Université Paris 6 08 Février 2016
MASTER ANDROIDE M1 – RP

Projet de résolution de problèmes, génération de mots-croisés

Introduction et objectifs

L'objet de ce projet est de développer et tester des méthodes de satisfaction de contraintes pour la génération de mots croisés. Etant donné une grille vide incluant quelques cases noires ça et là, et un dictionnaire de mots admissibles (français ou anglais), il s'agit de compléter la grille de manière à ce que tous les mots formés horizontalement ou verticalement dans la grille appartiennent au dictionnaire (on ne s'intéresse pas ici à la génération des définitions). On peut aussi vouloir compléter de la même manière une grille partiellement remplie par l'utilisateur. Enfin on peut aussi vouloir éventuellement obtenir diverses solutions pour la même grille.

1. Modélisation par un CSP et résolution

1.1 Proposer une modélisation du problème comme un problème de satisfaction de contraintes. On prendra soin d'expliquer quelles sont les variables du problème, leur domaine et les contraintes à satisfaire. Montrer comment adapter cette modélisation si l'on ajoute la contrainte supplémentaire qu'un même mot ne peut apparaitre plus d'une fois dans la grille (contrainte que l'on pourra éventuellement inclure dans la résolution).

1.2 Réaliser un programme qui permette de charger une grille prédéfinie ou d'engendrer aléatoirement une grille de taille prédéfinie puis une interface qui permet de la visualiser. Cette inferface devra permettre de visualiser la grille initiale ainsi que tout instanciation partielle ou totale des variables. On devra aussi pouvoir sauver une solution trouvée.

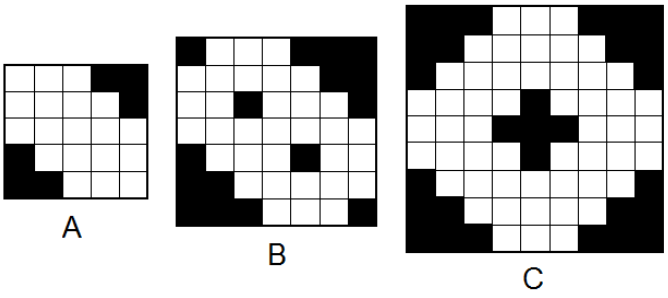
1.3 Développer une procédure d'initialisation établissant l'arc consistance du graphe des contraintes, avec un algorithme de type AC3. Développer ensuite une procédure de recherche d'une solution par retour arrière chronologique avec Forward Checking (FC). On précisera les heuristiques utilisées pour le choix de la variable à instancier, et on expliquera comment les instanciations sont ordonnées (en cas d'égalité dans les priorités on pourra faire des choix aléatoires pour créer de la diversité dans les exécutions répétées du programme sur une même grille).

1.4 Développer un algorithme de Conflict BackJumping (CBJ) susceptible d'accélérer éventuellement l'algorithme de forward checking proposé.

2. Expérimentation

L'objet de cette section est de tester les performances des algorithmes proposés dans la section précédente.

2.1 Appliquer AC3 puis un RAC avec Forward Checking sur les 3 grilles suivantes, et donner les temps moyens de résolution pour AC3, FC sans AC3 préalable, FC avec AC3 préalable, sur les grilles A, B,C suivantes :



On effectuera des tentatives de résolution en français et en anglais. Pour cela on pourra utiliser les dictionnaires joints, en commençant par les plus petits dictionnaires.

2.2 Etudier l'apport du Conflict BackJumping pour accélérer éventuellement les temps de résolution des grilles A, B, C.

2.3 En utlilisant la version la plus efficace que vous ayez obtenue, prolonger les expérimentations sur des grilles carrées plus grandes. On étudiera, en fonction de la taille de la grille, le temps moyen de résolution de votre algorithme, et le pourcentage de grilles que vous êtes parvenues à résoudre (sauvegarder les solutions). Pour obtenir des grilles on pourra utiliser les grilles proposées sur la page suivante : `http://puzzles.about.com/od/howtostutorials/ig/CrosswordGrids/` ou tout autre type de grilles, y compris des grilles tirées aléatoirement.

3. Extension au cas pondéré

On suppose maintenant que chaque mot du dictionnaire est muni d'un poids positif ou nul. Ces poids sont choisis dans l'ensemble [0,1] et peuvent, selon les cas, représenter l'importance qu'un utilisateur attache à un mot ou la fréquence d'occurence d'un mot dans un texte (livre, mémoire de thèse, page internet). On souhaite alors proposer un programme qui permette de rechercher la solution de poids maximum dans une grille, le poids d'une solution étant défini par agrégation des poids des mots qui figurent sur la grille.

3.1 Modéliser ce problème comme un CSP valué. On prendra soin de proposer une ou plusieurs structures de valuation adéquates pour ce problème et de discuter leur intérêt. En particulier on vérifiera que ces valuations sont monotones (par rapport à l'inclusion sur les interprétations).

3.2 Proposer alors un algorithme de type Branch and Bound pour rechercher une solution optimale.

3.3 Expérimenter cet algorithme pour la résolution d'instances de petite taille, puis essayer progressivement d'augmenter la taille des instances traitées pour voir comment évoluent les temps d'execution. Appliquer ce programme à la genération d'une grille de mots croisés qui évoque le master ANDROIDE en français comme en anglais. Pour cela on constituera un petit dictionnaire avec des mots clés utilisés fréquemment dans les cours du master ANDROIDE ainsi que le nom du master, tous de poids 1. On augmentera ce dictionnaire par un des dictionnaires donnés en attribuant un poids inférieur à 1 à ces mots supplémentaires utilisés éventuellement pour faciliter la complétion de la grille. On peut ainsi chercher une solution à toute grille en privilégiant l'utilisation des mots clés. Donner la ou les meilleurs grilles "ANDROIDE" ainsi obtenues.

3.4 *Question bonus facultative :* réaliser un programme qui prend en entrée un texte, qui calcule les fréquences d'occurence des mots de plus de trois lettres et qui utilise ces fréquences pour remplir une grille de mots croisés en privilégiant l'utilisation des mots les plus fréquents.