Travail pratique

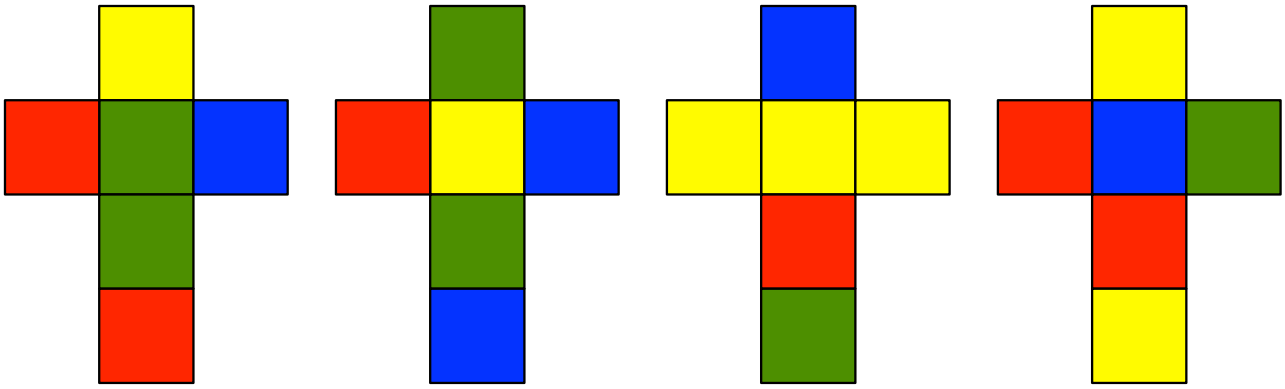
IFT 4001 / IFT 7020 Optimisation Combinatoire
Enseignant: Claude-Guy Quimper
Session: Automne 2024
Date de remise: 13 octobre 2023, 23h59
Équipes de 2 ou 3 personnes

- ⊘ Aucune communication entre les équipes n'est permise.
- ⊘ Vous n'êtes pas autorisés à publier cet énoncé, les fichiers qui l'accompagnent, ou votre solution sur GitHub ou tout autre medium.

Premier problème

Le premier problème consiste à résoudre le jeu des quatre cubes. Vous avez quatre cubes dont les faces sont colorées en jaune, en rouge, en bleu ou en vert. Vous devez aligner les cubes côte à côte de sorte à former un prisme rectangulaire de dimension 1 x 1 x 4. Les cubes doivent être disposés de façon à ce que les quatre couleurs apparaissent sur chacune des quatre faces rectangulaires du prisme.

Voici la façon dont les cubes sont colorés. Pour construire le jeu, découpez le contour des croix et pliez à angle de 90° le long des lignes. Vous obtiendrez les quatre cubes qu'il faut aligner.



Pour ce travail, vous devez modéliser avec le langage MiniZinc et résoudre ce problème avec le solveur Gecode ou Chuffed.

Les livrables

- Le code de programmation. Un seul fichier .mzn devrait suffire pour cette question;
- Un document PDF comportant;
  - Une page titre avec les noms et numéros de dossier de chaque membre de l'équipe;
  - Une description détaillée et complète du modèle (variables, domaines, contraintes). Il ne suffit pas de reproduire votre code de programmation ou de l'expliquer. Le lecteur de votre rapport doit comprendre votre modèle sans avoir jeté un coup d'oeil à votre programme. Inspirez-vous de la notation employée dans les exercices du chapitre 1. De plus, donnez une description, dans vos propres mots, de chaque variable et de chaque contrainte. Dites ce que représente l'affectation d'une variable X à une valeur v. Dites ce qu'encode chaque contrainte.
  - Une analyse de votre modèle: le nombre de variables, le nombre de valeurs (par domaine et au total), le nombre de contraintes et, s'il y a lieu, le nombre d'entrées dans les contraintes tableau.
  - La solution retournée par le solveur;
  - Le temps et le nombre de noeuds requis au solveur pour trouver votre solution.

