République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة هواري بومدين للعلوم و التكنولوجيا
Université des Sciences et de la Technologie
Houari Boumediene

Rapport du projet

Module : Métaheuristiques

Intitulé :

Résoudre le problème du N reines en utilisant les métaheuristiques .

Spécialité : Systèmes Informatiques Intelligents

Réalisé par :

— Djenane Nihad
— M'Barek Lydia

Travail demandé par :

Mme. MESSAOUDI Imane - Cours
Mme. Houacine Naila - TP

Promotion : 2022/2023

Table des matières

Table des figures

Liste des tableaux

Introduction générale

La première partie de notre rapport a présenté l'approche espace d'états pour résoudre le problème des N reines. Bien que cette approche soit efficace pour les problèmes de petite taille, elle devient rapidement inefficace pour les instances plus grandes. Pour surmonter cette limitation, nous allons maintenant nous tourner vers les métaheuristiques, une famille d'algorithmes d'optimisation puissants et flexibles qui peuvent être utilisés pour résoudre des problèmes difficiles tels que notre problème des N reines.

Les métaheuristiques sont une approche efficace pour résoudre des problèmes d'optimisation difficiles. Leur polyvalence, leur facilité de mise en œuvre et leur capacité à trouver des solutions approximatives rapidement en font une approche très intéressante pour résoudre notre problème.

Notre projet vise donc à résoudre le problème des N reines en utilisant des métaheuristiques, notamment l'algorithme génétique qui s'inspire de l'évolution naturelle, et le métaheuristique basé sur l'essaim de particules, une méthode d'optimisation coopérative entre les particules. Pour cela, notre plan sera structuré comme suit :

1. *Description de l'inspiration de chaque métaheuristique*
2. *Fonctionnement et algorithme de chaque métaheuristique*
3. *Expérimentation pour le paramétrage de chaque métaheuristique*
4. *Expérimentation sur la taille du problème pour chaque métaheuristique*
5. *Comparaison (graphes/tableaux, analyse et discussion) des métaheuristiques*
6. *Comparaison de la Partie 1 avec la Partie 2*
7. *Conclusion*

En suivant cette structure, nous allons présenter les métaheuristiques de manière approfondie, les comparer et évaluer leurs performances sur des instances du problème des N reines.

Chapitre 1

Résolution du problème N reines en utilisant l'algorithme génétique (AG)

1.1 L'inspiration de l'évolution naturelle dans l'algorithme génétique

L'algorithme génétique est une heuristique de recherche inspirée de la théorie de l'évolution naturelle par l'ADN de Charles Darwin. Elle consiste à trouver une solution en exécutant des opérations génétiques : la sélection, le croisement et la mutation. Nous utiliserons cet algorithme pour résoudre le problème des N-reines.

1.2 Modélisation de l'AG pour résoudre le problème des Nreines

Un chromosome est une solution potentielle dans l'algorithme génétique, composé de gènes représentant l'échiquier à une instance. On peut représenter un chromosome pour notre probleme sous forme de tableau, où l'indice correspond à la ligne et la valeur à la colonne où se trouve la reine. L'ensemble des chromosomes constitue une population. La transformation dans les gènes de cette population nous permet de trouver la solution optimale.

On représente une population par une liste, tel que chaque entrée soit un chromosome. La taille de la liste dépend de la taille de la population.

1.3 La description de l'AG

L'algorithme génétique suit plusieurs étapes, lesquelles sont expliquées ci-dessous :

1. **Générer des solutions aléatoires :**
   Nous commençons par la génération de chromosomes aléatoires en placer une seule reine pour chaque ligne avec une colonne choisie aléatoirement. Voici un exemple de solution générée aléatoirement pour N = 8

5	3	0	2	6	4	1	7

FIGURE 1.1 – Exemple d'une solution aléatoire.

   Obtenu à l'aide de cet algorithme

Input : $ProblemeSize$: entier
Output: $Resultat$: tableau d'entier
Créer une liste $liste$ vide;
for $i \leftarrow 0$ *to* n **do**
  Ajouter i à la fin de la liste $liste$;
end
Mélanger aléatoirement les éléments de la liste $liste$;
Initialiser un tableau $Resultat$ de taille n;
for $i \leftarrow 0$ *to* n **do**
  Placer l'élément à l'index i de la liste $liste$ dans la case i du tableau $resultat$;
end
return $Resultat$;

Algorithm 1: Algorithme de génération de solutions aléatoires pour le problème des N-Reines.

2. **Évaluer les solutions :**
En utilisant l'algorithme ci-dessous, nous évaluons la fitness de chaque solution potentielle en déterminant le nombre de reines qui sont en menaces.

Input : $solution$: tableau d'entier
Output: $fitness$: entier
$fitness \leftarrow 0$;
for *reine in solution* **do**
  if *il y a une reine sur la même ligne* **then**
    $fitness \leftarrow fitness + 1$;
    continue;
  end
  if *il y a une reine sur la même colonne* **then**
    $fitness \leftarrow fitness + 1$;
    continue;
  end
  if *il y a une reine sur le même diagonal* **then**
    $fitness \leftarrow fitness + 1$;
    continue;
  end
end
return $Fitness$;

Algorithm 2: Algorithme de calcul de fitness pour le problème des N-Reines.

Ilustration : Prenons l'exemple d'une population de six individus. Voici la première génération de chromosomes obtenue :

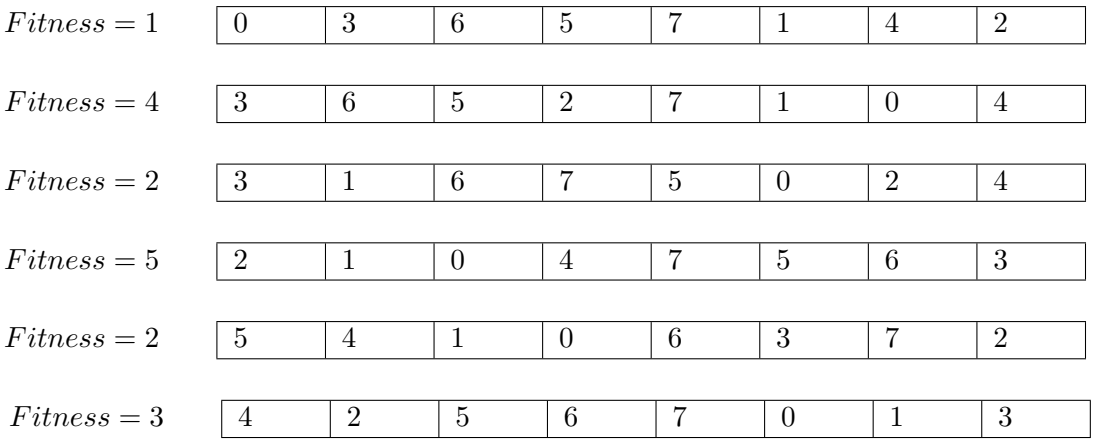

FIGURE 1.2 – Exemple d'une population initiale évaluée

NB : si la solution optimale est trouvée lors de la première génération de la population, alors l'algorithme s'arrête. Sinon, on continue avec le reste de l'algorithme.
Dans notre exemple, la solution optimale est quand la valeur du fitness est égale a 0.

3. **La sélection :**
   Cette étape nous permet de sélectionner les individus les plus performants de la population en fonction de leur fitness. Il existe plusieurs méthodes de sélection, mais nous avons choisi de sélectionner les individus ayant le meilleur fitness.
   Ilustration : En se basant sur l'exemple précédent, les chromosomes qui sont sélectionnés sont le premier et le cinquième.
4. **Le croisement :**
   L'opération de croisement consiste à créer de nouveaux chromosomes à partir de la population P' en combinant les gènes de deux parents sélectionnés lors de l'étape de sélection. Cette combinaison est réalisée en utilisant des points de croisement, qui déterminent les positions où les gènes des parents sont échangés pour produire des enfants. Ainsi, les enfants héritent des caractéristiques de leurs deux parents et présentent des solutions potentielles différentes de celles de leurs parents.

