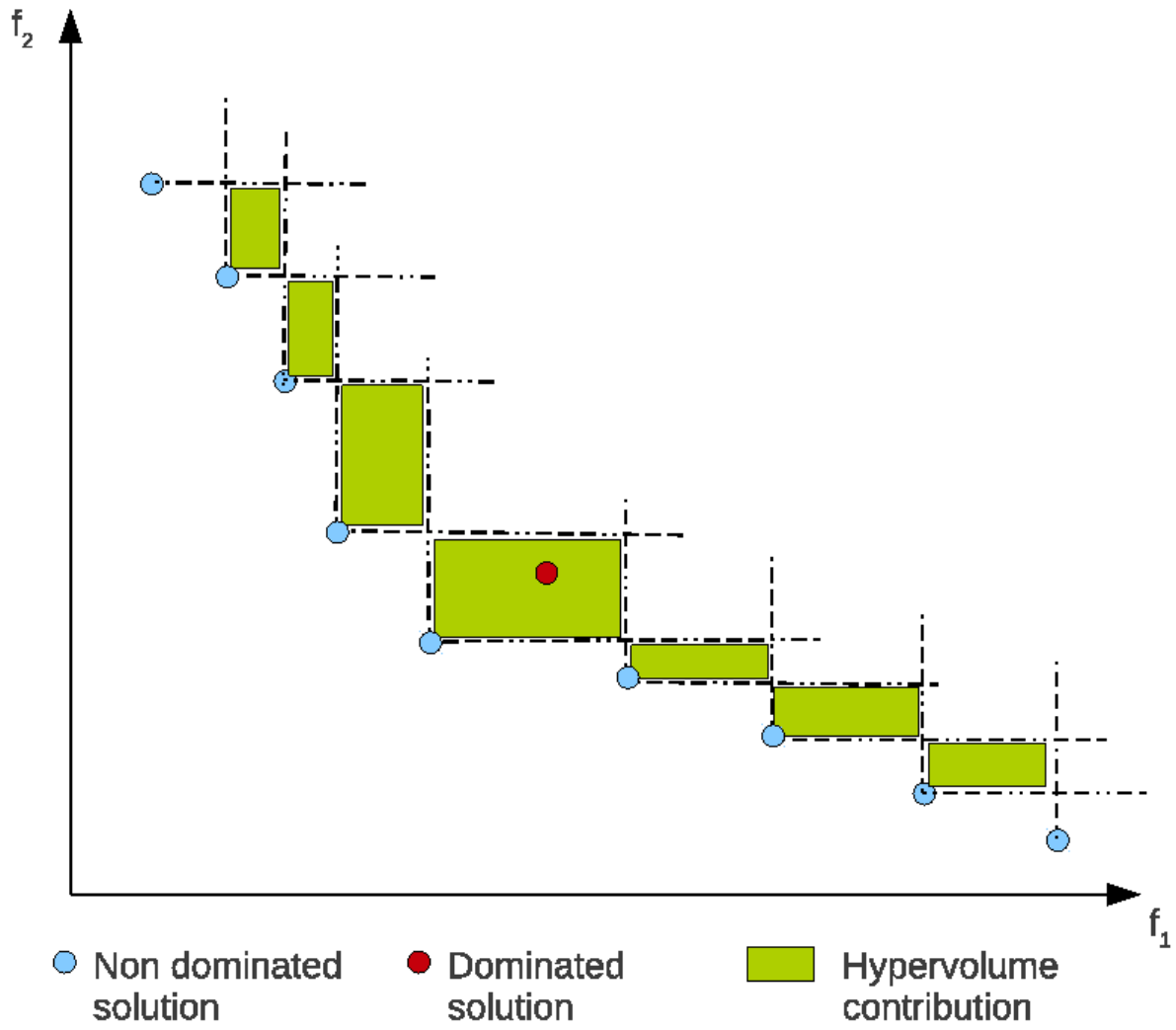
## Hypervolume indicator

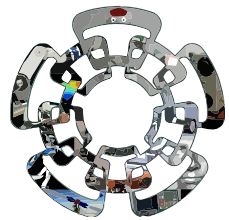
- Computes the whole set of solutions
- Complexity increases with the number of objectives
- Time consuming
- Select the optimal set of $p$ solutions

