

Algorithmes multi-objectifs : MOPSO et Micro-GA

Esmé James & Wilfried Pouchous



SOMMAIRE

- I - Explication MOPSO**
- II - Explication Micro-GA**
- III - Résultats**
- IV - Comparaisons**
- V - Conclusion**
- VI - Démo**

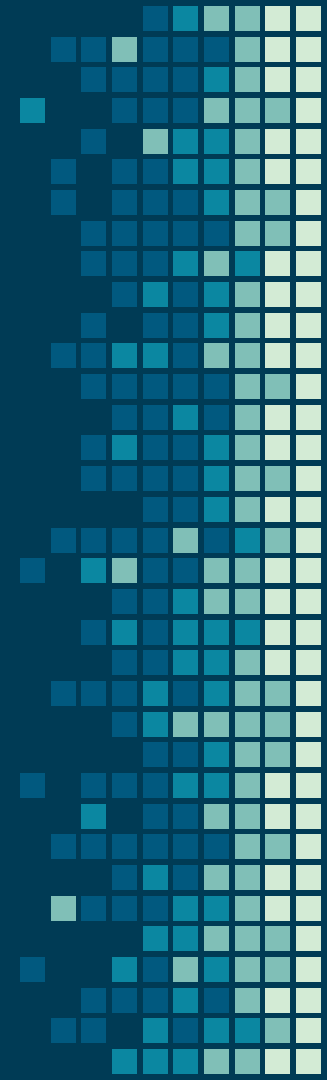


1.

MOPSO

Multi-objective Particle Swarm Optimisation

Coello et Pulido

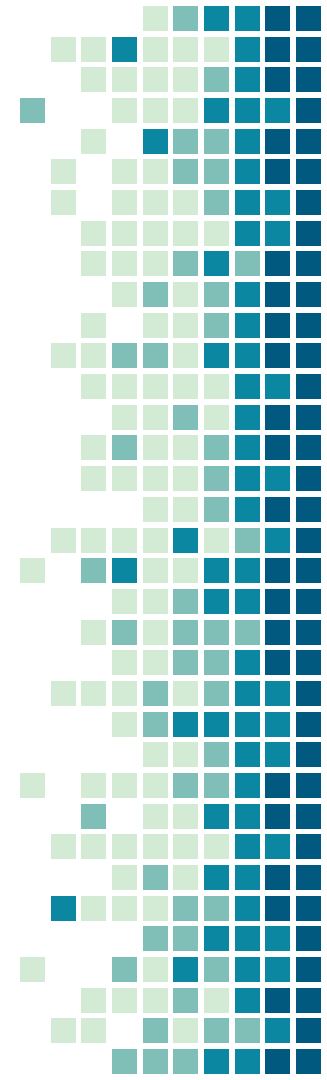


Contexte de la recherche

Problèmes d'**efficacité** des algorithmes multi-objectifs généralement liés au temps d'exécution

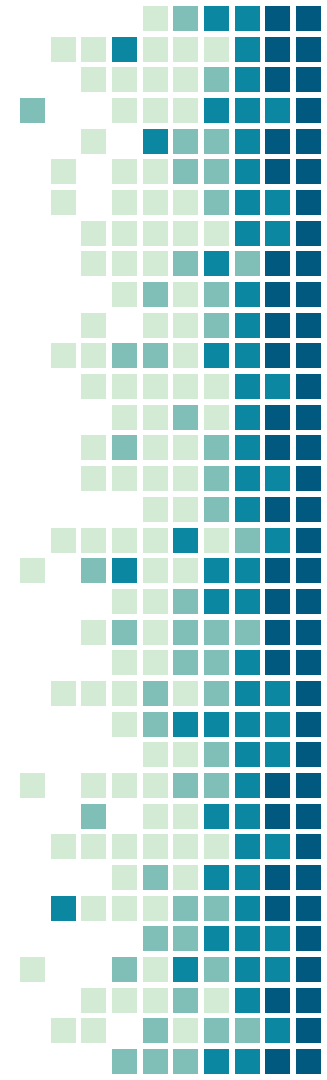
2 responsables à cela :