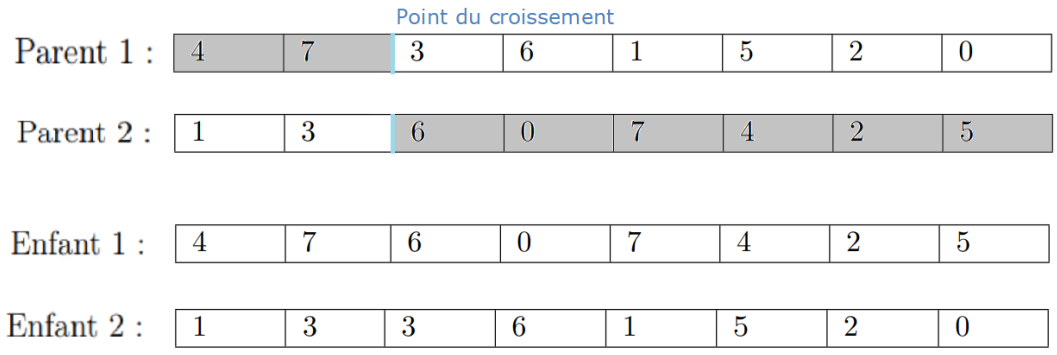

FIGURE 1.3 – Exemple de croisement de deux parents

5. **La mutation :**
   Cette étape nous permet d'appliquer certaines transformations dans les gènes pour obtenir un chromosome différent des parents avec un taux de mutation probable. Nous générons

un nombre aléatoire, si ce nombre est inférieur au taux de mutation, alors nous permetu deux gène choisie aléatoirement ; sinon, aucun changement ne sera effectué.

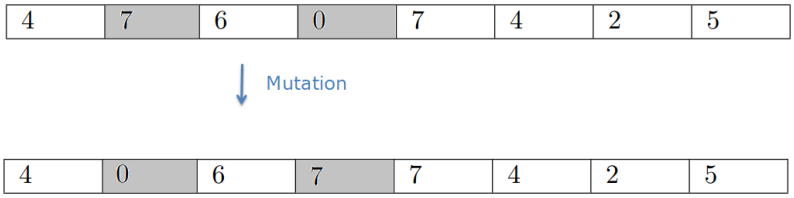

FIGURE 1.4 – Exemple de mutation d'un gène dans un chromosome

6. **Le remplacement :**
   L'algorithme génétique permet d'obtenir de nouveaux chromosomes. Par conséquent, la taille de la population sera supérieure à la taille maximale de la population. Pour limiter la taille, nous passons par la phase de remplacement qui élimine les chromosomes inutiles. Il existe plusieurs stratégies de remplacement, nous avons opté pour la stratégie consistant à remplacer les chromosomes les moins performants tout en conservant les individus ayant le meilleur fitness.

NB : Nous répétons les étapes deux à six jusqu'à ce que nous atteignions la condition d'arrêt. Soit la solution optimale trouvée soit, le nombre maximal d'itérations atteint.

1.4 Pseudo-code de l'AG

Input : $PCossover$, $PMutation$, $PopulationSize$, $ProblemeSize$, MAX_ITER
Output: $SGbest$: tableau d'entier
$Population \leftarrow$ Initialize_population ($PopulationSize$, $ProblemeSize$);
Evaluate_population($Population$);
SGbest $\leftarrow$ GetBestSolution($Population$);
while *(fitness* $\neq 0$ *and nombreIter* $< MAX\_ITER$ *)* **do**
    parents $\leftarrow$ selectParents(Population, PopulationSize);
    Children $\leftarrow$ {};
    for *(Parent1, Parent2) in Parents* **do**
        child1 $\leftarrow$ crossover(parent1, parent2, PCossover);
        child2 $\leftarrow$ crossover(parent1, parent2, PCossover);
        Children $\leftarrow$ Mutate(child1, PMutation);
        Children $\leftarrow$ Mutate(child2, PMutation);
    end
    EvaluatePopulation(Children);
    Sbest $\leftarrow$ GetBestSolution(Children);
    if *Sbest* $>$ *Sgbest* **then**
        Sgbest $\leftarrow$ Sbest;
    end
    Population $\leftarrow$ Replace(Population, Children);
end
return SGbest;

Algorithm 3: Algorithme génetique pour le problème des N-Reines.

1.5 Définition des paramètres empiriques dans l'AG

L'algorithme génétique utilise plusieurs paramètres. Pour assurer le bon fonctionnement de l'algorithme, nous devons définir la valeur la plus appropriée pour chacun de ces paramètres. Pour ce faire, nous mesurons la qualité de la solution en faisant varier les paramètres pour trois instances du problème des n reines (n = 50 , n = 100 et n = 200) et en évaluant la performance de l'algorithme pour chacune d'entre elles. Ensuite, nous choisissons celui qui a obtenu le taux de satisfaction le plus élevé pour les trois instances.
Il est à noter que :

$$\text{Taux de satisfaction} = \frac{\text{Nombre de reines en sécurité} \times 100}{\text{Taille du problème}}$$

$$\text{Nombre de reines en sécurité} = \text{Taille du problème} - \text{Fitness}$$

1.5.1 Taille de la population

Ci-dessous sont présentées les variations du temps d'exécution et de la fonction d'évaluation (fitness) pour les trois instances en fonction de la taille de la population.

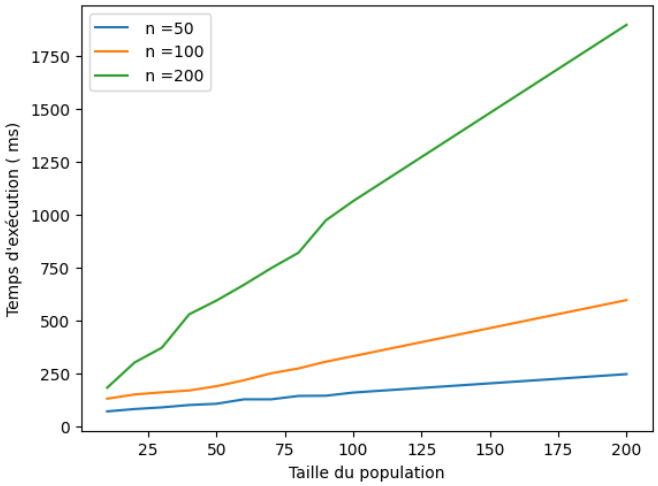

FIGURE 1.5 – Variation du temps d'exécution en fonction de la taille de la population

Instance 1 : n = 50

Taille population	10	20	30	40	50	60	70	80	90	100
Fitness	3	3	3	2	2	2	2	2	2	2
Taux de satsification	94%	94%	94%	96%	96%	96%	96%	96%	96%	96%

TABLE 1.1 – Variation de la fonction d'évaluation en fonction du taille de population pour l'instance 1 : n = 50

Instance 2 : n = 100

Taille population	10	20	30	40	50	60	70	80	90	100
Fitness	9	9	9	8	7	7	7	7	6	6
Taux de satsification	91%	91%	91%	92%	96%	93%	93%	93%	94%	94%

TABLE 1.2 – Variation de la fonction d'évaluation en fonction du taille de population pour l'instance 2 : n = 100

Instance 3 : n = 200

Taille population	10	20	30	40	50	60	70	80	90	100
Fitness	32	30	24	25	24	23	23	22	21	21
Taux de satsification	84%	85%	88%	87.5%	88%	88.5%	88.5%	89%	89.5%	89.5%

TABLE 1.3 – Variation de la fonction d'évaluation en fonction du taille de population pour l'instance 3 : n = 200

Analyse les resultats :

L'analyse des résultats montre que le temps d'exécution augmente de manière linéaire à mesure que la taille de la population augmente, plutôt que de manière exponentielle. En ce qui concerne le fitness, les tableaux indiquent que celui-ci diminue à mesure que la taille de la population augmente, convergeant vers une solution optimale.

