# Université des Sciences et de la Technologie HOUARI BOUMEDIENE Département d'informatique



TP

META-HEURISTIQUE

# « PROJET DE REALISATION

# DU TAQUIN (3X3) (PARTIE 2) »

***OBJECTIFS***

Dans ce Projet, on va continuer la partie 1 de réalisation du programme pour résoudre le jeu de taquin de (3X3), mais avec une nouvelle approche, celle de l’intelligence en Essaim et les Algorithmes Evolutionnaires

* La résolution du problème se fait par 3 approches différentes :
  1. L’algorithme GA (Genetic Algorithm)
  2. L’algorithme PSO (Particle Swarm Optimization)
  3. L’algorithme BSO (Bee Swarm Optimization)
* Les outils utilisés pour réaliser ce projet sont :
  1. Langage de programmation Java
  2. L’interface graphique Java-Swing
* References

***Introduction***

En intelligence artificielle, la résolution de problèmes par la recherche comprend la formulation d'un problème comme une question de recherche, puis construire sur cette formule un arbre de recherche qui sera parcouru pour trouver la cible. Ce processus de parcours utilise des techniques de recherche qui peuvent être divisées en deux grandes catégories.

Dans la première partie du Projet on a implémenter les techniques de recherche **non** **informées** tels que le **Breath First Search** ou le **Depth First Search** et techniques de recherche **informées** conçu à l’aide d’heuristiques tels que l’algorithme de recherche **A\*.**

Dans cette deuxième partie, on vise à élaborer la notion de **« Meta Heuristique ».** Contrairement aux méthodes **heuristiques**, Les **méta heuristiques** ne sont pas conçues pour un problème donné. Elles sont plutôt générales, adaptables et applicables à plusieurs problèmes.  
Tout comme les méthodes heuristiques, elles permettent d’obtenir une solution réalisable dont le coût est proche de la solution en un temps raisonnable.

Pour cela on va étudier dans cette partie l’un des méthodes **méta heuristiques** les **plus populaires** notamment (Genetic Algorithm **(GA),**Particle Swarm Optimization Algorithm **(PSO),** et Bee Swarm Optimization Algorithm **(BSO)**)

**Définition d’une métaheuristique:**

Une métaheuristique est un algorithme d’optimisation visant à résoudre des problèmes d’optimisation difficiles pour lesquels on ne connaît pas de méthode classique plus efficace. Les métaheuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global. Elles se comportent comme des algorithmes de recherche, tentant d’apprendre les caractéristiques d’un problème afin d’en trouver une approximation de la meilleure solution. Il existe un grand nombre de métaheuristiques différentes, allant de la simple Recherche Locale à des algorithmes complexes de recherche globale. Ces méthodes utilisent cependant un haut niveau d’abstraction, leur permettant d’être adaptées à une large gamme de problèmes différents. En d’autres termes, ces algorithmes se veulent des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changements profonds dans l’algorithme employé.

Il n’existe pas de définition qui fasse l’unanimité, mais tous s’accordent sur les points suivants :

• Les métaheuristiques sont des stratégies permettant de guider la recherche d’une solution optimale.

• Le but visé par les métaheuristiques est d’explorer l’espace de recherche efficacement afin de déterminer des solutions optimales.

• Les techniques qui constituent des algorithmes de type métaheuristique vont de la simple procédure de Recherche Locale à des processus d’apprentissage complexes.

• Les métaheuristiques peuvent contenir des mécanismes qui permettent d’éviter d’être bloqué dans des régions de l’espace de recherche.

• Les concepts de base des métaheuristiques peuvent être décrit de manière abstraite, sans faire appel à un problème spécifique.

• Les métaheuristiques peuvent faire appel à des heuristiques qui tiennent compte de la spécificité du problème traité, mais ces heuristiques sont contrôlées par une stratégie de niveau supérieur.

• Les métaheuristiques peuvent faire usage de l’expérience accumulée durant la recherche de l’optimum, pour mieux guider la suite du processus de recherche.