## Hypervolume contribution

- Considers only the contribution of one solution in maximizing the hyervolume
- Complexity acceptable
- Easy implementation
- the chosen set can be different than the optimal set of $p$ solutions ! !
- But, error ratio $< 35\%$

# Hypervolume contribution



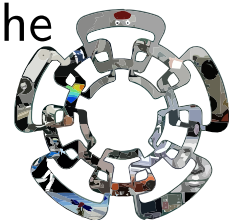
# Previous algorithms

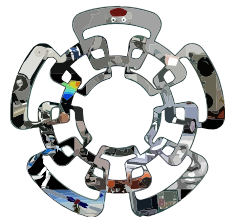
Since 2002, MOAIS algorithms have been shown to be efficient in MOO. Nevertheless, no results on true AIS have been provided for hard problems (DTLZ class).

## Some facts

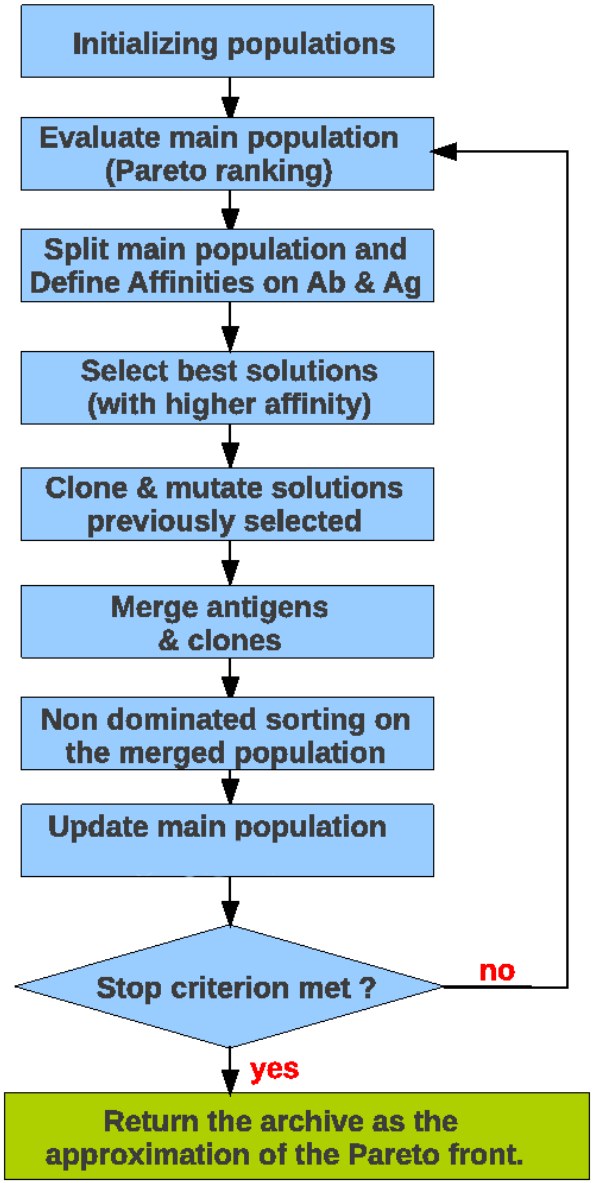
- True MOAIS only tested on easy MOP
- MOAIS tested on hard problem uses recombination
- MOAIS never combined with Hypervolume before
- A few comparison with state-of-the-art algorithms

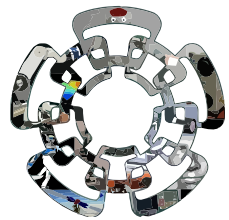
The most common idea adopted in MOAIS is to define two sets of solutions. Some cells are considered as the worst solutions, while the others are considered as good solutions. Most of the algorithm uses the clonal selection principle and classical Pareto dominance ranking.