En se basant sur les résultats de notre expérimentation, les meilleures valeurs pour les instances n = 50, n = 100 et n = 200 sont respectivement 40, 80 et 80. Afin de mieux comparer ces résultats, nous illustrons le graphique à barres suivant qui montre la variation du taux de satisfaction en fonction de la taille de la population pour les trois instances testées :

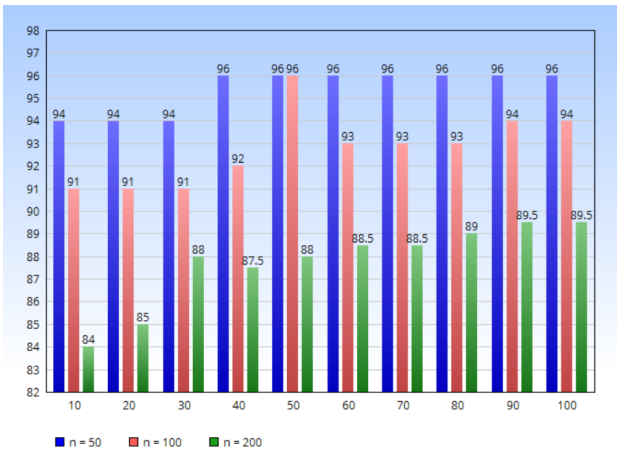

FIGURE 1.6 – Variation du taux de satisfaction en fonction de la taille de la population.

En outre, le tableau suivant présente la moyenne des taux de satisfaction :

Taille population	10	20	30	40	50	60	70	80	80	100
Taux de satsification moyen	89%	90%	91%	91.5%	93.6%	92%	92.5%	92.6%	93%	93%

TABLE 1.4 – Taux de satisfaction moyen en fonction de la taille de la population

Nous en concluons que la valeur la plus appropriée pour la taille de la population est de 50.

1.5.2 Maximum d'itération

Est un paramètre d'arrêt de l'algorithme. La figure suivante illustre la variation du temps d'exécution en fonction du nombre maximal d'itérations pour les trois instances testées :

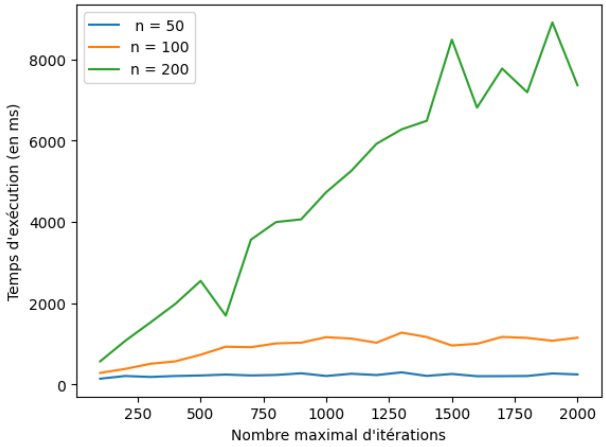

FIGURE 1.7 – Variation du temps d'exécution en fonction de nombre maximal d'itérations.

Instance 1 : n = 50

Maximum d'itération	100	200	300	400	500	600	700	800	900	1000
Fitness	2	1	1	1	0	0	0	0	0	0
Taux de satsification	96%	98%	98%	98%	99.99%	99.99%	99.99%	99.99%	99.99%	99.99%

Maximum d'itération	1100	1200	1300	1400	1500	1600	1700	1800
Fitness	1	1	0	0	0	0	0	0
Taux de satsification	99.99%	99.99%	99.99%	99.99%	99.99%	99.99%	99.99%	99.99%

Maximum d'itération	1900	2000
Fitness	0	0
Taux de satsification	99.99%	99.99%

TABLE 1.5 – Variation de la fonction d'évaluation (fitness) en fonction du nombre maximal d'itérations pour l'instance 1 avec n = 50.

Instance 2 : n = 100

Maximum d'itération	100	200	300	400	500	600	700	800	900	1000	1100	1200
Fitness	7	4	2	1	1	1	1	1	1	1	1	0
Taux de satsification	93%	96%	99%	99%	99%	99%	99%	99%	99%	99%	99%	99.99%

Maximum d'itération	1300	1400	1500	1600	1700	1800	1900	2000
Fitness	0	0	0	0	0	0	0	0
Taux de satsification	99.99%	99.99%	99.99%	99.99%	99.99%	99.99%	99.99%	99.99%

TABLE 1.6 – Variation de la fonction d'évaluation (fitness) en fonction du nombre maximal d'itérations pour l'instance 2 avec n = 100.

Instance 3 : n = 200

Maximum d'itération	100	200	300	400	500	600	700	800	900	1000	1100
Fitness	24	14	10	7	4	5	3	4	2	2	2
Taux de satsification	88%	93%	95%	96.5%	98%	97.5%	98.5%	98%	99%	99%	99%

Maximum d'itération	1200	1300	1400	1500	1600	1700	1800	1900	2000
Fitness	2	2	1	1	1	1	1	1	0
Taux de satsification	99%	99%	99.5%	99.5%	99.5%	99.5%	99.5%	99.5%	99.99%

TABLE 1.7 – Variation de la fonction d'évaluation (fitness) en fonction du nombre maximal d'itérations pour l'instance 3 avec n = 200.

Les meilleures valeurs de paramètre pour les instances n = 50 , n = 100 et n = 200 sont respectivement de 500, 1200 et 2000.
Pour une meilleure visualisation, nous avons regroupé ces valeurs dans un seul graphe à barres.

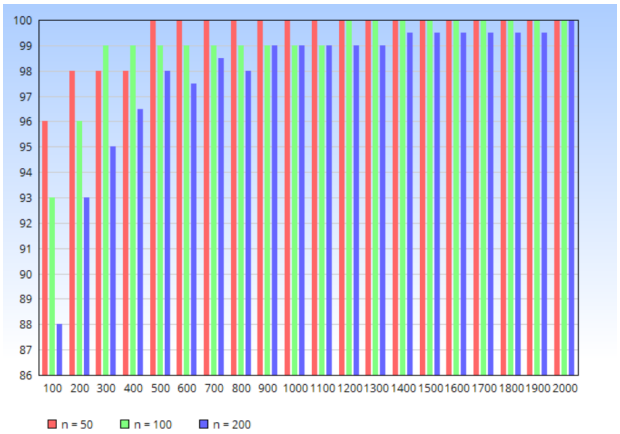

FIGURE 1.8 – Variation du taux de satisfaction en fonction du nombre maximal d'itérations.

1.5.3 Taux de croissement

Le taux de croisement est un paramètre probabiliste qui détermine la probabilité. Le graphique suivant décrivent la variation du temps d'exécution en fonction du taux de croisement.

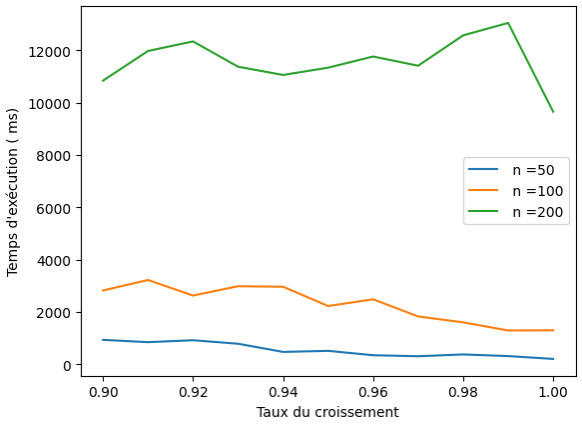

FIGURE 1.9 – Variation du temps d'exécution en fonction du taux de croissement

En se basant sur le graphique, on peut clairement constater que le choix du taux de croisement a un impact significatif sur le temps d'exécution de l'algorithme. Un taux de croisement trop faible peut réduire la diversité de la population et conduire à une convergence prématurée, tandis qu'un taux de croisement trop élevé peut entraîner une exploration inefficace de l'espace de recherche et augmenter considérablement le temps d'exécution. Par conséquent, le choix d'un taux de croisement optimal est essentiel pour obtenir de bonnes performances de l'algorithme génétique en terme temps d'execution.
Concernant les variations de la fonction fitness, les résultats sont présentés dans les tableaux suivants.

