TP2 - Modélisation en programmation par contraintes

Remise le vendredi 12 novembre (avant minuit) sur Moodle pour tous les groupes.

Consignes

- Le devoir doit être fait par groupe de 2 au maximum. Il est fortement recommandé d'être 2.
- Lors de votre soumission sur Moodle, donnez vos 4 modèles minizinc (.mzn) à la racine d'un seul dossier compressé (matricule1_matricule2_TP2.zip).
- Indiquez vos noms et matricules en commentaires au dessus des fichiers .mzn soumis.
- Il n'y a aucun rapport à remettre pour ce devoir.
- Toutes les consignes générales du cours (interdiction de plagiat, etc.) s'appliquent pour ce devoir.

Logiciels requis

Si ce n'est pas déjà fait, vous aurez besoin de télécharger MiniZinc et son environnement de développement.

Conseils

Ce TP est volontairement *challenging* pour obtenir la totalité des points. Voici quelques conseils pour le mener à bien :

1. Prenez-y vous tôt. Il y a également le temps d'apprentissage de Minizinc à considérer.
2. Travaillez efficacement en équipe, et répartissez vous bien les tâches.
3. Tirez le meilleur parti des séances de laboratoire encadrées afin de demander des conseils.
4. Prenez avantage des contraintes globales offertes par la librairie de Minizinc.

Bonne chance !

Enoncé

Pour ce deuxième TP, il vous est demandé de modéliser différents problèmes, de difficulté croissante, sur MiniZinc. Un tutoriel se trouve à même la documentation, vous pourrez passer au travers avant d'attaquer les exercices. Pour chaque exercice, un fichier de base `.mzn`, ainsi que différentes configurations à résoudre `.dzn` vous seront fournies. Vous êtes fortement encouragé à utiliser l'API de minizinc et d'identifier les contraintes globales les plus pertinentes pour résoudre les problèmes.

Vos modèles devraient être capables de résoudre chaque configuration en moins de 1 minute, en utilisant le solveur conseillé pour chaque problème [1]. Lors de la soumission, envoyez seulement vos quatre modèles Minizinc.

1. Et je compte très large, tout peut être résolu en moins de 10 secondes sur mon laptop 2,6 GHz Intel Core i5.

1 Infestation dans le jardin - easy level (3 pts)

Votre employeur aimerait obtenir votre aide pour résoudre un problème d'infestation d'insectes dans son jardin, qu'il modélise à l'aide d'une grille (Figure 1). Il voudrait utiliser des pesticides naturels afin de le remettre en bonne santé. Pour éliminer un insecte, il faut mettre une unité du pesticide naturel choisi. Il doit choisir exactement le bon nombre d'unités de pesticide à mettre dans chaque case du jardin. Pour ce faire, il devrait compter un à un les insectes de chaque case, ce qui est une tâche impossible.

Il s'est donc muni d'un outil permettant de compter le nombre d'insectes total dans une rangée ou une colonne. La grille suivante représente le jardin en question. La figure permet aussi d'illustrer le fonctionnement de l'outil.

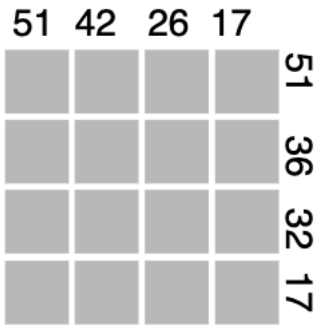

FIGURE 1 – Représentation du jardin et des mesures prises par l'outil.

L'outil indique ici par exemple que la première colonne contient 51 insectes au total, que la deuxième rangée en contient 36 et ainsi de suite. De plus, avant de réaliser que cette tâche était impossible à faire à la main, votre employeur avait déjà commencé à compter le nombre d'insectes dans quelques cases du jardin. Il vous fournit donc le nombre d'insectes exact pour certaines cases. Une dernière information qui vous aidera à résoudre ce problème est qu'il est impossible qu'un même nombre d'insectes se retrouve plus d'une fois dans la même rangée ou la même colonne.

Implémentez un modèle générique qui permet d'indiquer à votre employeur le nombre exact d'insectes dans chaque case. Utilisez le solveur `Gecode`. Trois configurations vous sont données dans des fichiers `.dzn`. Votre modèle devrait pouvoir trouver une solution correcte pour chaque configuration.

2 Tournoi d'échec - normal level (3 pts)

Le centre communautaire local a entendu parler de vos récentes prouesses en matière de résolution de problèmes. L'équipe a besoin de votre aide pour retrouver le score perdu d'un joueur lors de leur dernier tournoi d'échec. Pour vous aider à résoudre le problème, vous avez accès aux informations suivantes :

1. Il y avait cinq joueurs en tout.
2. Chaque joueur a joué une seule fois contre chacun des autres joueurs.
3. On a accordé aux joueurs 3 points pour une victoire, 1 point pour une nulle et 0 point pour une défaite.
4. Les scores finaux des quatre premiers joueurs sont 1, 2, 5 et 7.

Implémentez un modèle pour résoudre ce problème spécifique. Utilisez le solveur `Gecode`. Il n'y a pas de fichier de configuration pour ce problème. Tout est dans la base de code du fichier `.mzn` pour ce problème. Vous devrez vous assurer vous-même que la solution retournée par votre modèle est correcte.

