Ruzzle : à la recherche de la plus belle grille

Didier Müller, Lycée cantonal de Porrentruy

Ruzzle[1] est un jeu de lettres aux règles simples : une grille de quatre fois quatre lettres, et deux minutes pour trouver le plus de mots possibles, avec pour seul impératif que les cases se touchent. On ne peut pas utiliser deux fois la même case, ce qui fait que la longueur maximale d'un mot sera de 16.

Pour pimenter le jeu, *Ruzzle* reprend également les cases bonus « lettre compte double », « lettre compte triple », « mot compte double » et « mot compte triple » du Scrabble. Pour gagner, il faudra obtenir le plus gros score !

Dans cet article, on se contentera de trouver tous les mots possibles, sans s'occuper des bonus. Mais notre but est ambitieux : trouver la grille qui contient le plus de mots, celle que nous appellerons la « plus belle » grille.

Ressources trouvées sur le web

Dictionnaire

Premier outil indispensable : un dictionnaire de mots français, avec des lettres non accentuées, contenant des mots au singulier et au pluriel, et des verbes conjugués à tous les temps. Par chance, ce dictionnaire existe sur le web. Il faut juste l'adapter à nos besoins, à savoir éliminer les mots d'une seule lettre, les mots composés et les mots de plus de 16 lettres.

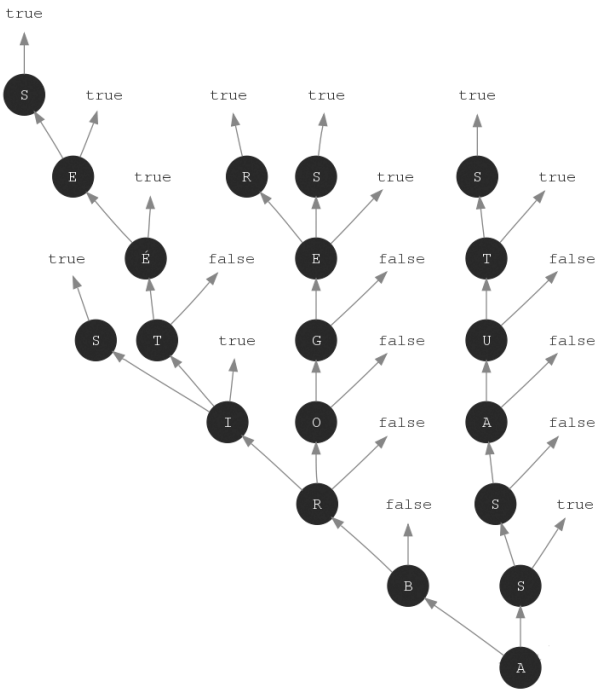
Ce dictionnaire purgé contient encore 328'465 mots[2]. Mais il faut préciser que ce n'est pas le même qu'utilise *Ruzzle*. En effet, *Ruzzle* donne plus de mots que n'en trouve le programme.

Structure de données du dictionnaire

Dans ce projet, il faut non seulement chercher tous les mots, mais encore les trouver très vite, en quelques millisecondes. En effet, il faudra faire cette recherche très souvent, d'où la nécessité d'avoir une représentation performante du lexique.

Un *trie* ou *arbre préfixe* est un arbre numérique ordonné qui est utilisé pour stocker une table associative où les clés sont généralement des chaînes de caractères. Contrairement à un arbre binaire de recherche, aucun nœud dans le *trie* ne stocke la chaîne à laquelle il est associé. C'est la position du nœud dans l'arbre qui détermine la chaîne correspondante.

Pour tout nœud, ses descendants ont en commun le même préfixe. La racine est associée à la chaîne vide. Des valeurs ne sont pas attribuées à chaque nœud, mais uniquement aux feuilles et à certains nœuds internes se trouvant à une position qui désigne l'intégralité d'une chaîne correspondant à une clé.



Ci-contre, un trie construit avec les mots : abri, abris, abrité, abritée, abritées, abroge, abroger, abroges, as, assaut et assauts.

Chercher tous les mots d'une grille

Un programme très efficace existe sur le web qui permet de trouver tous les mots d'une grille[3]. Il utilise évidemment un Trie. La liste des mots est trouvée en quelques millièmes de secondes[4].

