Sorbonne Université 03 Mars 2020
MASTER ANDROIDE M1 – RP

Projet de résolution de problèmes : Satisfaction de contraintes pour le Master 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 *master mind*. Nous considérerons une version de ce jeu dans laquelle le code secret à découvrir se compose de n caractères alphanumériques (chiffres ou lettres **deux à deux distincts**) pris parmi l'ensemble D_p constitué des p premiers éléments de l'ensemble $\{0, \ldots, 9, A, \ldots, Z\}$ (on pourra se restreindre au cas où $4 \leq n \leq 18$ et $p = 2n$). Le jeu consiste pour un joueur (le codeur) à choisir un code qu'il garde secret et pour l'autre joueur (le décodeur) à deviner ce code. Pour obtenir de l'information le décodeur peut faire un essai, c'est-à-dire proposer un code et le codeur devra alors indiquer d'une part combien de caractères du code proposé sont corrects et bien placés et d'autre part combien de caractères sont corrects mais mal placés (par exemple si le décodeur propose A0B1 alors que le code secret est 0ABC on aura 1 bien placé et 2 mal placés). Le décodeur peut alors tenter un nouvel essai et ainsi de suite jusqu'à ce qu'il tombe sur le code secret qui engendrera 4 bien placés et la partie s'arrête. Le but est bien sûr de chercher à obtenir le bon code en utilisant le moins d'essais possible (voir l'illustration du jeu d'origine à la Figure 1).

FIGURE 1 – Exemple de jeu de Master mind à $n = 4$ (4 positions) et $p = 8$ (8 couleurs, pas forcément distinctes dans la version originale du jeu). Les pions rouges à côté d'une combinaison indiquent le nombre de billes de la bonne couleur et bien placées, les pions blancs le nombre de billes de la bonne couleur mais mal placées). Sur cet exemple, le joueur a trouvé le code secret (correspondant à 4 pions rouges) après 6 tentatives.

Dans ce projet, on s'intéresse à réaliser un programme qui, à chaque étape du jeu, est capable de proposer un nouveau code 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).

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

Partie 1 : modélisation et résolution par CSP

Dans cette partie, on décide de modéliser le problème de la recherche d'un code compatible avec l'information disponible par un CSP à n variables $X_1, \ldots, X_n$ de domaine D_p.

1.1 Pour générer un code compatible avec l'information disponible à l'itération courante on envisage trois algorithmes :

- A1 : engendrer et tester
- A2 : retour arrière chronologique
- A3 : retour arrière chronologique avec forward checking (le forward checking n'utilisant que les contraintes all-diff)

Implanter ces trois algorithmes au coeur d'un algorithme itératif de détermination du code secret. Déterminer les temps moyens de détermination du code secret sur 20 instances de taille $n = 4$ et $p = 8$. Etudier ensuite l'évolution du temps moyen de résolution et du nombre moyen d'essais nécessaires lorsque n et p augmentent (on pourra se limiter au cas où $p = 2n$ et ne faire que 10 essais par taille). Représenter sous forme de graphique les résultats obtenus.

1.2 On considère une variante du problème où le code peut admettre jusqu'à deux occurences du même caractère (on relâche donc la contrainte stipulant que les caractères doivent être deux à deux distincts). Montrer comment modifier l'algorithme A3 en un algorithme A4 pour s'adapter à cette nouvelle variante. Comparer les performances moyennes (en temps et nombre d'essais) de A3 et A4 pour le cas $n = 4$ et $p = 8$.

1.3 En revenant au problème initial (les caractères du code sont deux à deux distincts) proposer une amélioration du forward checking qui permette de tenir compte d'autres contraintes que la contrainte all-diff pour améliorer A3. Proposer et tester les performances de votre algorithme (que l'on nommera A5) par rapport à A3.

Partie 2 : modélisation et résolution par algorithme génétique

Dans cette partie, on décide de résoudre le problème de la recherche d'un code compatible avec un algorithme génétique (cas où les caractères du code sont deux à deux distincts).

Le but de l'algorithme génétique est de générer après chaque nouvel essai un ensemble E de codes compatibles avec l'ensemble des essais précédents. L'ensemble E de codes 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 code compatible après avoir atteint le nombre maximal de générations. Dans ce cas, vous pouvez continuer la recherche d'un code compatible jusqu'à ce qu'un timeout soit atteint (de 5 minutes par exemple). Si à la fin de ce timeout aucun code compatible n'a été généré, 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. La probabilité d'un code d'être sélectionné comme parent sera proportionnelle à sa valeur adaptative ("fitness"). La valeur adaptative d'un code devra être liée aux nombres d'incompatibilités avec les essais précédents. Dès qu'un code compatible est trouvé, il sera ajouté à l'ensemble E.

L'avantage de l'utilisation d'un ensemble de codes compatibles est que l'on peut ensuite tenter de déterminer parmi cet ensemble la meilleure prochaine tentative.

2.1 Coder une première version de l'algorithme génétique. 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.

2.2 Etudier ensuite d'autres façons de choisir la prochaine tentative parmi E. Par exemple :

- Choix du code présentant le plus de similarité avec les autres codes compatibles.
- Choix du code présentant le moins de similarité avec les autres codes compatibles.
- Estimation du nombre de codes compatibles restants si un code était tenté, ...

Comparer ces différentes options en terme de temps de calcul et nombre d'essais et déterminer la meilleure option.

2.3 Etudier ensuite l'évolution du temps moyen de résolution et du nombre moyen d'essais nécessaires lorsque n et p augmentent et comparer avec la méthode présentant les meilleurs résultats de la Partie 1. Représenter sous forme de graphique les résultats obtenus.

Organisation et dates

- le langage d'implémentation 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 **3 mai 2020** à minuit. 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 7 mai 2020. Prévoyez de pouvoir faire une démonstration sur machine de vos programmes.