



---

# S2.02 : EXPLORATION ALGORITHMIQUE D'UN PROBLEME

---

Ruddy Audigier, Mehdi El Allam, Chahid Diouri



## Table des matières

|  |    |
|--|----|
| Table des matières .....                       | 1  |
| Explication des jeux de test.....              | 2  |
| Méthode pour tester .....                      | 2  |
| 1 <sup>er</sup> jeu de test.....               | 2  |
| 2 <sup>ème</sup> jeu de test.....              | 3  |
| Fiches des heuristiques de niveau 1 .....      | 5  |
| Fiche de l'équilibre progressif .....          | 5  |
| Fiche des extrêmes en premier .....            | 9  |
| Fiche du n-swap .....                          | 13 |
| FICHE DU N-OPT .....                           | 18 |
| Fiche du Draft (heuristique inventé).....      | 22 |
| Fiches des heuristiques de niveau 2 .....      | 26 |
| FICHE DU N-OPT (réadapter) .....               | 26 |
| Fiche des extrêmes en premier (réadapter)..... | 29 |
| Fiches des heuristiques de niveau 3 .....      | 33 |
| FICHE DU N-OPT (réadapter) .....               | 33 |

# **Explication des jeux de test**

## **Méthode pour tester**

L'algo devait répartir un total de X Joueurs. Ces joueurs sont à l'image d'un matchmaking ordinaire dans lequel les niveaux et les personnages sont aléatoires car dans les matchmaking il y'a des gens bons d'autres mauvais, voire très mauvais....

Je vais donc ici pour tester tous les heuristiques avec le jeu de test dans le cas général, prendre au départ une liste de 250 personnages, puis j'enlèverai les 50 derniers joueurs pour en avoir plus que 200 et je répèterai la même manipulation pour avoir 150 et 100 joueurs.

J'ai décidé de faire mes jeux de test ainsi car comme ça on pourra voir la variation sur le temps d'exécution et le score selon le nombre de lignes ainsi que les joueurs qui ne sont pas inclus dans les équipes.

Ensuite, avec mon deuxième jeu de test, je vais tester avec 106 pour avoir une idée de l'efficacité de tous les heuristiques dans le cas où ma répartition n'est vraiment pas bonne car cela peut arriver dans la vraie vie (les vrais joueurs savent ce que ça fait de tomber sur des joueurs nuls car il n'y a que ça de connecter). J'ai pris 106 car l'important ici n'est pas de tester la vitesse d'exécution mais surtout le score.

Et ceci sur tous les jeux de test que je vais faire pour cette SAE. Je souhaite préciser que les jeux de test fait ici donne une image mais les résultats peuvent être légèrement absurde parfois selon les tests.

## **1<sup>er</sup> jeu de test**

### ***Description :***

Pour ce jeu de test, on a généré de manière aléatoire et uniforme un jeu de test par un algorithme en C# pour avoir le cas de base et être le plus proche de la réalité car on ne peut jamais deviner le niveau des joueurs qui se connecte.

Voici ci-dessous la répartition des niveaux principaux au sein de ce jeu de test :

(La répartition est en pourcentage)

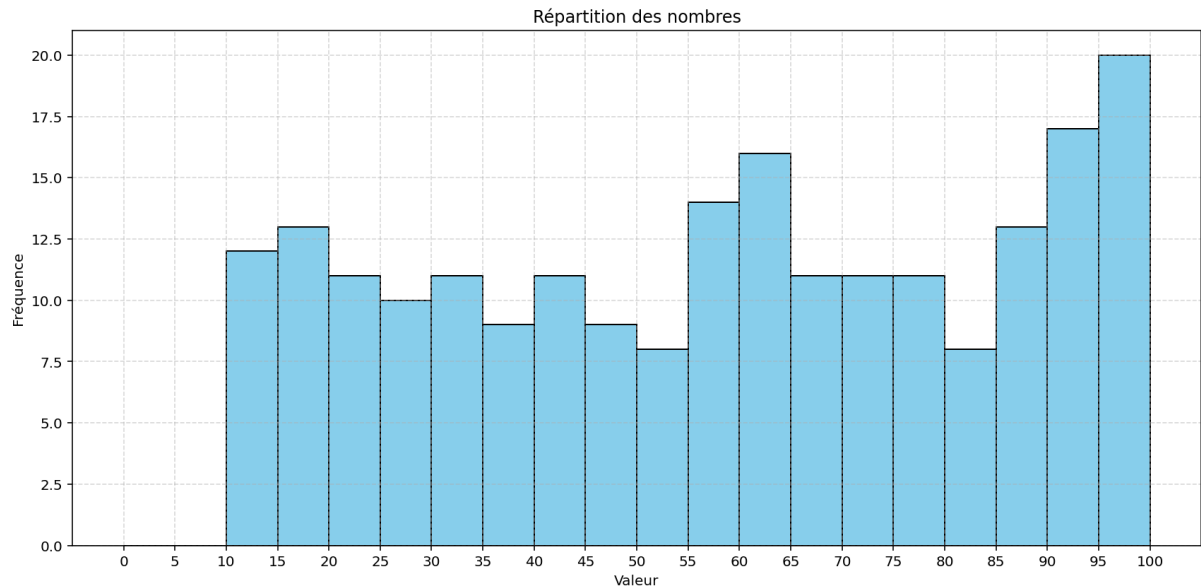


Figure 1 : Histogramme représentant la répartition du niveau des personnages du premier jeu de test

## 2<sup>ème</sup> jeu de test

### **Description :**

On souhaitait un jeu de test qui est un cas particulier. En effet, ce 2<sup>ème</sup> jeu de test correspond au cas où les joueurs ont un niveau très mal réparti et ont majoritairement un niveau inférieur à 50. Ce jeu de test a été effectué, comme le premier, grâce à un algorithme en C#.

Voici ci-dessous la répartition des niveaux principaux au sein de ce jeu de test :

(La répartition est en pourcentage)

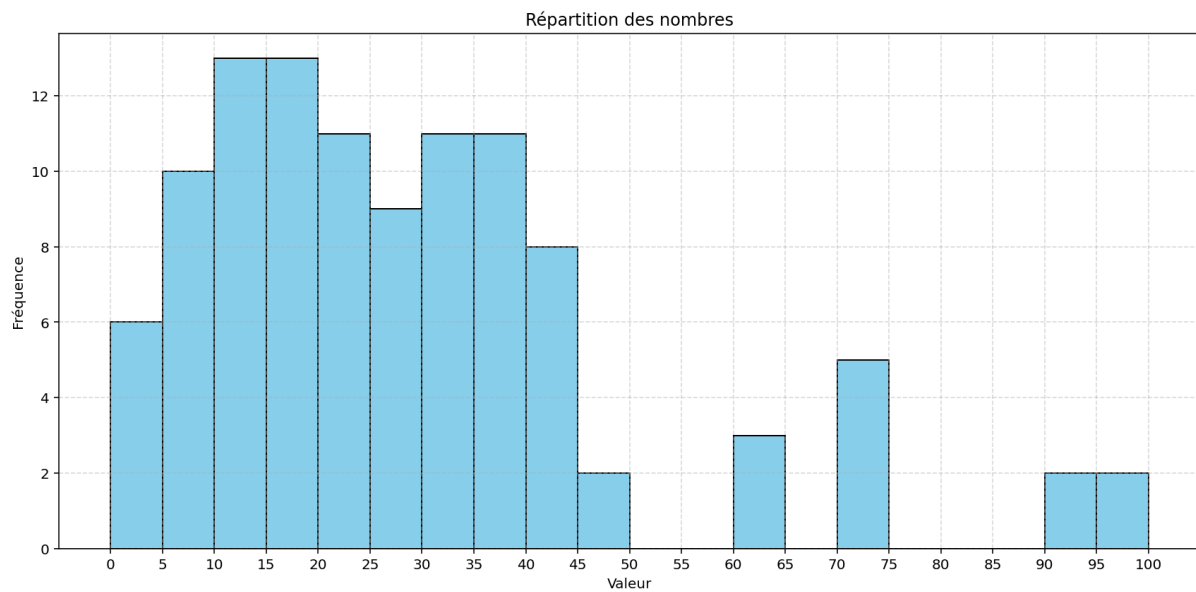


Figure 2 : Histogramme représentant la répartition du niveau des personnages du deuxième jeu de test

# **Fiches des heuristiques de niveau 1**

## **Fiche de l'équilibre progressif**

### ***Objectif de l'équilibre progressif :***

Sélectionner le premier personnage puis on teste et on compare la moyenne de l'équipe si on rajoutait chaque personnage. On prend celui qui donne le meilleur équilibre, c'est-à-dire la moyenne la plus proche de 50.

