

Filières MP/MPI - ENS de Paris-Saclay, Lyon, Rennes et Paris - Session 2024

Page de garde du rapport de TIPE

NOM : MONASSON	Prénoms :
Classe : MPI [*]	Bruno
Lycée : Janson de Sailly	Numéro de candidat :
Ville : Paris	27606

Concours auxquels vous êtes admissible, dans la banque MP Inter-ENS (les indiquer par une croix) :

ENS Cachan	MP - Option MP		MP - Option MPI	
	Informatique	X		
ENS Lyon	MP - Option MP		MP - Option MPI	
	Informatique - Option M	X		Informatique - Option P
ENS Rennes	MP - Option MP		MP - Option MPI	
	Informatique	X		
ENS Paris	MP - Option MP		MP - Option MPI	
	Informatique			

Matière dominante du TIPE (la sélectionner d'une croix inscrite dans la case correspondante) :

Informatique	X	Mathématiques		Physique	
--------------	---	---------------	--	----------	--



Titre du TIPE : Algorithmes heuristiques pour le jeu << Wordle >>

Nombre de pages (à indiquer dans les cases ci-dessous) :

Texte	5	Illustration	6	Bibliographie	1
-------	---	--------------	---	---------------	---

Attention, les illustrations doivent figurer dans le corps du texte et non en fin du document !

Résumé ou descriptif succinct du TIPE (6 lignes, maximum) :
 Mon projet TIPE consiste en la conception et l'implémentation d'algorithmes pour le jeu << Wordle >>. J'ai d'abord développé des stratégies naïves puis je me suis intéressé à d'autres stratégies plus complexes, reposant sur la théorie de l'information entre autres. Enfin, j'ai étudié le compromis entre temps d'exécution et performance et mené une analyse théorique de mes algorithmes.

A Paris	Signature du professeur responsable de la classe préparatoire dans la discipline	Cachet de l'établissement
Le 05/06/2024		LYCEE JANSON DE SAILLY 106, rue de la Pompe 75775 PARIS Cedex 16 Tél. 01.55.73.28.58 - Fax 01.55.73.28.50 075 0899 C
Signature du (de la) candidat(e)		

Algorithmes heuristiques pour le jeu *Wordle*

Bruno Monasson, candidat n° 27607

TIPE ENS 2024

1 Introduction

1.1 Présentation du Wordle

Le *Wordle* est un jeu dont l'objectif est de trouver un mot mystère de cinq lettres en au plus six essais, inspiré du jeu télévisé *Motus*. À chaque essai (qui est nécessairement un mot valide), l'utilisateur est informé des lettres qu'il a réussi à deviner et si elles sont placées au bon endroit :

- Les lettres qui ne sont pas présentes dans le mot mystère sont coloriées en gris
- Les lettres présentes dans le mot mystère sont coloriées en vert si elles sont au bon emplacement, en jaune sinon

M	A	T	H	S
S	O	R	T	E
S	P	O	R	T

FIGURE 1 – Exemple de partie si le mot mystère est « SPORT » dans laquelle le joueur gagne au troisième essai

La difficulté de ce jeu réside dans le grand nombre de mots (de l'ordre de quelques milliers) à élaguer en quelques essais seulement. Ceci explique sa popularité qui a atteint plusieurs millions de joueurs quotidiens, faisant de « Wordle » le résultat le plus recherché en 2022 sur le moteur de recherche Google.

1.2 Définitions et notations

Définissons tout d'abord quelques notations générales et termes (les concepts plus spécifiques seront définis par la suite) :

- On notera Σ l'alphabet sur lequel nous travaillerons. En pratique on aura toujours $\Sigma = \{a, b, \dots, z\}$.
- La notation $|\cdot|$ désignera le cardinal d'un ensemble
- On notera $|u|_a$ le nombre d'occurrences de la lettre $a \in \Sigma$ dans le mot $u \in \Sigma^*$
- Si $u \in \Sigma^*$ et $0 \leq i \leq |u| - 1$, on notera $u_i \in \Sigma$ la lettre numéro i du mot u
- Tous les mots seront de la même longueur (la taille de la grille), notée ℓ . Ici, $\ell = 5$
- On notera \mathcal{D} un *dictionnaire*, i.e. un ensemble de mots de longueur ℓ . Formellement, $\mathcal{D} \subset \Sigma^\ell$. Nous travaillerons principalement sur $\mathcal{D}_{\text{anglais}}$ l'ensemble des mots de longueur 5 en anglais. Toutefois le Wordle se joue sur un dictionnaire réduit $\mathcal{D}_{\text{officiel}}$ avec uniquement des mots communs. On a $|\mathcal{D}_{\text{anglais}}| = 14855$ et $|\mathcal{D}_{\text{officiel}}| = 2315$.
- $u \in \Sigma^*$ est un mot *valide* pour le dictionnaire \mathcal{D} si $u \in \mathcal{D}$
- À chaque essai l'ordinateur colorie les cases comme expliqué dans l'introduction. Nous appellerons ce coloriage *réponse*. Pour simplifier les notations, nous assimilerons le gris au chiffre 0, le jaune à 1 et le vert à 2. L'ensemble des réponses est donc $\mathcal{R} = \{0, 1, 2\}^\ell$

1.3 Position du problème

Nous essayerons de mettre au point une stratégie minimisant le nombre d'essais moyen quand le mot mystère parcourt l'ensemble du dictionnaire. Étant donné un dictionnaire il existe nécessairement une stratégie optimale vu qu'il existe un nombre fini de stratégies possibles. Il est alors possible de trouver cette stratégie optimale à l'aide d'une recherche exhaustive. Dans de nombreux jeux tels que les échecs ou le Go, parcourir toutes les parties est impossible à cause de leur trop grand nombre. Néanmoins, il est possible de résoudre le Wordle sur les dictionnaires réduits tels que $\mathcal{D}_{\text{officiel}}$. Par exemple, en 2022 deux chercheurs du MIT ont résolu le jeu grâce à la programmation dynamique.