# Résolution avec L’algorithme GA

# (Genetic Algorithm) :

## Définition du l’algorithme GA :

L’algorithme génétique **(Genetic Algorithm )** ou **« GA »** est un cas particulier des **algorithmes évolutifs** qui débute avec **une population initiale** souvent générée de **manière aléatoire** et répète un **cycle d’évolution**.

L’algorithme génétique est inspirée de la **génétique** et de la **théorie de sélection naturelle** citée par **Charles Darwin au 19ème siècle** et développer par **John Holland** de l’université de Michigan. Les GA introduits par Holland s’appuient fortement sur le codage universel sous forme de chaîne 0/1 de longueur fixe. Un individu dans ce codage est appelé **chromosome.**

## Principe de l’algorithme :

L’idée de base des algorithmes génétiques est de simuler un **processus évolutionnaire** afin d’obtenir de meilleurs **individus (solutions),** les étapes de **l’algorithme GA** sont :

**1) L’initialisation :**

Un ensemble de solutions est initialisé au départ au **hasard** et cet ensemble sera modifié à chaque itération par l’aspect de **(la sélection naturelle).**

**2) La sélection :**  
 Sélectionner un sous ensemble d’individus **« parents »** pour débuter l’étape du  
**croisement** et de la **mutation**.

**3) Le Crossover :**

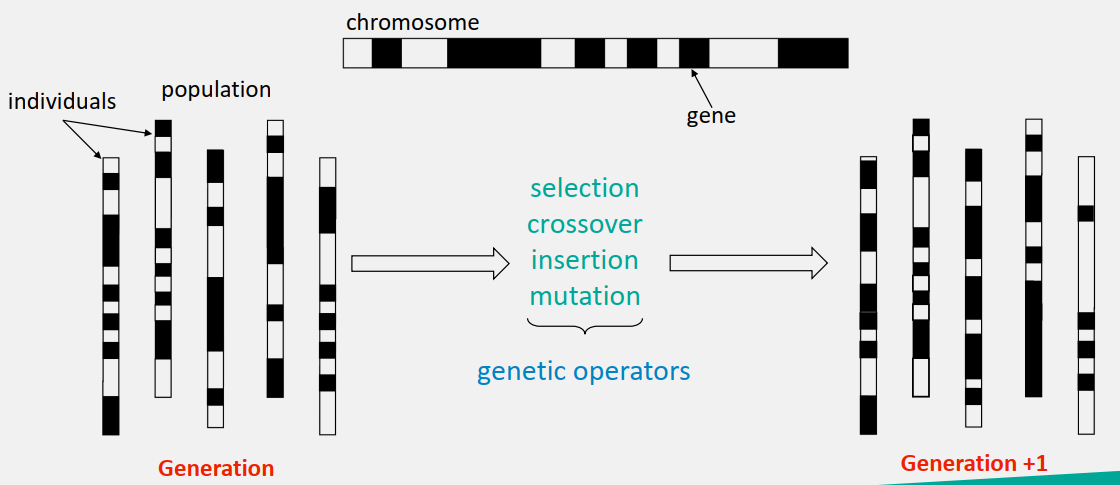
Dans cette étape on produit de nouveaux individus **(les enfants)** à partir d’un sous ensemble  
d’individus de la population **(les parents)**

Le but du croisement est de préserver **les meilleurs gènes** des parents, ainsi l’individu résultant du croissement a de grandes chances de ressembler aux parents.

**4) La mutation :**

Dans cette étape, un gène du **chromosome (solution) sera modifié**

*Shema de simulation de* ***Genetic Algorithm (GA)***



## Pseudo Code du l’algorithme GA :

# *Algorithme GA*

# BEGIN

# *Initialize* the entire population; *//(initialization)*

# Repeat

# *Select promising individuals from current population; //(selection)*

# *Reproduce selected individuals; //(crossover)*

# *Mutate reproduced individuals; //(mutation)*

# *Evaluate created individuals; //evaluation)*

# *Create new/updated population; //(insertion)*

# Until stopping criteria

# Return *Best from population;*

# END

**-Initialisation de la population pour le taquin 3\*3 :**

On a représenter la population par un ensemble de solutions aléatoires bonnes ou mauvaises . On l’a modéliser par une ArrayList de chemins que peut prendre le taquin pour arriver à l’état but .