3 Organisation d'un jardin intérieur - nigthmare level (4 pts)

Ayant trouvé le combat contre les insectes assez ardu, votre employeur songe à se tourner vers la conception d'un jardin intérieur. Il a déjà installé un certain nombre de lampes solaires à des endroits fixes. Jusqu'à présent, son jardin intérieur ressemble à la grille suivante :

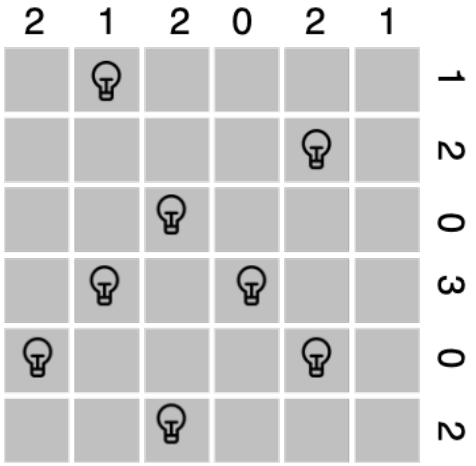

FIGURE 2 – Situation à résoudre.

Ne s'y connaissant pas beaucoup en la matière, il a demandé conseil à un botaniste pour savoir à quel endroit mettre les plantes. Après avoir visiter son jardin, le botaniste lui mentionne les critères suivants :

1. Pour des raisons techniques, un nombre fixe de plantes doit être installés sur chaque ligne et sur chaque colonne. Sur la figure, il s'agit des nombres écrits en haut et à droite de la grille.
2. Les plantes ne doivent pas se toucher horizontalement, verticalement ni en diagonale.
3. Chaque plante doit être directement à côté d'une lampe solaire (horizontalement ou verticalement).
4. Pour éviter d'avoir une lampe inutile, on impose également que chacune d'entre elles soit directement à côté d'au moins une plante (horizontalement ou verticalement).

Implémentez un modèle générique pour résoudre ce problème. Utilisez le solveur `Gecode`. Deux configurations vous sont données dans des fichiers `.dzn`. Vous devriez pouvoir trouver une solution faisable pour chaque configuration.

4 Planification de l'horaire d'infirmières - hell level (5 pts)

La crise sanitaire actuelle ayant semé le chaos dans le système de la santé, il est de plus en plus difficile de donner aux infirmières un horaire qui leur convient. Voulant remercier les infirmières de leur travail essentiel jusqu'à présent, le gouvernement fait appel à vous pour établir un horaire qui maximise la préférence des infirmières tout en respectant les exigences des hôpitaux.

Les critères à respecter sont les suivants :

1. Chaque période (`matin`, `soiree`, `nuit`) de travail est de 6 heures.
2. Il est possible pour une infirmière de prendre congé sur une journée, dès lors elle est notée en période repos. Aucune heure de travail n'est comptabilisée pour cette journée.

3. Afin de pouvoir mener à bien ses soins, l'hôpital a une demande minimale journalière qui doit être satisfaite pour chaque période, sauf pour le repos (`matin`, `soiree`, `nuit`).
4. Une infirmière ne peut travailler que sur une seule période par jour. Ainsi, elle sera soit identifiée en `matin`, `soiree`, `nuit` ou `repos` pour un jour donné.
5. Une infirmière ne peut travailler qu'au maximum 36 heures dans la semaine (du lundi au dimanche).
6. Une infirmière doit travailler au minimum 4 jours dans la semaine.
7. Une infirmière travaillant de nuit le jour j est obligatoirement en repos le jour $j+1$.
8. Une infirmière ne peut pas avoir un jour de travail isolé (c-à-d, on interdit la séquence suivante : `repos` - `travail` - `repos`).
9. Une infirmière ne peut pas travailler plus de 3 jours consécutifs.

Une base de code pour modéliser ce problème est déjà écrite pour vous. Voici quelques précisions sur les variables fournies :

1. T est le nombre de semaines de notre horizon temporel. Notez que pour T semaines, l'horaire devrait prendre en compte la planification de $7 \times T$ jours.
2. J est l'ensemble des jours de travail considérés. Chaque semaine commence un lundi. Dès lors $j = 1$ sera le lundi de la première semaine, $j = 7$ sera la dimanche de la première semaine, $j = 8$ sera le lundi de la deuxième semaine, etc.
3. $K = \{\texttt{matin}, \texttt{soiree}, \texttt{nuit}, \texttt{repos}\}$ est l'ensemble des périodes possibles.
4. I est l'ensemble des infirmières.
5. d_k^j est le nombre minimum d'infirmières nécessaires pour les opérations de la période $k \in K$ lors du jour $j \in J$.
6. $a_{i,k}^j$ est un coefficient indiquant le degré de préférence de l'infirmière $i \in I$ pour travailler la période k lors de la journée j.

Implémentez un modèle qui maximise la somme des préférences des infirmières. Utilisez le solveur `Gecode` ou `Coin-BC`[2]. Trois configurations vous sont données, vous pouvez les utiliser afin de tester votre modèle.

Vos missions sont maintenant achevées !

2. Petit détail technique : Coin-BC est un solveur qui n'est pas basé sur la programmation par contraintes mais sur la programmation en nombres entiers. Pour ce type de problème, et en fonction de votre modèle, ce solveur pourrait donner de meilleures performances. Ce type de solveur ne prend généralement que des contraintes linéaires, mais MiniZinc réalise une linéarisation en interne avant l'appel de ce solveur.