### ***Méthode implémentée :***

J'ai ajouté dans la classe personnage, une méthode nommée PlusProcheLvl qui renvoie le niveau du personnage qui ramène la moyenne de l'équipe le plus vers 50. En effet, ça compare le niveau du personnage avec lequel on l'appelle puis le niveau du 2<sup>ème</sup> joueur mis en paramètre et il faut mettre le niveau de l'équipe aussi en paramètre.

### ***Exemple cas de figure***

Exemple :

- Équipe A : [40, 45, 60]
- Reste des personnages non pris [20, 15, 64, 82]

La moyenne de l'équipe A est environ égal à 48,3.

Si on prend le personnage niveau 20 :

La moyenne = 41.25

Si on prend le personnage niveau 15 :

La moyenne = 40

Si on prend le personnage niveau 64 :

La moyenne = 52.25

Si on prend le personnage niveau 82 :

La moyenne = 56.75

Ici, on prend le personnage niveau 64 car c'est celui qui permet d'obtenir la moyenne la plus proche de 50.

Et on continue cela jusqu'à ce qu'il y ait plus assez de personnage afin de former une équipe.

### ***Pourquoi choisir ou non cette heuristique ?***

#### *Points positifs :*

- Elle est très forte pour fabriquer les premières équipes de manière optimale
- Assez rapide à programmer

#### *Désavantages :*

- Elle est assez longue à exécuter
- Elle est très mauvaise pour constituer les dernières équipes

### ***Test cas général***

#### *Test à 250 Personnages*

Pour ce test, l'algo devait répartir un total de 250 Joueurs.

**Le score de répartition a été de 1686.6**, avec un résultat par équipe surprenant, le score par équipe était soit excellent soit exécrable. En effet, le résultat est passé de **1 ou 0,1** pour les premières équipes à **2500** pour les dernières.

Il y'a eu donc 2 personnages qui sont dans aucune équipe.

Malgré ce score qui ne semble pas mauvais, (surtout pour 250 joueurs, soit 62 équipes complètes). Le temps de répartition était très rapide, l'algorithme de l'équilibre progressif a pris **1 milliseconde** pour effectuer la répartition ce qui est tout simplement **quasiment instantané**.

### *Test à 200 personnages*

Pour ce test, l'algo devait répartir un total de 200 Joueurs soit « seulement » 50 de moins que le précédent, et les résultats en termes de temps sont étonnants !

**Le score de répartition a été de 33 838.9**, avec un résultat par équipe autant surprenant qu'avant puisque le score par équipe était soit excellent soit exécrable. En effet, le résultat est passé de **1 ou 0,1** pour les premières équipes à **2500** pour les dernières.

Il y'a eu donc 0 personnages qui sont dans aucune équipe.

Le temps de répartition était encore une fois très rapide, l'algorithme de l'équilibre progressif pris **0 milliseconde** pour effectuer la répartition ce qui est instantané (ou presque).

### *Test à 100 personnages*

Pour ce test, l'algo devait répartir un total de 100 Joueurs soit 100 de moins que le précédent, et les résultats en termes de temps sont fou !

**Le score de répartition a été de 6213.4**, avec un résultat par équipe surprenant, le score par équipe est passé de **0 à 2400**.

Il y'a eu donc 0 personnages qui sont dans aucune équipe.

Malgré ce score qui est mauvais, (surtout pour 100 joueurs, soit 25 Teams complètes).

Le temps de répartition est quant à lui bien meilleure, cette heuristique a pris **1 milliseconde** pour effectuer la répartition ce qui est, encore une fois, très rapide, cependant on peut voir une légère incohérence avec le test d'avant puisque c'est une milliseconde de plus.

### *Conclusion test*

On peut conclure tous ces tests avec ce graphique réalisé grâce à d'autres tests :

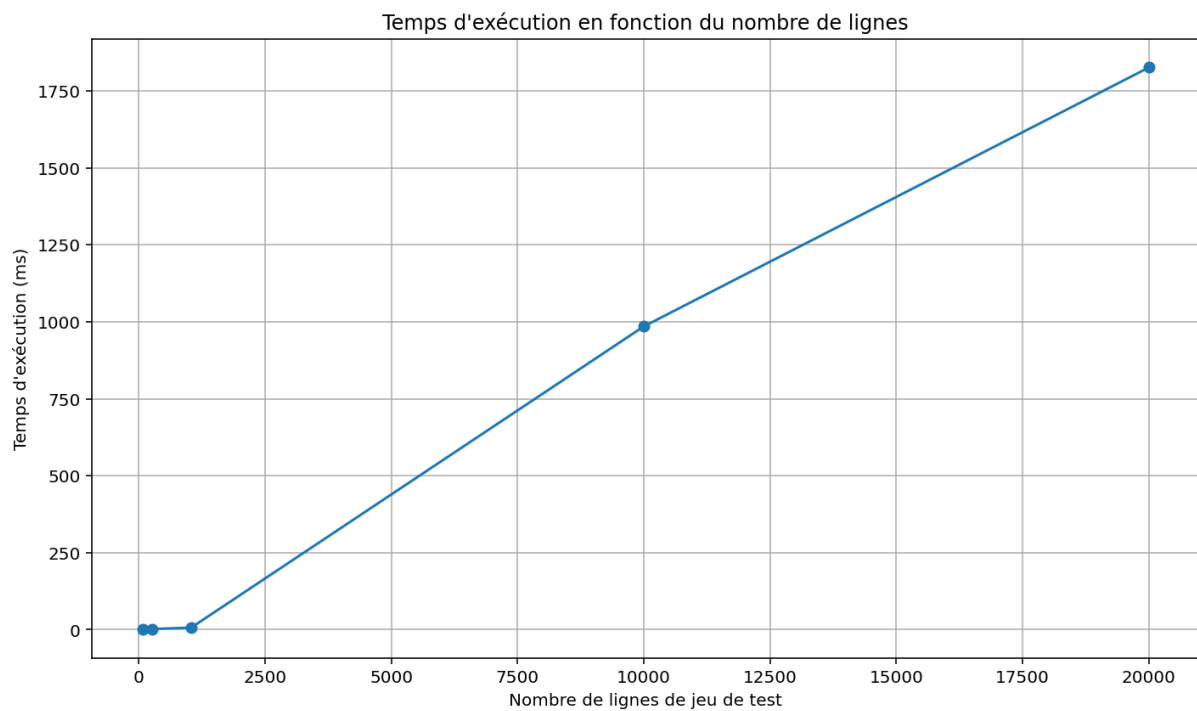


Figure 3 : Graphique linéaire représentant la vitesse d'exécution pour équilibre progressif

## ***Test cas faible***

Le test a donné comme score : **15 763.9**, et en temps : **0 milliseconde**.

## ***Comparaison et analyse des performances***

Les tests réalisés montrent clairement que le l'équilibre progressif n'est pas très performant en tant que répartition puisque les premières équipes vont être parfaites et les dernières vont être catastrophiques. On peut constater que cet algorithme est très rapide mais pas efficace sur les scores qu'elle obtient et ceci dans tous les cas.

## ***Et comparé aux autres heuristiques ?***

En résumé, ça vaut vraiment le coup ?

L'algorithme de l'équilibre progressif peut être intéressante au niveau de sa vitesse d'exécution mais malgré ce point fort cette heuristique est trop peu performante en réalité pour être réellement intéressante. L'idée peut sembler excellente à première vue et pourtant vraiment mauvaise en application.