Exemple de population : {P0,P1,P2......}

P0 : Left Up Right Up Left Down Down

P1 : Right Down Left Down Right Up Up

P2 : Down Right Up Right Down Left Up

Avant la sélection on a attribuer à chaque membre de la population sa valeur de la fonction fitness(Distance de Manhatten) puis on a trier toute notre population selon leurs valeurs de la fitness function .

**-Sélection :**

Sélectionner deux parents ayant les meilleurs valeurs selon la fonction fitness.

-**Croisement** :

Pour le croisement on a utiliser le Single Point .

Exemple :

P1 : Left Up Right || Up Right Down

P2 : Right Up Left || Down Down Left

Pour le croisement on va combiner les solutions des deux parents séléctionner :

C1 : Left Up Right || Down Down Left

C2 : Right Up Left || Up Right Down

-**Mutation** :

La mutation sert à ajouter de la diversité à la population on a procédé comme suit pour faire la mutation : on attribut une valeur Var Rate qui nous permettra de fixer combien nous allons faire de mutation pour chaque géne .

Exemple :

Var Rate=2 :

C1 : Left Up Right || Down Down Left

C1 va devenir : Up Left Down Right Down Left

on a muter le 1er géne avec le 2eme et le 3eme avec le 4eme .

-**Evaluation** :

L'évaluation sera en appliquant la fonction fitness aux fils généré lors de la mutation

On va garder à chaque itération la meilleur solution ayant la meilleur fitness et la remplacer dans notre population .

**-Stopping Criteria** :

On va arrêter notre algorithme une fois qu’on a trouvé la meilleur solution qui nous permettra d’atteindre le but en ayant la meilleur valeur fitness.

# Résolution avec L’algorithme PSO

# (Particle Swarm Optimization) :

## Définition du l’algorithme PSO :

L’optimisation par essaims particulaires **(Particle Swarm Optimization)** est une méthode qui s’inspire de **la biologie** pour résoudre des **problèmes d’optimisation**. Comme **les réseaux de neuronnes artificiels,** les **algorithmes génétiques** ou les algorithmes **de colonies de fourmis**, le Particle Swarm Optimization **(PSO)** est un algorithme **bio-inspiré**. Il repose sur les principes d’**auto-organisation** qui permettent à un groupe d’organismes vivants d’agir ensemble de manière complexe, à partir de **“règles” simples**. Le PSO s’inspire du modèle développé par **Craig Reynolds** pour **simuler** le déplacement grégaire de **certains animaux** (troupeaux de bovins, volées d’oiseaux…).

Comme nous le montre la figure ci-dessous une volée de oiseaux créant des sous-groupes puis se rassemblant en une volée entière:



*Groupes d'oiseaux à Milan prise par la Professeur Habiba DRIAS*

## Principe de l’algorithme :

**Métaphore:** Chaque solution unique de l'espace de recherche simule une particule, qui peut être

un Oiseau, un Poisson ,…**Chaque particule(Solution)**

- **A une position** dans l'espace de recherche à un instant donné t, qui représente concrètement une solution.

- **Se déplace** (vole) dans l'espace de recherche, le but étant de rencontrer des solutions optimales.

- **Se caractérise** par une vitesse qui dirige son vol.

- **Est évalué** par une fonction de fitness.

**Processus PSO:**

1-**Initialiser** un groupe de particules **(solutions) aléatoirement**.

2-l'objectif est de **rechercher la meilleure solution** dans l’espace de recherche.

