INFO-F-302 Informatique Fondamentale Projet : Problèmes de Satisfactions de Contraintes et Utilisation de l'Outil ChocoSolver

El Haman Abdeselam Abdeselam Minhas Prabhdeep

Année académique 2016-2017

1 Problèmes d'échecs

1.1 CPS

Soient n, k1, k2 et k3 des entiers positifs. Dans le problème d'échecs, nous considérons un échiquier de dimensions n x n, ainsi que k1 tours, k2 fous et k3 cavaliers.

Ces pièces ont une différente manière de se déplacer. Les tours peuvent aller verticalement ou horizontalement. Les fous, quant à eux, attaquent aux diagonales. Enfin, les cavaliers se déplacent en 'L', c'est-à-dire de deux cases dans une direction combinées avec une case perpendiculairement.

Nous pouvons compter deux problèmes dans les problèmes d'échecs, le problème d'indépendance ainsi que le problème de domination.

- Le problème d'indépendance consiste à déterminer s'il est possible d'assigner à chacune des pièce une position distincte sur l'échiquier de sorte qu'aucune pièce ne menace une autre pièce.
- Le problème de domination consiste à déterminer s'il est possible d'assigner à chacune des pièces une position distincte sur l'échiquier de sorte que chaque case soit occupée ou menacée par au moins une pièce.

Afin d'exprimer ces deux problèmes par un CSP équivalent, nous avons commencé par écrire les variables, le domaine ainsi que les notations utilisées dans les contraintes pour ceux-ci.

La variable X est composée de toutes les coordonnées des pièces. Les tours sont notées t_i et t'_i, les fous f_i et f'_i et finalement les cavaliers h_i et h'_i.

Le domaine est compris entre 1 et n, n indiquant la taille de l'échiquier.

Pour les notations, nous avons la dimension de l'échiquier (n x n), ainsi que les notations pour représenter les différentes pièces (k1, k2, k3). De plus, m nous indique le nombre total de pièces que nous avons (k1 + k2 + k3). Enfin, y représente la suite de toutes les coordonnées des pièces, celles-ci sont alternées.

Ainsi, les coordonnées x sont représentées sans les guillemets et les coordonnées y avec. Les 2 x k1 premières sont les tours, les 2 x k2 suivantes les fous et les 2 x k3 dernières les cavaliers.

1.1.1 Question 1

La première question nous demandait d'exprimer une instance quelconque du problème d'indépendance. Nous avons 4 contraintes pour ce problème.

La première contrainte nous indique que deux pièces ne peuvent pas se trouver au même endroit sur l'échiquier. Nous l'avons écrite comme ceci, pour un ensemble (v1, ..., v2m) $\in D^{2m}$, il existe un i et j, différent et compris entre 1 et m, le nombre de pièce, tel que ces deux pièces ne peuvent pas avoir les mêmes coordonnées $(v_{2i-1} \neq v_{2j-1}) \vee (v_{2i} \neq v_{2j})$.
L'ensemble est jusque v2m car dans y nous avons le nombre de pièce multipliées par 2, pour avoir toutes les coordonnées. De la même manière, nous multiplions les indices i et j par deux, car nous les avons bornés jusque m, mais nous avons besoin de la seconde valeur dans y. Pour rappel, dans y, par exemple pour la première tour, nous avions mis t_1, suivit de $t'1$.

Pour les trois autres contraintes de ce problème, l'ensemble reste le même c'est seulement les conditions sur les deux pièces qui changent.

La seconde contrainte se porte sur les tours, elle indique qu'aucune pièce ne peut se trouver sur l'horizontal ou la verticale d'une tour, $(v_{2i-1} \neq v_{2j-1}) \wedge (v_{2i} \neq v_{2j})$. Pour ceci, les bornes de i sont entre 1 et k1, afin de ne prendre en compte que les tours, et j se trouve entre 1 et m. Ces deux pièces ne peuvent être à la même colonne et à la même ligne.

La troisième contrainte est sur la portée des fous, aucune pièce ne peut être sur les diagonales d'un fou, $((v_{2i-1} \neq v_{2j-1} + k) \vee (v_{2i} \neq v_{2j} + k)) \wedge ((v_{2i-1} \neq v_{2j-1} - k) \vee (v_{2i} \neq v_{2j} + k))$. Pour cette contrainte, le i est entre k1+1 et k1+k2, afin de n'avoir que les fous, étant donné que les k1 représentent les tours nous faisons un saut du nombre de tour présentes. Les bornes de j, par contre, restent les mêmes, de 1 à m. Dans cette contraintes, nous devons tenir en compte un autre indice qui définira les déplacements vers les diagonales, nous l'avons appelé k et celui-ci est compris entre {-n, n}. Nous vérifions donc dans les 4 sens s'il n'y a pas d'autre pièces présentes.