## **Fiche des extrêmes en premier**

### ***Objectif de l'algorithme les extrêmes en premier :***

Sélectionner dans la liste triée par ordre de niveau du personnage, le personnage pas déjà pris avec le plus bas niveau puis celui avec le plus haut niveau. Faire ça deux fois pour constituer une équipe. Puis crée autant d'équipes que possible (jusqu'à ce qu'il y ait moins que 4 personnages).

### ***Exemple cas de figure***

Exemple :

- Reste des personnages non pris [20, 41, 45, 51, 58, 60, 66, 68, 72]
- On prend 20 et 72 puis 41 et 68. Cela constitue une équipe
- Puis, on prend 45, 66, 51 et 60. Cela constitue une autre équipe
- Equipe A = [20, 72, 41, 68]
- Il reste donc [45, 51, 58, 60, 66]
- Equipe B = [45, 66, 51, 60]
- Il reste le personnage niveau 58 tout seul, sans équipe

### ***Pourquoi choisir ou non cette heuristique ?***

#### ***Points positifs :***

- Elle est très rapide à exécuter
- Elle donne de très bon score de manière générale
- Très facile à programmer

#### ***Désavantages :***

- Elle laisse de côté les personnages avec les niveaux les plus proches de 50

## ***Test cas général***

### *Test à 250 Personnages*

Pour ce test, l'algo devait répartir un total de 250 Joueurs.

**Le score de répartition a été de 1408.9**, avec un résultat par équipe surprenant, le score par équipe est passé de **9 pour les meilleures à 30 pour les pires**.

Il y'a eu donc 2 personnages qui sont dans aucune équipe.

Le temps de répartition était extrêmement rapide, l'heuristique des extrêmes en premier a pris **0 milliseconde** pour effectuer la répartition ce qui est tout simplement **instantané (ou presque)**.

### *Test à 200 personnages*

Pour ce test, l'algo devait répartir un total de 200 Joueurs soit « seulement » 50 de moins que le précédent, et les résultats en termes de temps sont étonnants !

**Le score de répartition a été de 1111.8**, avec un résultat par équipe surprenant, le score par équipe est passé de **142** pour les meilleures à **256** pour les pires.

Il y'a eu donc 0 personnages qui sont dans aucune équipe.

Le temps de répartition est parfait, cet algorithme a pris **0 milliseconde** pour effectuer la répartition ce qui reste élevé.

### *Test à 100 personnages*

Pour ce test, l'algo devait répartir un total de 100 Joueurs soit 100 de moins que le précédent, et les résultats en termes de temps sont fou !

**Le score de répartition a été de 876.2**, avec un résultat par équipe surprenant, le score par équipe allait de **25 à 42**.

Il y'a eu donc 0 personnages qui sont dans aucune équipe.

Le temps de répartition était beaucoup moins long, cet algorithme a pris **0 milliseconde** pour effectuer la répartition ce qui est largement assez rapide.

## Conclusion test

On peut conclure tous ces tests avec ce graphique réalisé grâce à d'autres tests :

(Le temps est en seconde)

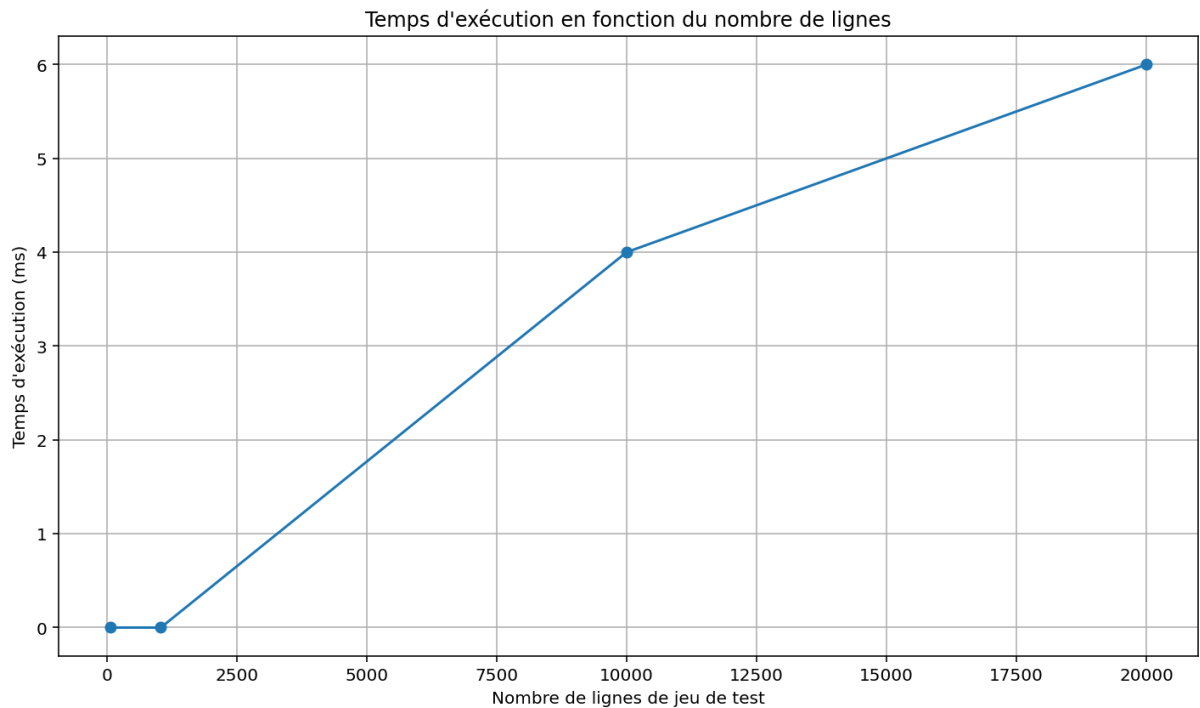


Figure 4 : Graphique linéaire représentant la vitesse d'exécution pour extrêmes en premier

## Test cas faible

Le test a donné comme score : **13 560.9**, et en temps : **0 milliseconde**.

## Comparaison et analyse des performances

Les tests réalisés montrent clairement que l'heuristique des extrêmes en premier donne de plutôt bons résultats, ce qui est étonnant du fait que l'idée soit simple et paraisse nul mais en pratique elle fonctionne très bien. Sauf dans le cas où la répartition est mauvaise, en effet, elle ne donne pas un résultat mauvais pour autant mais pas parmi les plus performant. Il faut souligner que cet algorithme est extrêmement rapide et peut

donc être utile sur un vrai jeu avec énormément de joueur pour pas qu'il y est un surplus de temps d'attente. Cela dit il risque d'être frustrant pour les joueurs puisque ça crée des équipes inégales.

### ***Et comparé aux autres heuristiques ?***

En résumé, ça vaut vraiment le coup ?

Cette heuristique peut être intéressante. C'est pour cela que nous l'avons monté au niveau 2 car elle est très performante en tant que vitesse comparée aux autres et obtient des résultats quasiment similaires voire meilleure.

# Fiche du n-swap

## ***Objectif du n-swap :***

Sélectionner n personnages dans des équipes différentes de façon que leur échange améliore le score global.

Comment choisir les membres à échanger ?

## ***Choix aléatoire***

- Choisir deux équipes aléatoires
- Choisir un personnage au hasard dans chacune
- Tenter l'échange

## ***Différent Swap possibles***

Le nombre définis le nombre de joueur par équipe que l'on va interchanger. Pour des raisons de complexité, nous choisirons entre le 1-swap et le 2-swap.

### ***1-Swap***

Le 1-swap explore un voisinage beaucoup plus petit (échanges simples entre deux personnages), donc c'est beaucoup plus rapide à tester.

### ***2-Swap***

Le 2-swap augmente exponentiellement le nombre de configurations à tester (beaucoup plus lent), ce qui peut vite devenir ingérable si tu as beaucoup de personnages ou d'équipes.

Notre choix sera donc d'implémenter le 1-Swap, ce swap prendra donc 1 joueur parmi 2 équipes distinctes.

## ***Exemple cas de figure***

Échanger 2 personnages de 2 équipes différentes.