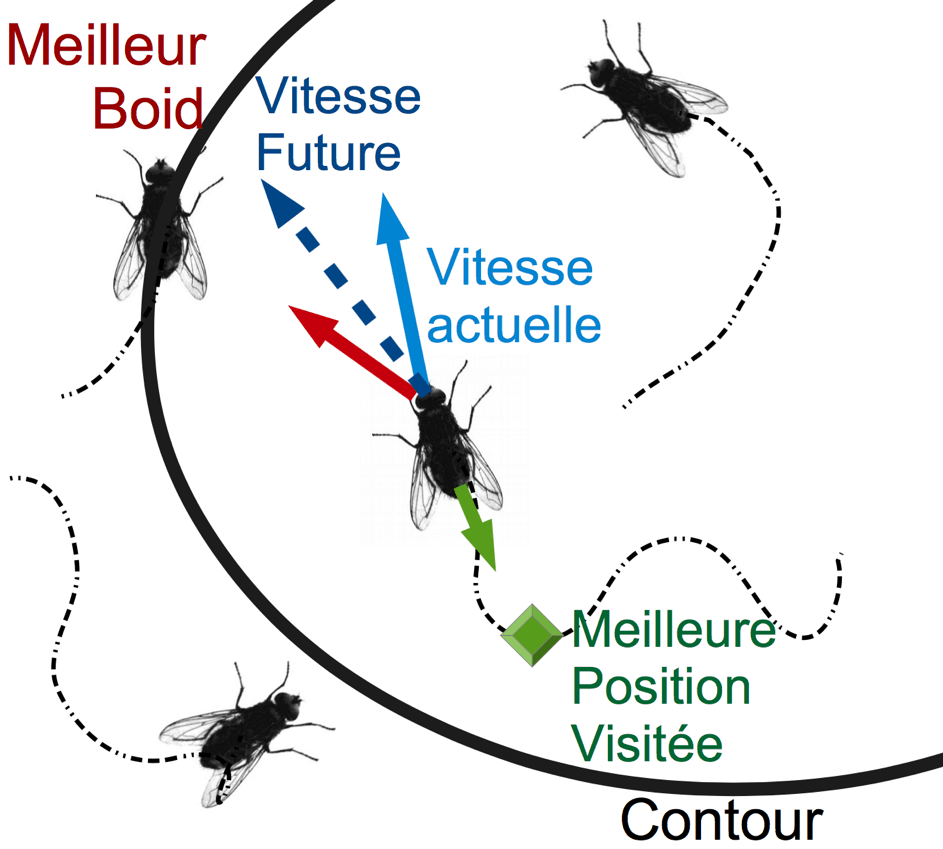
3-Il renouvelle la recherche **de nombreuses itérations** jusqu'à trouver une solution optimale ou une solution approximative de **haute qualité**.

À chaque itération et selon le mouvement du troupeau, chaque particule **met à jour sa position et sa rapidité**.

L'essaim de particules mémorise deux caractéristiques importantes :  
• **Pbest**, la meilleure solution (position) atteinte jusqu'ici par chaque particule courante.  
• **Gbest**, la meilleure solution obtenue jusqu'à présent par n'importe quelle particule de l'essaim.

Le mouvement d'une particule est décidé en combinant et en pondérant trois tendances de mouvement :

**• La vitesse** de la particule  
• La meilleure **performance** **de la particule**  
• La meilleure **performance** **des voisins**

****

**Formalisation :**

Une particule i de l’essaim dans un espace de dimension D est caractérisée, à l’instant t, par :

**– X :** sa position dans l’espace de recherche.

**– V :** sa vitesse.

**– Pbest :** la position de la meilleure solution par laquelle elle est passée.

**– Gbest:** la position de la meilleure solution connue de tout l’essaim.

Le déplacement de la particule i entre les itérations t et t+1 se fait selon les deux équations .

**Vi(t+1)=w\*Vi(t)+c1\*r1i(t)\*(Pbest i(t)– X(t)) +c2\*r2i\*(Gbesti(t)-Xi(t))**

**Xi(t+1)=Xi(t)+Vi(t+1)**

**– c1 et c2 :** deux constantes qui représentent les coefficients d’accélération, elles peuvent être non constantes dans certains cas selon le problème d’optimisation posé .

**– r1 et r2 :** deux nombres aléatoires tirés de l’intervalle [0,1].

La mise à jour prend en compte trois composants de déplacement :

La composante d’inertie : **w\*Vi(t), w** est le coefficient d‘inertie il control l‘influence du future mouvement .

