Jacobson, de l'Université de Pittsburgh ; nous avons utilisé des arbres et des structures encore plus condensées, les graphes acycliques orientés, où, dans les deux cas, les chemins représentent les mots ; dans ces structures, les lettres des mots ne sont plus des nœuds, mais des arêtes du graphe, de sorte que les débuts de mots similaires ne sont représentés qu'une fois. On les parcourt à l'aide d'es algorithmes de parcours récursif.

LA RECHERCHE D'ANAGRAMMES

Pour représenter le dictionnaire par un arbre, on associe une branche à chaque lettre, de sorte qu'un chemin jusqu'à un nœud terminal représente un mot du dictionnaire. Pour chercher si un mot, par exemple *abacab* est un mot du dictionnaire, on part de la racine de l'arbre et l'on se demande s'il existe des mots commençant par la lettre *a*. Si aucune arête partant de la racine n'est indexée par un *a*, il n'existe pas de mot commençant par un *a* ; si l'on trouve une arête indexée par la lettre *a*, on cherche s'il existe des chemins qui permettent de passer du *a* à un *b*. On répète alors l'opération pour les lettres suivantes : *a, c, a, b.*

L'ordinateur ainsi muni d'un dictionnaire peut-il dignement succéder à Rabelais ou à Vian? Examinons la recherche d'anagrammes du mot *esprit.* Pour résoudre ce problème, nous utilisons une méthode de recherche récursive, qui permet, quand on est dans une impasse ou que l'on a terminé d'explorer une série de solutions, de revenir en arrière jusqu'au dernier embranchement de l'arbre pour lequel la clause de vérification n'était pas encore invalidée.

Pour rechercher les anagrammes d'*esprit,* mot que l'on indique à l'ordinateur dans une variable *Mot-initial*, nous procédons de la façon suivante. Nous réservons un espace mémoire pour les résultats, le *Mot-résultat,* vide au début de l'exécution du programme. Puis nous remplissons un tableau d'une ligne et de 26 colonnes (pour les 26 lettres de l'alphabet), le *Panier,* en stockant dans chaque case le nombre d'apparitions des diverses lettres dans le mot dont on cherche les anagrammes. Par exemple, le *Panier* associé à ESPRIT est :

00001000100000010111000000.

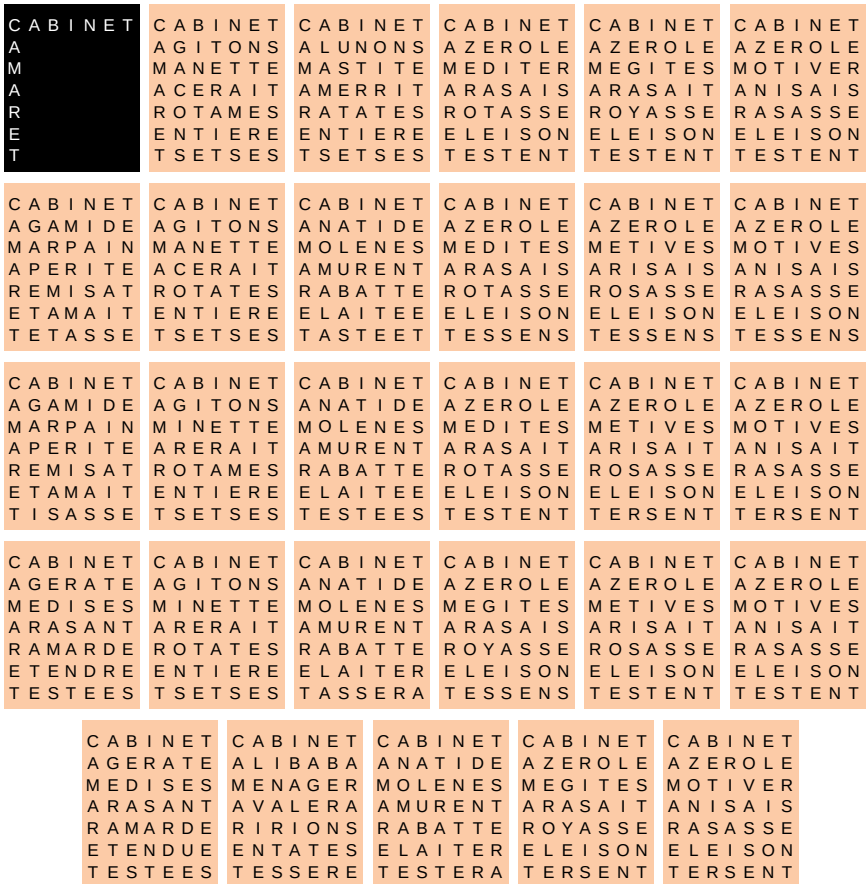
On regarde alors quelle est la lettre correspondant au nœud exploré. Puis on regarde dans le *Panier* la valeur de la case associée à cette lettre. Si ce nombre est nul, on ignore le sous-arbre dont le nœud exploré est la racine et on passe au nœud suivant. Si le nombre examiné dans le *Panier* n'est pas nul, on ajoute la lettre correspondant au nœud exploré à la fin de *Mot-résultat* et on diminue d'une unité, dans le panier, le nombre contenu dans la case examinée ; on applique l'algorithme général à chacun des nœuds dont le nœud exploré est le père. De plus, on vérifie si le nombre de lettres de *Mot-résultat* est égal au nombre de lettres de *Mot-initial.* Si oui, on a dans *Mot-résultat* une anagramme de *Mot-initial,* anagramme que l'on affiche (à ce stade, toutes les cases du panier sont nulles).

