

# Sujets TP d'application

## INT21 — 2024

# Par groupe de 2 ou 3 personnes

## TP simples :

- Faire un algo mémétique et le comparer à l'algo simple, tester sur des Weierstrass et sur Rastrigin.
- Programmer un niching et tester sur Rastrigin en grande dimension.
- Comparer différents algorithmes d'optimisation multi-objectifs.
- Programmer un algorithme multi-population (modèle en îlots), le comparer à un algo classique.

## TP de difficulté moyenne :

- Programmer et tester différentes mesures de diversité.
- Régression symbolique : A Living Benchmark for Symbolic Regression
- Labyrinthe avec ArenaBot.
- Optimisation d'un réseau de neurones.

## TP plus compliqués :

- Co-evolution coopérative : problème « Lamps ».
- Co-évolution compétitive : modèle prédateur/proie.
- Exploring Hawk-Dove games
- Evolution interactive d'ensembles de Julia.

**Sujet « sur mesure » sur demande**

## Contrôle

- **Présentation 10 min + 5 min questions**
- **Présenter méthode, résultats**
- **Analyse critique !**

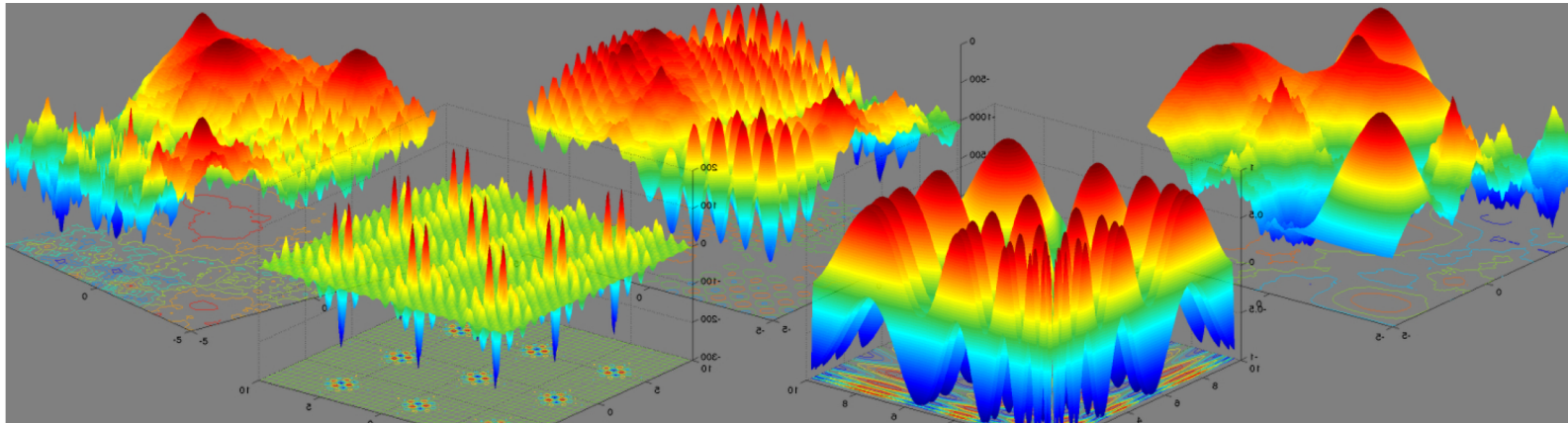
# Algorithme évolutionnaire mémétique

- **Optimisation évolutionnaire classique, mais dotée de capacités “d’apprentissage”. En général : rajout d’une optimisation locale lors de l’évaluation de chaque individu.**
- **Implémentation avec inspyred.**
- **Paramétrage et comparaison de l’approche mémétique avec l’approche classique.**