La composante cognitive**: c1\*r1i(t)\*(Pbest i(t)– X(t))**

La composante sociale **: c2\*r2i\*(Gbesti(t)-Xi(t))**

La particule a la capacité de mémoriser sa meilleure position du passé.  
• Il a une sorte de « nostalgie » de revenir à cette position.  
• Cette position est appelée **Pbest** et est mise à jour à chaque fois, pour chaque particule à l'aide de la formule suivante :

**Pbesti =Pbesti(t) if f(Xi (t+1)) ≥ f(Pbesti (t) )**

**Pbesti = Xi(t+1) sinon**

La particule a aussi un comportement social.

Il suit l'essaim dans sa meilleure position appelée **Gbest**, qui peut être déterminée à l'aide de le suivant :

**Gbest(t + 1) = argminPbesti (Pbesti(t+1)).**

En cas de **maximisation** de la fonction de fitness, **argmin** doit être remplacé par **argmax**.

## Pseudo Code du l’algorithme PSO :

***BEGIN***

***Initialize******N*** *particles:* ***positions*** *and* ***velocities****;*

***Evaluate*** *the particles* ***positions****;*

**for *each*** *particle* ***i*** *do* ***Pbesti = xi****;*

***calculate******Gbest*** *using the transition rule;*

**for *each*** *iteration do*

**for *each*** *particle* ***p*** *do*

***update******the velocity*** *and* ***the position****;*

***move*** *the particle and evaluate* ***its fitness****;*

***update******Pbest****;*

**endfor;**

*update* ***Gbest****;*

**endfor;**

**endfor;**

***END***

## Application de l’algorithme PSO pour le Taquin 3 \* 3 :

Les différentes **positions de tuiles** peuvent être traitées comme **les particules dans l'essaim** après avoir suivi le mouvement approprié, qui se trouve en **calculant la valeur de fitness** de tous **les mouvements possibles** dans **chaque état**. Nous entrons maintenant dans le meilleur état suivant et local qui a **la valeur optimale la** plus proche de **la valeur de fitness de l'état d'objectif,** comme dès que l'état d'objectif est atteint ou que le **nombre maximum d'itérations** est atteint, nous **arrêtons le processus** .Si l'objectif est atteint, nous stockons **le chemin complet** par lequel nous avons atteint **l'état d'objectif,** et ce chemin est la solution optimale pour le problème du **puzzle 8.**

## -représentation des paramètres de l’algorithme PSO dans le Taquin 3\*3:

## On a représenté la population par des solutions tirées aléatoires , les solutions sont des chemins (ensemble de règles à appliquer ).Dans notre cas une solution aléatoires sera comme suit : LEFT – UP – RIGHT …

**Algorithme PSO pour le Taquin 3\*3 :**

**Début**

**Initialisation de la population pour le taquin 3\*3 :**

On a représenter la population par un ensemble de solutions aléatoires bonnes ou mauvaises . On l’a modéliser par une ArrayList de chemins que peut prendre le taquin pour arriver à l’état but .

Exemple de population : {P0,P1,P2......}

P0 : Left Up Right Up Left Down Down

P1 : Right Down Left Down Right Up Up

P2 : Down Right Up Right Down Left Up .

**Evaluer les fitness** de chaque particule (Calcul des fitness de chaque particule )

Pbest := la solution ayant la meilleur fitness.

Gbest := Pbest //initialement G best recoit Pbest //

**Iteration=0;**

Si( **iteration>= Max\_iteration)** Alors **Stop**

Sinon **iteration++.**

**Pour chaque particule faire :**

**Mettre à jour les vitesses et les positions .**

Trouvez l'emplacement **du zero.**  
 Trouvez le nombre de **déplacements possibles** vers **le zero.**  
 **Si** c'est **la position la plus centrale** et que le nombre de **mouvements possibles** est **de**  **quatre**, Calculez la valeur de **fitness** pour tous **les quatre coups possibles** ainsi que la vitesse et leurs positions grâce aux formules vu précédemment .