Instance 1 : n = 50

Taux de croissement	0.9	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99	1
Fitness	4	3	3	3	1	1	0	0	0	0	0
Taux de satsification	92%	94%	94%	94%	99%	99%	99.99%	99.99%	99.99%	99.99%	99.99%

TABLE 1.8 – Variation de la fonction d'évaluation en fonction du taux de croisement pour l'instance 1 : n = 50

Instance 2 : n = 100

Taux de croissement	0.9	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99	1
Fitness	7	7	6	4	3	2	2	1	0	0	0
Taux de satsification	93%	93%	94%	96%	97%	98%	98%	99%	99.99%	99.99%	99.99%

TABLE 1.9 – Variation de la fonction d'évaluation en fonction du taux de croisement pour l'instance 2 : n = 100

Instance 3 : n = 200

Taux de croissement	0.9	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99	1
Fitness	12	10	7	9	6	5	3	2	2	1	0
Taux de satsification	94%	95%	96.5%	95.5%	97%	97.5%	98.5%	99%	99%	99.5%	99.99%

TABLE 1.10 – Variation de la fonction d'évaluation en fonction du taux de croisement pour l'instance 3 : n = 200

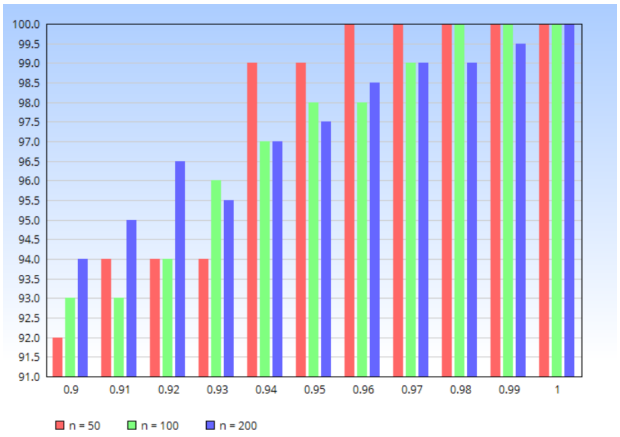

FIGURE 1.10 – Variation du taux de satisfaction en fonction du taux de croisement.

Ci-dessous est présenté le taux moyen de satisfaction des résultats :

Taux de croissement	0.9	0.91	0.92	0.93	0.94	0.95	0.96	0.97
Taux de satsification	93%	94%	94.8%	97.66%	98.16%	98.83%	99.33%	99.33%

Taux de croissement	0.98	0.99	1
Taux de satsification	99.66%	99.82%	99.99%

TABLE 1.11 – Taux de satisfaction moyen en fonction du taux de croissement

Nous concluons que la valeur la plus adaptée pour le taux du croissement est 1.

1.5.4 Taux du mutation

Est un paramètre probabiliste. Ses valeurs sont comprises entre zéro et un.

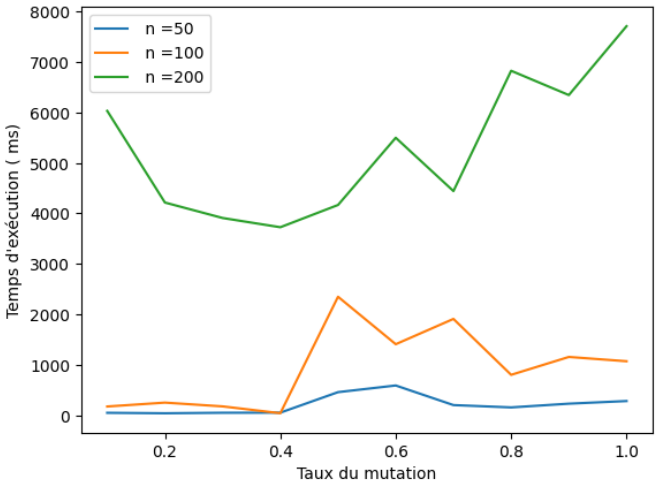

FIGURE 1.11 – Variation du taux de satisfaction en fonction du taux de croisement.

Instance 1 : n = 50

Taux du mutation	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Fitness	1	1	0	0	1	1	0	0	0	0
Taux de satsification	99%	99%	99.99%	99.99%	99%	99%	99.99%	99.99%	99.99%	99.99%

TABLE 1.12 – Variation de la fonction d'évaluation en fonction du taux du mutation pour l'instance 1 : n = 50

Instance 2 : n = 100

Taux du mutation	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Fitness	0	1	1	0	2	1	2	1	1	0
Taux de satsification	99.99%	99%	99%	99.99%	98%	99%	98%	99%	99%	99.99%

TABLE 1.13 – Variation de la fonction d'évaluation en fonction du taux du mutation pour l'instance 2 : n = 100

Instance 3 : n = 200

Taux du mutation	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Fitness	2	1	1	1	1	1	1	1	1	0
Taux de satsification	99%	99.5%	99.5%	99.5%	99%	99%	99%	99%	99%	99.99%

TABLE 1.14 – Variation de la fonction d'évaluation en fonction du taux du mutation pour l'instance 3 : n = 200

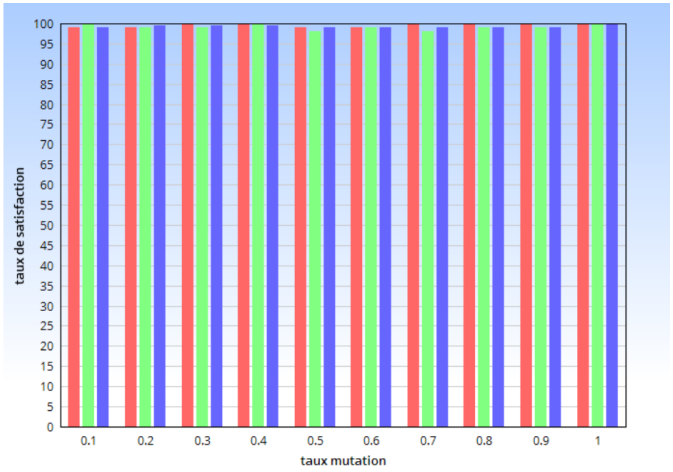

FIGURE 1.12 – Variation du taux de satisfaction en fonction du taux de croisement.

Le tableau suivant décrit le taux de satisfaction moyen des résultats :

Taux du mutation	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Taux de satsification moyen	99.33%	99.49%	99.49%	99.82%	98.83%	99.83%	99.16%	99.16%

Taux du mutation	0.9	1
Taux de satsification moyen	99.16%	99.99%

TABLE 1.15 – Taux de satisfaction moyen en fonction du taux du mutation

Nous en déduisons que la meilleure valeur du taux de mutation est 1.

1.6 conclusion :

Les valeur des paramètres empiriques qui donnent les meilleures performances pour le Algorithme genetiques sont :

Taille population	50
Maximum d'iteration	1000
taux du croissement	1
Taux du mutation	1

Chapitre 2

Résolution du problème N reines en utilisant optimisation par essaims de particules (PSO)

Dans notre recherche d'une solution efficace pour résoudre le problème des N-reines, nous avons étudié l'algorithme génétique dans le chapitre précédent comme une métaheurstique pour résoudre notre problème. Dans ce chapitre, nous allons découvrir une autre métaheurstique, l'optimisation par essaim de particules (PSO), pour résoudre le même problème, afin de pouvoir comparer les performances et les avantages de chaque approche et de choisir la méthode la plus adaptée.

2.1 L'inspiration animale dans l'optimisation par essaim de particules (PSO)

L'intelligence en essaims consiste à étudier et à construire des sociétés d'individus artificiels qui sont collectivement capables de fournir des décisions intelligentes complexes et riches grâce à des interactions simples.
L'optimisation par essaim de particules (PSO) est l'un des algorithmes les plus populaires en intelligence en essaims inspirée du comportement de regroupement d'un essaim d'oiseaux ou d'insectes. Les particules de l'algorithme PSO sont des points dans l'espace de recherche qui se déplacent dans cet espace en fonction de leur propre expérience de recherche et de l'expérience de recherche de leurs voisins. L'objectif est de trouver la meilleure solution possible en optimisant la fonction objectif.