Dans l'exemple de la recherche d'anagrammes du mot ESPRIT, le premier sous-arbre exploré sera celui qui correspond aux mots commençant par la lettre E, avec un *Panier* où la case de E est réduite d'une unité. Une fois qu'on aura exploré ce sous-arbre tout entier, on augmentera le compteur des E dans le *Panier* (on aura alors un *Panier* identique à celui du départ) et on tentera ensuite d'appliquer l'algorithme général au sous-arbre des mots commençant par la lettre suivante dont la case du *panier* n'est pas vide (la lettre I), avec *Mot-résultat* vide et le *Panier* égal à celui de départ, à l'exception de la case de la lettre I qui aura été réduite d'une unité. Puis on répétera l'opération avec les lettres P, R, S, T. L'avantage des algorithmes récursifs est qu'ils s'appliquent de la même façon à tout un arbre ou à un quelconque sous-arbre inclus dans l'arbre total. Ils s'appliquent notamment aux feuilles terminales, et la recherche d'anagrammes, par ce biais, se fait en un seul parcours de l'arbre dictionnaire.

Les anagrammes d'ESPRIT sont ainsi obtenues en une fraction seconde sur un ordinateur à base d'un processeur *Pentium* 120 MHz de la Société *Intel* : PETRIS, PISTER, PITRES, PRIEST, PRITES, REPITS, TRIPES.

On peut aussi étendre l'algorithme de sorte que le programme trouve des anagrammes de *Mot-initial* composées de plusieurs mots. À cette fin, on ajoute simplement un test sur le nœud exploré : ce nœud correspond-il à une fin de mot valide? Si oui, alors on ajoute un espace à la fin de mot-résultat et on applique l'algorithme général avec ce *Mot-résultat* sur l'arbre dictionnaire complet (à partir de sa racine), avec le *panier* réduit de la façon habituelle. Sinon, on ne fait rien de plus.

Notre méthode d'exploration combinatoire est exhaustive et conduit, si l'on n'y prend garde, à une inflation d'anagrammes composées de mots qui existent tous, mais dont l'association manque



3. Georges Pérec n'avait pas réussi à construire une grille ayant pour potence les mots CABINET (I horizontal) et CAMARET (1 vertical). L'ordinateur, lui, en a trouvé 28, qui sont indiquées sur cette figure.

souvent de pertinence. Il faut trier manuellement le bon grain de l'ivraie.

Les anagrammes produites à partir des noms et prénoms des personnes connues sont parfois amusantes. Deux des seuls qui y résistent, tant en français qu'en anglais, sont Johnny Halliday et Jean-Philippe Smet, le vrai nom du chanteur. En revanche, Boris Vian s'était donné une tâche facile, car notre programme lui trouve les anagrammes suivantes : ABRI OVNIS, BAR VOISIN, BISON RAVI, BOIRA VINS, VOIS RABIN, VISA ROBIN, VIRAI SNOB, VIBRA SOIN.

APPAIRA	APPAIRA	APPAIRE	APTOISE
GEORGES	GEORGES	GEORGES	GEORGES
ARIANES	ARRHANT	ARISENT	ARISENT
CERNERA	DERAMER	DESISTE	CELASSE
ACENSAI	ACINESE	ECENTES	ECENTES

APPAIRA	APPAIRA	APPAIRE	APTOISE
GEORGES	GEORGES	GEORGES	GEORGES
ARIANES	ORGANES	ARISENT	ARISENT
CERNERA	REGNERA	VERISTE	DELASSA
ACENSAS	ACENSAI	ECENTES	ECENTES

4. Quelques grilles solutions de celle qui est proposée en début d'article.

LES GRILLES AUTOMATIQUES

Revenons maintenant aux mots croisés en considérant une grille carrée de *N* cases de côté sans case noire, ce qui permet de simplifier la description de la méthode. Nous commençons par construire, à partir du dictionnaire général, un dictionnaire qui ne comporte que des mots de longueur *N*. Puis la génération s'effectue en deux étapes : on définit la taille de la grille, on place les mots que l'on désire y voir figurer (ce que l'on nomme la «potence» quand seules les premières ligne et colonne sont remplies, et le «canevas de grille» dans les autres cas) ; puis, dans le cas d'une potence, le générateur tente de placer un mot sur la seconde ligne, tout en vérifiant qu'il existe des mots, présents dans le dictionnaire, dans le sens vertical. Quand toutes les vérifications sont positives, le générateur tente de placer un mot sur la troisième ligne en effectuant les mêmes vérifications que précédemment, etc. En fin de travail, on obtient l'ensemble des grilles possibles correspondant à la potence.