Toutefois, cela ne sera pas le cadre de travail de ce TIPE. En effet, nous mettrons au point des algorithmes qui ne nécessitent pas de pré-calcul, prenant directement le dictionnaire en entrée à chaque partie. Cela nous permettra de tester nos algorithmes sur différents dictionnaires et notamment ceux complets tels que $\mathcal{D}_{\text{anglais}}$. La recherche exhaustive étant ainsi impossible car trop lente, nous nous intéresserons alors à l'élaboration de stratégies approchées et au compromis entre performances et temps d'exécution.

L'ensemble de ces algorithmes sera codé dans le langage C. Tous les fichiers codes sont disponibles sur cette page GitHub spécialement créée : <https://github.com/bmonasson/Wordle>.

2 Une première approche : heuristique naïve

2.1 Heuristique 1 (H1)

Ma première approche fut de mettre au point une heuristique. À chaque essai on parcourt l'entièreté des mots du dictionnaire et on joue le mot qui maximise la fonction **score**. Cette première heuristique correspond à la manière naïve de jouer :

- si on trouve une lettre et son emplacement (coloriée en vert) on continue de jouer cette lettre à cet emplacement
 - ↳ +15 points par lettre verte utilisée
- si on trouve une lettre mais pas son emplacement (coloriée en jaune) on continue de jouer cette lettre mais à d'autres emplacements
 - ↳ -5 points par lettre jaune non utilisée
- on évite de jouer des lettres si l'on sait, grâce aux essais précédents, qu'elles ne sont pas à cet emplacement dans le mot mystère (cela correspond aux lettres grises et aux lettres jaunes dans les emplacements qu'on a déjà essayé)
 - ↳ -5 points par lettre à un mauvais emplacement
- on essaye de jouer des mots avec le plus de lettres distinctes possibles pour obtenir le plus de renseignements possibles sur les lettres composant le mot mystère
 - ↳ +2 points par lettre distincte

Ces objectifs étant classés en ordre d'importance décroissante, on les pondère avec des coefficients de plus en plus faibles en valeur absolue. La valeur exacte de ces coefficients est arbitraire et ce qui importe est leur rapport. En effet, il est nécessaire de mettre des coefficients plus grands pour les premiers objectifs sinon les essais de l'ordinateur ne vont pas converger vers le mot mystère. Par exemple, si le dernier des quatre coefficients est démesurément grand, l'ordinateur risque de jouer uniquement des mots avec cinq lettres distinctes et de ne jamais trouver le mot mystère si celui-ci possède une lettre en double.

Le calcul du **score** sur deux exemples est explicité en annexe dans la figure 8. Il est intéressant de noter que ce calcul se fait en $\mathcal{O}(l)$. Alors le calcul du meilleur mot (celui qui maximise **score**) se fait en $\mathcal{O}(\ell \times |\mathcal{D}|)$. La complexité est donc linéaire en $|\mathcal{D}|$ (si on considère que ℓ est une constante).

2.2 Heuristique 2 (H2)

J'ai ensuite essayé d'améliorer cette heuristique H1 en prenant en compte la fréquence des lettres dans le dictionnaire. J'ai légèrement modifié la fonction **score** de l'heuristique H1 pour encourager l'ordinateur à jouer des lettres fréquentes. On définit ainsi la fonction **score2** qui prend en entrée $u \in \mathcal{D}$:

$$\text{score2}(u) = \text{score}(u) + \sum_{i=0}^{\ell-1} \text{frequence}(u_i)$$

où **frequence** est définie par $\text{frequence}(a) = \frac{1}{\ell \times |\mathcal{D}|} \sum_{u \in \mathcal{D}} |u|_a$, pour $a \in \Sigma$

On remarque que $\text{frequence}(a) \in [0, 1]$ et donc que ce facteur ne va pas avoir un poids capital dans le score total et va principalement servir pour départager les mots qui avaient le même score selon la précédente heuristique.

On notera qu'il est préférable de calculer dès le début de la partie `frequence(a)` pour tout $a \in \Sigma$ et d'enregistrer ces valeurs dans un tableau plutôt que de recalculer cela à chaque fois. Ce pré-calcul s'effectue en $\mathcal{O}(|\Sigma| + \ell \times |\mathcal{D}|)$. En effet, on crée un tableau de taille $|\Sigma|$ et on le remplit en parcourant toutes les lettres de tous les mots de \mathcal{D} . Alors la complexité de `score2` sera en $\mathcal{O}(\ell)$, comme pour `score1`. Le calcul nécessaire pour trouver le meilleur mot (qui s'effectue à chaque essai) se fait donc en $\mathcal{O}(\ell \times |\mathcal{D}|)$

Cette nouvelle heuristique a permis à l'ordinateur de trouver plus de mots rapidement mais a empiré un phénomène que nous allons mettre en avant dans le paragraphe suivant, rendant cette heuristique *in fine* moins performante pour de grands dictionnaires.