2.2 Modélisation de l'algorithme PSO pour résoudre le problème des N reines

Nous avons utilisé la structure de tableau pour modéliser les essaims, de telle sorte que chaque entrée représente une particule. La modélisation d'une particule se fait de la manière suivante :

- **Sa position actuelle :** un vecteur entier de taille le nombre de reines placées dans l'échiquier qui fait référence à la solution potentielle du problème des N-reines.
- **Fitness :** un entier qui nous permet d'évaluer chaque solution potentielle à l'aide de la fonction d'évaluation qui consiste à compter le nombre de reines en menace sur l'échiquier.

- **Vélocité :** un vecteur double qui exprime la vitesse de chaque reine sur le plateau d'échecs.
- **PBest :** un vecteur entier qui exprime la meilleure position visitée par une particule dans terme d'évaluation

2.3 La description de l'algorithme PSO

L'algorithme PSO prend en entrée le nombre de reines placées sur l'échiquier ainsi que différents paramètres. Il retourne en sortie la meilleure solution obtenue.

1. **Générer des solutions aléatoires :**
   Pour démarrer l'algorithme, nous devons produire des particules aléatoires dont le nombre dépend d'un paramètre appelé la taille de population (POP_SIZE) donné en entrée. Nous calculons pour chaque particule sa vitesse initiale.
   Après avoir généré les solutions, nous sélectionnons la meilleure position dans le groupe d'essaim GBest qui correspond à la position ayant la meilleure valeur de fitness.
   NB : L'algorithme PSO utilise certains paramètres empiriques, tels que le nombre de particules, dont la valeur est déterminée par expérimentation. Nous détaillerons ce processus dans les sections suivantes.
   Ilustartion : Nous avons généré aléatoirement des vecteurs de position pour un problème de 8 reines avec un nombre de particules égal à 4. Pour chaque solution, nous avons calculé ses performances ainsi que sa vitesse initiale qui est également aléatoire. La meilleure position initiale visitée par une particule (PBest) est la position courante.
   Voici les résultats :
   - **Particule 1 :**
     La position :

1	2	3	0	4	6	5	7

     Le véolocité :

-1	-2	3	-4	5	6	-7	0

     Fitness : 6
   - **Particule 2 :**
     La position :

4	0	7	6	5	3	1	2

     Le véolocité :

-1	2	-3	4	5	-6	7	0

     Fitness : 3
   - **Particule 3 :**
     La position :

5	4	7	6	2	0	3	1

     Le véolocité :

1	-2	-7	4	5	0	-6	-3

     Fitness : 5
   - **Particule 4 :**
     La position :

6	4	0	2	1	5	7	3

     Le véolocité :

-6	-5	4	-3	-2	1	7	0

     Fitness : 3

GBest correspond à la solution avec la meilleure évaluation. Dans ce cas, GBest est égal à la position de la particule 2.

2. **Mise à jour de la vitesse d'une particule :**
   Voici la formule générale pour calculer la vitesse :

$$v_i(t+1) = wv_i(t) + c_1 r_{1,i}(t)(P_{best,i}(t) - x_i(t)) + c_2 r_{2,i}(t)(G_{best}(t) - x_i(t)$$

3. **Mise à jour de la position d'une particule :**
   La mise à jour de la position des reines consiste à les déplacer en fonction de la vitesse calculée. La formule suivante permet de mettre à jour la nouvelle position :

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

4. **Mettre à jour PBest d'une particule :**
   Une particule doit conserver la meilleure position visitée (PBest), donc après avoir actualisé la position, la particule doit recalculer PBest en utilisant l'équation suivante :

$$\text{Pbest}_i(t) = \begin{cases} \text{x}_i(t+1) & \text{si } f(\text{x}_i(t+1)) \leq f(\text{Pbest}_i(t)) \\ \text{Pbest}_i(t) & \text{sinon} \end{cases}$$

   NB : Les étapes deux, trois et quatre sont répétées pour autant de particules existant dans le groupe d'essaim.

5. **Mise à jour GBest :**
   La mise à jour de la meilleure position trouvée par toutes les particules d'un même groupe "gbest" est mesurée après avoir terminé la mise à jour de toutes les particules dans la même itération.

2.4 Pseudo-code de l'algorithme PSO

Input: Size, C1, C2, W, MAX_ITER, POP_SIZE
Output: SGBest
Population ← Initialize_population (PopulationSize, ProblemeSize);
Evaluate_population(Population);
Initialize_PBest (PopulationSize);
SGbest ← GetBestSolution(Population);
nombreIter ← 0;
while *(nombreIter < MAX_ITER and SGbest.fitness ≠ 0)* **do**
  for *Particule in population* **do**
    Mettre à jour la vitesse ;
    Mettre à jour la position ;
    Mettre à jour Pbest ;
  end
  Mettre à jour GBest ;
  $I \leftarrow I + 1$;
end
return SGbest;

Algorithm 4: Optimisation par essaim de particules pour le problème des N-reines

2.5 Définition des paramètres empiriques pour PSO

Pour ajuster les paramètres, nous allons suivre les mêmes étapes que l'algorithme génétique. Nous avons choisi trois instances, examiné la variation de la fonction d'évaluation pour chaque paramètre, et enfin comparé les résultats obtenus. Nous avons ensuite retenu la combinaison de paramètres la plus performante.

Rappel :

$$\text{Taux de satisfaction} = \frac{\text{Nombre de reines en sécurité} \times 100}{\text{Taille du problème}}$$

$$\text{Nombre de reines en sécurité} = \text{Taille du problème} - \text{Fitness}$$

2.5.1 Taille du population

Permet de délimiter la taille d'un groupe d'essaimage. Autrement dit, le nombre de solutions aléatoires à générer. Nous vous présentons ci-dessous le résultat d'une expérience menée sur trois instances différentes :

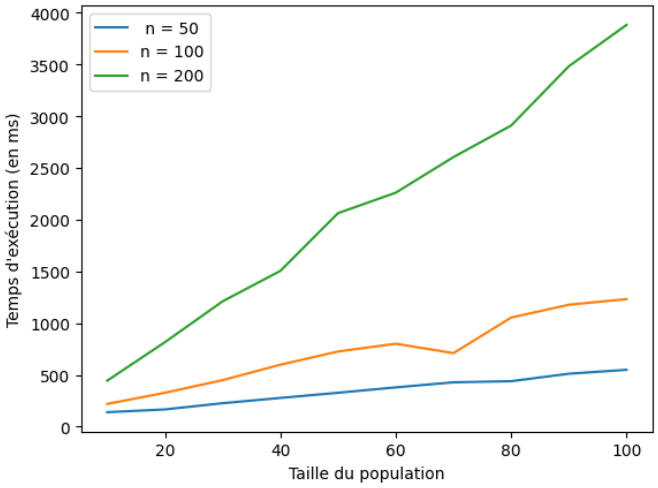

FIGURE 2.1 – Variation du temps d'exécution en fonction de la taille de la population

Instance 1 : n = 50

Taille du population	10	20	30	40	50	60	70	80	90	100
Fitness	15	15	14	14	14	14	14	14	14	14
Taux de satsification	70%	70%	72%	72%	72%	72%	72%	72%	72%	72%

TABLE 2.1 – Variation de la fonction d'évaluation (fitness) en fonction de la taille de la population pour l'instance 1 : n = 50.

Instance 2 : n = 100

Taille du population	10	20	30	40	50	60	70	80	90	100
Fitness	38	37	36	36	36	36	36	35	35	34
Taux de satsification	62%	63%	64%	64%	64%	64%	64%	65%	65%	66%

TABLE 2.2 – Variation de la fonction d'évaluation (fitness) en fonction de la taille de la population pour l'instance 2 : n = 100.

Instance 3 : n = 200

Taille du population	10	20	30	40	50	60	70	80	90	100
Fitness	83	83	80	80	80	79	79	79	79	78
Taux de satsification	58.5%	58.5%	60%	60%	60%	60.5%	60.5%	60.5%	60.5%	61%

TABLE 2.3 – Variation de la fonction d'évaluation (fitness) en fonction de la taille de la population pour l'instance 3 : n = 200.