# Niching : Identification des différents optima

- **Fonctions multi-modales :**

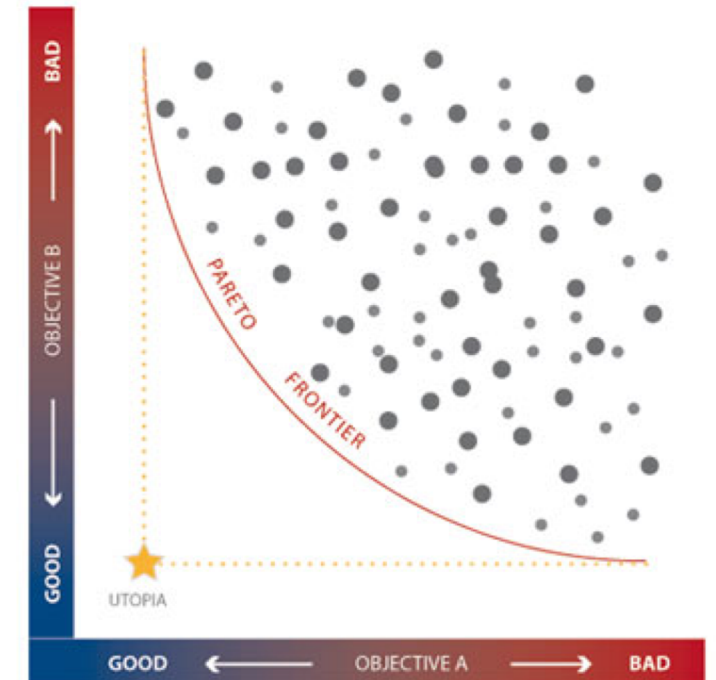


- **L'optimisation évolutionnaire classique converge vers un seul optimum local.**
- **Solution : Forcer la population à converger vers plusieurs optima plutôt qu'un seul.**
- **Implémenter à l'aide d'inspyred une ou plusieurs méthodes de niching.**
- **Tester les méthodes sur des fonctions multi-modales et comparer leurs performances avec l'optimisation évolutionnaire classique (code TP1).**



# Comparaison de différents algorithmes d'optimisation multi-objectifs

- Comparer les performances entre NSGA-II, NSGA-III et MOEA/D
  - Temps de calcul,
  - Qualité des fronts obtenus (programmer différentes métriques).
- Utilisation de différents problèmes multi-objectifs, avec un nombre d'objectifs variable.
- Lien code : <https://pymoo.org>
- Benchmarks : <https://pymoo.org/problems/multi/zdt.html>



Niveau de difficulté ★

# Algorithme multi-populations : modèle à îlots

- Point de départ : <https://pythonhosted.org/inspyred/examples.html#island-models>
- Comparer avec un modèle mono-population utilisant le même nombre d'évaluations.
- Faire varier le nombre d'îlots, les fréquences de communication.
- Sur différentes fonctions-test (Rastrigin, Weierstrass).

# Mesurer et suivre la diversité

- Programmer une mesure de diversité de population.
  - Mesures **géométriques** : somme, max, moyenne des **distances** entre toutes les paires possibles d'individus (définir une distance sur l'espace de recherche : **phénotypique**, **génotypique**, **fitness** ?)
  - Mesures **statistiques** (variance, écart-type, caractéristiques calculées sur les histogrammes des valeurs de gènes, entropie de Shannon)
- Affichage à chaque génération (petite interface de visualisation).
- Tester et comparer les mesures de diversité (différents paramétrages d'AE sur quelques fonctions-test).

<https://aarongarrett.github.io/inspyred/reference.html#benchmarks-benchmark-optimization-functions>

# Régression symbolique : A Living Benchmark for Symbolic Regression

Utiliser le code <https://cavalab.org/srbench/> pour tester PySR et Gplearn (accessibles par pip install) sur :

- <https://github.com/cavalab/srbench>
- <https://cavalab.org/srbench/user-guide/#reproducing-the-experiment>

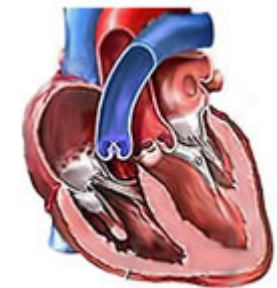
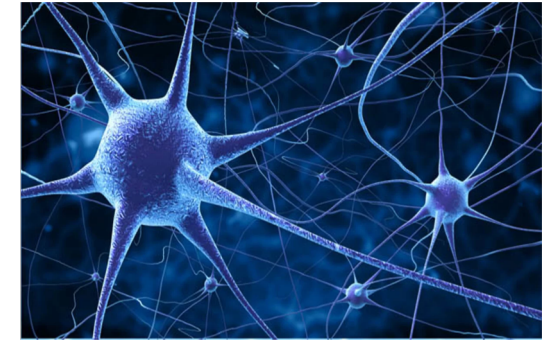
Niveau de difficulté