Recherche de la plus belle grille

Petit calcul grossier : il existe $26^{16} = 4{,}36 \cdot 10^{22}$ grilles différentes. Même si l'on enlève les grilles ne contenant que des consonnes, il sera impossible de les tester toutes ! Et il va falloir compter sur la chance. Ce qui veut dire qu'il sera quasiment impossible d'être sûr qu'il n'y a pas de plus belle grille que celle que nous trouverons.

Généralement, dans une partie réelle de *Ruzzle*, la grille proposée compte entre 250 et 350 mots, parfois plus.

Grille aléatoire

Première idée qui vient à l'esprit : générer un grand nombre de grilles aléatoirement. Cela donnera une première idée. Mais ne faisons pas n'importe quoi. Quand on joue souvent à *Ruzzle*, on constate que des lettres n'apparaissent presque jamais : Z, W, K, J, Y, notamment. On va d'abord calculer les fréquences des lettres dans notre dictionnaire. Les lettres seront tirées au sort en tenant compte de fréquences que nous avons calculées :

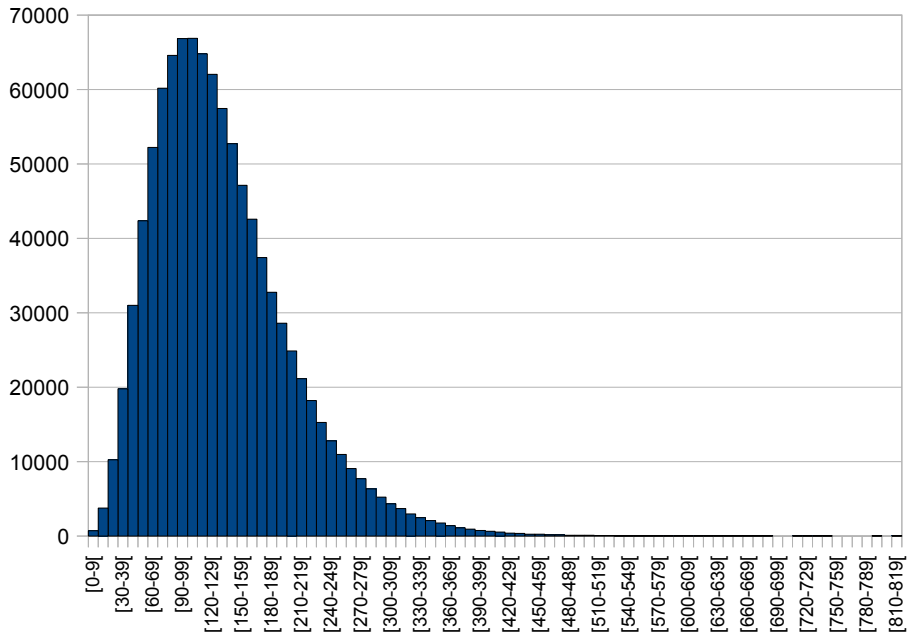
E	S	A	I	R	N	T	O	L	U	C	M	D
14.89%	10.21%	9.71%	9.40%	8.67%	7.34%	6.82%	5.82%	4.01%	3.60%	3.40%	2.54%	2.36%
P	G	B	F	H	Z	V	Q	Y	X	J	K	W
2.35%	1.60%	1.40%	1.36%	1.16%	1.07%	0.96%	0.50%	0.34%	0.25%	0.18%	0.05%	0.01%

On peut remarquer que cette table est un peu différente de celle des fréquences des lettres de la langue française[6]. La raison est simple : dans *Ruzzle*, chaque mot n'apparaît qu'une seule fois.

La meilleure grille trouvée avec cette méthode est `PETESRNIEIATMSEL` (on a écrit les 16 lettres ligne par ligne), qui contient 827 mots. C'est *grosso modo* le double de ce qu'on avait vu sur le jeu réel.

Dans une autre expérience, nous avons généré un million de grilles en utilisant les fréquences ci-dessus. Cela a mis environ 4 heures. Un rapide calcul nous permet d'estimer le temps que nous mettrions pour tester naïvement toutes les grilles possibles : environ 20'000 milliards d'années !

Sur l'histogramme ci-dessous, on a mis en abscisse le nombre de mots trouvés (classes de largeur 10) et en ordonnée le nombre de grilles correspondants. On voit que la grande majorité des grilles comptent entre 50 et 200 mots. Il y a donc un clair décalage entre le jeu réel et notre simulation. Explications possibles : soit *Ruzzle* utilise un dictionnaire beaucoup plus fourni que le nôtre, soit les grilles ne sont pas générées totalement au hasard, soit les deux...