2.3 Résultats et limitations

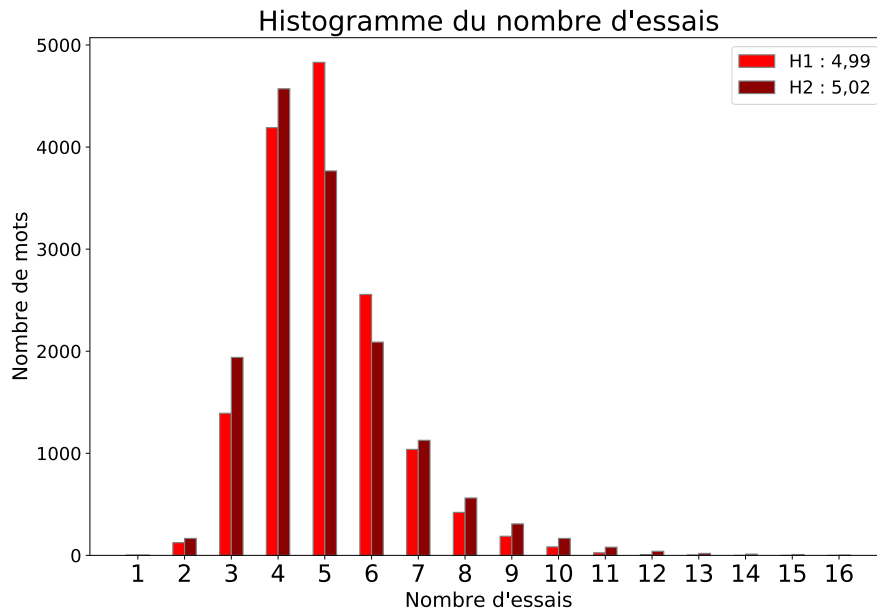


FIGURE 2 – Résultats de l'heuristique 1 et 2 sur $\mathcal{D}_{\text{anglais}}$

Les deux heuristiques H1 et H2 ont obtenu des résultats plutôt satisfaisants étant donné leur rudimentarité, trouvant le mot mystère en cinq essais en moyenne. Toutefois comme nous pouvons le voir sur la figure 2, et de manière plus précise sur les résultats complets en annexe, un nombre non négligeable de mots sont trouvés en plus de dix essais. On voit aussi que la deuxième heuristique trouve plus de mots rapidement (en 3 ou 4 essais) mais aussi beaucoup plus en 10 essais ou plus, comparé à la première heuristique.

Étudions l'exemple suivant, illustré dans la figure 3, pour comprendre l'origine de ce phénomène : Imaginons que, grâce aux essais précédents, nous avons déterminé les quatre dernières lettres du mot mystère qui est sous la forme « ?ONDE ». Il semble alors que nous soyons très proches de trouver la solution. Toutefois, il y a en réalité plusieurs mots mystère possibles : BONDE, FONDE, CONDE, MONDE, PONDE, RONDE, SONDE, TONDE. et l'heuristique va toutes les essayer l'une après l'autre. Dans cette situation il est préférable de faire des essais qui n'ont aucune chance d'être le mot mystère mais qui permettent d'éliminer plus de candidats simultanément.

Cela m'a donc amené à mettre au point un deuxième algorithme qui met en pratique cette autre stratégie.

3 Une seconde approche : théorie de l'information de Shannon

3.1 Algorithme d'élimination (E2)

Définissons tout d'abord la notion de *candidat*. Lors d'une partie, un candidat est un mot qui pourrait être le mot mystère, c'est à dire qu'il ne contredit pas les informations récupérées jusque là. Par exemple, lors de la partie de la figure 1, après avoir joué « SORTE » deux candidats sont « STORY » ou « SPORT » mais « SHORT » n'est pas un candidat (le H a été colorié en gris au premier essai) et « STOCK » non plus (ne

...					...				
?	O	N	D	E	?	O	N	D	E
B	O	N	D	E	P	A	R	C	S
F	O	N	D	E	M	O	T	I	F
C	O	N	D	E	T	O	N	D	E
M	O	N	D	E					
P	O	N	D	E					
R	O	N	D	E					
S	O	N	D	E					
T	O	N	D	E					

Élimine : **P, R, C, S**
Élimine : **M, F, B**

FIGURE 3 – Illustration du phénomène de "l'effet tunnel"

contient pas le R colorié en jaune au deuxième essai). Notons \mathcal{C} un ensemble des candidats (le contexte sera toujours explicite).

