Mini-Projet de E-Commerce

Approche méta-heuristique pour le problème de détermination du gagnant dans les enchères combinatoires

ABDELALI Asma Nihad
M1 SII G1, 171731052943

Gray Wolf Optimizer pour le problème de détermination du gagnant dans les enchères combinatoires.

Problème des enchères combinatoires

Le problème est défini par la soumission d'un ensemble d'objets aux enchères et que plusieurs individus cherchent à acheter un sous-ensemble d'objets à un certain prix.
Le problème parvient lors que plusieurs sous ensembles contiennent le même objet.
Le but est de maximiser le gain, soit la somme des valeurs des offres acceptées.

Exemple :

Nous disposons de N Items et M Offres

I= {1, 2 … N}

B = {B1, B2 … Bm}

Chaque offre Bi = (V O1 O2 … Ox) où V est la valeur de l'ensemble d'objets O1 à Ox.

Le but est de maximiser la somme des V sous la contrainte que chaque item Li ce trouve dans une seule offre au plus.

La fonction de fitnesse est alors définie comme suit :

$$F(V) = \sum_{i=1}^{L} Price(B_i) = \sum_{i=1}^{L} P_i$$

Gray Wolf Optimizer

Une méta-heuristique développée par Seyedali Mirjalili.

Hiérarchie d'une meute de loups

Les loups vivent en meutes, et s'ordonnent dans une hiérarchie stricte. à la tête s'y trouve l'alpha, suivi par les loups beta, delta puis les oméga respectivement

Stratégie de chasse d'une meute de loups

Lorsque une meute de loups décide de chasser une proie chaque individu la garde en vue, et ajuste sa propre position par rapport à la position de l'alpha et des loups bêta et delta.
Lorsque tous les loups encerclent la proie, c'est la fin de la traque et le début de l'attaque.

Modéliation mathématique des problèmes.

- **L'espace de recherche** c'est l'ensemble de toutes les enchères
- Une **solution** est un ensemble d'enchères et aussi la proie recherchée par les loups.
- La meute de loups représente un **ensemble de solutions**
- **L**es trois meilleures solutions sont respectivement l'alpha, le bêta et delta, le reste des loups seront des oméga.
- La **fonction de fitness** est la maximisation de la somme des gains des enchères du problème.
- Dans le monde réel les loups se repèrent à leurs sens cependant il n'est pas possible de voir la **solution (proie)** dans un monde mathématique, nous assumons alors que l'alpha, le bêta et le delta ont la meilleure connaissance de la position de la proie.

The Random Key Encoding

Le Random Key Encoding consiste en un ensemble de réels qui représentent l'ordre dans lequel les offres vont êtres considérées.
Par exemple pour une ensemble d'offres B1, B2, B3 et un r = {0.6, 0.85, 0.23}.
On va considérer les Bids selon l'ordre suivant B2, B1, B3.

Le RKE nous aide à définir nos solutions. (Individus de la meute)

Mise à jour de la position

$$\vec{D} = |\vec{C}.\overrightarrow{X_p}(t) - \vec{X}(t)|$$

$$\vec{X}(t+1) = \overrightarrow{X_p}(t) - \vec{A}.\vec{D}$$

A et C sont des vecteurs de coefficients, Xp(t) est la position de la proie à l'instant t et X(t) indique la position du loup actuel à l'instant t.

$$\vec{A} = 2\vec{a}.\overrightarrow{r_1} - \vec{a}$$

(valeurs entre −a et a) Ce paramètre oblige le loup de se rapprocher ou de s'éloigner de la solution.

$$\vec{C} = 2.\overrightarrow{r_2}$$

(valeurs entre 0 et 2) C>1 plus d'exploration, C<1 plus d'exploitation.

Où **a** est initialement égal à 2 et décrémenté au long des itérations, **r1** et **r2** sont des nombres au hasard, ils permettent au loup de se repositionner.

