Mots Casés

Vassili Tremoureux Marie Noiret Lucie Barbieux

# Intro

Le but de l’application est d’effectuer un solveur de mots casés. L’utilisateur doit rentrer la forme de la grille ainsi que la tous les mots contenus dans la liste. A l’application ensuite de trouver la bonne place pour chaque mot dans la grille, et donc donner la solution de celle-ci.

Dans la méthode de résolution à la main, il faut essayer de déterminer dans un premier temps s’il y a un seul mot plus petit que les autres ou à l’inverse un plus grand. Parfois les grilles disposent d’une lettre déjà placée. Une fois que le premier mot est placé, il suffit alors de déterminer pour chaque mot qui croise le premier, la taille de celui-ci, puis d’aller chercher dans la liste un mot qui a la même lettre au croisement. Ensuite il faut répéter cette opération pour tous les croisements trouvés.

# Choix de la méta-heuristique

Afin de choisir la méta-heuristique la plus approprié à mettre en œuvre pour ce projet, il suffit de réfléchir à la façon d’adapter la méthode au sujet précis.

Recuit

Le principe du recuit, en métallurgie, est de chauffer un matériau, puis de le refroidir de façon contrôlée. Ainsi, les atomes se replacent de la manière dont on le souhaite, ce qui selon la matière en question et la façon dont c’est réalisé (temps de chauffe, température atteinte, temps de refroidissement…) rend le matériau plus ou moins résistant. Quand on l’applique à un algorithme de méta-heuristique, cela permet de trouver les extrema d’une fonction.

Si on décide de l’appliquer au cas précis, cela voudrait donc dire qu’une fois un mot placé, il faudrait ne changer qu’une lettre. Bien évidemment, cela n’est pas possible puisque le mot en perdrait alors tout son sens. Ce n’est donc pas le recuit qu’il faut utiliser pour notre problème.

Colonie de fourmis

Inspiré du comportement des fourmis, il s’agit d’un algorithme méta-heuristique d’optimisation. Le principe est qu’une fourmi se lance sur un graphe et dépose des phéromones sur son passage. Les fourmis suivantes vont tenir compte du chemin pris par la première pour prendre le leur, avec une certaine part de hasard. Le chemin ayant le plus de phéromones sera celui qui est optimisé. Ne voyant pas comment un tel principe pourrait être appliqué à notre sujet, nous avons décidé de ne pas s’attarder sur cette solution.

Algorithme génétique

Utilisant le principe de sélection naturelle, cette méthode permet de s’approcher un maximum de la solution souhaitée. La solution est approchée par « bonds » successifs, comme dans une procédure de [séparation et évaluation](http://fr.wikipedia.org/wiki/S%C3%A9paration_et_%C3%A9valuation).

Pour pouvoir utiliser cet algorithme dans le cas précis, il faut considérer comme gènes non pas les lettres du mot, mais un mot en entier. Dans ce cas précis, cette méta-heuristique peut s’appliquer. C’est donc celle que nous avons choisie de mettre en œuvre.

# Choix techniques

Afin de réaliser ce projet, nous avons choisi d’utiliser le langage C++. Il permet de gérer des objets, et c’est un langage que nous commençons à maîtriser. Il nous a donc paru plus simple de partir sur un langage connu du groupe entier.

Pour gérer la partie graphique de l’application, nous avons décidé de nous tourner vers la libraire graphique SMFL. C’est une librairie avec laquelle nous n’avons jamais travaillé encore, mais qui est très bien documentée et qui semble plus simple d’utilisation et plus performante que la très connue SDL. De plus elle est multi-langage, ce qui rend son apprentissage intéressant par la suite, pour d’autres projets. Pour finir, c’est une librairie multimédia qui possède cinq modules : système, fenêtrage, graphisme, audio et réseau.

# Mise en application

Interactions avec l’utilisateur

Rappelons que pour que l’application fonctionne, l’utilisateur doit lui passer certaines informations. Nous avons pris le parti de faire communiquer l’utilisateur et l’application via de simples fichiers textes.

Le premier concerne la grille et sa forme. Dans ce fichier, il y a deux lignes qui permettent à l’utilisateur de savoir ce qu’il faut écrire en dessous (respectivement les lignes 0 et 2). Ces lignes sont automatiquement ignorées par le programme. Il faut donc veiller à ce qu’elles soient à la bonne position. A la ligne 1, on retrouve la taille de la grille complète, le premier chiffre concernant le x (et donc la largueur) et le deuxième concernant le y (soit la hauteur). A partir de la ligne 3, l’utilisateur doit indiquer les coordonnées des différentes cases qui sont inaccessibles. Une ligne équivaut à une coordonnée. Pour en écrire d’autres, il faut faire un saut de ligne. Il est important de respecter ce formalisme, car il est prévu comme tel dans l’application.