À chaque essai on cherche donc à jouer le mot qui va éliminer le plus de candidats possibles. Les candidats éliminés dépendent entièrement de la réponse du jeu à l'essai. Par exemple si on joue l'essai $u \in \mathcal{D}$ et que la réponse du jeu est $(2, 2, \dots, 2)$ (c'est à dire que les l cases sont coloriées en vert), alors nécessairement le mot mystère est u . Néanmoins cette réponse du jeu a très peu de chance de se produire car elle advient uniquement si u est le mot mystère. De manière générale si on joue $u \in \mathcal{D}$ et la réponse du jeu est $r \in \mathcal{R}$ alors en notant \mathcal{C} l'ensemble des candidats avant l'essai u et \mathcal{C}_r l'ensemble des candidats après la réponse r , le nombre de candidats éliminés par l'essai u sera de $|\mathcal{C}| - |\mathcal{C}_r|$. De plus, la réponse à l'essai u est r si et seulement si le mot mystère appartient à \mathcal{C}_r . Alors la probabilité que la réponse soit r est $\frac{|\mathcal{C}_r|}{|\mathcal{C}|}$ (tous les mots ont une équiprobabilité d'être le mot mystère). Cela nous amène donc à la formule suivante

$$\text{nombre_éliminés2}(u, \mathcal{C}) = \sum_{r \in \mathcal{R}} \frac{|\mathcal{C}_r|}{|\mathcal{C}|} (|\mathcal{C}| - |\mathcal{C}_r|)$$

qui est la moyenne du nombre de candidats éliminés par l'essai u . J'ai utilisé cette fonction comme heuristique pour mettre au point l'algorithme Élimine 2 (E2), qui fonctionne exactement comme les algorithmes de la première partie mais avec la fonction `nombre_éliminés2` à la place de `score`.

Étant donné r on peut calculer \mathcal{C}_r en $\mathcal{O}(\ell \times |\mathcal{C}|)$. Ainsi la complexité de `nombre_éliminés2` est $\mathcal{O}(3^\ell \ell \times |\mathcal{C}|)$ car $|\mathcal{R}| = 3^\ell$. Le calcul du meilleur mot se fait donc en $\mathcal{O}(3^\ell \ell \times |\mathcal{C}| \times |\mathcal{D}|)$. Si on considère que l est une constante cette complexité est en $\mathcal{O}(|\mathcal{C}| \times |\mathcal{D}|)$ mais le préfacteur constant est assez important. De plus, j'ai noté empiriquement que $|\mathcal{C}|$ décroissait exponentiellement d'essais en essais. Cela est assez intuitif car chaque essai va éliminer une fraction des candidats. J'ai fait une étude probabiliste pour démontrer ce point important (dans un cadre restreint) en [annexe](#), tout en le soulignant expérimentalement.

3.2 Amélioration de cet algorithme grâce à la théorie de l'information de Shannon (E3)

Dans les années 1940, Claude Shannon définit l'entropie d'une variable aléatoire X , mesurée en bits :

$$H = - \sum_{x \in \Omega} p(x) \log_2 p(x)$$

où \log_2 désigne le logarithme en base 2, et en considérant que $0 \log_2 0 = 0$ par continuité (car $x \log_2 x \rightarrow 0$ quand $x \rightarrow 0$).

La valeur de H correspond au nombre de bits nécessaires en moyenne pour décrire une réalisation de la variable aléatoire. On peut aussi considérer l'entropie comme l'espérance de l'information obtenue lorsque on observe une réalisation de X , à savoir $I(X) = -\log_2 p(X)$.

Ici, $\Omega = \mathcal{R}$ et comme nous l'avons expliqué dans le paragraphe 3.1, si $r \in \mathcal{R}$ la probabilité que la réponse r soit renvoyée par le jeu est $p(r) = \frac{|\mathcal{C}_r|}{|\mathcal{C}|}$, en reprenant les mêmes notations. Il en découle que, dans notre contexte,

$$H = - \sum_{r \in \mathcal{R}} \frac{|\mathcal{C}_r|}{|\mathcal{C}|} \log_2 \frac{|\mathcal{C}_r|}{|\mathcal{C}|}$$

Cela nous amène à la définition de `nombre_éliminés3` :

$$\text{nombre_éliminés3}(u, \mathcal{C}) = \sum_{r \in \mathcal{R}} \frac{|\mathcal{C}_r|}{|\mathcal{C}|} (\log_2 |\mathcal{C}| - \log_2 |\mathcal{C}_r|)$$

On remarque la similarité avec la fonction `nombre_éliminés2` mais l'usage du logarithme permet de réduire l'effet des cas dégénérés, c'est à dire ayant une probabilité très faible, ce qui va effectivement rendre l'algorithme légèrement meilleur comme nous allons le voir dans le prochain paragraphe. La complexité de cette fonction est la même que celle de `nombre_éliminés2`.

3.3 Résultats et limitations

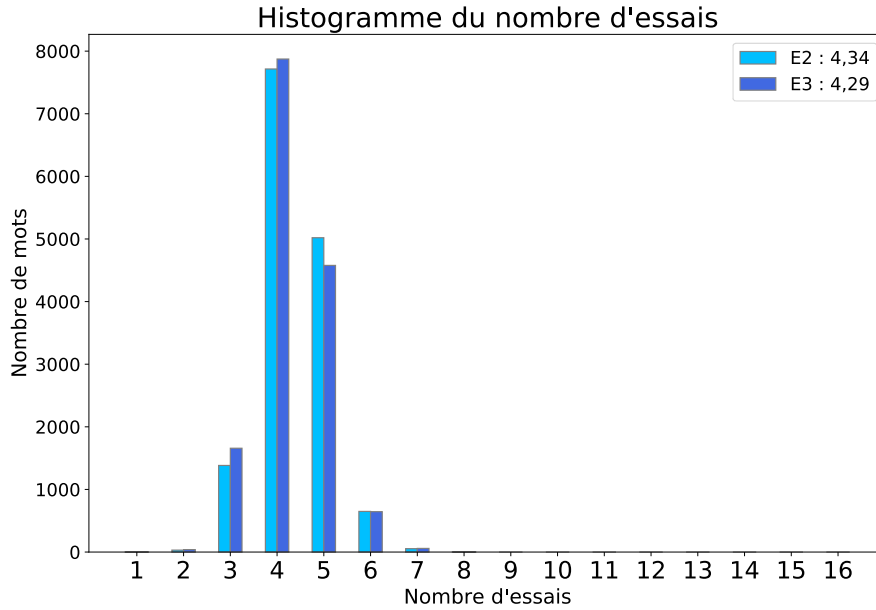


FIGURE 4 – Résultats des algorithmes E2 et E3 lorsque le mot mystère parcourt $\mathcal{D}_{\text{anglais}}$