D'ou on obtient les formules suivantes :

$$\overrightarrow{D_\alpha} = |\vec{C_1}.\overrightarrow{X_\alpha} - \vec{X}|, \overrightarrow{D_\beta} = |\vec{C_2}.\overrightarrow{X_\beta} - \vec{X}|, \overrightarrow{D_\delta} = |\overrightarrow{C_3}.\overrightarrow{X_\delta} - \vec{X}|$$

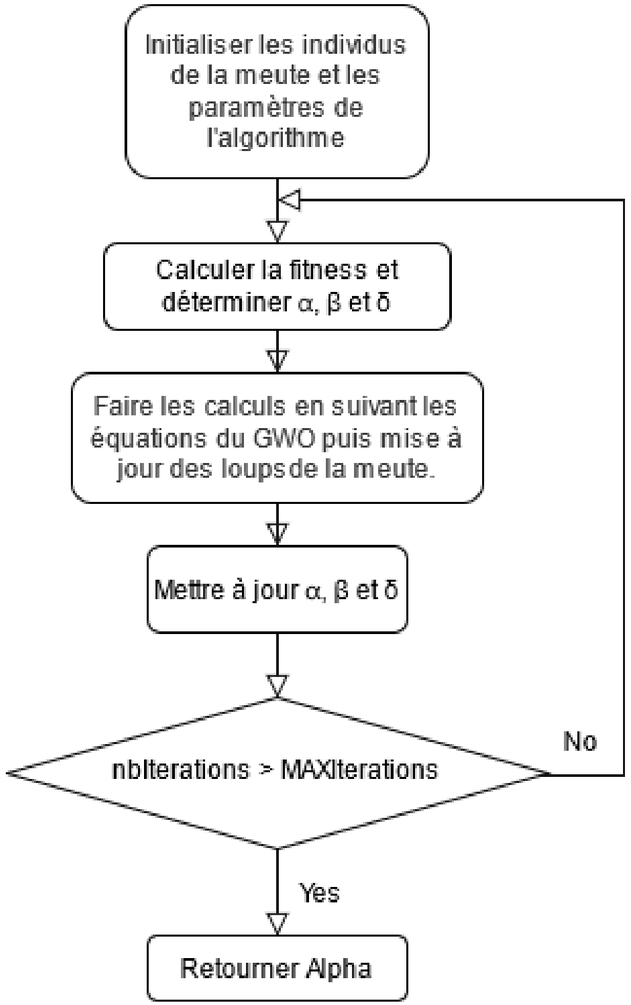
$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \vec{A_1}.(\overrightarrow{D_\alpha}), \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2}.(\overrightarrow{D_\beta}), \overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3}.(\overrightarrow{D_\delta})$$

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3}$$

Où X(t+1) est la position du loup à l'instant t+1

Ces équations nous permettent de se positionner dans un espace à trois dimensions, dans le monde réel ces **repositionnements hasardeux** sont dus au terrain et aux obstacles.
Par contre dans un monde abstrait on ne peut pas faire ces multiplications sur un ensemble d'enchères, nous utilisons la position afin de **modifier une partie de la solution trouvée**.

Algorithme développé



- **Initialisations**

  Initialiser les individus de la meute et les paramètres de l'algorithme en respectant les conflits entre les enchères.

- **Calcul de la position**

  Faire les calculs en suivant les équations du GWO puis mise à jour des solutions des Omega.

- **Nouveau départ**

  Si les solutions trouvées sont pires que les précédentes en générer de nouvelles puis mettre à jour la hiérarchie.

Pseudo Algorithme

Gray Wolf Optimization Pseudo-Code

Entrée: Taille de la meute, maxIterations : Entier;

Sortie: Ensemble d'enchères.

Debut

```
wolfPack.Initialiser(Taille de la meute); // En utilisant le Random Key Encoding
Alpha = Collections.Max(wolfPack);          wolfPack.remove(Alpha);
Beta = Collections.Max(wolfPack);           wolfPack.remove(Beta);
Delta= Collections.Max(wolfPack);           wolfPack.remove(Delta);
Initialize a, r1, r2, A, C;   //En utilisant les équations vues précédemment
While(iter<maxIteration){
    for (wolf: wolfPack) {
        wolf.calculerPosition();
        wolf.majSolution();
    }
    a = 2 - iter* ((2.0) / maxIterations);
    Update r1, r2, A, C;
    Update Alpha, Beta, Delta; // Remettre Alpha, Bêta, Delta dans la meute et récupérer les nouveau 3 meilleures solutions
    iter ++;
}
return Alpha;
```