# Main loop


Initializing populations
Evaluate main population (Pareto ranking)
Split main population and Define Affinities on Ab & Ag
Select best solutions (with higher affinity)
Clone & mutate solutions previously selected
Merge antigens & clones
Non dominated sorting on the merged population
Update main population
Stop criterion met ?
no
yes
Return the archive as the approximation of the Pareto front.



# Spliting the population

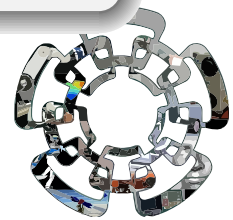
The main population is divided in two populations :
Antigens (archive), only composed of non-dominated solutions.
Antibodies, other solutions.

## Unconstrained problems

- Antigens : non-dominated individuals
- Antibodies : the others

## Constrained problems

- Antigens : Non-dominated and feasible individuals
- Antibodies : the others



# Defining affinity

## On antibodies (bad solutions)

- Presence of antigens :
  Affinity is the inverted Euclidean distance between the antibody and an antigen chosen randomly : $Aff(Ab_i) = \frac{1.0}{Eucl\_Dist(Ab_i, Ag)}) \ \forall i$
- No antigens :
  Affinity is based on the antibody's rank :
  $Aff(Ab_i) = \frac{1.0}{(Rank(Ab_i)+1)}) \ \forall i$

## On antigens (good solutions)

- Affinity is based on Hypervolume contribution :
  $Aff(Ag_i) = Hyperv\_Cont(Ag_i) + max_j(Aff(Ab_j)) \ \forall i$

Extreme solutions, antibodies as well as antigens, are set to the maximum affinity values.

# Clonal selection principle

## Number of candidates to be cloned

- Defined by a parameter $nb\_cl$
- $NC = n * 10\% + nb\_cl * n * 30\%$

## Number of clones to create

- $P_1$ represents the set of extreme solutions
- $P_2$ represents the other solutions
- A coefficient is applied to $P_j$ depending on the generation
- $NCC(A_{i,j}) = P_j * \frac{Aff(A_{i,j})}{\sum_{i=0}^{n_j} Aff(A_{i,j})} \quad \forall i, j$

  where :
  $A_{i,j}$ is the $i^{th}$ antigen or the $i^{th}$ antibody of the set $j$,
  $P_j$ is the total number of clones for the set $j$,
  $n_j$ is the number of candidates in the set $j$.

# Mutation

## Mutation probability

Controls the probability to mutate one variable of a vector. The mutation probability is defined by $\frac{1}{nreal}$. Should be improved in a future work.