Ces deux algorithmes ont des biens meilleurs résultats que les premières heuristiques, trouvant le mot mystère en 4,3 essais en moyenne seulement. Nous pouvons aussi noter sur les résultats que la deuxième version utilisant la théorie de l'information (E3) est légèrement meilleure que la première version (E2).

Toutefois ces deux algorithmes sont bien plus lents que la première heuristique, comme nous l'avons mis en avant lors des calculs de complexité. En pratique, j'ai codé ces algorithmes en C et je les ai fait tourner sur un mini mac M1. Les deux premiers algorithmes ont mis quelques minutes à peine à tourner sur l'intégralité du dictionnaire $\mathcal{D}_{\text{anglais}}$ (14855 mots), alors que les algorithmes d'élimination ont pris environ 4 jours pour finir les mêmes calculs (en évitant de faire le calcul du premier mot à chaque fois qui prenait 30 minutes à lui seul).

Cela m'a donc poussé à chercher un compromis entre performances et temps d'exécution.

4 Compromis entre performances et temps d'exécution

4.1 Algorithme hybride

Mon idée pour cet algorithme était d'essayer de tirer parti des deux algorithmes précédents, en combinant la vitesse des heuristiques et la performance, notamment en fin de partie, des algorithmes d'élimination. J'ai donc mis au point un algorithme hybride qui prend en entrée un entier k et qui joue selon la stratégie heuristique (H1) pour les k premiers essais avant de basculer vers la stratégie éliminatoire (E2).

Comme nous pouvons le voir sur les résultats de la figure 5, plus k est grand, plus les résultats empiriques se rapprochent de ceux de H1. Toutefois, il y a un très net gain de temps comme nous allons le souligner dans la conclusion.

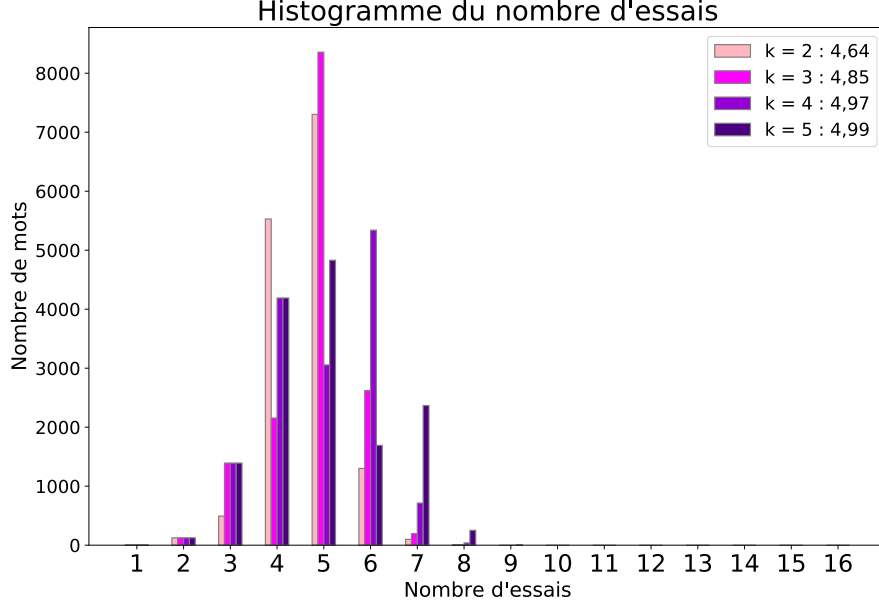


FIGURE 5 – Résultats de l'algorithme hybride sur $\mathcal{D}_{\text{anglais}}$

Néanmoins, j'ai été un peu déçu par les résultats de cet algorithme et c'est pour cela que j'ai pensé à une autre stratégie pour réduire le temps de calcul voir ci-dessous. Sur le dictionnaire réduit $\mathcal{D}_{\text{officiel}}$, les résultats sont encore pires car la transition entre la stratégie H1 et E2 ne fonctionne pas du tout, voir [Table 4](#) et commentaires en annexe.