Analyse les resultats :

D'après le graphe, on constate que le temps d'exécution augmente en fonction de la taille de la population pour les trois instances considérées. Cependant, en examinant les tableaux correspondants, il est clair que la fonction d'évaluation (fitness) converge très lentement vers la solution optimale.
Les meilleures valeurs de paramètre pour les instances n = 50 , n = 100 et n = 200 sont respectivement de 30, 100 et 100.
Pour une meilleure visualisation, nous avons regroupé ces valeurs dans un seul graphe à barres.

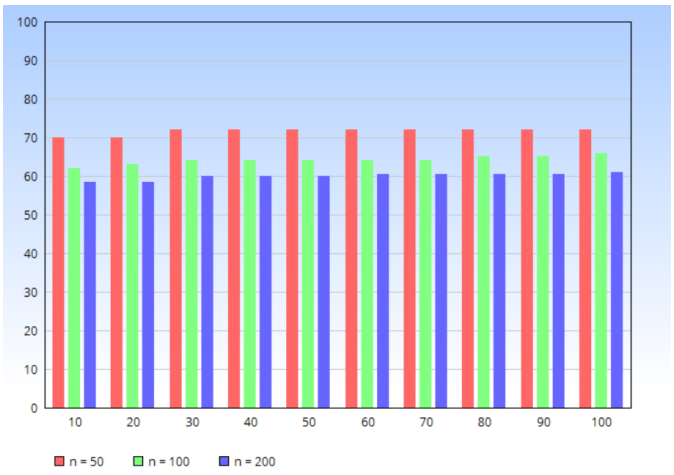

FIGURE 2.2 – Variation du temps d'exécution en fonction de la taille de la population

Le tableau suivant illustre le taux de satisfaction moyen des résultats :

Taille du population	10	20	30	40	50	60	70	80	90	100
Taux de satsification	62.5%	63.83%	65.33%	65.33%	65.33%	65.5%	65.6%	65.8%	60.8%	66.33%

TABLE 2.4 – Taux de satisfaction moyen en fonction de la taille de la population

Nous concluons que la valeur la plus adaptée pour la taille du population est 100.

2.5.2 Maximum d'itération

Est un paramètre qui participe à la condition d'arrêt.Ci-dessous, vous trouverez les résultats de l'expérimentation pour les différentes valeurs de ce paramètre pour les trois instances :

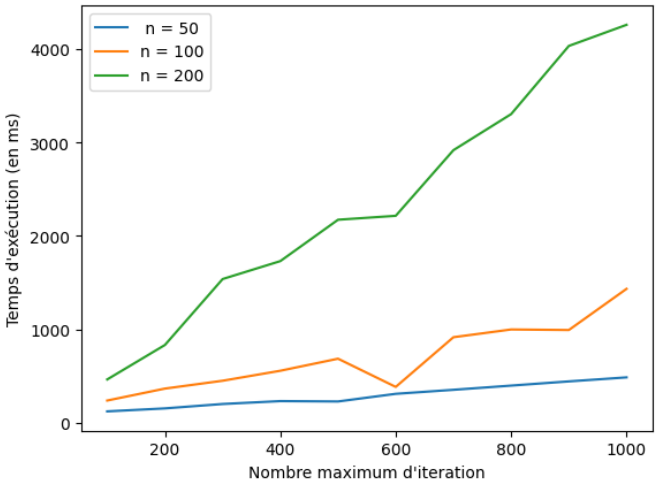

FIGURE 2.3 – Variation du temps d'exécution en fonction du nombre maximal d'itérations.

Instance 1 : n = 50

Maximum d'itération	100	200	300	400	500	600	700	800	900	1000
Fitness	14	14	14	14	14	13	13	13	13	12
Taux de satsification	72%	72%	72%	72%	72%	74%	74%	74%	74%	76%

TABLE 2.5 – Variation de la fonction d'évaluation (fitness) en fonction du nombre maximal d'itérations pour l'instance 1 : n = 50.

Instance 2 : n = 100

Maximum d'itération	100	200	300	400	500	600	700	800	900	1000
Fitness	37	37	36	36	35	35	35	35	35	34
Taux de satsification	63%	63%	64%	64%	65%	65%	65%	65%	65%	66%

TABLE 2.6 – Variation de la fonction d'évaluation (fitness) en fonction du nombre maximal d'itérations pour l'instance 2 : n = 100.

Instance 3 : n = 200

Maximum d'itération	100	200	300	400	500	600	700	800	900	1000
Fitness	79	78	76	78	77	77	77	78	77	77
Taux de satsification	60.5%	61%	62%	61%	61.5%	61.5%	61.5%	61%	61.5%	61.5%

TABLE 2.7 – Variation de la fonction d'évaluation (fitness) en fonction du nombre maximal d'itérations pour l'instance 3 : n = 200.

Analyse les resultats :

On peut également observer que, de la même manière que pour l'augmentation de la taille de population, le temps d'exécution augmente pour les différentes valeurs du nombre maximum d'itearation. De plus, la convergence vers la solution optimale demeure très lente.
La meilleur valeur pour les 3 instances est 1000, On peut clairement l'observer dans le graphique à barres ci-dessous :

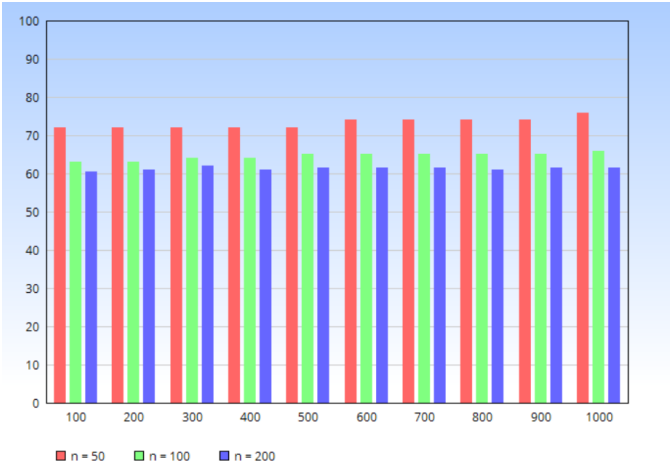

FIGURE 2.4 – Variation du temps d'exécution en fonction du nombre maximum d'iteration.

Le taux de satisfaction moyen :

Maximum d'itération	100	200	300	400	500	600	700	800	900	1000
Taux de satsification	65.16%	65.33%	66%	65.66%	66%	66.83%	66.83%	66.66%	66.83%	67.83%

TABLE 2.8 – Taux de satisfaction moyen en fonction du nombre maximal d'itérations.

Alors , la valeur la plus adaptée pour le nombre maximun d'itération est 1000.

2.5.3 Parametre W :

Les résultats de l'expérimentation menée pour déterminer la meilleure valeur de ce paramètre sont les suivants :

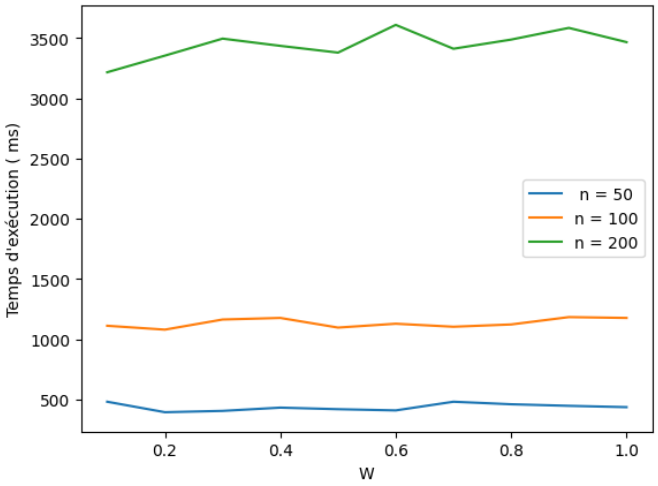

FIGURE 2.5 – Variation du temps d'exécution en fonction du nombre maximal d'itérations.

Instance 1 : n = 50

W	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Fitness	13	14	14	14	14	13	13	14	13	13
Taux de satsification	74%	72%	72%	72%	72%	74%	74%	72%	74%	74%

TABLE 2.9 – Variation de la fonction d'évaluation (fitness) en fonction du parametre W pour l'instance 1 : n = 50.