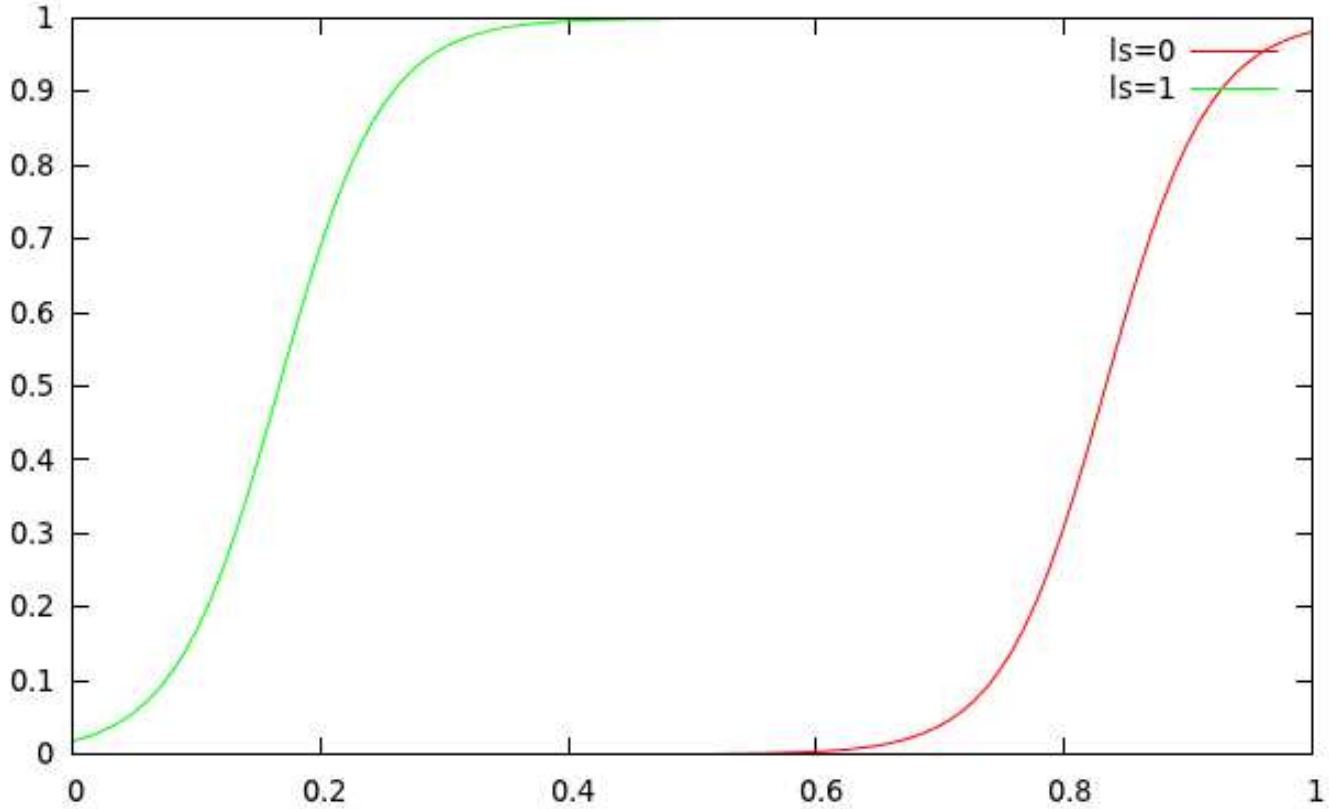
## Hybrid mutation step-size

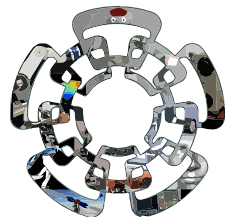
The mutation is defined by two classical Gaussian mutations.
GL, Gaussian local mutation, which defined small step size mutation.
GG, Gaussian global mutation, which defined larger step-size.
Each time a variable has to be mutated, we compute the following value :

$$p\_mut\_type = \frac{1.0}{(1.0 + exp(-2.0 * (x + p)))}$$

where :
$x = -6.0 + (t/ngen) * 12.0$, increasing with the number of generations.
$p = -4.0 + ls * 8.0$, with ls a parameter which determines the tradeoff between $GL$ and $GG$.

# Gaussian mutation type probability



# Gaussian Mutation

## GL vs GG

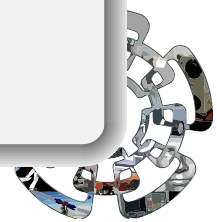
Local Gaussian mutation is performed following this formula :