4.2 Algorithme de coupe aléatoire

Ici, l'idée est d'utiliser l'algorithme E2 mais de restreindre la recherche du meilleur mot à un sous ensemble du dictionnaire extrait aléatoirement. L'intuition est que l'information dévoilée par le meilleur mot de ce sous ensemble sera proche de celle dévoilée par le meilleur mot en absolu. On effectue cette stratégie uniquement si $|\mathcal{C}| > 100$ car quand $|\mathcal{C}| \leq 100$, le calcul du meilleur mot est rapide (on rappelle que la complexité de ce calcul est en $\mathcal{O}(|\mathcal{C}| \times |\mathcal{D}|)$) et il est nécessaire d'être très précis en fin de partie. Ainsi à chaque essai l'algorithme tire au hasard un sous dictionnaire $\mathcal{D}_h \subset \mathcal{D}$ tel que $\frac{|\mathcal{D}|}{|\mathcal{D}_h|} = d$, avec d fixé comme paramètre, et calcule le meilleur essai parmi \mathcal{D}_h . Cet algorithme est donc de type Monte Carlo car son temps d'exécution est toujours le même mais les résultats dépendent du tirage de \mathcal{D}_h . Toutefois, j'ai noté expérimentalement que le nombre d'essais moyen de cet algorithme a très peu varié quand je l'ai lancé plusieurs fois pour la même valeur de d . De même pour $d \in \{2, 5, 10\}$ les résultats sont très proches. En effet, le nombre d'essais moyens sur $\mathcal{D}_{\text{officiel}}$ varie uniquement de 7 millièmes comme nous pouvons le voir dans les résultats en annexe.

Comme nous allons le voir dans la conclusion, cela permet d'avoir des résultats très proches de ceux de E2 tout en divisant le temps de calcul.

5 Conclusion

5.1 Comparaison de mes résultats avec d'autres algorithmes

Voici un [graphe](#) recensant les résultats de mes différents algorithmes et d'autres algorithmes (en gris) comme ceux des chercheurs du MIT ou du WordleBot qui est l'algorithme du New York Times, et de résultats d'humains comme ceux de mon père ou de l'utilisateur moyen, sur le dictionnaire officiel $\mathcal{D}_{\text{officiel}}$. On notera que les résultats du WordleBot, de mon père et de l'utilisateur moyen correspondent aux valeurs renseignées par le New York Times sur une période de quinze jours (13-28 avril 2024) et donc correspondent à quinze parties seulement et non pas 2315 comme pour les autres algorithmes.

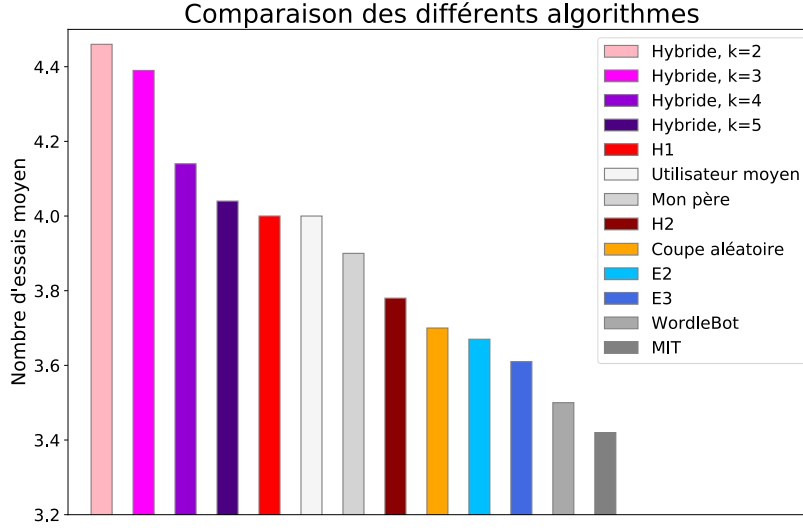


FIGURE 6 – Résultats des différents algorithmes sur $\mathcal{D}_{\text{officiel}}$

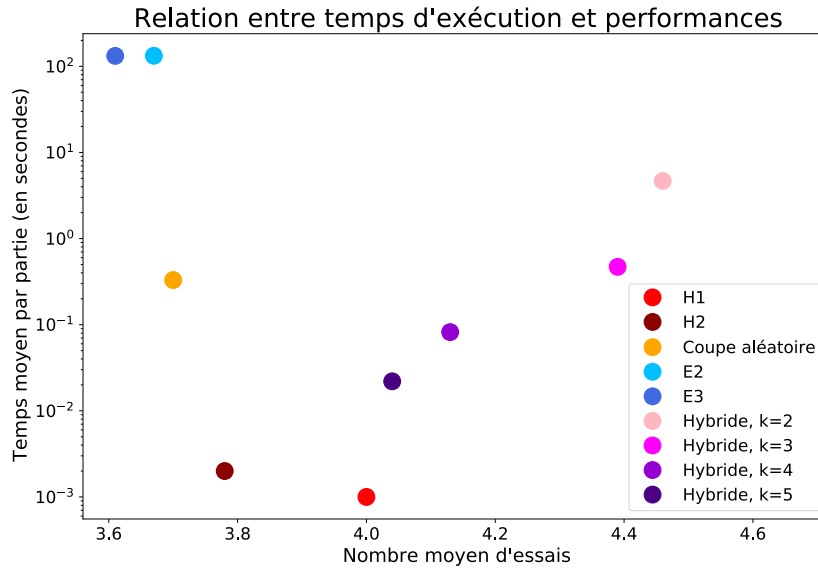


FIGURE 7 – Relation entre temps d'exécution et performances des différents algorithmes sur $\mathcal{D}_{\text{officiel}}$

5.2 Comparaison entre résultats et temps d'exécution de mes algorithmes

J'ai tracé sur la figure 7 la relation entre le temps d'exécution (en échelle logarithmique) et les performances des différents algorithmes. À part pour les algorithmes hybrides qui ne fonctionnent pas, on voit sur la partie gauche de la figure qu'un (petit) gain en performances s'accompagne d'une augmentation approximativement exponentielle du temps de calcul.