Fin;

Pseudo code de la mise à jour de la Solution

Procédure UpdateSolution()

Var : i, cpt, encheresAremplacer : Entier

Début

```
Wolf w = this;
encheresAremplacer_ = abs(position) % w.nombreEncheres; // Le nombre d'enchères dans cette solution.
for (int i=d; d<w.nombreEncheres; d++){// vider une partie des enchères et actualiser les Conflicts
        w.Encheres.set(i, null);
        w.actualiserListeConflicts();
}

for(i =0, cpt=0; j<nombreEncheresTotal cpt<encheresAremplacer  ; j++){
        if( ! EncheresTotales.get(i).estEnConflict(this.Encheres){
                this.AjouterEnchère( EncheresTotales.get(i))  // En admettant la fonction Ajouter met à jour le total des gains et  la liste des conflits
                cpt ++;
        }
}
if (this.getGain() < w.getGain())// Si la nouvelle solution est meilleure alors la prendre
          this=w;
    else //Sinon générer une nouvelle solution
          this.générerRandom()
```

Fin.

Programme
Développé et résultats expérimentaux

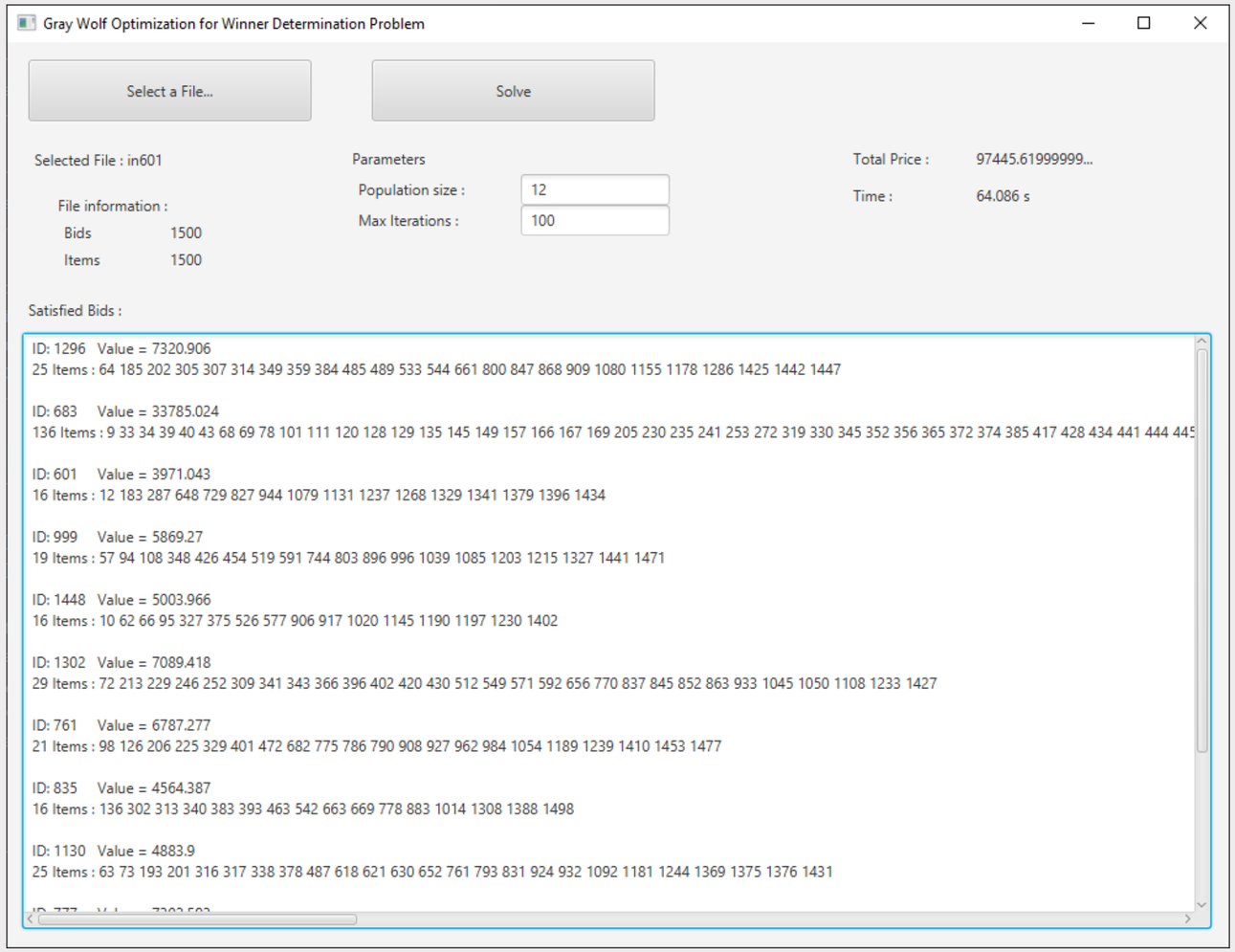
Interface Graphique

Le programme développé nous permet :

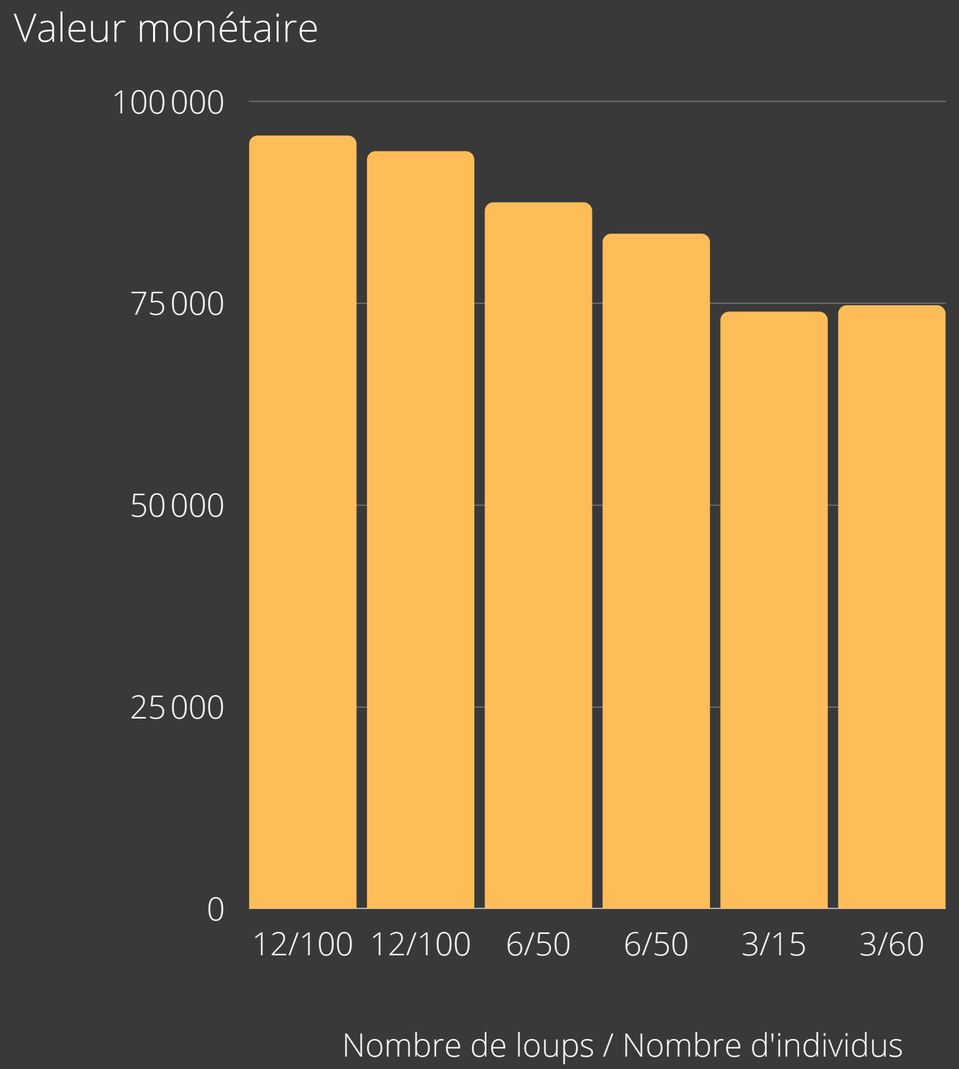
- De choisir un fichier à résoudre.
- A préciser les paramètres.