- Ranger les populations
- Mécanisme pour maintenir la diversité



But de l'algorithme

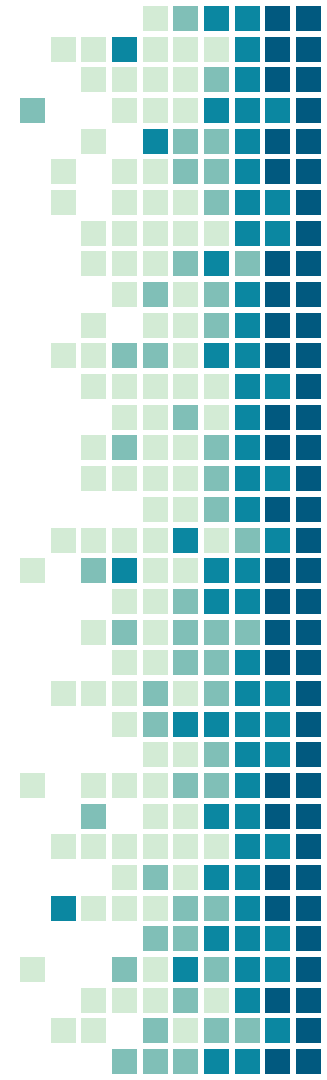
*“ Proposer une **extension** de l'algorithme **PSO** afin de lui permettre de résoudre des **problèmes multi-objectifs**. ”*



PSO : Particle Swarm Optimisation

Basé sur simulation du déplacement d'oiseaux

- Population de particules qui se déplacent selon des règles simples
 - Mouvement influencé par les meilleures positions locales connues et guidé vers les meilleures positions connues
- ⇒ Déplacement de l'essaim vers solutions optimales



MOPSO : les bases

- Un essaim de particules **Po** avec
 - Position
 - Vitesse
 - Mémoire
- Une mémoire externe (Repository) **Rep**
 - Stockage des vecteurs non-dominés

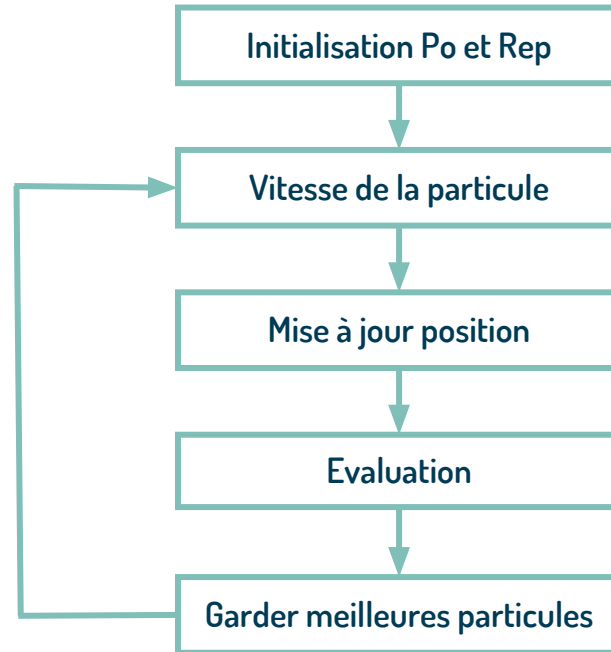


MOPSO : Étapes

- 1) Initialisation de la population **Po**
- 2) Stockage des particules non-dominées dans l'archive **Rep**
- 3) Générer hypercubes et localiser les particules
- 4) Boucle jusqu'à **Gmax**
 - a) Calculer la vitesse de chaque particule :
$$V = W*V + R1*(Pbest - Po) + R2*(RepH - Po)$$
 - b) Mise à jour de la position des particules en fonction de leur vitesse
 - c) Évaluer les particules
 - d) Garder les nouvelles meilleurs particules non-dominées dans **Rep**
 - e) Si **Rep** est plein, enlever des particules des zones peuplées des hypercubes



MOPSO : Flowchart



2.

