Premiers pas avec `Choco`

Arnaud Malapert

4 octobre 2012

1 Installation d'Eclipse et Maven

1. Installer le Java Development Kit (JDK6)
   - Tutoriel : Windows
   - Download link : Java SE 6 Update 35 (JDK)
   - dans un terminal, entrez la commande `java -version`.
2. Installer l'environnement de développement Eclipse
   - Tutoriel : Linux &Windows
   - Download link : Eclipse IDE for Java Developers
3. Installer Maven
   - Tutoriels : Multiplateformes & Windows (détaillé)
   - Download link : Maven 3
   - *ATTENTION aux valeurs des variables d'environnement.*
   - dans un terminal, entrez la commande `mvn -v`.
4. Installer le plugin Maven m2eclipse.
   - *plugin pré-installé dans la dernière version d'eclipse (juno)*
   - Tutoriel : Linux & Windows

2 Premiers pas avec Maven

Maven est un outil open-source de build pour les projets Java très populaire, conçu pour supprimer les tâches difficiles du processus de build. Maven utilise une approche déclarative, où le contenu et la structure du projet sont décrits, plutôt qu'une approche par tâche utilisée par exemple par Ant ou les fichiers make traditionnels. Cela aide à mettre en place des standards de développements au niveau d'une société et réduit le temps nécessaire pour écrire et maintenir les scripts de build.
Pour plus d'information, vous pourrez lire une introduction à Maven.

Dans notre cas, nous utiliserons Maven pour :

1. récupérer sur internet et lier au projet les fichiers jar du solveur `Choco` ;
2. construire un jar exécutable du projet ;
3. construire une archive du projet.

Pour construire le jar et l'archive du projet, vous pouvez taper commande `mvn assembly:single` à la racine du projet.
Rappelons qu'un fichier JAR (Java ARchive) est un fichier ZIP utilisé pour distribuer un ensemble de classes Java. Ce format est utilisé pour stocker les

définitions des classes, ainsi que des métadonnées, constituant l'ensemble d'un programme.

Nous allons maintenant importer le projet `choco-sandbox` dans Eclipse :

1. Récupérer l'archive contenant le projet `choco-sandbox`.
2. Extraire le contenu de l'archive dans votre workspace
3. Importer le projet le projet dans Eclipse :
   Import project ...→ Maven → Existing Maven Projects

Nous allons maintenant exécuter le programme dans Eclipse.

- ouvrez le fichier source `src/main/java/sandbox/SendMoreMoney.java`.
- clic droit → Run As → Java Application.

Nous allons maintenant faire un build avec Maven et exécuter le programme en ligne de commande :

- ouvrez un terminal et aller à la racine du projet
- entrez la commande `mvn assembly:single`
- allez dans le répertoire `target` et observez son contenu
- entez la commande `java -jar choco-sandbox-0.0.1-SNAPSHOT-jar-with-dependencies.jar`

3 Solveur de contraintes `Choco`

Choco est une bibliothèque java pour la programmation par contraintes hébergée sur sourceforge. Vous êtes invités à télécharger dès à présent la documentation.

4 SEND + MORE = MONEY

On cherche à associer un nombre différent à chaque lettre de façon à ce que l'équation SEND + MORE = MONEY soit vérifiée.

Question 4.1. Exécutez le programme et analysez la trace obtenu.

Question 4.2. Nous allons maintenant tester plusieurs stratégie de branchement (c.f. p.35 de la documentation) grâce aux fonctions statiques proposées par `choco.cp.solver.search.BranchingFactory`. Testez différents ordres statiques sur les variables (`lexicographic`) avec les heuristiques de sélection de valeur `minVal` et `maxVal`

5 Séquence magique

Une séquence magique de longueur n est une séquence d'entiers $x_1, \ldots, x_n$ compris entre 0 et $n-1$ telle que le nombre i ($i = 0, \ldots, n-1$) apparaisse exactement x_i fois dans la séquence (voir CSPLib). Par exemple, la séquence $6, 2, 1, 0, 0, 0, 1, 0, 0, 0$ est magique car 0 apparaît 6 fois, 1 apparaît deux fois ...

Question 5.1. Pourquoi x_i est-il strictement inférieur à n ?

Question 5.2. Proposez un modèle naïf basé uniquement sur des contraintes `occurrence`

Question 5.3. Proposez une contrainte redondante basée sur l'observation suivante :

$$6 + 2 + 1 + 0 + 0 + 0 + 1 + 0 + 0 + 0 = 10$$

Analysez son impact sur la résolution. Rappel : une contrainte redondante est impliquée par une ou plusieurs autres. En d'autres termes, il s'agit d'une contrainte qui n'est pas nécessaire, mais dont on espère qu'elle va aider à la résolution.

Question 5.4. Proposez une contrainte redondante basée sur l'observation suivante à propos des $i \times x_i$:

$$\underbrace{6}_{0,0,0,0,0,0} , \underbrace{2}_{1,1}, \underbrace{1}_{2}, \underbrace{0}, \underbrace{0}, \underbrace{0}, \underbrace{1}_{6}, \underbrace{0}, \underbrace{0}, \underbrace{0}$$

Analysez son impact sur la résolution. Rappel : une contrainte redondante est impliquée par une ou plusieurs autres. En d'autres termes, il s'agit d'une contrainte qui n'est pas nécessaire, mais dont on espère qu'elle va aider à la résolution.

Question 5.5. Essayez d'améliorer la résolution en utilisant d'autres stratégies de branchement (c.f `BranchingFactory`).

Question 5.6. Combien y a t'il de séquences magiques pour $n = 10, 50, 100$? Que remarquez vous ? Proposez un algorithme polynomial pour générer une séquence magique ? Est-ce que cet algorithme est valable pour tout n ?