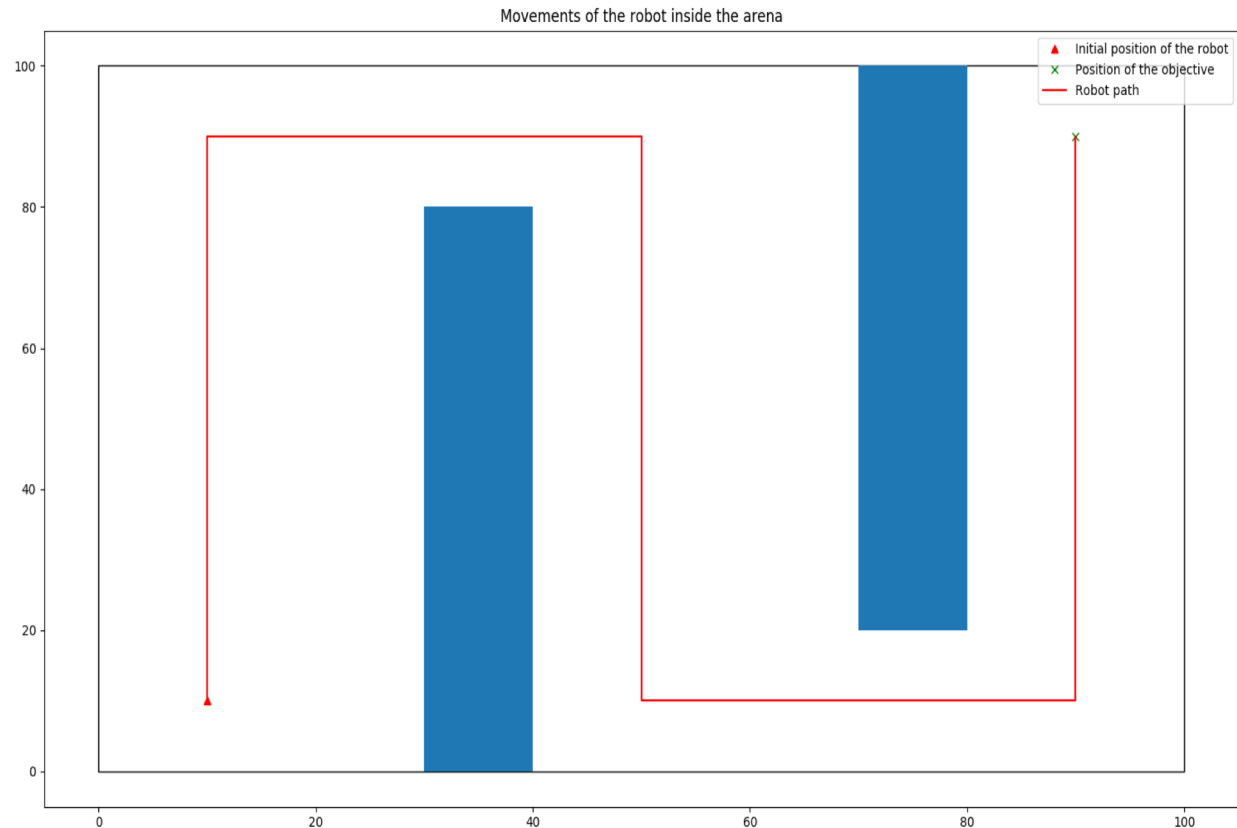
# Optimisation de réseaux de neurones

- **Prérequis : une bonne connaissance en apprentissage de réseaux de neurones artificiel (Keras ou PYTorch)**
- **Jeu de donnée : OpenML.org**
- **Apprendre les poids et la structure d'un réseau de neurones par optimisation évolutionnaire.**
- **Réduire au maximum la complexité du réseau tout en gardant un niveau de prédiction satisfaisant (multi-objectifs ?).**