**sinon** s'il s'agit de la position **la plus au coin** que le nombre de **déménageur**s est **de deux**, Calculez la valeur de **fitness** **pour les deux** ainsi que la vitesse et leurs positions t**.**  
  **Sinon,** si c'est **la position centrale** limite que le nombre de mouvements **est à trois,** calculez la valeur de fitness pour tous **les trois coups possibles** ainsi que la vitesse et leurs positions **.**

**Pbest :=** la meilleur solution trouvé **.**

**Fin pour**

**Mettre à jour Gbest := Pbest**

# Résolution avec L’algorithme BSO

# (Bee Swarm Optimization) :

## Définition du l’algorithme BSO :

La méta heuristique d’**optimisation par essaim d’abeilles** est une nouvelle méta heuristique inspirée de **la métaphore des comportements collectifs des abeilles** pour la résolution des problèmes.

L’applications de **l’intelligence en essaim** est en pleine expansion, dans **l’optimisation** **combinatoire**,car elle offre une méthode alternative pour la conception des **systèmes intelligents** pour lesquels l’**autonomie**, l’**émergence** et le **fonctionnement distribué** remplacent **contrôle**, **pré programmation** et **centralisation**.

L’algorithme **BSO (Bee Swarm Optimization)** a été élaborée par **une équipe de recherche du département d’informatique** à l’université de Science et Technologie Houari Boumediene **(USTHB)** constituée par **(les professeurs H.Drias, Sadeg et Yahi en 2005)** et qui s’inspire du comportement réel des abeilles durant leur recherche de nourriture.

## Principe de l’algorithme :

## -représentation des paramètres de l’algorithme GA dans le Taquin 3\*3:

## On a représenté la population par des solutions tirées aléatoires , les solutions sont des chemins (ensemble de règles à appliquer ).Dans notre cas une solution aléatoires sera comme suit : LEFT – UP – RIGHT …

## 1) Une des abeilles BeeInit initie le processus en déterminant un sommet appelée Sref. Ce sommet désigne la région d’exploitation qui s’appelle SearchArea dans laquelle les abeilles intensifieront leurs recherches .

## 2) La région d’exploration est constituée d’un ensemble de sommets. Chacun d’eux est obtenu à partir du sommet Sref et ce, en utilisant une certaine stratégie.

## 3) chaque abeille se voit assignée un sommet (de SearchArea), depuis lequel elle démarre sa recherche dans le voisinage de la zone qui lui a été assigné.

## 4) A l’issue de sa recherche, chaque abeille k mémorise le meilleur sommet qu’elle a visité à la kème entrée qui lui est réservée dans une table Dance.

## 5) Mais si au bout d’un certain temps l’essaim constate que la qualité ne s’améliore plus (optimum local) le critère de diversité sera appliqué pour s’éloigner des régions déjà exploitées

## 6) Un des sommets de Dance deviendra le nouveau Sref, ainsi l’effort d’exploitation sera dirigé vers cette nouvelle zone prometteuse.

## 7) Pour éviter qu’un même sommet puisse être considéré deux fois comme référence, une liste Taboo peut être utilisée .