Exemple :

- Équipe A : [40, 45, 60, 70]
- Équipe B : [30, 50, 55, 65]

On échange 60 (A) ↔ 55 (B)

Nouvelles équipes :

- A : [40, 45, 55, 70] → moyenne plus proche de 50 ?
- B : [30, 50, 60, 65]

Si le score diminue, on garde l'échange.

## ***Pourquoi choisir cette heuristique ?***

### *Points positifs*

- Elle améliore progressivement la solution en ciblant les pires équipes.
- Elle permet d'échapper à certaines mauvaises répartitions initiales sans tout reconstruire

### *Désavantages :*

- Elle est longue à exécuter
- Elle est complexe à programmer

## ***Test cas général***

### *Test à 250 Personnages*

Pour ce test, l'algo devait répartir un total de 250 Joueurs.

**Le score de répartition a été de 1686.6**, avec un résultat par équipe surprenant, le score par équipe était entre soit **20 et 30**.

Il y'a eu donc 2 personnages qui sont dans aucune équipe.

Malgré ce score qui ne semble pas mauvais, (surtout pour 250 joueurs, soit 62 teams complètes). Le temps de répartition était très long, le N-Swap a pris **1 minute** pour effectuer la répartition ce qui est tout simplement **colossale**.

### *Test à 200 personnages*

Pour ce test, l'algo devait répartir un total de 200 Joueurs soit « seulement » 50 de moins que le précédent, et les résultats en termes de temps sont étonnants !

**Le score de répartition a été de 1100.4**, avec un résultat par équipe surprenant, le score par équipe était entre **20 et 30**.

Il y'a eu donc 0 personnages qui sont dans aucune équipe.

Malgré ce score qui ne semble pas mauvais, (surtout pour 200 joueurs, soit 50 Teams complètes).

Le temps de répartition était nettement moins long, le N-Swap a pris **25 secondes** pour effectuer la répartition ce qui reste élevé.

### *Test à 100 personnages*

Pour ce test, l'algo devait répartir un total de 100 Joueurs soit 100 de moins que le précédent, et les résultats en termes de temps sont fou !

**Le score de répartition a été de 873.7**, avec un résultat par équipe surprenant, le score par équipe était entre **30.0 et 40**.

Il y'a eu donc 0 personnages qui sont dans aucune équipe.

Malgré ce score qui ne semble pas mauvais, (surtout pour 100 joueurs, soit 25 Teams complètes).

Le temps de répartition était beaucoup moins long, le N-Swap a pris **1.2 secondes** pour effectuer la répartition ce qui est raisonnable.

## Conclusion test

On peut conclure tous ces tests avec ce graphique réalisé grâce à d'autres tests :

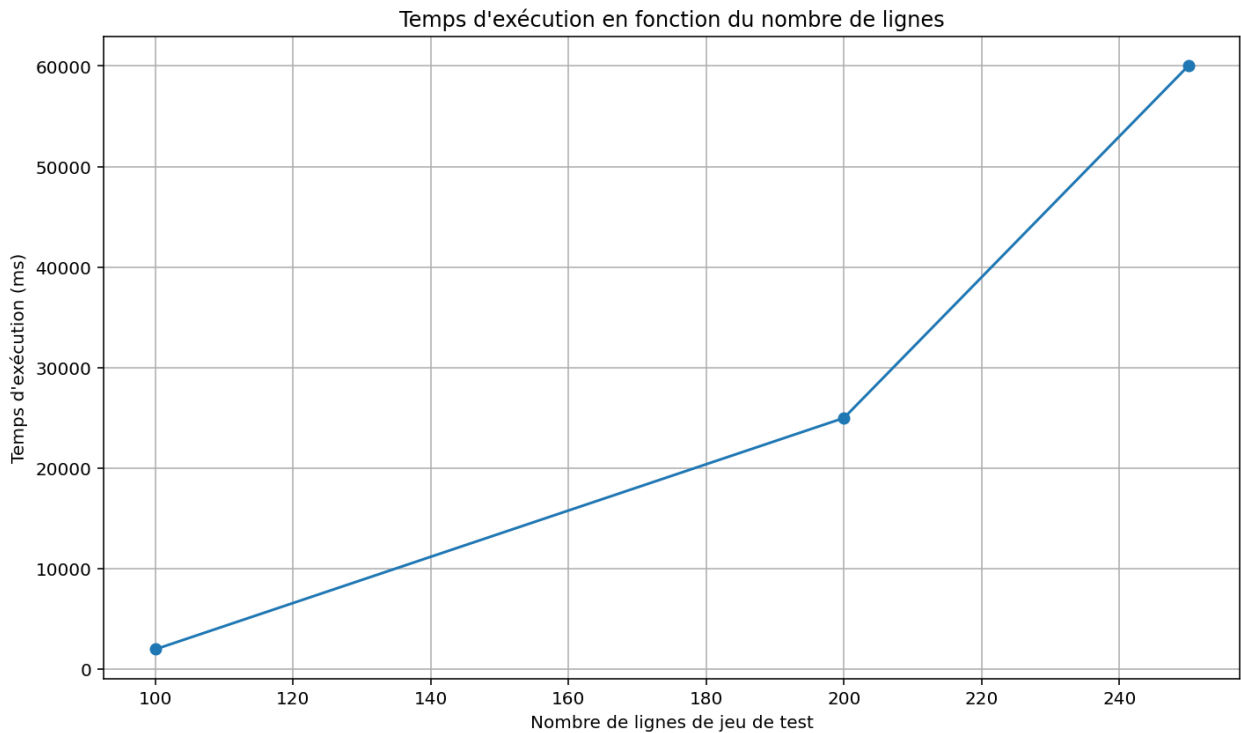


Figure 5 : Graphique représentant le temps d'exécution de NSwap

## Test cas faible

Le test a donné comme score : **12 251.6**, et en temps : **4 secondes**.

## Comparaison et analyse des performances

Les tests réalisés montrent clairement que le n-swap présente une croissance du temps d'exécution très rapide en fonction du nombre de personnages. Si les scores obtenus restent relativement bons et assez stables pour chaque taille d'échantillon, le coût

computationnel devient un réel problème dès que le nombre de personnages dépasse les 150.

### ***Et comparé aux autres heuristiques ?***

En résumé, ça vaut vraiment le coup ?

Le n-swap est une heuristique d'amélioration permettant d'optimiser une répartition d'équipes existante en échangeant jusqu'à n personnages. Malgré son efficacité pour améliorer et équilibrer les équipes, son temps d'exécution augmente rapidement avec le nombre de personnages, rendant son usage coûteux pour de grands ensembles. Dans le cadre de cette SAE, il se révèle utile car il permet d'avoir certes pas optimale mais convenable. Cependant le fait de ne pas pouvoir traiter un grand nombre de personnage, le rend obsolète si on veut pouvoir traiter un grand nombre de joueurs. On peut aussi relever un résultat très bon au deuxième test, relevant l'efficacité de l'algorithme à s'adapter aux différents cas. Cependant, ce score a été obtenue en 4 secondes, ce qui est assez long pour 106 joueurs.

# FICHE DU N-OPT

## ***Objectif du N-Opt :***

Réorganiser la composition de deux équipes entières pour amélieore le score global.

Contrairement au n-swap, on ne fait pas simplement un échange, mais on recombine totalement les membres de deux équipes.

## ***Comment fonctionne le N-Opt :***

- 1- Constituer des équipes « random »
- 2- Choisir deux équipes différentes
- 3- Regrouper leurs membres
- 4- Générer toutes les combinaisons possibles de X équipes à partir des K personnages
- 5- Évaluer le score de chaque combinaison
- 6- Conserver les combinaisons qui réunis, font le meilleur score

## ***Différents Swap possibles***

Là encore, à l'image du n-Swap, On peut décider de mélanger plus ou moins d'équipes. Néanmoins, plus le nombre d'équipes mélangé est élevé, plus il y'a de combinaisons ce qui fait croitre le temps du test des combinaison à vitesse grand V. C'est pourquoi nous allons nous contenter de mélanger 2 équipes et de tester leurs combinaisons. Nous allons donc implémenter le 1-Opt.