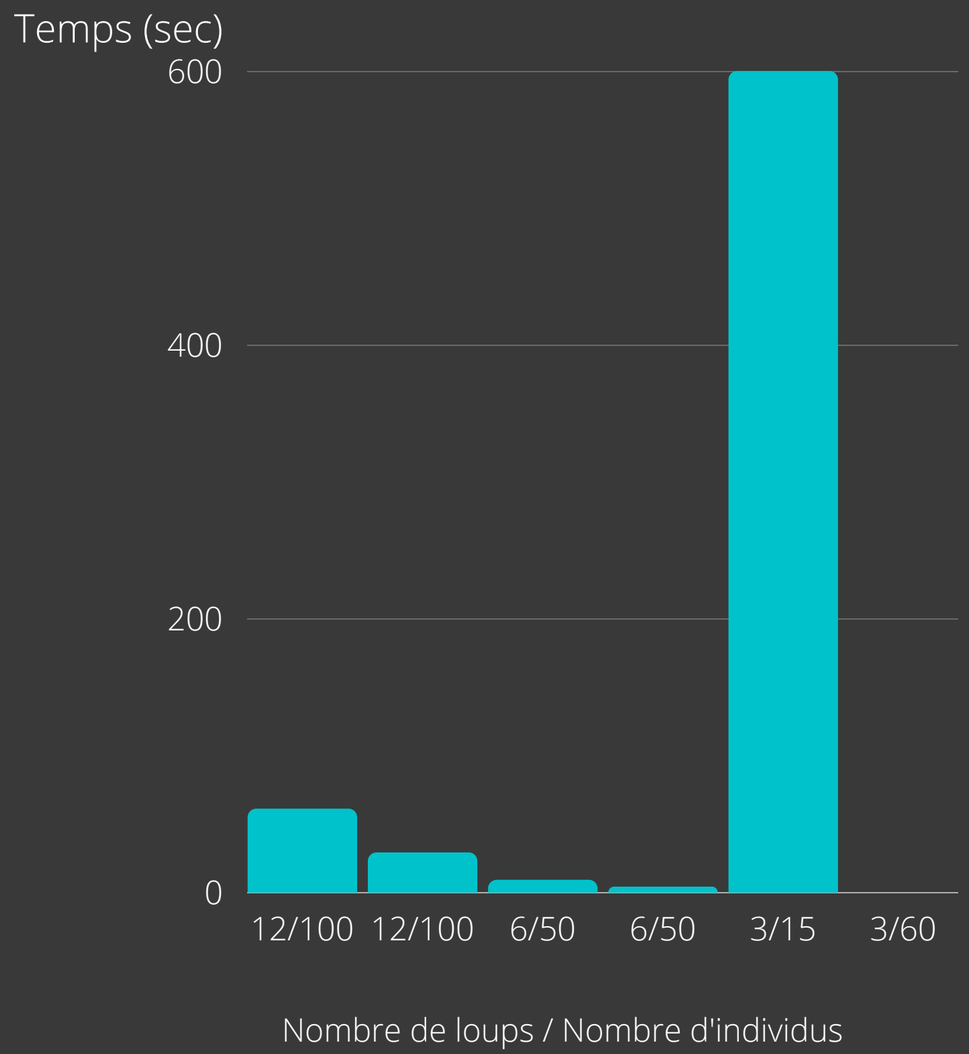
Après résolution de l'instance on affiche :

- Le prix total des enchères satisfaites
- Le temps d'exécution.
- L'id de chaque enchère satisfaite, sa valeur et les items qu'elle contient.