## 2022-05-15_171233

***Bee Swarm in real Life***

## 2022-05-15_171120

*Shema de simulation de* ***Bee Swarm***

## 2022-05-15_110633

*Shema de simulation de* ***Bee Swarm Optimization Algorithm (BSO)***

## Pseudo Code du l’algorithme BSO:

# *Algorithme BSO*

# Var

# *Int  iteration = 0 ;*

# *Boolean  Optimal\_found=false;*

# BEGIN

# 

# *Let Sref be the solution found by BeeInit ;*

# While *(iteration < MaxIter AND not Optimal\_found) Do*

# *Insert Sref in Taboo List;*

# *Determine SearchPoints from Sref;*

# *Assing a solution of SearchPoints to each Bee;*

# Foreach *Bee k do*

# *Search starting with the assigned solution;*

# *Store the result in Dance;*

# EndFor*;*

# *Assing the best solution to Sref;*

# EndWhile

# END

## Application de l’algorithme BSO pour le Taquin 3 \* 3 :

# On commence par une abeille (BeeInit) qui initie le processus de recherche par un état

# Initial (Sref).

# Avant d’entamer la recherche, on vérifie si l’état but est atteint, si oui alors on quitte l’algorithme

# Et on affiche le chemin vers le but, sinon on commence la recherche de solution

# L’abeille (BeeInit) détermine la région d’exploitation (SearchArea) en créant des abeilles filles

# (Bees)

# Chaque abeille fille est assigne a un point de recherche (SearchPoint) qui désigne un état du

# Taquin

# Chaque abeille fille démarre la recherche de la solution (Localsearch) d’après son point de

# recherche

# Chaqu’une des abeilles intensifieront la recherche en mémorisant à chaque fois le meilleur état du

# Taquin qu’elle a visite ,

# La meilleure solution est sélectionnée par chaque abeille en utilisant l’heuristique de

# (distance de Manhattan)

# Chaqu’une des abeilles après avoir trouver sa meilleur solution, elle sauvegarde sa meilleur

# solution dans une table Dance

# 

# Apres Avoir la meilleur solution de chaque abeille dans la table Dance ,On la trie et on prend la

# Meilleure solution dans la table Dance (On peut considérer la table Dance comme

# (PriorityQueue) qui met la meilleure solution en tête de queue.

# La meilleur Solution pris de la table Dance devient le Nouveau (Sref)

# Pour qu’un état ne sera pas référencé plus qu’une fois, chaque solution visiter sera mise dans

# une List Taboo

# On termine l’algorithme soit par un but attient (si Optimal\_Found sera True) ou nombre

# maximale d’itérations est dépassé

## Comparaison du temps d‘execution pour résoudre une instance du probléme pour les 3 méta heuristiques:

|  |  |  |  |
| --- | --- | --- | --- |
| EXEMPLE | Temps GA (en ns) | Temps PSO (en ns) | BSO (en ns0 |
| 283164705 | 228949700 | 157261200` | 7010400 |
| 281043765 | 229227200 | 92628400 | 11244100 |
| 845306712 | 503546600 | 126887600 | 19113500 |
| 503612784 | 245331500 | 88492700 | 136484800 |
| 518603247 | 189699300 | 179288500 | 140548300 |

## 

# En comparant entre tous les algorithmes étudié (BFS DFS A\* Manhattan Distance GA PSO BSO )on remarque que A\* et Manhatten ditance sont les mieux addaptés au probléme du taquin . en se penchant vers les résultats trouvés avec les métaheuristiques en terme de temps d’execution on trouve qu’il nous donnent d’assez bon résultat et nous permettent d’atteindre parfois des solutions optimale en un temps trés réduit.

**Conclusion :**

# Ce projet nous a permis d’approfondir nos connaissances en se familiarisant avec des approches sophistiquées pour résoudre des problèmes complexes. Ces derniers, souvent NP-Complets, n’étant pas solvables avec les techniques classiques (exactes). Pour cela, les méthodes méta-heuristiques sont utilisées cherchant à trouver un compromis entre les ressources utilisées, temps d’exécution et la qualité de la solution. De plus, l’étude approfondie du problème du taquin nous a donné la chance de découvrir son importance capitale dans l’étude théorique de la complexité.

***Reference***

***1-Algorithmique Moderne Analyse Et Complexité*** *(livre de Professeur* ***Habiba Drias****)*

*2-* [***https://www.researchgate.net/***](https://www.researchgate.net/)*(site internet pour les recherches scientifiques)*

*3-* [***https://www.researchgate.net/publication/284032167\_An\_Optimum\_Solution\_to\_8-\_Puzzle\_Problem\_using\_Particle\_Swarm\_Optimization\_PSO***](https://www.researchgate.net/publication/284032167_An_Optimum_Solution_to_8-_Puzzle_Problem_using_Particle_Swarm_Optimization_PSO)*(article sur PSO Algorithm)*