## ***Exemple cas de figure***

Répartition initiale :

Équipe A : [40, 45, 60, 70]

Équipe B : [30, 50, 55, 65]

On récupère les 8 joueurs et on teste toutes les façons possibles de les répartir en 2 équipes de 4 joueurs.

Exemples de répartitions testées :

A : [40, 30, 45, 50]

B : [60, 70, 55, 65]

-----

A : [40, 45, 55, 65]

B : [30, 50, 60, 70]

On calcule le score de chaque répartition candidate et on garde la meilleure.

Si la meilleure une des répartitions est meilleure que la solution actuelle, on remplace et on recommence.

### ***Pourquoi choisir cette heuristique ?***

*Points positifs :*

- Le N-OPT explore un voisinage plus large que le 1-swap.
- Il peut complètement « casser » des groupes pour en faire des nouveaux d'un coup.

### ***Test cas général***

*Test à 250 Personnages*

Pour ce test, l'algo devait répartir un total de 250 Joueurs.

**Le score de répartition a été de 1401.2**, avec un résultat par équipe surprenant, le score par équipe était entre **16 et 22 avec majoritairement des 20.2 et des 18.1**.

Il y'a eu donc 2 personnages qui sont dans aucune équipe.

Ce score qui ne semble pas mauvais, (surtout pour 250 joueurs, soit 62 teams complètes). Le temps de répartition était légèrement long, le N-OPT a pris environ **15 secondes** pour effectuer la répartition.

### *Test à 200 personnages*

Pour ce test, l'algo devait répartir un total de 200 Joueurs soit « seulement » 50 de moins que le précédent, et les résultats en termes de temps sont étonnants !

**Le score de répartition a été de 1101.8**, avec un résultat par équipe entre **20 et 26**.

Il y'a eu donc 0 personnages qui sont dans aucune équipe.

Ce score est plutôt bon pour 200 personnages

Le temps de répartition était de **5 secondes**.

### *Test à 100 personnages*

Pour ce test, l'algo devait répartir un total de 100 Joueurs soit la moitié de moins que le précédent, et les résultats en termes de temps sont étonnants !

**Le score de répartition a été de 875.7**, avec un résultat par équipe entre **30 et 42**.

Il y'a eu donc 0 personnages qui sont dans aucune équipe.

Ce score est plutôt bon pour 100 personnages

Le temps de répartition était de **0.6 secondes**.

### *Conclusion test*

On peut conclure avec ce graphique sur le temps d'exécution réalisé grâce à d'autres tests :

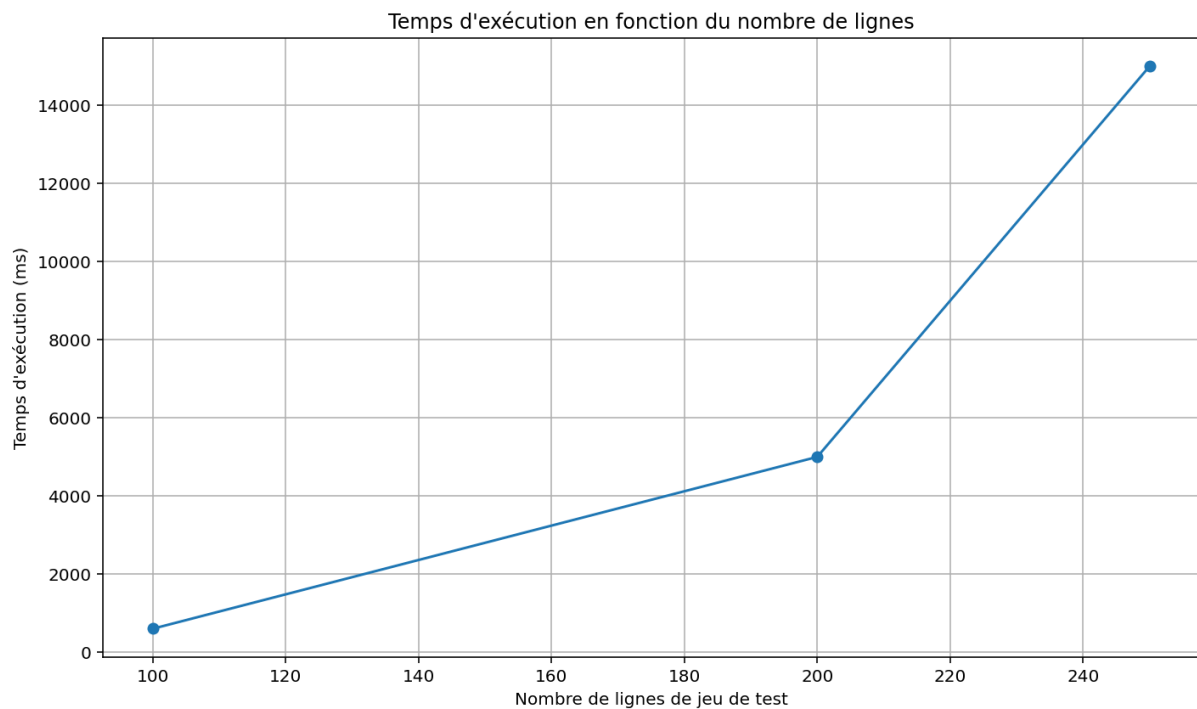


Figure 6 : Graphique représentant le temps d'exécution de N-OPT

### ***Test cas faible***

Le test a donné comme score : **12 253.7**, et en temps : **219 millisecondes**.

### ***Comparaison et analyse des performances***

Les tests réalisés montrent clairement que le n-OPT présente une croissance du temps d'exécution assez rapide. Néanmoins les scores obtenus restent relativement bons et deviennent même meilleur sur un grand échantillon. De plus, cette heuristique est intéressante plus le niveau est élevé puisque grâce au contrainte il y a moins de combinaison possible ce qui accélère le processus tout en gardant l'efficacité. Cela sera donc intéressant de la monter au niveau 2 et au niveau 3 pour voir son efficacité à son plein potentiel. De plus, cela est intéressant car ça donne une très bonne répartition des équipes sans avoir de niveau très haut avec des niveaux très bas.

### ***Et comparé aux autres heuristiques ?***

En résumé, ça vaut vraiment le coup ?

Le n-OPT est une heuristique d'amélioration permettant d'optimiser une répartition d'équipes existante en échangeant les personnages. Malgré son efficacité pour améliorer et équilibrer les équipes, son temps d'exécution augmente rapidement avec le nombre de personnages, rendant son usage coûteux pour de grands ensembles. Dans le cadre de cette SAE, il se révèle utile car il permet d'avoir un résultat certes pas optimal mais convenable. Néanmoins, comme dit précédemment au niveau 2 ou 3 elle sera beaucoup plus intéressante. On peut souligner que cette heuristique a, comme le N-Swap, un score très performant. Néanmoins, le N-OPT l'a fait en 219 millisecondes comparé à 4 secondes pour le N-Swap.

## Fiche du Draft (heuristique inventé)

### **Objectif du draft :**

Création du nombre d'équipe nécessaire, je place un personnage dedans, je test tous les personnages je prends le meilleur puis je passe à l'équipe suivante jusque que tout soit full. Sur le papier cette heuristique semble prometteuse.