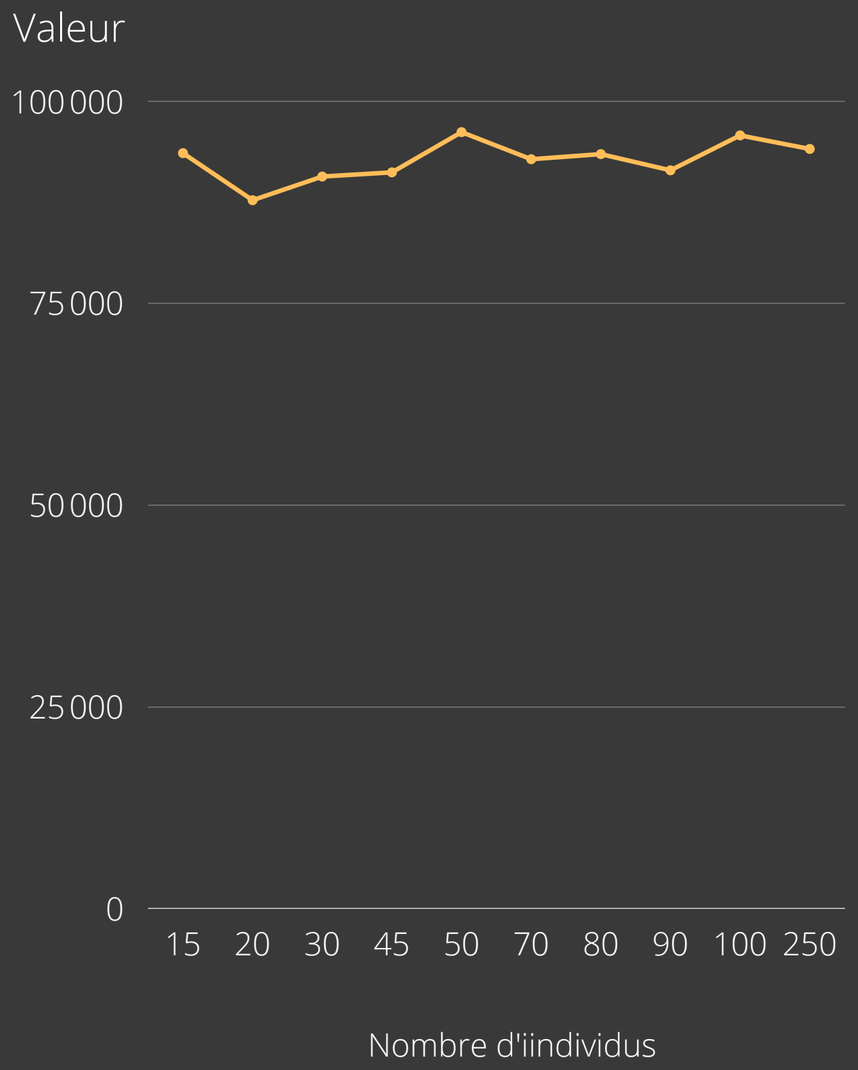


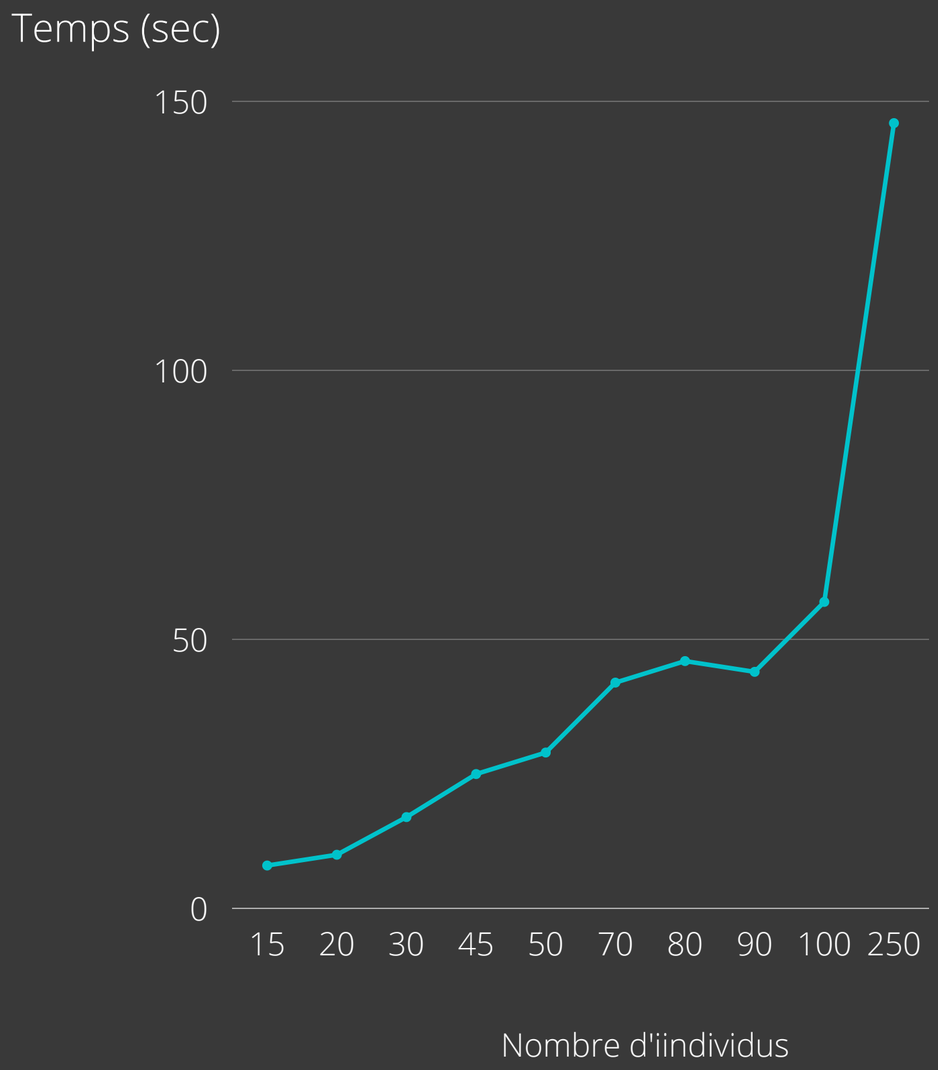
Résultats Expérimentaux sur des benchmarks