Multi-objective MicroGA

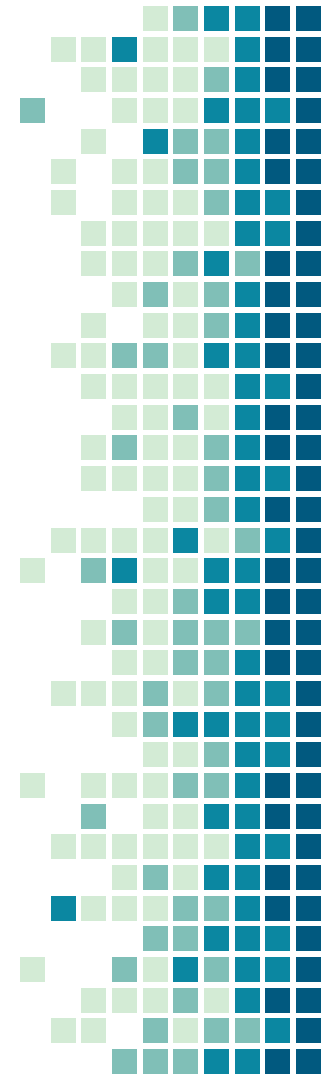
Coello et Pulido

Contexte de la recherche

Même contexte cité précédemment

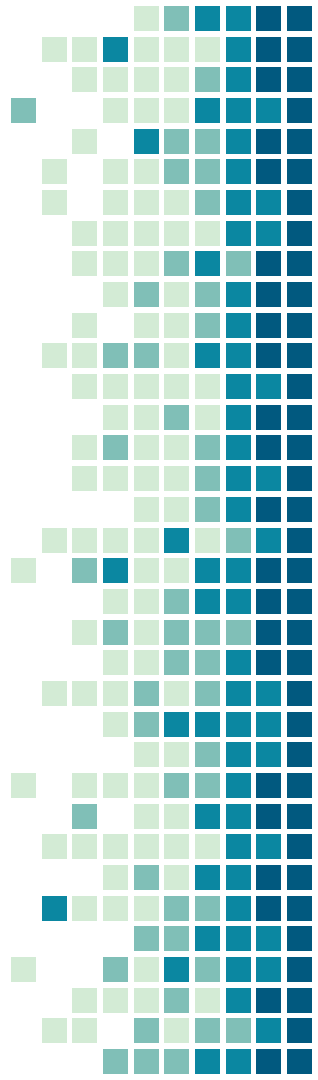
Solution : fichier externe pour stocker les individus non-dominés (archive)

⇒ **Micro-GA**



But de l'algorithme

*“ Montrer qu’un algorithme **Micro-GA** bien construit et bien paramétré est suffisant pour générer le front Pareto d’un **problème multi-objectifs**. ”*



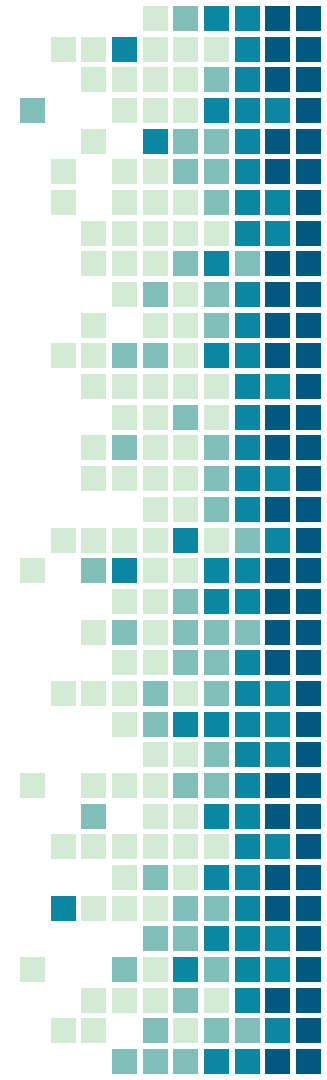
Qu'est-ce que Micro-GA ?