Le second fichier concerne la liste des mots qui sont à placer dans la grille. Un traitement est fait à la lecture du fichier : les caractères spéciaux sont supprimés, et tout est passé en majuscule. En effet, dans les grilles de mots casés, tous les caractères sont en majuscule, et cela pourrait poser des problèmes si certains mots contenaient des majuscules et d’autres non. De même pour les caractères spéciaux.

La modification de ces fichiers est donc laissée à l’utilisateur. Il doit ensuite renseigner le chemin des fichiers au début du programme. Il s’agit d’un chemin relatif, partant de l’endroit où est placée la racine du projet. Un dossier nommé « data » est prévu pour y placer ce fichier, mais il n’y a aucune obligation. L’utilisateur devra donc rentrer par exemple « data/nomDuDocument.txt ».

Pour finir, on demande à l’utilisateur la taille de la population qu’il souhaite générer et le taux de mutation.

Traitement des données

Une fois les données entrées par l’utilisateur, il faut bien évidemment les traiter. Dans un premier temps, on charge en mémoire la grille. Il est assez difficile d’effectuer des vérifications, et il faut donc croire l’utilisateur sur parole. On importe donc un premier temps la structure de la grille, avec les cases libres et celles qui sont prises. Ensuite on parcoure ce tableau et on stocke dans une collection chaque espace de mot disponible, c’est-à-dire la position en x et en y dans le tableau, l’orientation et le nombre de caractères.

Dans un second temps, il y a une fonction qui s’occupe d’importer les différents mots à placer et qui retraite toutes les données. On effectue aussi d’autres vérifications comme par exemple vérifier qu’un mot n’est pas trop long pour le nombre d’espaces disponible ; par exemple un mot de dix lettres alors qu’il n’existe au maximum que des espaces de neuf lettres disponibles.

L’étape finale de ce traitement est de déterminer quelles sont les cases qui sont des intersections. Elles serviront par la suite pour la fonction d’évaluation.

Mise en place de l’algorithme

Quand on utilise l’algorithme génétique, il faut plusieurs choses. Dans un premier temps, un tirage aléatoire. Pour cette étape, on parcourt le tableau des différents espaces de mots disponibles, on récupère sa taille, puis on tire au sort un mot dans le tableau contenant les mots de la même taille. On fait de même pour tous les espaces disponibles de la liste. Puis on réitère l’opération autant de fois que cela a été spécifié par l’utilisateur.

Une fois que ce tirage au sort sur toute la population effectué, il faut essayer de trouver la grille qui se rapproche le plus de la solution. Il faut alors passer par une fonction d’évaluation, qui va faire ressortir les meilleurs éléments. En effet cette fonction se concentre sur les cases de type intersection et vérifie les lettres qui sont contenues dedans. Si elles sont identiques, la fonction ajoute alors une certaine valeur au score global. Si le score obtenu par une grille atteint le score maximum possible, alors il suffit d’arrêter le programme, puisque cela signifie que l’on a trouvé la ou une des solution(s).

Si ça n’est pas le cas, il faut alors essayer de trouver la solution qui s’approche le plus du résultat, et pour ça faire appel à une roue biaisée, dont le but est d’accorder une place plus importante à certains éléments, pour qu’ils aient plus de chance d’être tirés au sort. Il faut donc se baser sur la fonction d’évaluation pour qui attribue la note afin de donner une meilleure place aux solutions les plus probables. A l’issue du passage dans la roue biaisée la population est divisée par deux. Cette nouvelle population repasse dans la fonction d’évaluation puis par la roue biaisée jusqu’à la solution soit trouvée ou qu’il ne reste qu’une solution.

Après la roue biaisée viens le croisement. On croise deux solutions entre elles, pour en former deux nouvelles. Ce croisement se fait sur fait sur un ou plusieurs points, déterminés aléatoirement.

Afin d’arriver plus facilement à la solution idéale, le taux de mutation se doit d’être assez importante. Le principe de la mutation est qu’une fois la roue biaisée faite, certains éléments de la population ont un gène qui mute, c’est-à-dire qu’il est de nouveau tiré au sort.