Examinons plus précisément comment on place les lettres dans les cases. Considérons le stade où l'on essaie de placer une lettre dans la case *(i, j)* de la grille (*i* est le numéro de la ligne, et *j* celui de la colonne), après avoir correctement rempli toutes les cases précédentes. Nous notons *Placer(nœud, i, j)* la fonction récursive qui place une lettre dans la case considérée.

Cette fonction consiste d'abord à tenter de placer dans la case *(i, j)* la lettre correspondant au premier nœud situé derrière le dernier nœud exploré dans l'arbre (son «fils»). On vérifie alors qu'en ajoutant cette lettre horizontalement, il existe bien un début de mot vertical dans l'arbre dictionnaire. Si tel est le cas, on place la lettre trouvée dans la case *(i, j)* et on place à la case suivante, *(i, j+1)*, ou bien, si *j* est égal à *N*, dans la case *(i+1, 1)*. Cela revient à appeler *Placer(fils-de-nœud, i, j+1)* si *j* est différent de *N* et *placer(racine-du-dictionnaire, i+1, 1)* si *j* est égal à *N*. Si *i* et *j* sont tous deux simultanément égaux à *N*, alors on affiche la grille ; sinon, on poursuit l'algorithme.

«Ensuite, quoi qu'il arrive», on tente de placer dans la case *(i, j)* la lettre correspondant au premier frère du dernier nœud exploré (ce qui revient à appliquer l'algorithme avec ce premier frère comme nœud à explorer sans changer les paramètres *i* et *j*, c'est-à-dire à appeler *Placer(premier-frère, i, j)*. Si le dernier nœud exploré n'a pas de frère, alors on quitte l'algorithme en cours. Comme cet algorithme s'appelle lui-même, quitter l'algorithme en cours signifie qu'on remonte d'un niveau dans la récursion. L'algorithme, conformément à la clause «ensuite quoi qu'il arrive», cherche alors la lettre suivante dans la case précédente, à savoir la case *(i, j-1)* ou, si *j* est égal à *N*, dans la case *(N, j-1)*. Ainsi, en parcourant l'arbre de la racine vers les feuilles et du premier mot vers le dernier de la liste alphabétique, on explore toutes les grilles carrées de *N* cases de côté sans cases noires qu'il est possible de construire avec le dictionnaire utilisé.

Nous avons ultérieurement modifié cet algorithme afin qu'il accepte des cases noires, puis avec des lettres imposées dans certaines cases, mais le principe reste le même. Cet algorithme construit, nous avons cherché les grilles de sept cases horizontalement et de cinq cases verticalement, avec le canevas de grille GEORGES horizontal et PEREC vertical , le premier E de GEORGES servant de premier E de PEREC. Les grilles trouvées sont indiquées sur la figure 4.

Dans le cas des grilles carrées, avec seulement la première ligne passée en potence, on constate que, pour une même potence, on obtient assez souvent des grilles symétriques, les mêmes mots apparaissant en horizontal et en vertical, à la même position. Les grilles obtenues sont groupées par paquets dont les éléments diffèrent peu : par exemple, une grille de sept cases de côté comportera ACCEDAS en 1 vertical et SASSEES en VII horizontal, et l'autre aura ACCEDAT et TASSEES aux mêmes positions, le S commun ayant été remplacé par un T. Si l'on ne veut publier que des grilles originales, on devra choisir l'une d'entre elles.

Aidé par l'ordinateur, on oublie le cauchemar péréquien : on obtient facilement des grilles de neuf cases de côté avec trois cases noires, et l'on bat de deux cases le record de Robert Scipion, avec des grilles de 12 par 10 comportant seulement huit cases noires. Quant aux grilles de dix cases de côté avec seulement cinq cases noires, on en obtient une ribambelle. Naturellement, moins il y a de cases noires, plus le temps de calcul augmente.

Nous avons pris comme base la potence utilisée par Pérec dans sa tentative infructueuse de trouver une grille de sept cases de côté sans case noire. Le résultat a été obtenu en 347 secondes sur un ordinateur à base de processeur *Pentium* 120 MHz. La potence est représentée dans la première case de la figure 3, et les résultats (28 grilles) sont placés dans les cases restantes. On observe une fréquence élevée de S, de nombreux imparfaits du subjonctifs.

Qui trouvera maintenant des grilles parfaites de neuf cases de côté ?

Bernard IQUEAUX est responsable du développement de la Société *Phoenixx Systèmes*, qui commercialise les programmes de génération présentés dans l'article (*Minitel* : 3615 CRUCI et adresse *Internet* : http://www.biqueaux@adiabasoft.com).

Guy HACHETTE, *Des mots pour jouer*, éditions Jibena.

Georges PÉREC, *Les mots croisés*, éditions POL et Mazarine.

Josiane et Patrick BURGEL, *Le guide de l'Anagramme prémonitoire*, éditions Guy Tredaniel.

Andrew APPEL et Guy JACOBSON, *The World Fastest Scrabble Program*, in *Communications of the ACM*, vol. 31, n°5, mai 1988.