$$x_i^* = x_i + (max_i - min_i) * 0.1 * \mathcal{N}(0, st_1)$$

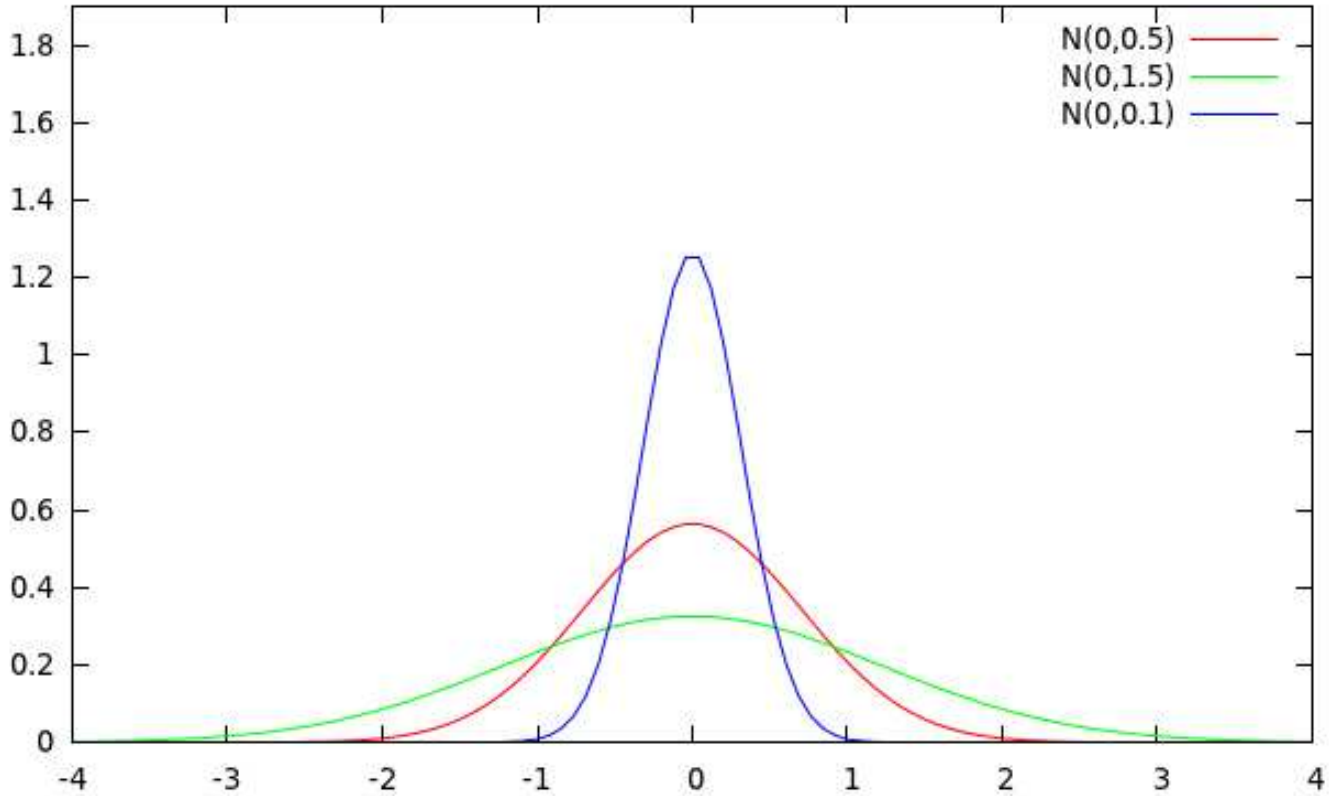
Global Gaussian mutation is performed following this formula :

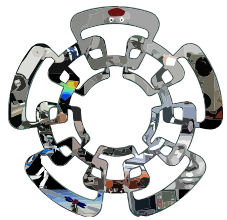
$$x_i^* = x_i + (max_i - min_i) * 0.1 * \mathcal{N}(0, st_2)$$

where :
$max_i, min_i$ are the bounds of the decision variable,
$x_i$ is the variable to be mutated,
$\mathcal{N}(0, x)$ is the Normal distribution with mean 0 and standard deviation $x$,
$st_1 \in [0.1, 0.5]$, parameter controlling local Gaussian mutation step,
$st_2 \in [0.5, 1.5]$, parameter controlling global Gaussian mutation step.