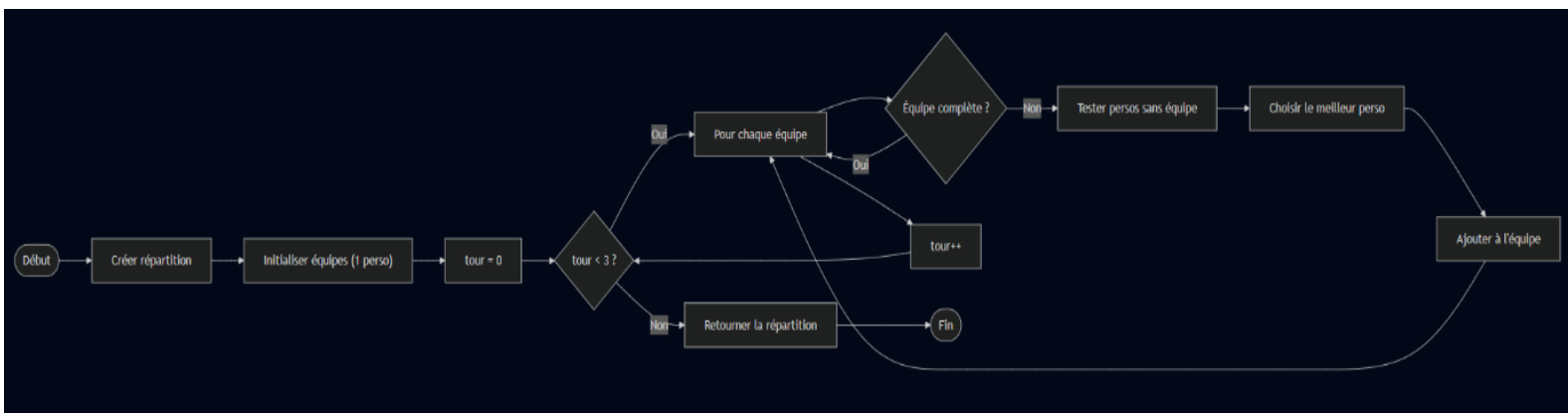


Figure 7 : Diagramme d'activité pour la Draft

### **Exemple cas de figure**

## ***Pourquoi choisir cette heuristique ?***

### *Points positifs*

- L'idée a l'air bonne
- Facile à programmer et à comprendre

### *Désavantages :*

- Elle ne donne pas de si bons résultats que ça

## ***Test cas général***

### *Test à 250 Personnages*

Pour ce test, l'algo devait répartir un total de 250 Joueurs.

**Le score de répartition a été de 4226.2**, avec un résultat par équipe surprenant, le score par équipe est passé **27 à 300**.

Il y'a eu donc 2 personnages qui sont dans aucune équipe.

Malgré ce score qui est moyen, (surtout pour 250 joueurs, soit 62 teams complètes). Le temps de répartition était très long, la Draft a pris **8 millisecondes** pour effectuer la répartition ce qui est correct.

### *Test à 200 personnages*

Pour ce test, l'algo devait répartir un total de 200 Joueurs soit « seulement » 50 de moins que le précédent, et les résultats en termes de temps sont étonnants !

**Le score de répartition a été de 3978.1**, avec un résultat par équipe surprenant, le score par équipe est passé **27 à 300**.

Il y'a eu donc 0 personnages qui sont dans aucune équipe.

Malgré ce score qui est moyen, (surtout pour 200 joueurs, soit 50 Teams complètes).

Le temps de répartition était nettement moins long, la Draft a pris **5 millisecondes** pour effectuer la répartition ce qui reste élevé.

### *Test à 100 personnages*

Pour ce test, l'algo devait répartir un total de 100 Joueurs soit 100 de moins que le précédent, et les résultats en termes de temps sont fou !

**Le score de répartition a été de 3242.4**, avec un résultat par équipe surprenant, le score par équipe est passé **27 à 300**.

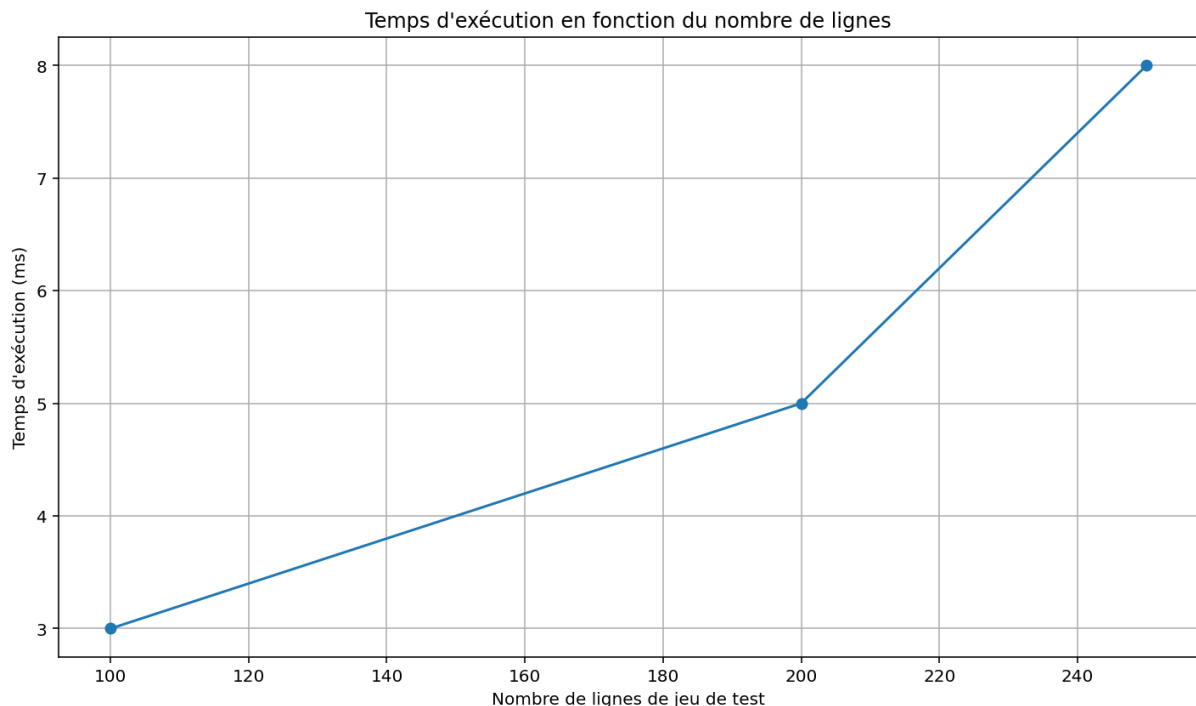
Il y'a eu donc 0 personnages qui sont dans aucune équipe.

Malgré ce score est moyen, (surtout pour 100 joueurs, soit 25 Teams complètes).

Le temps de répartition était beaucoup moins long, la Draft a pris **3 millisecondes** pour effectuer la répartition ce qui est raisonnable.

### *Conclusion test*

On peut conclure tous ces tests avec ce graphique réalisé grâce à d'autres tests :



*Figure 8 : Graphique représentant le temps d'exécution pour Draft*

## ***Test cas faible***

Le test a donné comme score : **12 937.8**, et en temps : **1 milliseconde**.

## ***Comparaison et analyse des performances***

Les tests réalisés montrent clairement que la Draft donne des résultats très rapide mais pas autant que d'autres heuristiques. De plus, elle donne des résultats pas si bons sauf dans le cas d'une mauvaise répartition.

## ***Et comparé aux autres heuristiques ?***

En résumé, ça vaut vraiment le coup ?

Cet algorithme n'est pas performant a cause du score. Elle n'a pas donc trop d'avantages comparé aux autres malgré une idée qui semblait bonne au départ.

## Fiches des heuristiques de niveau 2

### FICHE DU N-OPT (réadapter)

#### ***Objectif du N-Opt :***

Réorganiser la composition de deux équipes entières pour améliorer le score global.

Contrairement au n-swap, on ne fait pas simplement un échange, mais on recombine totalement les membres de deux équipes. Comme au niveau 1 mais maintenant on prend en compte le rôle principal pour obtenir une équipe valide. Donc quand on mélange, on échange que les mêmes rôles entre eux.

#### ***Test cas général***

##### *Test à 250 Personnages*

Pour ce test, l'algorithme devait répartir un total de 250 Joueurs.

**Le score de répartition a été de 1876** avec un résultat par équipe surprenant, le score par équipe était entre 20 et 25.

Il y'a eu donc 2 personnages qui sont dans aucune équipe.

Ce score qui ne semble pas mauvais, (surtout pour 250 joueurs, soit 62 équipes complètes). Le temps de répartition n'était pas très long, le N-OPT a pris **5 secondes** pour effectuer la répartition ce qui est raisonnable.