Cette méthode naïve a vite montré ses limites. Il va falloir trouver des métaheuristiques[5] plus « intelligentes » pour améliorer nos performances. Toutes les méthodes que nous allons voir ont quelque chose en commun : on va partir d'une grille aléatoire, puis on va essayer d'améliorer petit à petit cette grille pour arriver à un maximum local.

Petite définition préalable : deux grilles sont « voisines » si elles ne différent que d'une lettre.

Méthode de la plus grande pente

1. Générer une grille aléatoire.
2. Parmi les 400 (25 lettres x 16 cases) grilles voisines, choisir celle qui contient le plus de mots.
3. Si on a trouvé une meilleure grille, retourner à 2. Sinon STOP.

Le résultat dépend beaucoup de la grille de départ. La plus belle grille trouvée avec cette méthode est `STIPENALARIMSTES`, qui compte 1388 mots. Par rapport à la méthode naïve de la grille aléatoire, l'amélioration est spectaculaire.

Méthode avec tabous

Le problème de la méthode de la plus grande pente est que l'on se coince toujours dans un maximum local, et qu'il est impossible d'en sortir. Pour pallier cela, on peut ajouter une liste de tabous : on garde en mémoire les dernières grilles visitées, afin de ne pas revenir dessus. Si la liste est suffisamment longue, cela évite les blocages. La longueur de la liste des tabous est d'ailleurs le seul paramètre avec lequel on peut jouer.

1. Générer une grille aléatoire.
2. Parmi les 400 grilles voisines, choisir la plus belle qui n'est pas dans la liste des tabous.
3. Si la liste des tabous est pleine, retirer de cette file la grille la plus ancienne.
4. Mettre la grille précédente en queue de la liste des tabous.
5. Retourner à 2, tant qu'on n'a pas décidé de s'arrêter.

La plus belle grille trouvée avec cette méthode est `SMLPEIAERTNRSAIT`, qui contient 1473 mots.

Recuit simulé

Le recuit simulé introduit plus de hasard que dans la méthode avec tabous. On progressera toujours d'une grille à l'une de ses voisines, mais celle-ci sera tirée au sort. De deux choses l'une : soit la nouvelle grille est meilleure, et on la garde comme grille courante, soit elle est moins bonne, et on ne la gardera qu'avec une certaine probabilité qui diminuera avec le temps.

1. Choisir une « température » de départ T.
2. Générer une grille aléatoire. Appelons-la $G1$.
3. Copier la grille $G1$ dans une grille $G2$, puis effacer une case et placer une nouvelle lettre tirée au sort en tenant compte des fréquences.
4. Calculer $\Delta = \text{score}(G1) - \text{score}(G2)$
5. Si $\Delta \leq 0$, $G1 \leftarrow G2$; le cas échéant, mettre à jour le meilleur score et la meilleure grille. Aller à 8.
6. Générer un nombre réel aléatoire entre 0 et 1, que nous appellerons r.
7. Si $r < \exp(-\Delta/T)$, $G1 \leftarrow G2$.
8. Diminuer T.
9. Retourner à 3, tant qu'on n'a pas décidé de s'arrêter.

Le terme *température* du point 1 fait référence au procédé sidérurgique du recuit : on chauffe fortement un métal, puis on fait descendre la température par paliers.

Dans le recuit simulé, la décroissance de la température est un paramètre important avec lequel il faut jouer. Si la température décroît rapidement, on arrivera vite à un bon résultat, mais on risque de se coincer plus rapidement dans un maximum local. En effet, quand T diminue, $\exp(-\Delta/T)$ diminue aussi, et on retiendra moins volontiers une plus mauvaise grille. On fait généralement décroître la température par paliers : on laisse la température stable un certain nombre d'itérations (encore un paramètre délicat à régler), puis on multiplie T par un coefficient $0 < c < 1$ (dernier paramètre à choisir). Remarquons encore que plus Δ est grand, plus la probabilité d'acceptation d'une mauvaise grille est faible.

La plus belle grille trouvée avec cette méthode (après environ 2000 essais) est `CRESAMINILATLSER`, qui contient 1634 mots. On a utilisé les paramètres suivants :
$T_0 = 70$, paliers de longueur 10, $c = 0.95$, 1000 itérations au maximum.