# Gaussian mutation



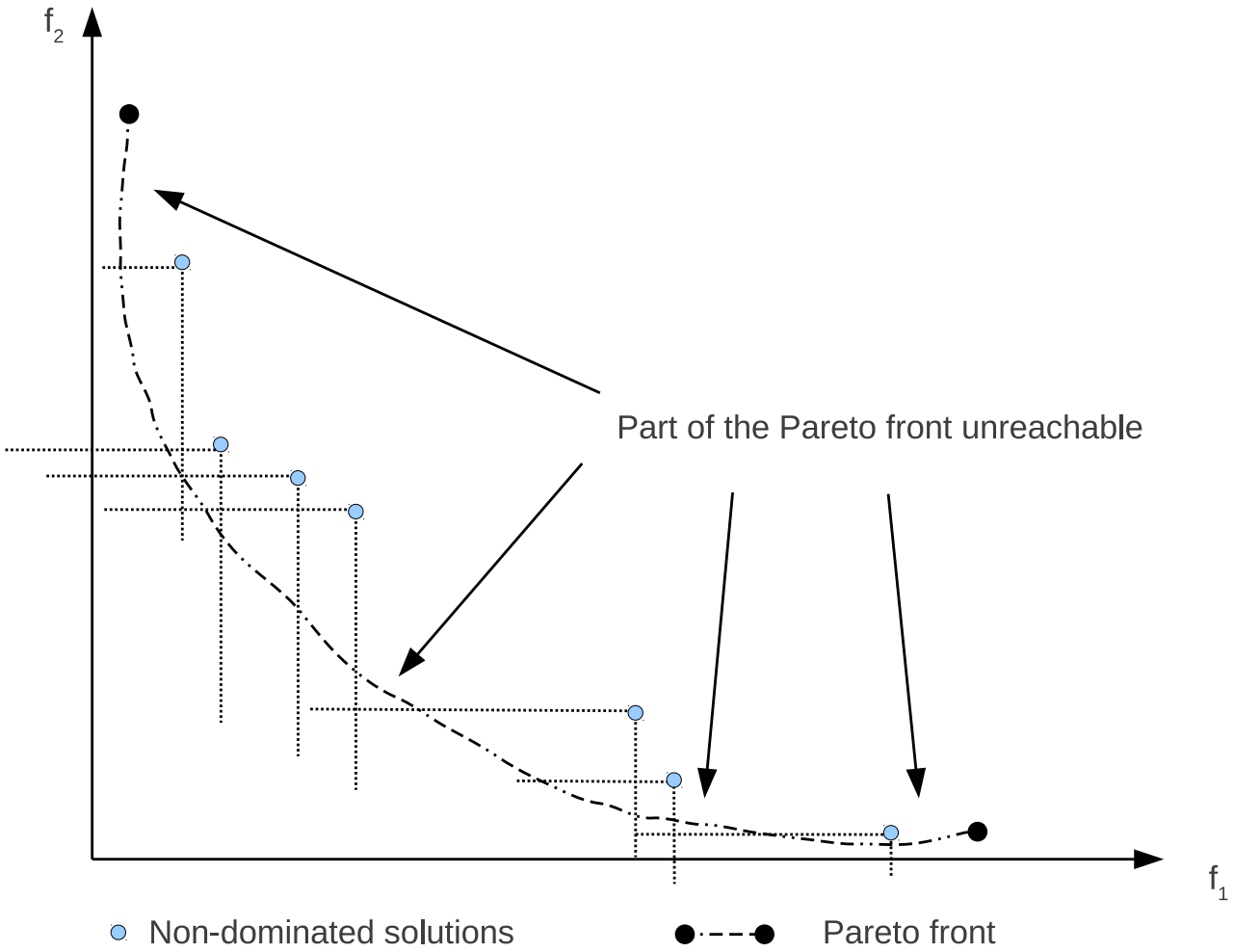



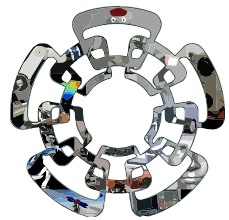
# Update the main population

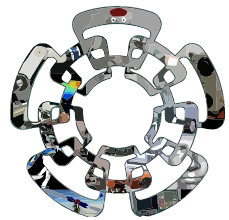
## Archiving process

- Fill the main population with successive ranks from the pool $P$.
- If the addition of the individuals of the current rank is greater than $n$, two cases occur :
  - if ($curgen < \frac{2}{3} ngen$), perform Hypervolume discard process (the individual which has the lowest hypervolume contribution is discarded. The procedure is repeated until reaching a rank of size $n - n_{prev}$ which will be added to the main population). The aim here is to find a good spread.
  - Otherwise, only accept individuals that dominate solutions already in the archive by replacing them. The aim here is to maximise the hypervolume.

# Update main population





# Comparison : Some indicators

## Hypervolume

Indicates the behavior of the algorithms for both convergence and spread. The reference point is computed according to both the algorithms and the true Pareto front.

## IGD

Measures how far the elements in the Pareto optimal set are from those in the set of non-dominated vectors found.

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$$

where :
$n$ is the number of vectors in the Pareto optimal set.
$d_i$ : Euclidean distance between each of these solutions and the nearest member of the set of the non-dominated vectors found

# Comparison : Some indicators

## Two sets coverage

Indicator that counts how many solutions of a population dominate solutions from another population.

$$I_c(A, B) = \frac{|\{ \vec{x} \in A \mid \exists\, \vec{y} \in B \ : \ \vec{x} \prec \vec{y} \}|}{|A|}$$