##### *Test à 200 personnages*

Pour ce test, l'algorithme devait répartir un total de 200 Joueurs soit « seulement » 50 de moins que le précédent, et les résultats en termes de temps sont étonnants !

**Le score de répartition a été de 1628.9**, avec un résultat par équipe entre 20 et 30.

Il y'a eu donc 0 personnages qui sont dans aucune équipe.

Ce score est plutôt bon pour 200 personnages

Le temps de répartition était de **2.3 secondes**.

### *Test à 100 personnages*

Pour ce test, l'algo devait répartir un total de 100 Joueurs soit la moitié de moins que le précédent, et les résultats en termes de temps sont étonnants !

**Le score de répartition a été de 1210.2**, avec un résultat par équipe entre 30 et 40.

Il y'a eu donc 0 personnages qui sont dans aucune équipe.

Ce score est plutôt bon pour 100 personnages

Le temps de répartition était de **0.2 secondes**.

### *Conclusion test*

On peut conclure tous ces tests avec ce graphique réalisé grâce à d'autres tests :

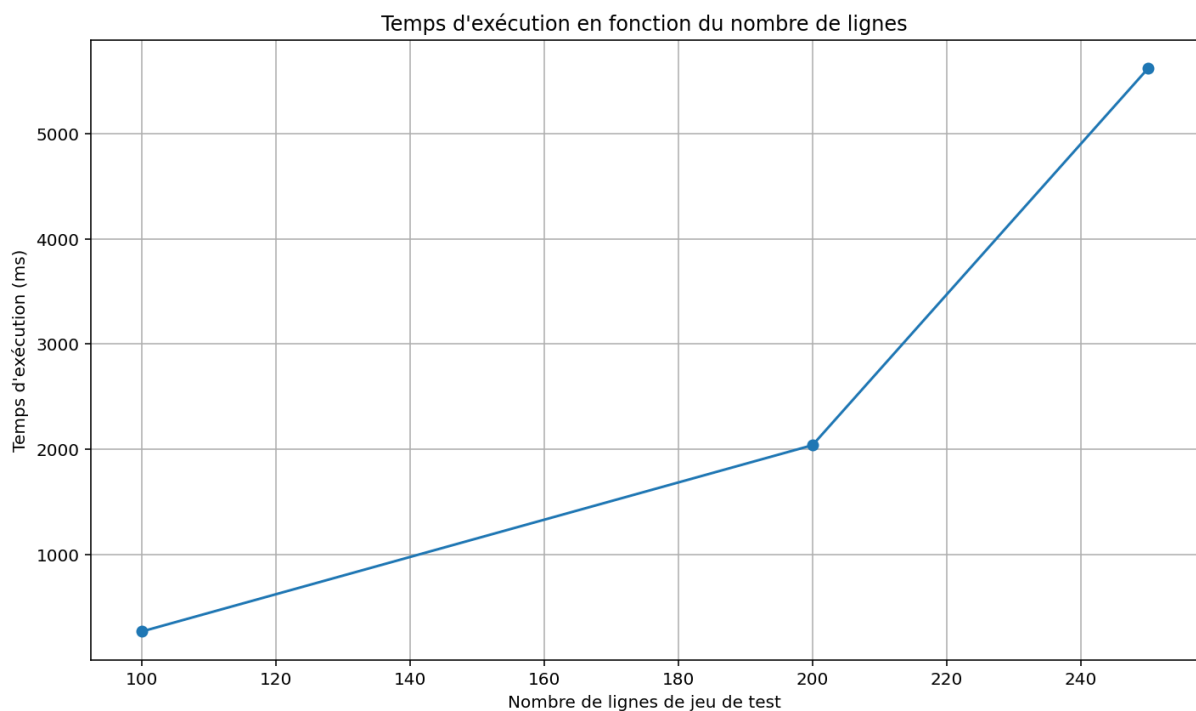


Figure 9 : Graphique représentant le temps d'exécution pour N-OPT lvl2

### ***Test cas faible***

Le test a donné comme score : **11 396.9**, et en temps : **111 millisecondes**.

# Fiche des extrêmes en premier Niveau 2 (réadapter)

## *Fonctionnement*

Pour adapter l'algorithme Extrême en Premier au niveau 2, nous avons décidé d'agir sur la liste de départ en faisant en sorte de trier les éléments pour que des répartitions valides soit sélectionné dès le départ.

Concrètement pour ce faire j'ai trié les personnages de mon jeu d'essai par groupe de classe ( 3 listes au total : tanks, support et dps). J'ai trié ces trois listes dans l'ordre croissant. J'ai ensuite créé une liste intermédiaire dans laquelle j'ai alterné tank et support jusqu'à vider les deux listes afin de créer une liste qui ressemble à ça :

T-S-T-S-T-S-T-S

Une fois que c'est fait et que tous mes supports et tanks sont dans la liste, j'ai ajouté à la suite tous mes dps dans l'ordre croissant. J'ai ensuite transposé l'entièreté de ma liste complète de personnages retrié dans mon tableau sur lequel je vais appliquer l'extrême premier. Ce tri permet de garder le concept du Extrême premier ( trié par ordre croissant ) mais en faisant en sorte au moment de prendre les 2 premiers et 2 derniers éléments d'avoir des groupes valides. En plus vu que les Tanks supports ont été triés par ordre croissant et les dps aussi, cette configuration permet bel et bien d'avoir 2 perso bas niveau et 2 perso haut niveau. Grâce au tri préalable, la création des équipes se fera pareil que pour l'extrême premier de base. J'ai ajouté un test de vérification du nombre de tanks ;dps et support avant l'ajout d'une équipe à la répartition car s'il n'y a plus assez de personnages d'une certaine classe, les équipes ne doivent pas être créées.

## *Test à 250 Personnages*

Pour ce test, l'algo devait répartir un total de 250 Joueurs.

**Le score de répartition a été de 1671.4**, avec un résultat par équipe surprenant, le score par équipe est passé de **4 à 30**.

Il y'a eu donc 2 personnages qui sont dans aucune équipe.

Le temps de répartition était extrêmement rapide, l'heuristique des extrêmes en premier a pris **4 millisecondes** pour effectuer la répartition ce qui est tout simplement **instantané (ou presque)**.

### *Test à 200 personnages*

Pour ce test, l'algo devait répartir un total de 200 Joueurs soit « seulement » 50 de moins que le précédent, et les résultats en termes de temps sont étonnants !

**Le score de répartition a été de 1172.8**, avec un résultat par équipe surprenant, le score par équipe est passé de **10** pour les meilleures à **20** pour les pires.

Il y'a eu donc 0 personnages qui sont dans aucune équipe.

Malgré ce score qui est mauvais, (surtout pour 200 joueurs, soit 50 Teams complètes).

Le temps de répartition est parfait, cet algorithme a pris **5 millisecondes** pour effectuer la répartition ce qui est instantané et pas beaucoup moins lent que le test à 100 personnages.

### *Test à 100 personnages*

Pour ce test, l'algo devait répartir un total de 100 Joueurs soit 100 de moins que le précédent, et les résultats en termes de temps sont fou !

**Le score de répartition a été de 1369.9**, avec un résultat par équipe surprenant, le score par équipe allait de **25 à 121**.

Il y'a eu donc 0 personnages qui sont dans aucune équipe.

Malgré ce score qui ne semble pas mauvais, (surtout pour 100 joueurs, soit 25 Teams complètes).

Le temps de répartition était beaucoup moins long, cet algorithme a pris **4 millisecondes** pour effectuer la répartition ce qui est largement assez rapide.