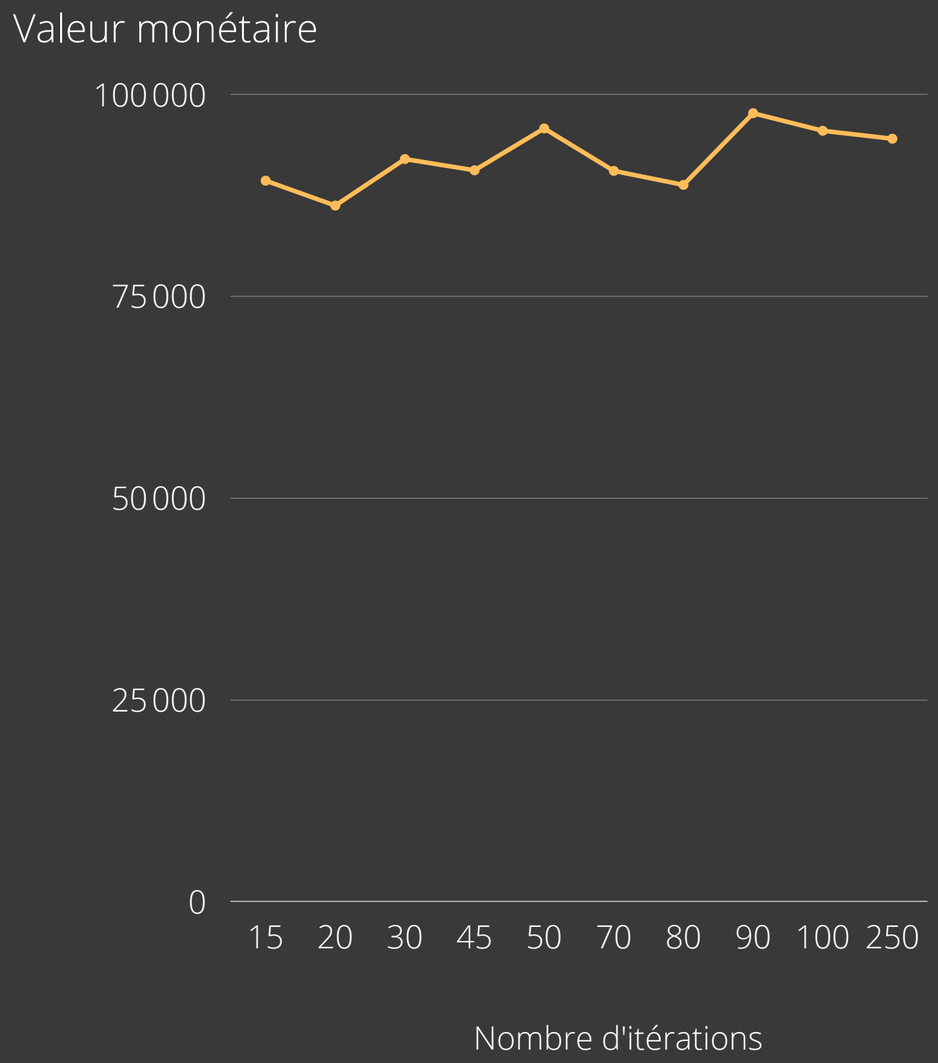


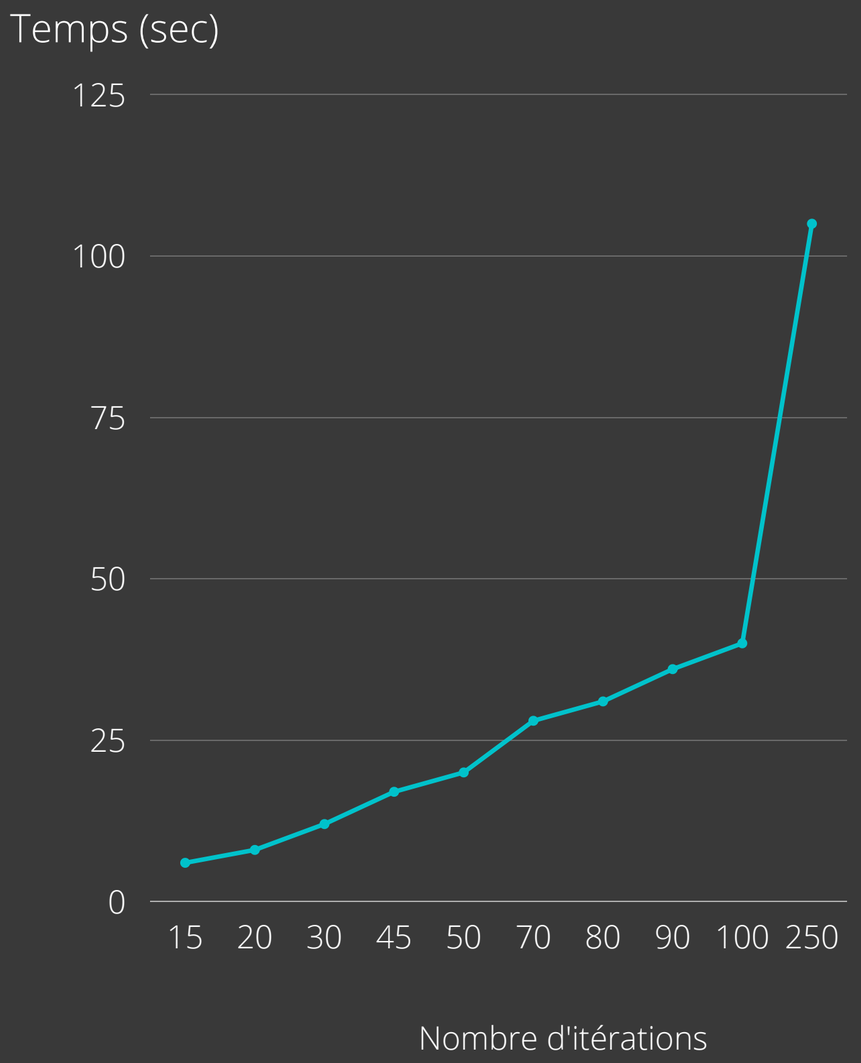
Représentation du gain des enchères et du temps d'exécution lors des variations du nombre d'individus dans la meute





Représentation du gain des enchères et du temps d'exécution lors des variations du nombre d'itérations





Conclusion

Ce qu'on peut améliorer?

Références

- S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey Wolf Optimizer, Advances in Engineering Software , vol. 69, pp.
- Dalila Boughaci - Metaheuristic Approaches for the Winner Determination Problem in Combinatorial Auction
- Jie-sheng Wang & Shu-Xia Li - An Improved Grey Wolf optimizer Based on Differential Evolution and elimination Mechanism