## Spread

Measures the extent of spread achieved among the obtained solutions :

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{n-1} |d_i - \bar{d}|}{d_f + d_l + \bar{d}\ (n-1)}$$

where :
$d_i$ is the Euclidean distance between consecutive solutions,
$\bar{d}$ is the mean of these distances,
$d_f$ and $d_l$ : Euclidean distances to the extreme solutions of the PF.

# Algorithm parameters

## MOISA-HV : Setting up parameters

- a : trade-off between global search and local search.
  0.0 allows a more global search and 1.0 allows a more local search.
- b & c : Parameters which define mutation stepsize for Gaussian mutation.
  0.0 allows smaller step-size mutation whereas 1.0 sets the mutation to a larger step-size.
- d : Parameter which defines the number of candidates to be cloned.
  0.0 represents 10% of the size of the population. 1.0 represents 40% of the population's size.

## NSGA-II : parameters

Usual parameters are used for problem originally implemented in NSGA-II. For other problems, parameters where chosen depending on the number of objectives, variables and constraints that are efficient in other problems.

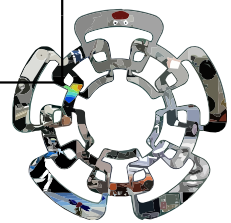
# MOISAHV vs NSGA-II

## Methodology

Quick search on the parameters to find (more or less) the best values. Each parameter is tested with 6 values uniformly distributed on their domain of definition. NSGA-II is run with its classical parameters.

| Name | # Var | # Obj | # Const | S(NSGAII) | S(MOISA-HV) |
|---|---|---|---|---|---|
| ZDT1 | 30 | 2 | 0 | 0 | 5 |
| ZDT2 | 30 | 2 | 0 | 0 | 5 |
| ZDT3 | 30 | 2 | 0 | 2 | 3 |
| ZDT4 | 10 | 2 | 0 | 5 | 0 |
| ZDT6 | 10 | 2 | 0 | 0 | 5 |
| KUR | 3 | 2 | 0 | 0 | 5 |
| SCH1 | 1 | 2 | 0 | 1 | 4 |
| SCH2 | 1 | 2 | 0 | 0 | 5 |