Par un hasard incroyable, on a trouvé une autre grille composée des mêmes lettres et qui contient les mêmes 1634 mots : `RTNSEAIESLMRLIAC` (cette grille s'obtient en lisant la grille ci-contre de droite à gauche et de bas en haut).

Algorithme génétique

Dernière approche : un algorithme reprenant (plus ou moins) les principes de l'évolution, d'où son nom d'algorithme génétique. L'idée est d'observer une « population » de grilles que l'on va croiser deux par deux (*crossover*). Ce crossover consiste à couper deux chaînes, croiser deux segments, et les recoller. Par exemple, les deux chaînes :

`ABCDEFGHIJ|KLMNOP`		`ABCDEFGHIJ|PONMLK`
et	deviennent	et
`ZYXWVUTSRQ|PONMLK`		`ZYXWVUTSRQ|KLMNOP`

1. Générer 2*n* grilles aléatoires.
2. Classer et numéroter ces 2*n* grilles selon leur score, du meilleur au moins bon.
3. Croiser les grilles 2*k*–1 et 2*k*, pour *k* allant de 1 à *n*.
4. Effectuer une *mutation* pour chaque grille : avec une certaine probabilité, effacer une case et placer une nouvelle lettre tirée au sort en tenant compte des fréquences.
5. Classer et numéroter les 2*n* grilles obtenues selon leur score, du meilleur au moins bon.
6. Dupliquer la grille 1 (la meilleure) et placer ce doublon en position 2*n*, après avoir éliminé la grille 2*n* (la moins bonne).
7. Le cas échéant, mettre à jour le meilleur score et la meilleure grille.
8. Retourner à 3, tant qu'on n'a pas décidé de s'arrêter.

Les paramètres avec lesquels on peut jouer sont la taille de la population et la probabilité de mutation.

La plus belle grille trouvée avec cette méthode est `APEDRMLREIAESNTS`, qui compte 1508 mots. On a observé pendant 600 générations une population de 80 grilles, avec une probabilité de mutation de 10%.

Petite analyse des résultats

Les résultats complets sont disponibles sur [2]. On a gardé les 4 meilleures résultats obtenus avec chacune des méthode.

- Ce qui frappe d'abord, c'est que toutes ces excellentes grilles ne contiennent que les lettres A, C, E, I, L, M, N, P, R, S et T. Les voyelles O et U sont étrangement absentes. Inversement, on ne s'attendait pas forcément à trouver C et P.
- Sans surprise, les deux premières méthodes décrites dans cet article arrivent en bas du classement. On n'a pas dépassé 900 mots avec les grilles aléatoires et on reste en dessous des 1400 mots avec le méthode de la plus grande pente.
- On a dépassé les 1400 mots avec les trois autres métaheuristiques. C'est le recuit simulé qui a donné les meilleurs résultats, mais les deux autres méthodes pourraient probablement fournir de meilleures grilles, si l'on jouait davantage avec les paramètres.

À vous de jouer !

Comme on l'a déjà dit au début de l'article, il est très probable qu'il existe de plus belles grilles. Pourquoi ne pas vous laisser prendre au jeu et tenter de les découvrir ? Envoyez-moi vos découvertes à didier.muller@ju.educanet2.ch. Je mettrai à jour le tableau de [2], page où vous trouverez par ailleurs les programmes Python utilisés.

On peut aussi s'amuser à trouver les plus belles grilles avec contraintes. Par exemple :

- « E » est interdit dans la grille,
- « E » est la seule voyelle autorisée,
- contenir le plus de mots de 4 lettres,
- etc.

Références

[1] Site officiel de Ruzzle, <www.ruzzle-game.com/>
[2] Müller Didier, « Ruzzle : à la recherche de la plus belle grille », <www.nymphomath.ch/info/algo/corriges/ruzzle/>
[3] Ferraro Tyler, « Ruzzle-Solver », <https://github.com/TylerFerraro/Ruzzle-Solver>
[4] Mannino Miro, « Ruzzle Solver Algorithm », <http://miromannino.com/ruzzle-solver-algorithm>
[5] Müller Didier, « Le problème des *n* dames pour illustrer les métaheuristiques », Bulletin 113 de la SSPMP, 2010
[6] Müller Didier, « Analyse des fréquences en français », <www.nymphomath.ch/crypto/stat/francais.html>