Instance 2 : n = 100

W	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Fitness	35	35	34	35	36	34	35	34	34	34
Taux de satsification	65%	65%	66%	65%	64%	65%	65%	66%	66%	66%

TABLE 2.10 – Variation de la fonction d'évaluation (fitness) en fonction du parametre W pour l'instance 2 : n = 100.

Instance 3 : n = 200

W	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Fitness	80	79	79	78	79	79	78	77	76	77
Taux de satsification	60%	60.5%	60.5%	61%	60.5%	61%	60.5%	61.5%	62%	61.5%

TABLE 2.11 – Variation de la fonction d'évaluation (fitness) en fonction du parametre W pour l'instance 3 : n = 200.

Analyse les resultats :

Les résultats de l'expérimentation montrent que, pour les instances considérées et avec une limite maximale d'itérations fixée à 1000, le paramètre W n'a pas d'impact significatif sur le temps d'exécution de l'algorithme.

Les meilleures valeurs de paramètre pour les instances n = 50 , n = 100 et n = 200 sont respectivement de 0.1, 0.3 et 0.9.

Le graphique à barres suivant montre la variation du taux de satisfaction en fonction du parametres W :

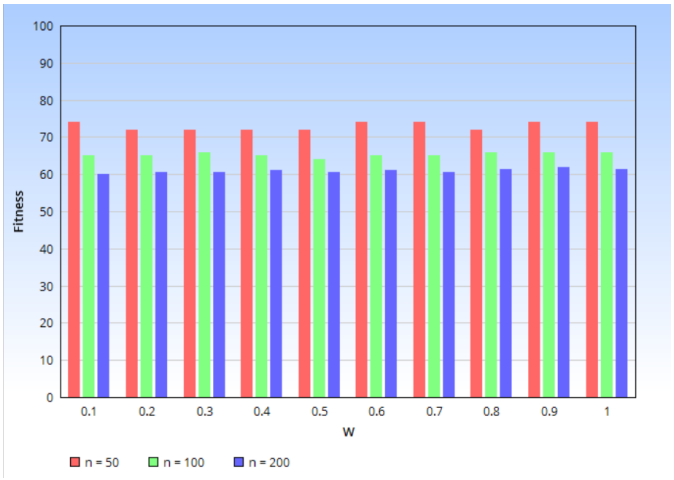

FIGURE 2.6 – Variation du temps d'exécution en fonction du parametre W.

Le taux de satisfaction moyen :

W	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Taux de satsification	66.33%	66.16%	66.16%	66%	65.5%	66.66%	66.5%	66.5%	67.33%	67.16%

TABLE 2.12 – Taux de satisfaction moyen en fonction du parametre W.

Nous en concluons que la valeur la plus appropriée pour ce parametres est de 0.9.

2.5.4 Parametre C1 :

Le paramètre empirique C1 prend ses valeurs dans l'intervalle [0, 1]. Les résultats de l'expérimentation menée pour fixer sa valeur sont les suivants

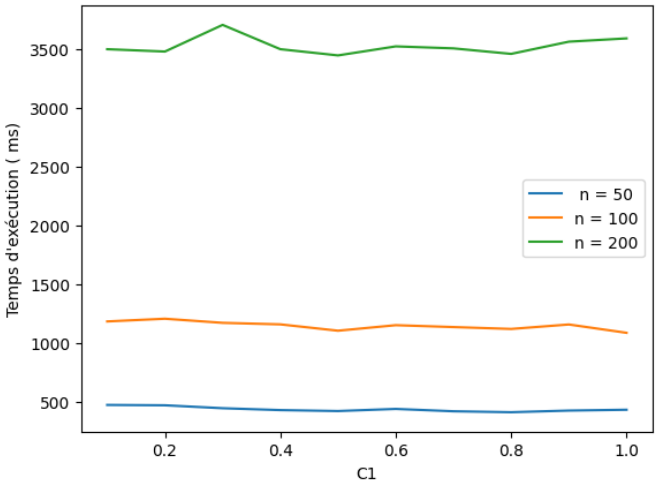

FIGURE 2.7 – Variation du temps d'exécution en fonction du parametre C1.

Instance 1 : n = 50

C1	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Fitness	12	12	13	12	13	12	13	13	13	13
Taux de satsification	76%	76%	74%	76%	74%	76%	74%	74%	74%	74%

TABLE 2.13 – Variation de la fonction d'évaluation (fitness) en fonction du parametre C1 pour l'instance 1 : n = 50.

Instance 2 : n = 100

C1	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Fitness	34	33	34	33	34	34	34	34	33	33
Taux de satsification	66%	67%	66%	67%	66%	66%	66%	66%	67%	67%

TABLE 2.14 – Variation de la fonction d'évaluation (fitness) en fonction du parametre C1 pour l'instance 2 : n = 100.

Instance 3 : n = 200

C1	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Fitness	79	78	77	78	78	78	79	79	78	78
Taux de satsification	60.5%	61%	61.5%	61%	61%	61%	60.5%	60.5%	61%	61%

TABLE 2.15 – Variation de la fonction d'évaluation (fitness) en fonction du parametre C1 pour l'instance 3 : n = 200.

Analyse les resultats :
Les meilleures valeurs pour les instances n = 50, n = 100 et n = 200 sont respectivement 0.3, 0.2 et 0.3

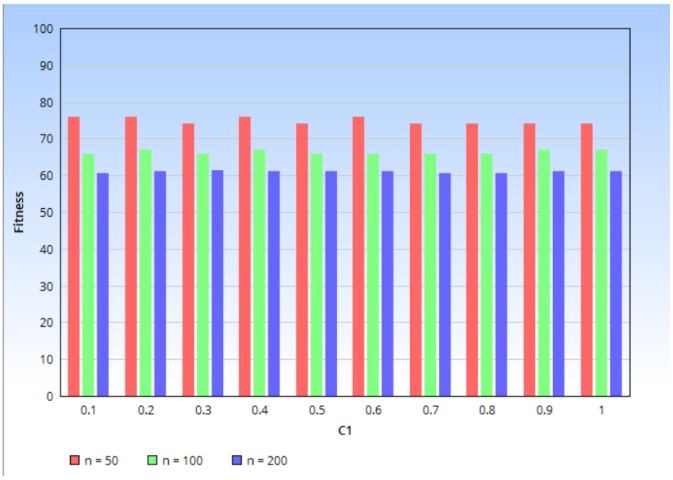

FIGURE 2.8 – Variation du temps d'exécution en fonction du parametre C1.

Le taux de satisfaction moyen :

C1	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Taux de satsification	67.5%	68%	67.16%	68%	67%	66.66%	66.83%	66.83%	67.33%	67.33%

TABLE 2.16 – Taux de satisfaction moyen en fonction du parametre C1.

2.5.5 Parametre C2 :

C2 prend ses valeurs dans le même intervalle que C1 comme indiqué dans les figures suivant :

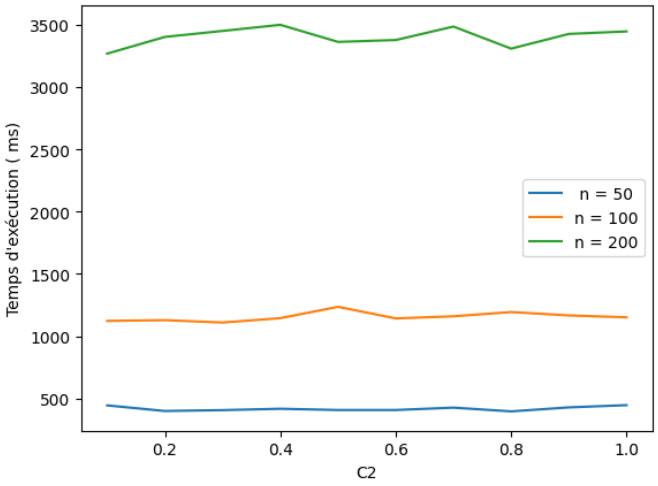

FIGURE 2.9 – Variation du temps d'exécution en fonction du parametre C2.