Enfin, pour la dernière contrainte sur les cavaliers, nous devons vérifier qu'aucune pièce ne se trouve sur les déplacements en L, $((v_{2i-1} \neq v_{2j-1} + k) \vee (v_{2i} \neq v_{2j} + l)) \wedge ((v_{2i-1} \neq v_{2j-1} + l) \vee (v_{2i} \neq v_{2j} + k))$. Pour cela, la borne inférieure de i est maintenant k1+k2+1, nous sautons donc toutes les tours et les fous. Et donc la borne supérieure est bien le nombre total de pièce m. En plus de ceci, nous avons un k compris entre {-2, 2} et un l entre {-1, 1}, afin de pouvoir indiquer le déplacement vers une direction de deux case suivit d'un mouvement

perpendiculaire.

1.1.2 Question 2

Une instance quelconque du problème domination devait être exprimée pour la seconde question. Nous avons 2 contraintes pour celle-ci, où la première reste la même que pour la première question, 2 pièces ne peuvent pas être sur la même coordonnée dans l'échiquier.

La deuxième contrainte indique qu'une pièce est soit :
— Occupée par une pièce
— Menacée par une tour
— Menacée par un fou
— Menacée par un cavalier
Les indices i et j sont bornés entre 1 et n, nous les passons en paramètre aux 4 ensembles exprimant les conditions citées ci-dessus.

Le premier ensemble signifie que pour une case dont les coordonnées sont {i, j}, il existe une pièce quelconque ayant les même coordonnées, $(\exists k \in \{1, ..., m\}, v_{2k-1} = i \wedge v_{2k} = j)$. L'indice k, appartenant à {1, m}, représente donc une pièce.

Dans le second ensemble, nous voulons qu'une tour aie la même ligne ou la même colonne que la case {i, j}, $(\exists k \in \{1, ..., k1\}, v_{2k-1} = i \vee v_{2k} = j)$. L'indice k symbolise les tours que nous avons, ses bornes étant entre 1 et k1.

De même le troisième ensemble indique qu'il existe un fou dans les diagonales d'une case {i, j}, $(\exists k \in \{k1+1, ..., k1+k2\}, p \in \{-n, n\}, ((v_{2k-1} = i+p \wedge v_{2k} = j+p) \vee ((v_{2k-1} = i-p \wedge v_{2k} = j+p))))$. Ici, vu que nous considérons les fous, les bornes de k sont comprises entre k1+1 à k1+k2. Et comme pour la première question, nous devons intégrer un autre indice, que nous avons appelé ici p, afin de se déplacer dans les diagonales d'une case.

Enfin le dernier ensemble exprime qu'il y a un cavalier aux déplacements L d'une case se trouvant aux indices {i, j}, $(\exists k \in \{k1 + k2 + 1, ..., m\}, p \in \{-2, 2\}, l \in \{-1, 1\}, ((v_{2k-1} = i+k \wedge v_{2k} = j+l) \vee ((v_{2k-1} = i+l \wedge v_{2k} = j+k))))$. Avec le même raisonnement qu'avant, les indices k, des cavaliers, sont compris entre k1+k2+1 et m. De plus, pour faire les déplacements en L nous avons besoin de deux autres indices, appelés ici p et l.

1.2 ChocoSolver

1.2.1 Question 3

Cette question nous demande d'implémenter un programme qui, étant donné une instance du problème de domination ou d'indépendance, en calcule une solution à l'aide de ChocoSolver.

Un format d'entré et de sortie est fixé, où nous pouvons passer en paramètre les différentes options que nous voulons pour exécuter le programme. Afin de parser les arguments, nous avons utilisé la librairie Argparse4j. Pour cela, un fichier jar est donné qu'il faut rajouter dans le buildpath du projet en maven.

Certains paramètres comme la dimension, le nombre de tour, cavalier, fous, sont obligatoire. En ce qui concerne le type de problème demandé, nous vérifions qu'il y a bien qu'un seul paramètre entré, soit un "-i" (indépendance) soit un "-d".

Pour le projet, nous avons décidé de créer une structure générique, où nous avons les classes tour, fous et cavalier héritant d'une classe Pièce. Cette classe pièce contient les méthodes que nous voulons implémenter pour tous les types de pièces existantes, et nous permet de nous renvoyer les coordonnées x et y de celles-ci. De plus, dans cette classe nous avons créé une méthode qui, prenant en paramètre des coordonnées x et y, peut vérifier si une pièce y existe déjà.

Les trois autres classes comportent les mêmes méthodes. Dans checkIndependency, nous vérifions qu'aucune pièce ne se trouve aux déplacements que peux effectuer la pièce en question. Cette fonction parcourt toutes les pièces, vérifie d'abord qu'aucune autre n'est présentes aux coordonnées x et y, puis applique les différentes contraintes que nous avons exprimée dans le CSP plus haut. Par exemple, pour les tours se serait vérifier que la position x et y de la tour est différente de toutes les pièces.

La seconde méthode présente dans les 3 sous-classes, s'appele inDomain, et prend en paramètre une case avec ses coordonnées. Elle nous renvoit, bien évidemment, le domaine tel que la case (X,Y) se trouve dans le domaine de la pièce en question. Par exemple pour les fous, on vérifie pour les diagonales de cette case la pièce fou s'y trouve.

Une fois que tous les arguments sont initialisés, nous créons un Model ainsi que toutes les pièces demandées par l'utilisateur.

1.2.2 Question bonus

1.2.3 Question 4

2 Surveillance de musée