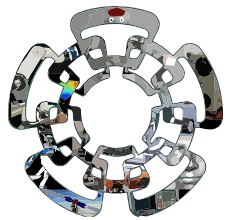
Bi-objective unconstrained problems



# MOISAHV vs NSGA-II

| Name | # Var | # Obj | # Const | S(NSGAII) | S(MOISA-HV) |
|---|---|---|---|---|---|
| BNH | 2 | 2 | 2 | 0 | 5 |
| OSY | 6 | 2 | 6 | 5 | 0 |
| SRN | 2 | 2 | 2 | 0 | 5 |
| TNK | 2 | 2 | 2 | 2 | 3 |

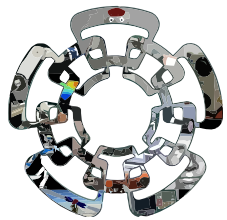
Bi-objectives constrained problems



# MOISAHV vs NSGA-II

| Name | # Var | # Obj | # Const | S(NSGAII) | S(MOISA-HV) |
|---|---|---|---|---|---|
| BNH4 | 2 | 3 | 2 | 0 | 5 |
| VNT1 | 2 | 3 | 0 | 0 | 5 |
| VNT2 | 2 | 3 | 0 | 1 | 4 |
| VNT3 | 2 | 3 | 0 | 0 | 5 |
| DTLZ1 | 12 | 3 | 0 | 5 | 0 |
| DTLZ2 | 12 | 3 | 0 | 1 | 4 |
| DTLZ3 | 12 | 3 | 0 | 4 | 1 |
| DTLZ4 | 12 | 3 | 0 | 3 | 2 |
| DTLZ7 | 22 | 3 | 0 | 1 | 4 |

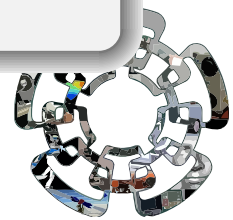
Three objectives problems

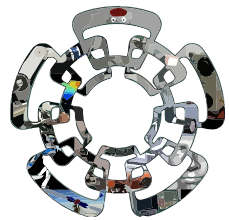


# MOISAHV vs NSGA-II

## Results analysis

- On bi-objective unconstrained problems, according to the hypervolume, MOISA-HV performs better.
- Some problems remain hard to solve for MOISA-HV (ZDT4, OSY, some DTLZs).
- Problems when solving problems having a discontinuous Pareto front (ZDT3).
- Difficulties when solving problems having PF with steep slope (OSY).
- Graphically, we can see that convergence towards the PF is achieved for all problems.





# Conclusion

## Achieved goals

- ✓ Present a simple and robust algorithm based on simple rules
- ✓ Respect of the MOAIS canonical algorithm
- ✓ Test the algorithm on a large set of problem
- ✓ Convergence to PF is achieved for all problems

## Drawbacks

- 4 parameters have to be tuned
- Parameters domains have to be redefined
- Hypervolume discard process does not select relevant solutions when PF have steep slope

# Future work

## Further research

- Investigate self-adaptive parameters
  Mutation step-size and mutation type parameters should be self-adaptive. A method to detect when the algorithm reaches the PF should be found.
- Mutation probability
  It has been shown that mutation probabiltiy can be greater than $\frac{1}{nreal}$ at the beggining at the search and be more efficient on multi modal problems.
- Non-uniform mutation
  By applying a coefficient to the Local Gaussian mutation, non-uniform mutation would improve precision of the results after detecting that most of the solutions are on the Pareto front.
- Comparator
  Improve its complexity in order to be able to compute more results faster. Add the convergence rate indicator.

# Questions ?

Thank you for your attention