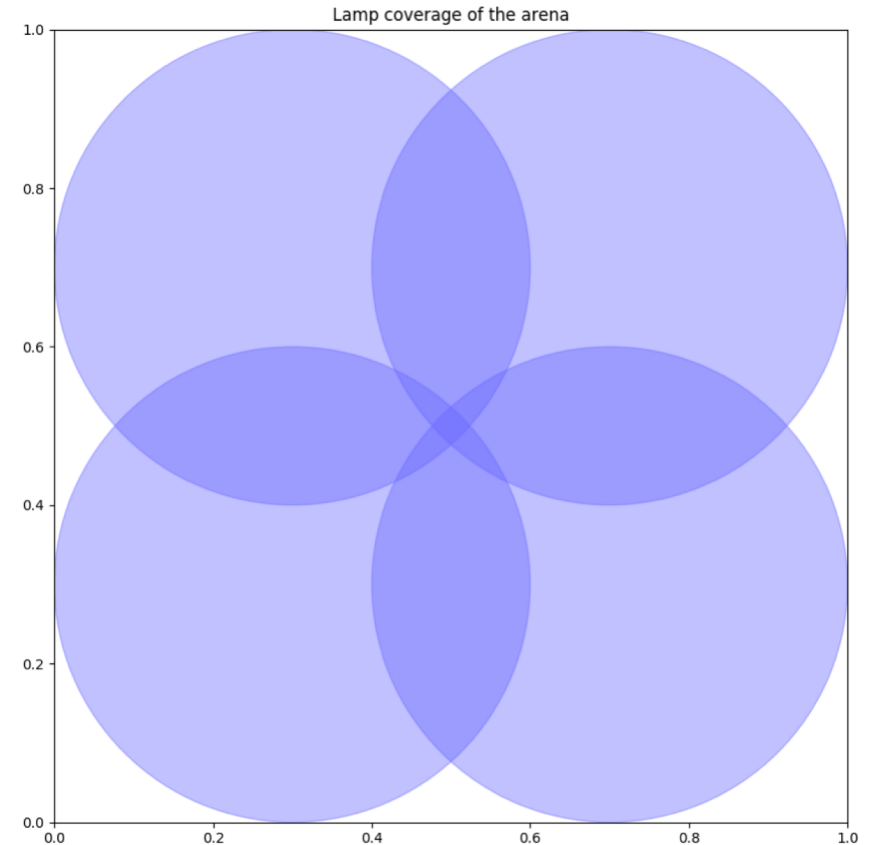
# Labyrinthe

- A l'aide d'inspyred, écrire un algo GP pour aider un "robot" à trouver son chemin dans un labyrinthe
- Le robot va recevoir une liste de commandes ("move 80", "rotate 90", "move 20", ...) et la fitness évalue la distance de l'objectif.
  - Le robot ne peut pas passer à travers les murs.
  - La fonction dans **arenabot.py** calcule la fitness pour une liste de commandes donnée.



# Co-evolution coopérative

- Avec un cas d'étude "lamps", utiliser inspyred (ou autre) pour créer un "cooperative co-evolutionary algorithm"
- Chaque individu est un lampe (un cercle), et l'objectif global c'est de couvrir le carré avec le minimum de lampes. La fitness de chaque lampe est son recouvrement du carré.



# Co-évolution compétitive : prédateur/proie

- Point de départ = modèle à base d'agents wolf/sheep :  
[https://github.com/projectmesa/mesa-examples/tree/main/examples/wolf\\_sheep](https://github.com/projectmesa/mesa-examples/tree/main/examples/wolf_sheep)
- Deux populations : les loups et les moutons.
- Génomes = caractéristiques individuelles des agents (loups ou moutons).
- Evaluation = chaque individu a un niveau d'énergie qui correspond à sa fitness.
- Sélection / Croisement / mutation évolutionnaire pour chaque population (à programmer !).

# Exploring Hawk-Dove games

<https://www.nature.com/articles/s41598-017-04284-6#:~:text=While%20hawks%20are%20willing%20to,c%20H%20%2C%20while%20doves%20retreat.>

Article | [Open access](#) | Published: 06 July 2017

## Evolutionary dynamics of N-person Hawk-Dove games

[Wei Chen](#), [Carlos Gracia-Lázaro](#), [Zhiwu Li](#) , [Long Wang](#) & [Yamir Moreno](#)

[Scientific Reports](#) **7**, Article number: 4800 (2017) | [Cite this article](#)

**7715** Accesses | **20** Citations | **22** Altmetric | [Metrics](#)

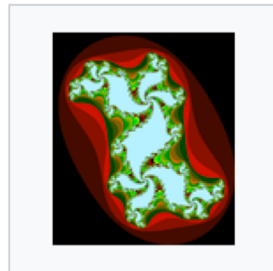
Niveau de difficulté



# Exploration interactive d'images fractales : ensembles de Julia.

- **Ensembles de Julia :**

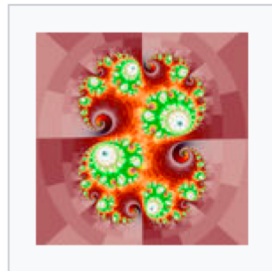
Comportement de la suite  $z_{n+1} = z_n^2 + c$  dans le plan complexe.



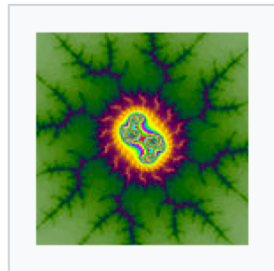
Ensemble de Julia de  $z^2 + c$  avec  $c = 0.3 + 0.5i$ .



Ensemble de Julia pour  $c = 0.285 + 0.01i$  (courbes de niveau du nombre d'itérations).



Le même ensemble avec une coloration différente (nombre d'itérations et [pavage](#) selon le secteur d'arrivée).



Détail de l'ensemble de Julia pour  $c = -1.417022285618$ .



Détail de l'ensemble de Julia pour  $c = -0.038088 + 0.975i$ .



Ensemble de Julia pour  $c = 0.285 + 0.013i$ .

- **Génome :  $c = x + iy + \text{table de couleurs} + ?$**
- **Evaluation interactive par un slider.**
- **Présentation d'une population, vote et itération suivante.**