Deuxième problème

Le deuxième problème que vous devez résoudre est un problème d'optimisation. Vous devez concevoir un horaire de travail pour les N employés d'une boutique devant accomplir A activités. L'horaire d'un employé doit respecter ces règles:

- Un employé doit travailler entre 6 et 8 heures (incluant les pauses, excluant le dîner);
- Un employé doit avoir une pause de 15 minutes, un dîner d'une heure et une autre pause de 15 minutes dans sa journée (dans cet ordre);
- Lorsqu'il entame une activité, un employé doit travailler pendant au moins une heure;
- Un employé peut seulement changer d'activité à la suite d'une pause ou du dîner;
- Un employé doit travailler avant et après une pause ou un dîner.

On représente un horaire de travaille avec une séquence de nombres où chaque nombre couvre une période de 15 minutes. Le nombre 0 signifie que l'employé ne travaille pas, c'est-à-dire qu'il n'a pas commencé à travailler, qu'il est en pause, qu'il dîne ou qu'il a terminé de travailler. Les nombres de 1 à A représentent des activités. La séquence suivante représente un employé qui commence sa journée par une période de 2 heures de travail à l'activité 1, suivie d'une pause de 15 minutes, suivie d'une heure et demie de travail à l'activité 2, suivie d'un dîner d'une heure, suivi de 2 heures de travail à l'activité 1, suivies d'une pause de 15 minutes, suivie d'une heure et de demie de travail à l'activité 2.

000001111111102222220000111111110222222000000

Pour chaque période t de 15 minutes de la journée et pour chaque activité a, on a une demande de d_{ta} employés, un sur-coût de C_{ta} et un sous-coût de c_{ta}. Si $d_{ta} = C_{ta} = c_{ta} = 0$ pour toutes les activités au moment t, alors aucun employé ne doit travailler. La boutique est fermée. Si $d_{ta} > 0$ pour une activité a, alors au moins un employé doit travailler sur cette activité. Soit x_{ta} le nombre d'employés travaillant à l'activité a au temps t dans votre solution. Si $x_{ta} > d_{ta}$, alors une pénalité de $C_{ta}(x_{ta} - d_{ta})$ est imputée à la solution. Si $x_{ta} < d_{ta}$ alors une pénalité de $c_{ta}(d_{ta} - x_{ta})$ est imputée à la solution.

Vous devez écrire un fichier `horaire.mzn` dans lequel on retrouve un modèle MiniZinc pour ce problème. Dans chaque fichier `bench_A` qui vous est remis, vous retrouverez les données d'une instance du problème avec *A* activités. À partir de ces données, créez un fichier `bench_A.dzn` compatible avec votre fichier `horaire.mzn`. On ne vous demande pas de remettre un programme faisant la conversion de fichier ni même d'en écrire un. Si c'est plus rapide pour vous, vous pouvez manuellement créer votre fichier dzn.

À titre indicatif, voici une solution (non optimale) pour le problème bench_3 à 3 activités et 6 employés.

```
000000000000000000000000000000000000033333333303333000011110111111111111000000000000000000000000
000000000000000000000000000000000000000022222222203333333333333000033330111100000000000000000000
000000000000000000000000000000000000000033333033333333000022222222222220333300000000000000000000
000000000000000000000000000000000033333033333333000022222220333333333330000000000000000000000000
000000000000000000000000000000000000011111111110333333000033333330333333000000000000000000000000
000000000000000000000000000000000000000000000000000011111103333333333330000333301111000000000000
```

Si le solveur prend trop de temps pour retourner une solution, considérez les options suivantes.

- Changez les heuristiques de recherche.
- Ajoutez une limite de 10 minutes au temps de résolution du solveur. Ainsi, vous n'obtiendrez pas une solution optimale, mais probablement une bonne solution.

- Modifiez votre modèle afin d'obtenir un filtrage plus fort. Un bon modèle a peu de variables et l'affectation d'une variable à une valeur filtre beaucoup les autres domaines.
- Si vous aviez à résoudre ce problème à la main, quelles décisions prendriez-vous pour construire un horaire? Assurez-vous qu'il existe une variable dans votre modèle permettant au solveur de prendre cette décision en lui affectant une valeur.
- Vous devriez pouvoir résoudre (à l'optimum) chaque instance en moins de 10 minutes.

Indice: Les contraintes vues au chapitre 1, combinées aux méta-contraintes du chapitre 3, sont suffisantes pour résoudre ce problème.

Vous devez aussi fournir une analyse de votre modèle, c'est-à-dire le nombre de variables, de contraintes et de valeurs dans les domaines en utilisant la notation Θ. Pour votre analyse, supposez qu'il y a P périodes dans une journée, N employés et A activités et donnez le nombre de contraintes, de variables, de valeurs et d'entrées dans les contraintes tableau (si applicable) en fonction de N, P, et A.