Instance 1 : n = 50

C2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Fitness	13	12	13	12	13	13	13	13	13	13
Taux de satsification	74%	76%	74%	76%	74%	74%	74%	74%	74%	74%

TABLE 2.17 – Variation de la fonction d'évaluation (fitness) en fonction du parametre C2 pour l'instance 1 : n = 50.

Instance 2 : n = 100

C2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Fitness	34	34	34	34	33	34	34	33	34	34
Taux de satsification	66%	66%	66%	66%	67%	66%	66%	67%	66%	66%

TABLE 2.18 – Variation de la fonction d'évaluation (fitness) en fonction du parametre C2 pour l'instance 2 : n = 100.

Instance 3 : n = 200

C2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Fitness	79	77	77	76	78	78	78	78	77	77
Taux de satsification	60.5%	61.5%	61.5%	62%	61%	61%	61%	61%	61.5%	61.5%

TABLE 2.19 – Variation de la fonction d'évaluation (fitness) en fonction du parametre C2 pour l'instance 3 : n = 200.

Les meilleures valeurs de paramètre pour les instances n = 50 , n = 100 et n = 200 sont respectivement de 0.2, 0.5 et 0.4.

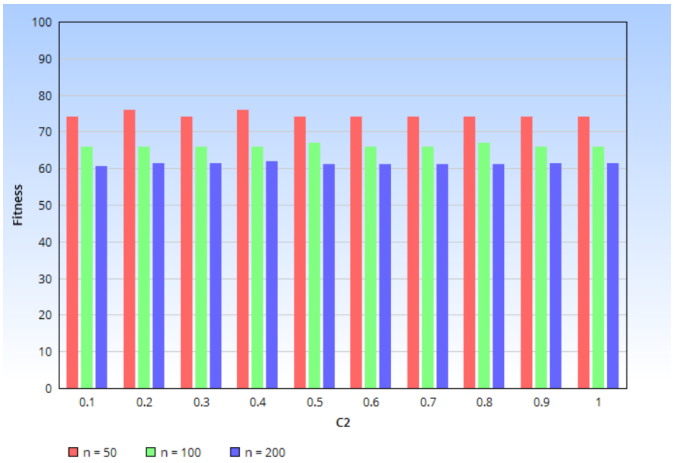

FIGURE 2.10 – Variation du temps d'exécution en fonction du parametre C2.

Le taux de satisfaction moyen :

C2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Taux de satsification	66.83%	67.83%	67.16%	68%	67.33%	67%	67%	66.73%	67.16%	67.16%

TABLE 2.20 – Taux de satisfaction moyen en fonction du parametre C2.

Nous concluons que la valeur la plus adaptée pour C2 est 0.4.

2.6 conclusion :

Les valeur des paramètres empiriques qui donnent les meilleures performances pour le Algorithme PSO sont :

Taille population	100
Maximum d'iteration	1000
W	0.9
C1	0.2
C2	0.4

Chapitre 3

Comparaison des résultats

Dans ce chapitre, nous allons comparer les performances de deux métaheuristiques : l'algorithme génétique et l'optimisation par essaim de particules (PSO). En outre, nous allons également comparer ces deux métaheuristiques avec les méthodes exactes qui ont été implémentées dans la partie 1 de ce rapport.

3.1 Comparaison des performances de l'algorithme génétique et de PSO :

Dans cette première partie du chapitre, nous allons comparer les performances des deux métaheuristiques en mesurant la qualité de la solution et le temps d'exécution en fonction de la taille du problème. Nous allons examiner comment chacune des deux métaheuristiques s'est comportée pour résoudre le problème des n-reines et déterminer laquelle a donné de meilleurs résultats en termes de qualité de solution et de temps d'exécution.

Les tableaux suivants présentent l'évolution du temps d'exécution et de la qualité de la solution (fitness) en fonction de la taille de la population pour chaque métaheuristique.

Taille du probleme	10	20	30	40	50	60	70	80	90	100	200
Temps d'execution	1	5.6	8.2	8.8	14.4	59.2	62.8	34.2	64.2	116.2	1725
Fitness	0	0	0	0	0	0	0	0	0	0	1

TABLE 3.1 – Résultats de l'algorithme génétique en fonction de la taille du probleme

Taille du probleme	10	20	30	40	50	60	70	80	90	100	200
Temps d'execution	16	224	348	508	585	904	1029	941	1057	1235	3638
Fitness	0	2	6	9	14	17	21	26	29	34	78

TABLE 3.2 – Résultats de l'algorithme PSO en fonction de la taille du problem

Afin de visualiser les résultats de manière comparative, voici la figure suiantes :

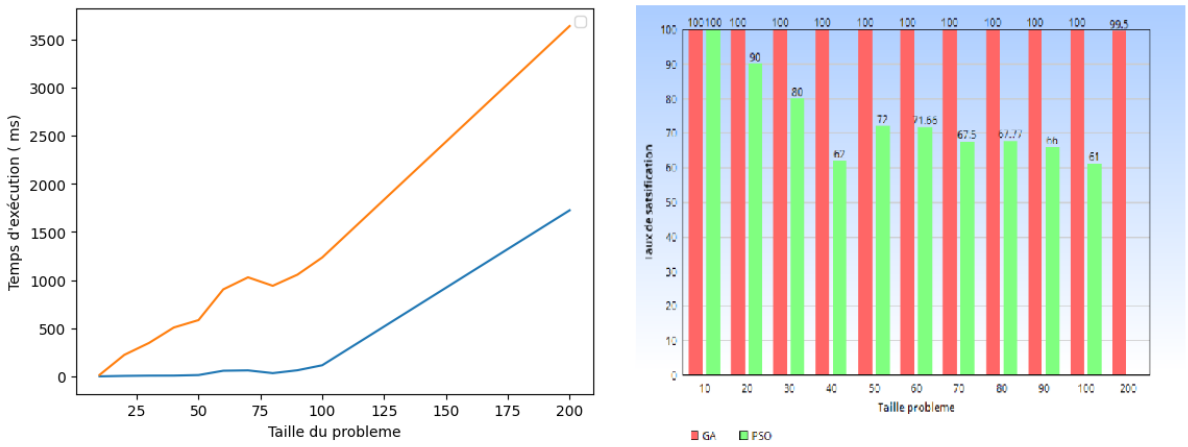

FIGURE 3.1 – Graphique comparatif des temps d'exécution de l'algorithme génétique et de PSO pour différentes tailles de problèmes

D'après les figures, il est clairement observé que le PSO prend beaucoup plus de temps que l'algorithme génétique. En ce qui concerne la qualité de la solution, en examinant le graphe à barres, il est constaté que l'algorithme génétique est capable de trouver des solutions optimales dans un temps raisonnable jusqu'à la taille 100, tandis que le PSO converge vers la solution optimale très lentement.

Conclusion :
L'utilisation de l'algorithme génétique pour le probleme de N-reines est plus adaptée que l'algorithme PSO.

3.2 Comparaison des performances des métaheuristiques et des méthodes exactes

Suite à nos expérimentations dans la première partie du rapport, nous avons conclu que les méthodes exactes sont très coûteuses en termes de temps d'exécution et de mémoire, surtout pour des valeurs élevées de n (par exemple, pour n=30, cela peut prendre plus de 30 minutes). En revanche, les métaheuristiques sont beaucoup plus rapides, bien qu'elles puissent parfois conduire à des solutions non optimales.

Conclusion

En conclusion, il est évident que les méthodes exactes sont très coûteuses en termes de temps d'exécution et de mémoire. Les métaheuristiques, quant à elles, sont des alternatives efficaces, offrant des résultats satisfaisants en termes de qualité de solution dans des temps raisonnables. Cependant, le choix de la métaheuristique utilisée peut avoir une influence significative sur la qualité de la solution obtenue. Il est donc important de choisir la méthode la plus adaptée en fonction du problème étudié.

Il convient toutefois de souligner que les métaheuristiques ont leurs limites et ne garantissent pas toujours la solution optimale. En effet, leur approche peut parfois conduire à des solutions non optimales. De plus, leur performance peut être influencée par la complexité du problème et la qualité de la fonction objectif utilisée. Par conséquent, il est important d'utiliser les métaheuristiques avec d'autres méthodes lorsque cela est possible pour améliorer la qualité de la solution.