GA : Genetic Algorithm

Micro :

- Très petite population (**4** dans l'article)
- Processus de réinitialisation (après la **convergence nominale**)

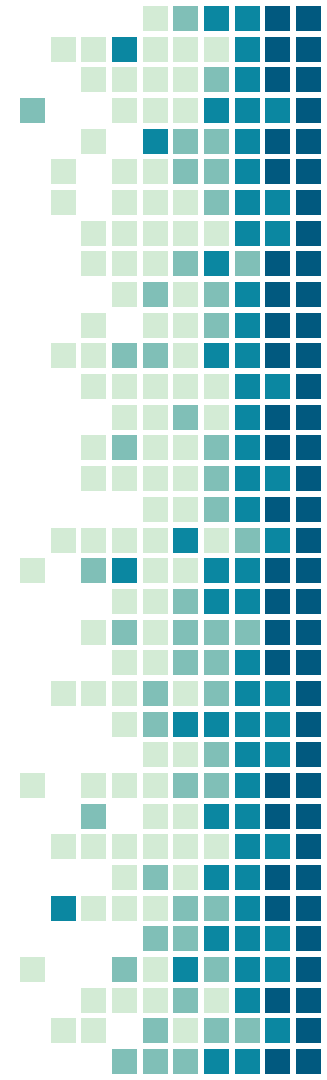
Convergence nominale : État où les tous les individus sont identiques ou très proches



Micro-GA : les bases

Deux mémoires

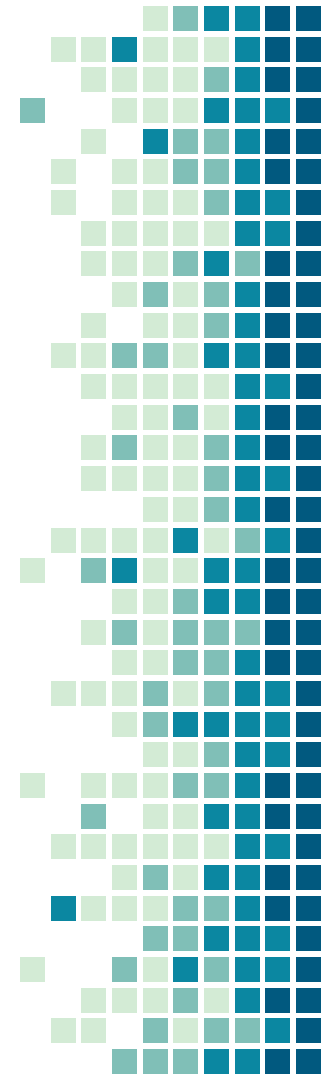
- Mémoire externe **E** : Archive les membres du front de Pareto
- Mémoire population **M**
 - Partie remplaçable **Mr**
 - Partie non-remplaçable **Mn**



Micro-GA : 3 types d'élitismes

- Conservation d'un individu non-dominé à chaque fin de cycle de Micro-GA
- Remplacement d'individus de la population remplaçable venant de la population Micro-GA
- Remplacement d'individus de la population remplaçable venant de la mémoire externe E (**cycle de remplacement**)

⇒ Principe de la sélection naturelle



Micro-GA : Étapes

- 1) Génération d'une population aléatoire **M** (composée de **Mr** et **Mn**)
- 2) Début de la boucle de Micro-GA (jusqu'à **Gmax**)
 - a) Création de la population initiale **Pi** à partir de **Mr** et **Mn**
 - b) Boucle jusqu'à la **convergence nominale**
 - (1) Opérations de sélection, croisements et mutations
 - (2) Sauvegarde d'un individu non-dominé (**élitisme n° 1**)
 - c) Sélection de 2 individus non-dominés de **Pi**
 - d) Comparaison des 2 individus avec la mémoire externe **E** et ajout dans **E** s'ils sont toujours non-dominés (si **E** plein, remplacer 2 individus encombrés de **E** par les nouveaux individus via la **Crowding Distance**)