5.3 Comparaison de mes résultats sur différents dictionnaires

J'ai aussi testé mon algorithme E2 sur différents dictionnaires, par exemple un dictionnaire français $\mathcal{D}_{\text{français}}$ de 2315 mots comme $\mathcal{D}_{\text{officiel}}$. J'ai été surpris de voir à quel point les résultats sur cet autre dictionnaire sont proches de ceux sur $\mathcal{D}_{\text{officiel}}$, comme nous pouvons le voir sur le tableau 7 en annexe.

plus que $|C_{N-g,\ell-v}| = (N-g)^{\ell-v} = (N-g)^g$ candidats restants. Ainsi le nombre de candidats a été divisé par le rapport $\rho = \frac{|C_{N,\ell}|}{|C_{N-g,\ell-v}|} = \frac{N^\ell}{(N-g)^g}$. Pour N suffisamment grand devant ℓ ceci est minimal pour $\ell = g$ (toutes les cases sont coloriées en gris). Ainsi pour $N = 26, \ell = 5$ le nombre de candidats est divisé par au moins 2,91 au premier essai.

Calculons maintenant la valeur moyenne du logarithme du rapport ρ (cela est plus représentatif que la moyenne de ρ). Notons X la variable aléatoire qui compte le nombre de cases coloriées en gris. Alors $X \hookrightarrow \mathcal{B}(\ell, p)$ (la loi binomiale de paramètres ℓ et p) où $p = \frac{N-1}{N}$. En effet, la probabilité qu'une lettre soit coloriée en vert est $\frac{1}{N}$ et la probabilité qu'une lettre soit coloriée en gris est $\frac{N-1}{N}$. On s'intéresse ici à $\mathbb{E} \left(\log \frac{N^\ell}{(N-X)^\ell} \right)$ que nous pouvons noter $\alpha_{N,\ell}$. Selon le théorème de transfert, (on suppose $N > \ell$)

$$\alpha_{N,\ell} = \sum_{g=0}^{\ell} \binom{\ell}{g} \left(\frac{N-1}{N} \right)^g \left(\frac{1}{N} \right)^{\ell-g} \log \frac{N^\ell}{(N-g)^g} = \sum_{g=0}^{\ell} \binom{\ell}{g} \frac{(N-1)^g}{N^\ell} [\ell \log N - g \log(N-g)]$$

Voici quelques valeurs : $\exp(\alpha_{26,5}) \simeq 5,048$, $\exp(\alpha_{21,5}) \simeq 7,135$, $\exp(\alpha_{21,3}) \simeq 2,365$.

Ce phénomène se constate empiriquement. Sur la figure 9, j'ai tracé le nombre de candidats restants en fonction du nombre d'essais pour cinq mots aléatoires de $\mathcal{D}_{\text{anglais}}$: « sweep », « usual », « mauve », « morph » et « motto » (en gris), et la moyenne pour ces cinq mots (en noir).

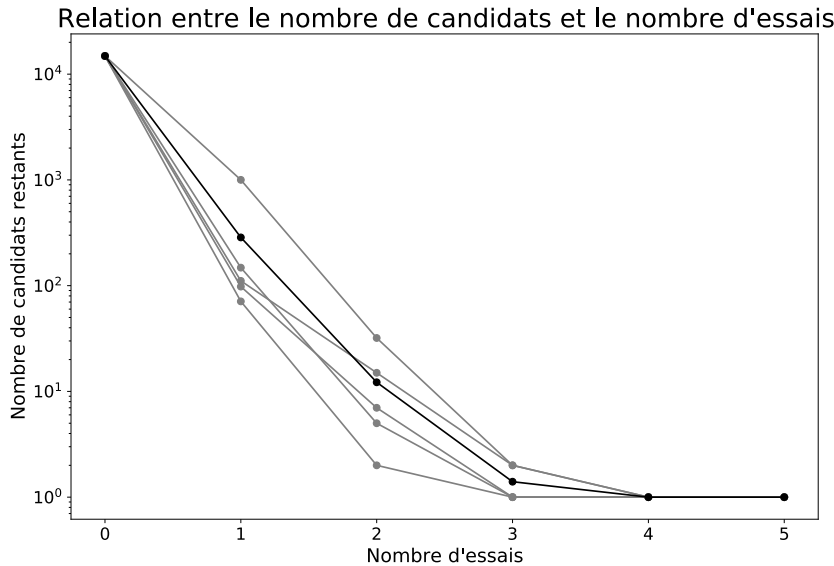


FIGURE 9 – Décroissance exponentielle du nombre de candidats

7.3 Résultats détaillés

On voit sur le tableau 4 que les algorithmes hybrides ne fonctionnent pas du tout sur le dictionnaire réduit $\mathcal{D}_{\text{officiel}}$. Cela s'explique par la très mauvaise transition entre les deux stratégies. En effet, lors de l'essai numéro $k+1$, ici en gras, l'algorithme E2 trouve très peu de mots car il préfère éliminer plus de candidats pour être certain de gagner dans les coups suivants. Toutefois cela conduit à un nombre moyen d'essais élevé.