### *Conclusion test*

On peut conclure tous ces tests avec ce graphique réalisé grâce à d'autres tests :

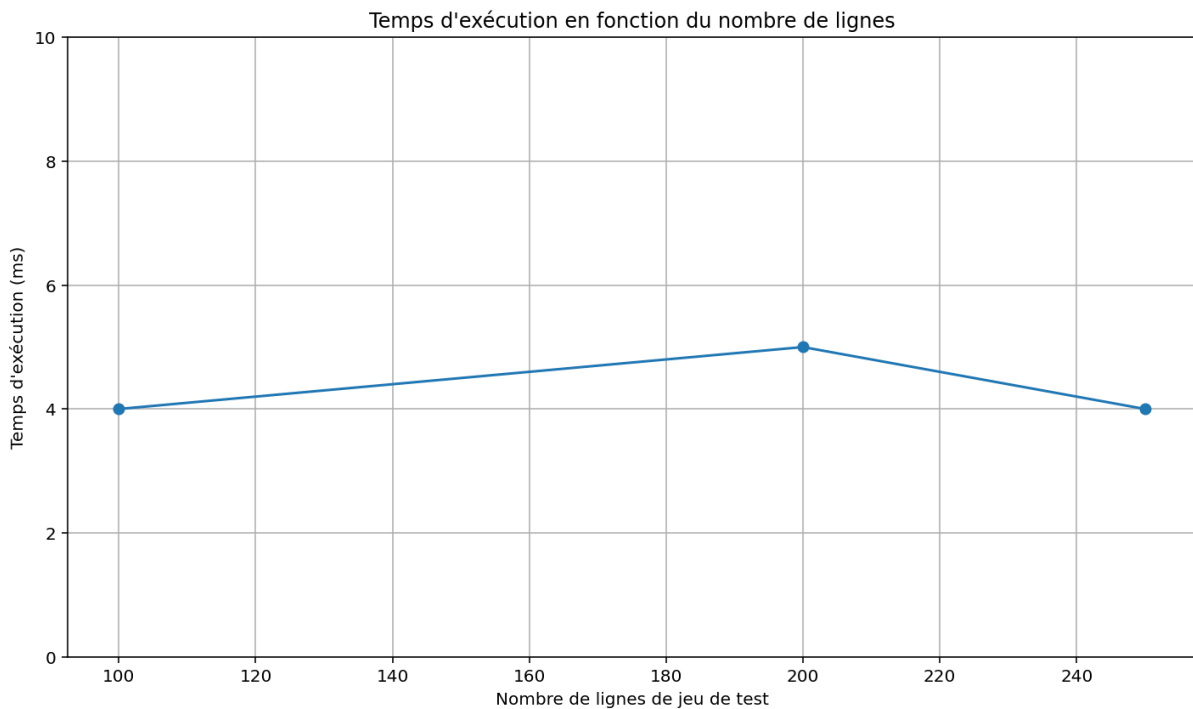


Figure 10 : Diagramme représentant le temps d'exécution pour les extrêmes en premier lvl2

### ***Test cas faible***

Le test a donné comme score : **14 990.9**, et en temps : **0 milliseconde**.

### ***Comparaison et analyse des performances***

Les tests réalisés montrent clairement que l'algorithme des extrêmes en premier possède toujours l'avantage d'être très rapide en temps d'exécution et assez bon en tant que score. Cependant, il y a toujours le problème de répartition au sein des équipes.

### ***Et comparé aux autres heuristiques ?***

En résumé, ça vaut vraiment le coup ?

Cette heuristique est très rapide mais moins performant que le N-OPT qu'on a vu. En effet, le N-OPT est un poil plus long mais est très efficace et a une meilleure

répartition au sein des équipes. Ce qui permettra aux joueurs de ne pas éclater leur clavier par terre à cause de leurs coéquipiers (on connaît la sensation).

# Fiches des heuristiques de niveau 3

## FICHE DU N-OPT (réadapter)

### *Test cas général*

#### *Test à 250 Personnages*

Pour ce test, l'algo devait répartir un total de 250 Joueurs.

**Le score de répartition a été de 1027.6**, avec un résultat par équipe surprenant, le score par équipe était entre 0 et 1.

Il y'a eu donc 8 personnages qui sont dans aucune équipe. A cause des rôles.

Ce score est excellent pour ce problème et le nombre de joueurs.

Le temps de répartition n'était pas très long, le N-OPT a pris **33 secondes** pour effectuer la répartition ce qui est un peu long.

#### *Test à 200 personnages*

Pour ce test, l'algo devait répartir un total de 200 Joueurs soit « seulement » 50 de moins que le précédent, et les résultats en termes de temps sont étonnants !

**Le score de répartition a été de 810.6**, avec un résultat par équipe entre **0 et 1**.

Il y'a eu 8 personnages qui sont dans aucune équipe.

Ce score est très bon pour 200 personnages

Le temps de répartition était de **16 secondes**.

#### *Test à 100 personnages*

Pour ce test, l'algo devait répartir un total de 100 joueurs.

**Le score de répartition a été de 410.4**, avec un résultat par équipe entre **0 et 1**.

Le temps de répartition était super court, le N-Opt a pris **1 seconde** pour effectuer la répartition ce qui est vraiment raisonnable.

### *Conclusion test*

On peut conclure avec le graphique juste en-dessous :

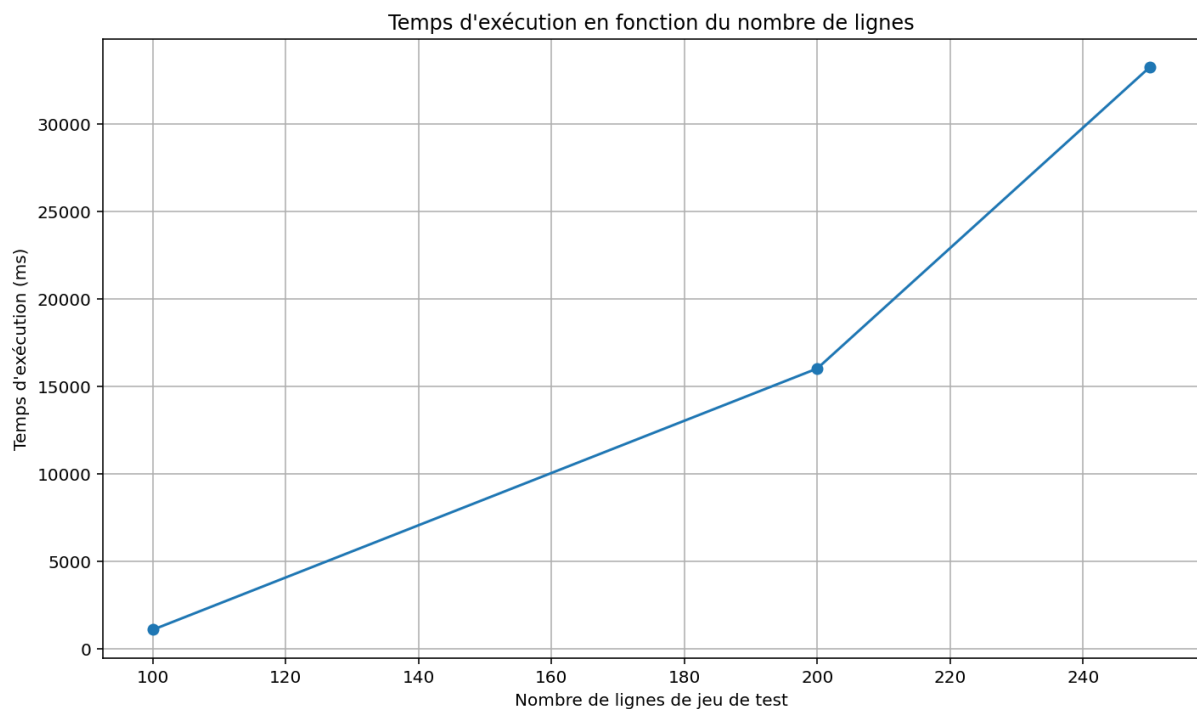


Figure 11 : Graphique représentant le temps d'exécution du N-OPT lvl3

### *Test cas faible*

Le test a donné comme score : **7455.4**, et en temps : **1.4 seconde**.

### *Comparaison et analyse des performances*

Les tests réalisés montrent clairement que l'algorithme N-OPT est très bons pour répartir les personnages. Son seul point faible réside dans sa lenteur d'exécution. Mais,

au final, je pense que les joueurs resteront plus sur un jeu qui met certes un peu plus de temps à répartir les équipes mais qui une fois lancé est excellent et où ils peuvent se régaler avec des bons coéquipiers.