Micro-GA : Étapes

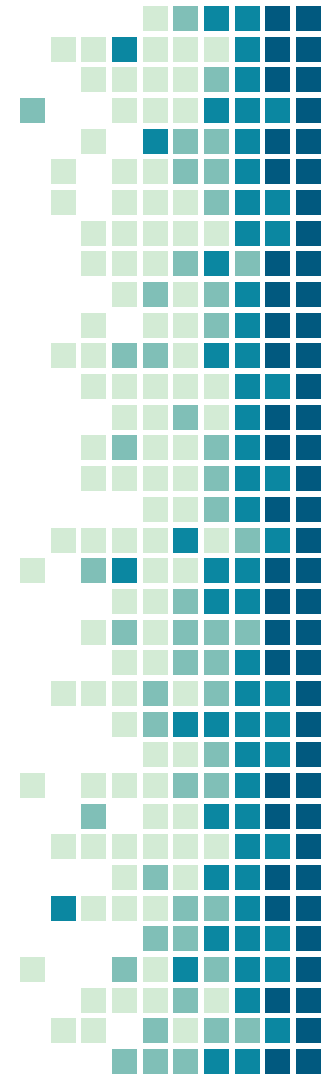
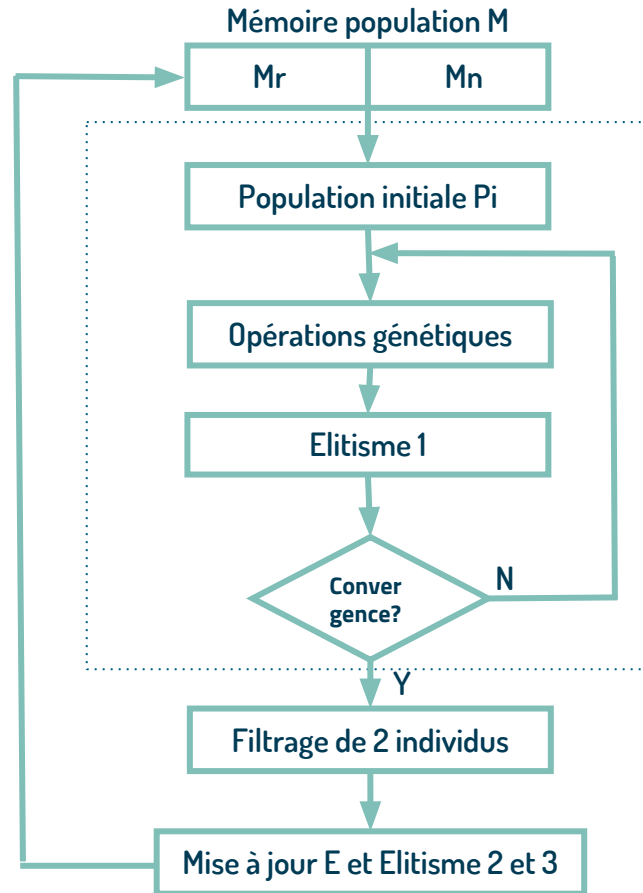
- e) Comparaison des 2 individus avec 2 individus aléatoires de **Mr** de la mémoire population **M** et remplacement s'ils sont toujours non-dominés (élitisme n°2)

- f) Remplacement des individus de **Mr** par des individus de **E** à intervalle régulier, appelé cycle de remplacement (élitisme n°3)



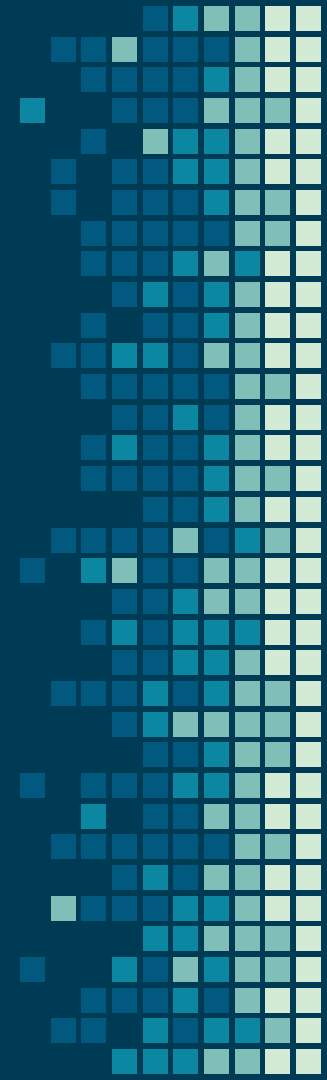
Micro-GA : Flowchart

Cycle Micro - GA



3.