Les livrables

- Votre fichier `horaire.mzn` et vos fichiers `bench_A.dzn`.
- Votre programme doit afficher la solution et sa valeur objective.
- Un document PDF contenant:
    - Une description détaillée et complète du modèle (variables, domaines, contraintes, fonction objectif). Il ne suffit pas de reproduire votre code de programmation ou de l'expliquer. Le lecteur de votre rapport doit comprendre votre modèle sans avoir jeté un coup d'oeil à votre programme. Inspirez-vous de la notation employée dans les exercices du chapitre 1. De plus, donnez une description, dans vos propres mots, de chaque variable et de chaque contrainte. Dites ce que représente l'affectation d'une variable X à une valeur v. Dites ce qu'encode chaque contrainte;
    - Une analyse de votre modèle;
    - Une description des heuristiques utilisées (ou mentionnez que vous avez utilisé les heuristiques par défaut);
    - Les solutions trouvées;
    - Une mention justifiant si ces solutions sont optimales ou une mention disant que le solveur n'a pas réussi à démontrer que les solutions sont optimales;
    - Le solveur utilisé;
    - Le temps et le nombre de noeuds visités requis au solveur pour trouver chaque solution.

La qualité de la langue est évaluée dans les travaux. Il y a une pénalité d'un demi point par faute et une pénalité maximale de 10 points.

Références

La page web du langage MiniZinc:
https://www.minizinc.org

The MiniZinc Handbook (incluant un tutoriel à la section 2):
https://docs.minizinc.dev/en/stable/

Comment bien présenter un modèle?

Lorsque l'on présente un modèle d'optimisation dans un rapport, on le présente comme un modèle mathématique. Un lecteur ne connaissant pas MiniZinc devrait être en mesure de comprendre ce modèle et de l'implémenter dans le langage de son choix (MiniZinc, Prolog, Essence, …)

1. Privilégiez la notation arithmétique plutôt que la notation NomDeLaContrainte(Variables)
2. Utilisez la notation ∀ plutôt que la syntaxe `forall`.
3. Bien que dans un code de programmation, on privilégie les noms descriptifs, en mathématique, les variables sont généralement représentées par un seul symbole.
4. Utilisez la notation dom(X) pour définir le domaine d'une variable.

Voici un extrait de code MiniZinc et la façon dont il doit être présenté dans un rapport.

```
set of int: lignes = 1..n;
set of int: colonnes = 1..n;
int: SOMME_MAGIQUE = n * (n * n + 1) div 2;
array[lignes, colonnes] of var 1..n * n: x;
forall(i in lignes)(sum([x[i,j] | j in colonnes]) = SOMME_MAGIQUE;
```

Code MiniZinc

$$
\begin{aligned}
L &= \{1, \ldots, n\} && \text{Ensemble des lignes} \\
C &= \{1, \ldots, n\} && \text{Ensemble des colonnes} \\
S &= \frac{n(n^2+1)}{2} && \text{Somme magique} \\
\mathrm{dom}(x_{i,j}) &= \{1, \ldots, n^2\} && \forall i \in L, j \in C \\
\sum_{j \in C} x_{i,j} &= S && \forall i \in L
\end{aligned}
$$

Présentation demandée dans le rapport

Écrire en notation mathématique ne veut pas dire qu'il faut décomposer les contraintes. En effet, la contrainte AllDifferent n'est pas équivalente aux $O(n^2)$ contraintes de différence (≠) puisque les algorithmes de filtrage de ces deux contraintes ne sont pas les mêmes et le nombre de contraintes n'est pas le même: 1 vs $O(n^2)$.

Les heuristiques de recherche ne font pas partie du modèle mathématique. Vous n'avez donc pas à les décrire mathématiquement, mais vous devez les décrire dans vos propres mots. Encore une fois, l'heuristique doit être compréhensible par quelqu'un qui ne connaît pas MiniZinc. Plutôt que de dire « Nous utilisons les heuristiques `smallest` et `indomain_min` sur le vecteur *x* », vous devez plutôt écrire « Le solveur branche sur la variable dans le vecteur *x* ayant la plus petite valeur dans son domaine et l'affecte à cette valeur ».