De plus, on voit sur le tableau 5 que sur cinq calculs différents, le nombre total d'essais ne varie que très peu. En effet, en notant N_e le nombre total d'essai, $N_e \approx 8577$ (c'est la valeur moyenne) et $\Delta N_e = 8598 - 8564 = 34$. D'où $\frac{\Delta N_e}{N_e} = 0,4\%$. Cela confirme donc mon intuition mais j'imagine que ce taux très faible dépend grandement du dictionnaire sur lequel on travaille. Il est aussi intéressant de noter que les performances de l'algorithme aléatoire ne dépendent pas tant que ça du choix de d , comme nous pouvons le voir sur la table 6

Cela semble indiquer que jouer un mot presque optimal, mais non optimal, dans les premiers coups va avoir peu d'influence sur le score moyen. Ce qui semble alors plus important est la gestion des essais quand il ne reste plus que quelques candidats en fin de partie.

	Heuristique 1	Heuristique 2
1 essai	1	1
2 essais	125	166
3 essais	1392	1939
4 essais	4190	4571
5 essais	4830	3766
6 essais	2556	2089
7 essais	1038	1127
8 essais	421	562
9 essais	187	308
10 essais	83	166
11 essais	25	80
12 essais	5	40
13 essais	2	19
14 essais	0	12
15 essais	0	8
16 essais	0	1
Nombre total de mots	14855	14855
Nombre total d'essais	74181	74526
Nombre moyen d'essais	4,9937	5,0169

TABLE 1 – Résultats complets de H1 et H2 sur $\mathcal{D}_{\text{anglais}}$

	Elimine 2	Elimine 3
1 essai	1	1
2 essais	31	39
3 essais	1383	1658
4 essais	7715	7874
5 essais	5020	4578
6 essais	649	645
7 essais	54	58
8 essais	2	2
Nombre total de mots	14855	14855
Nombre total d'essais	64460	63731
Nombre moyen d'essais	4,3393	4,2902

TABLE 2 – Résultats complets de E2 et E3 sur $\mathcal{D}_{\text{anglais}}$

	$k = 2$	$k = 3$	$k = 4$	$k = 5$
1 essai	1	1	1	1
2 essais	125	125	125	125
3 essais	493	1392	1392	1392
4 essais	5529	2155	4190	4190
5 essais	7302	8357	3055	4830
6 essais	1301	2621	5340	1692
7 essais	99	195	714	2367
8 essais	5	9	38	252
9 essais	0	0	0	6
Nombre total de mots	14855	14855	14855	14855
Nombre total d'essais	68895	71995	73804	74128
Nombre moyen d'essais	4,6378	4,8465	4,9683	4,9901

TABLE 3 – Résultats complets des algorithmes hybrides sur $\mathcal{D}_{\text{anglais}}$

Enfin, j'ai construit un dictionnaire de mots français de cinq lettres de la même taille que $\mathcal{D}_{\text{officiel}}$ pour voir si la langue avait une influence sur les résultats. Comme le montre le tableau 7 les performances de l'algorithme E2 sont à peine altérées par le choix de la langue.

	$k = 2$	$k = 3$	$k = 4$	$k = 5$
1 essai	1	1	1	1
2 essais	80	80	80	80
3 essais	83	625	625	625
4 essais	980	231	999	999
5 essais	1039	1072	155	471
6 essais	130	292	413	29
7 essais	2	14	39	96
8 essais	0	0	3	14
Nombre total de mots	2315	2315	2315	2315
Nombre total d'essais	10319	10170	9582	9345
Nombre moyen d'essais	4,4575	4,3931	4,1391	4,0367
Temps total (s)	10802,5	1078,7	190	50,3

TABLE 4 – Résultats complets des algorithmes hybrides sur $\mathcal{D}_{\text{officiel}}$

1 essai	0	0	0	0	0
2 essais	34	36	36	33	37
3 essais	763	751	761	749	755
4 essais	1379	1390	1381	1381	1375
5 essais	139	138	137	151	148
6 essais	0	0	0	1	0
Nombre total de mots	2315	2315	2315	2315	2315
Nombre total d'essais	8568	8575	8564	8598	8579
Nombre moyen d'essais	3,7011	3,7041	3,6994	3,7140	3,7058

TABLE 5 – Résultats complets de l'algorithme aléatoire sur $\mathcal{D}_{\text{officiel}}$ pour $d = 10$

	$d = 2$	$d = 5$	$d = 10$	$d = 20$	$d = 100$
1 essai	0	0	0	0	0
2 essais	34	36	36	34	34
3 essais	761	746	751	748	732
4 essais	1393	1402	1390	1384	1368
5 essais	127	131	138	149	180
6 essais	0	0	0	0	1
Nombre total de mots	2315	2315	2315	2315	2315
Nombre total d'essais	8558	8573	8575	8593	8642
Nombre moyen d'essais	3,6968	3,7032	3,7041	3,7119	3,7330
Temps (en secondes)	1009,5	819,2	756,4	730,4	716,8

TABLE 6 – Résultats complets de l'algorithme aléatoire sur $\mathcal{D}_{\text{officiel}}$ pour différentes valeurs de d

	$\mathcal{D}_{\text{officiel}}$	$\mathcal{D}_{\text{français}}$
1 essai	0	1
2 essais	27	37
3 essais	793	835
4 essais	1412	1299
5 essais	83	141
6 essais	0	2
Nombre total de mots	2315	2315
Nombre total d'essais	8496	8493
Nombre moyen d'essais	3,6700	3,6687

TABLE 7 – Résultats complets de E2 sur $\mathcal{D}_{\text{officiel}}$ et $\mathcal{D}_{\text{français}}$