Résultats



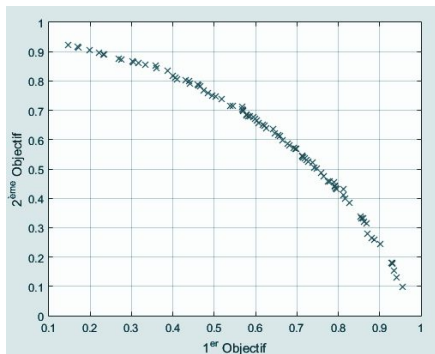
Fonctions de tests

- Fonseca & Fleming **FON**
- Poloni **POL**
- Kursawe **KUR**
- Zitzler–Deb–Thiele 1 **ZDT1**
- Zitzler–Deb–Thiele 2 **ZDT2**
- Zitzler–Deb–Thiele 3 **ZDT3**

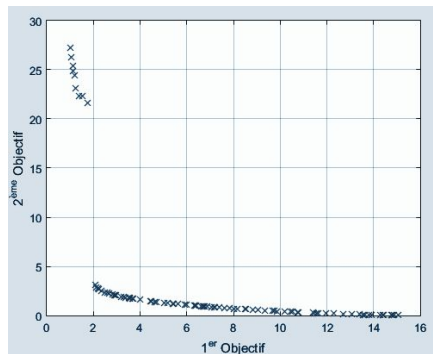


Résultats MOPSO

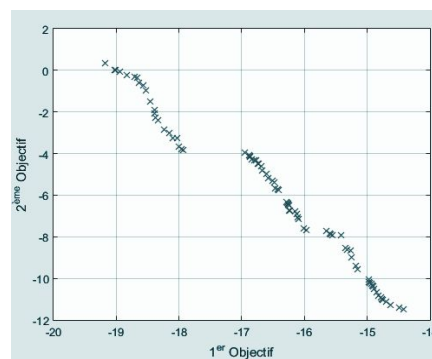
FON



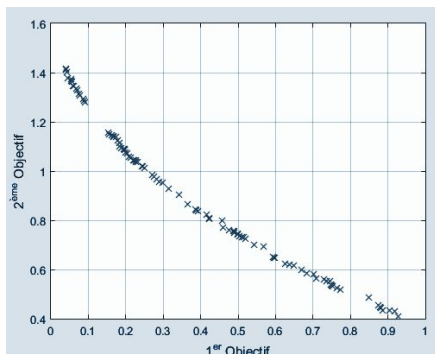
POL



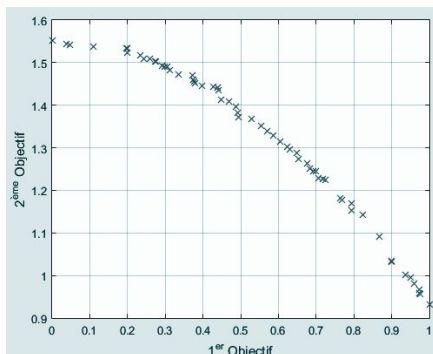
KUR



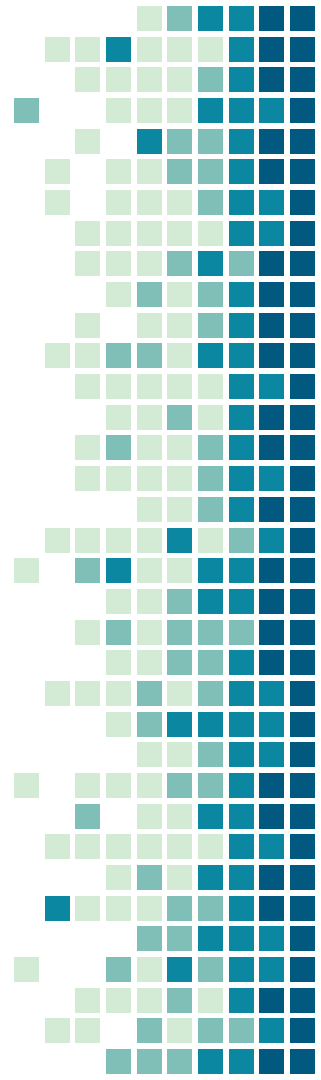
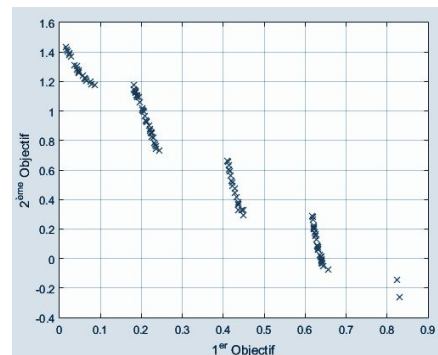
ZDT1



