Sorbonne Université 04 Mars 2022
MASTER ANDROIDE M1 – RP

Projet de résolution de problèmes : Satisfaction de contraintes pour le Wordle Mind

Introduction et objectifs

L'objet de ce projet est de développer et tester des méthodes de satisfaction de contraintes et un algorithme génétique pour la résolution d'un problème de *wordle mind*. Nous considérons une version de ce jeu dans laquelle on doit découvrir un mot caché du dictionnaire ("appelé mot secret") qui se compose de n lettres. Le jeu consiste pour le joueur à deviner le mot. Pour obtenir de l'information le joueur peut proposer un mot du dictionnaire (voir fichier dico.txt) et le programme lui indique combien de caractères du mot proposé sont corrects et bien placés et d'autre part combien de caractères sont corrects mais mal placés (par exemple si le décodeur propose le mot "tarte" alors que le mot secret est "dette" on aura 2 bien placés et 1 mal placé (attention, contrairement à la version originale de "wordle" la réponse n'indique pas quelle lettre est bien placée et quelle lettre est mal placée). Le joueur peut alors tenter un nouvel essai et ainsi de suite jusqu'à ce qu'il tombe sur le mot secret qui engendrera n bien placés et la partie s'arrête. Le but est bien sûr de chercher à découvrir le mot secret en utilisant le moins d'essais possibles.

Dans ce projet, on s'intéresse à réaliser un programme qui, à chaque étape du jeu, est capable de proposer un nouveau mot à essayer qui soit compatible avec toutes les informations accumulées lors des essais précédents (on s'interdit de tester des combinaisons incompatibles avec l'information disponible, même si elle sont informatives).

Question 1. Expliquer pourquoi l'utilisation d'un tel programme permet de créer une séquence d'essais qui converge nécessairement vers la solution (le mot caché).

Partie 1 : modélisation et résolution par CSP

Dans cette partie, on décide de modéliser le problème de la recherche d'un mot compatible avec l'information disponible par un CSP à n variables $X_1, \ldots, X_n$ de domaine $D = \{a, \ldots, z\}$. Pour engendrer un mot compatible avec l'information disponible à l'itération courante on envisage deux algorithmes :

- A1 : retour arrière chronologique
- A2 : retour arrière chronologique avec arc cohérence (forward checking ou plus si necessaire).

Question 2. Présenter et expliquer vos procédures de recherche d'une solution compatible puis implanter ces procédures au cœur d'un algorithme itératif de détermination du mot secret. Donner les temps moyens de détermination du mot secret sur 20 instances de taille $n = 4$. Etudier ensuite l'évolution du temps moyen de résolution et du nombre moyen d'essais nécessaires lorsque n augmente (on ira au moins jusqu'à $n = 8$).

Partie 2 : modélisation et résolution par algorithme génétique

Dans cette partie, on décide d'aborder le problème de la recherche d'un mot compatible à l'aide d'un algorithme génétique. Le but de cet algorithme serait, après chaque nouvel essai, d'engendrer un ensemble E de mots compatibles avec l'ensemble des essais précédents. L'ensemble E de mots compatibles présentera une taille maximale $maxsize$ et l'algorithme génétique s'arrêtera dès que la taille maximale est atteinte. Afin de limiter les temps de calcul, l'algorithme génétique pourra également s'arrêter après un nombre maximal de générations $maxgen$ (l'algorithme génétique pourra donc s'arrêter même si la taille de E est inférieure à $maxsize$). Notez qu'il se peut que l'algorithme ne retourne aucun mot compatible après avoir atteint le nombre maximal de générations. Dans ce cas, vous pouvez continuer la recherche d'un mot compatible jusqu'à ce qu'un timeout soit atteint (de 5 minutes par exemple). Si à la fin de ce timeout aucun mot compatible n'a été engendré, on peut considérer que la méthode a échoué.

La population évoluera grâce aux opérateurs de croisements et de mutations. Différents opérateurs de mutations sont possibles (changement aléatoire d'un caractère, échange entre deux caractères, inversion de la séquence de caractères entre deux positions aléatoires, ...) et chacun pourra être choisi avec une certaine probabilité. Notez que ces opérations peuvent engendrer des mots interdits car ils ne figurent pas dans le dictionnaire ; dans ce cas on choisira le mot le plus proche dans le dictionnaire (au sens d'une distance de votre choix, par exemple ordre alphabétique ou distance d'édition). La probabilité d'un mot d'être sélectionné comme parent sera proportionnelle à sa valeur adaptative ("fitness"). La valeur adaptative d'un mot devra être liée aux nombres d'incompatibilités avec les essais précédents. Dès qu'un mot compatible est trouvé, il sera ajouté à l'ensemble E.

Question 3. Coder un algorithme génétique dans lequel le choix de la prochaine tentative se fera aléatoirement parmi l'ensemble E. Fixer les paramètres de probabilité de mutation, taille de E, taille de la population et nombre de générations afin d'obtenir des résultats de bonne qualité en un temps acceptable. Etudier l'évolution du temps moyen de résolution et du nombre moyen d'essais nécessaires lorsque n augmente et comparer avec les résultats obtenus pour la partie 1. Représenter sous forme de graphique les résultats obtenus.

Partie 3 : détermination de la meilleure tentative (question bonus, optionnelle)

Il peut être intéressant d'évaluer a priori la valeur informative d'une tentative (essai d'un mot) pour réduire efficacement l'espace des solutions admissibles. Proposer une ou plusieur méthodes pour évaluer a priori l'utilité d'une tentative donnée, à la suite de celles déjà effectuées. Une telle évaluation peut être utilisée pour choisir la meilleure tentative dans une population de solutions compatibles engendrée par l'algorithme génétique. On peut aussi modifier l'algorithme de la partie 1 pour engendrer plusieurs solutions compatibles avec l'information disponible et choisir parmi elles la meilleure. Dans les deux cas, on cherchera à évaluer à quel point la sélection de la meilleure tentative peut accélérer la résolution du problème.

Organisation et dates

- le langage d'implantation recommandé est Python, toutefois, si ce langage ne vous convient pas d'autres options sont éventuellement admissibles (Java ou C/C++).
- les projets doivent s'effectuer en binôme. Informez votre chargé de TD des binômes constitués dès que possible (thibaut.lust@lip6.fr). En cas de difficulté à constituer un binôme, une

dérogation pourra être accordée à titre exceptionnel après demande motivée (par mail) auprès du chargé de TD avec cc au responsable du projet (patrice.perny@lip6.fr).

— le projet doit être rendu au plus tard le **18 Avril 2022** à 23h59. Votre livraison sera constituée d'une archive zip qui comportera les sources du programme (avec les instructions pour l'exécution) et un rapport au format pdf (de préférence rédigé en LaTeX). Le plan du rapport suivra le plan du sujet et résumera les choix méthodologiques, les principales réalisations et les tests numériques. L'archive zip, dont le nom contiendra les noms des deux personnes constituant le binôme, devra être soumise sur Moodle.

— la soutenance du projet est prévue en salle TME le 21 Avril 2022. Prévoyez de pouvoir faire une démonstration sur machine de vos programmes.