ZDT2



ZDT3



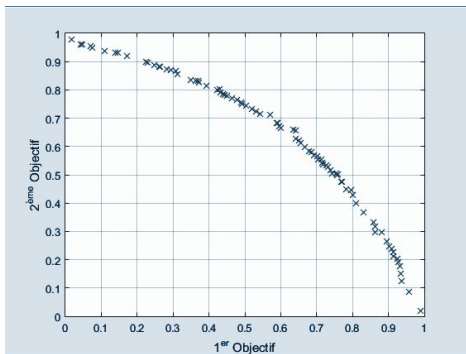
Résultats MOPSO : Remarques

- Convergence rapide dans la plupart des cas
- Utilise peu de ressources

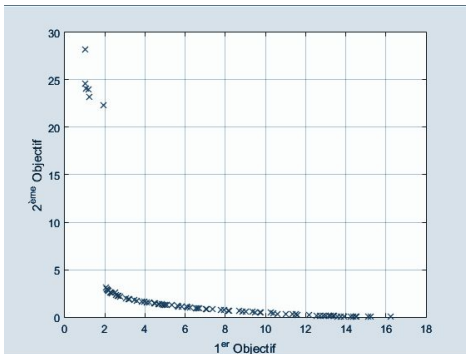


Résultats Micro-GA

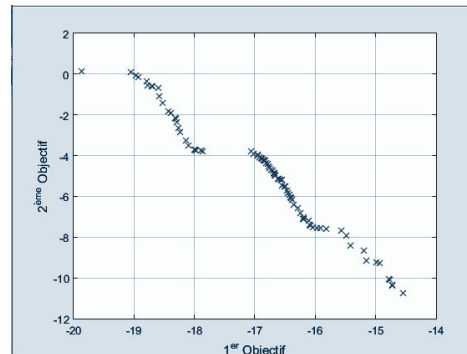
FON



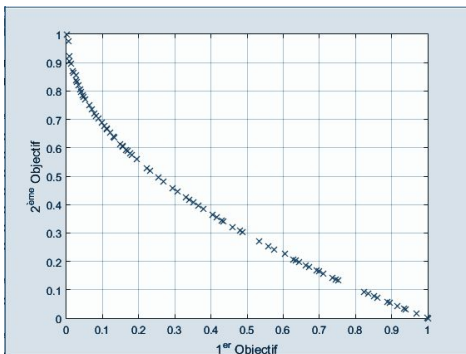
POL



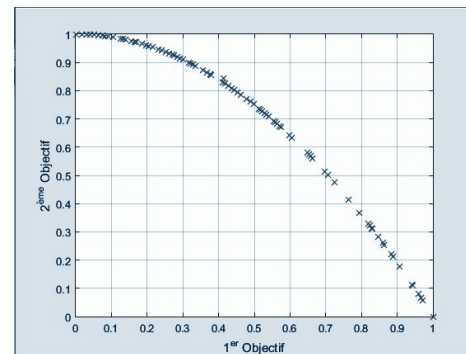
KUR



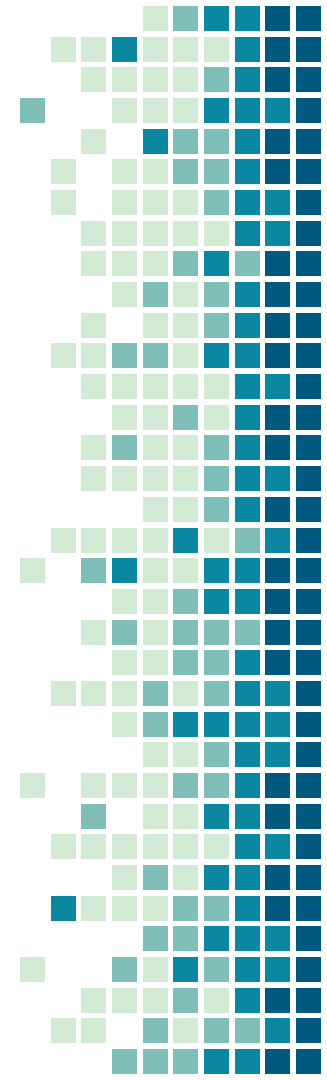
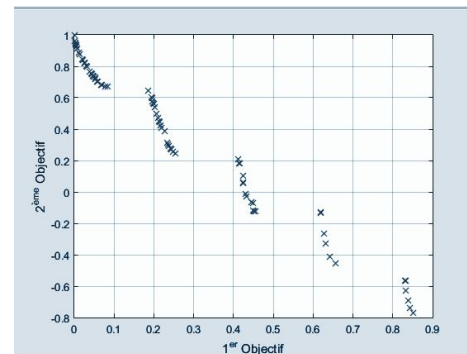
ZDT1



ZDT2



ZDT3



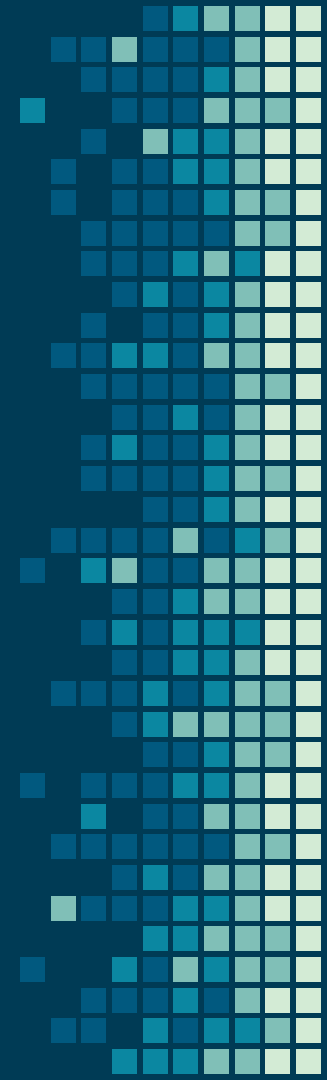
Résultats Micro-GA : Remarques

- Convergence rapide dans la plupart des cas
- Utilise peu de ressources



4.

Comparaison



MOPSO vs. Micro-GA

	MOPSO (temps en s)	Micro-GA (temps en s)
FON	10.56	10.80
POL	6.18	6.55
KUR	18.34	31.26
ZDT1	14.29	9.09
ZDT2	5.68	8.99
ZDT3	7.48	16.94



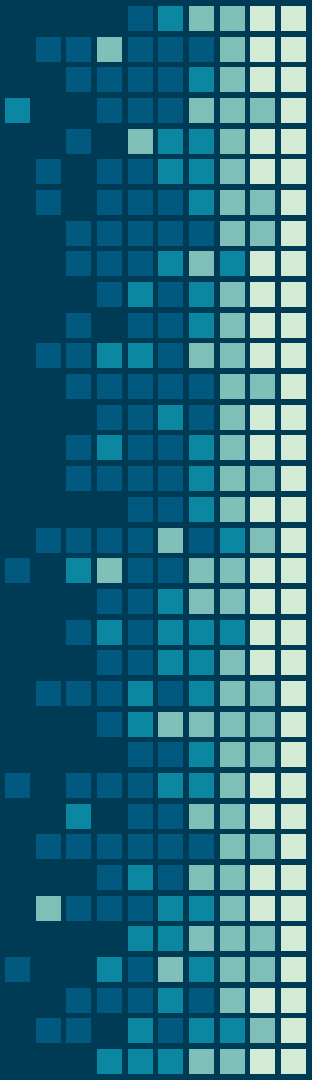
“

MOPSO et Micro - GA sont plus rapides que :

- *NSGA-II*
- *PAES*

5.

Conclusion



Conclusion

- Deux algorithmes compétitifs
- MOPSO plus rapide
- Micro-GA plus simple



MERCI

Des questions ?