Devoir 3 : Programmation par contraintes

Remise le 30 octobre sur Moodle (facultatif)

Consignes

- Les devoirs doivent être réalisés seul ou par binôme (de préférence par binôme).
- Soumettez un document (format pdf) reprenant votre devoir, ainsi que votre modèle MiniZinc (fichier `.mzn`).
- Indiquez vos noms et matricules dans le titre des fichiers soumis.
- Veillez à rendre un rapport **structuré**, **clair** et **concis**. Les lacunes de forme seront pénalisées.
- L'utilisation d'une IA générative est **STRICTEMENT INTERDITE**.

1 À la découverte de contraintes globales

Un des points forts de la programmation par contraintes est son vaste catalogue de contraintes globales. Pour cette question, il vous est demandé de parcourir la documentation MiniZinc afin de comprendre de nouvelles contraintes globales. Concrètement, donnez une explication en quelques lignes des contraintes suivantes et inventez pour chaque un exemple court pour illustrer :

1. `count(array[int] of var int: x, var int: y, var int: c)`
2. `global_cardinality(array[int] of var int: x, array[int] of int: cover, array[int] of var int: counts)`.
   Break up a `global_cardinality` constraint into multiple `count` constraints.
3. `cumulative(array[int] of var int: s,array[int] of var int: d, array[int] of var int: r, var int: b)`

Attention : il n'est pas suffisant de recopier la définition donnée dans la documentation. A la place, vous devez expliquer la sémantique des contraintes avec vos mots. Donnez également un exemple d'utilisation avec des variables/paramètres que vous définissez (soyez succincts, l'idée est plus importante que la rigueur de l'exemple). Pour cela, vous pouvez utiliser la syntaxe MiniZinc.

2 Un code secret de la plus haute importance

Vous êtes en mission pour désamorcer une bombe et vous avez pour cela besoin d'un code secret à cinq chiffres. Celui-ci est sécurisé par des règles bien particulières, et vous savez qu'il suit la logique suivante :

1. chaque chiffre a une valeur, entre 0 et 9, et une couleur, noir ou blanc (N ou B). Il y a donc vingt chiffres possibles ($0N$, $1N$, ..., $9N$, $0B$, $1B$, ..., $9B$) et les cinq chiffres du code sont à choix unique dans cette liste (il ne peut y avoir qu'un seul 1 noir, par exemple).
2. les chiffres sont en ordre croissant, avec le chiffre noir avant le blanc s'il y a deux fois la même valeur.
3. les 5 font exception pour la couleur : il n'y a pas de 5 noir ni blanc dans la liste des chiffres possibles, mais deux 5 verts.

Ainsi un code pourrait ressembler à $2N, 2B, 5V, 5V, 9B$.

Vous vous êtes infiltrés dans une salle de commande et après avoir injecté un logiciel espion dans l'ordinateur de contrôle vous obtenez pour le code les indications suivantes :

— deux chiffres du code ont la même valeur.
— les trois premiers chiffres du code sont les seuls chiffres adjacents ayant la même couleur.
— la somme des valeurs des trois derniers chiffres du code vaut 7.

Faites vite et bien ! Vous n'avez qu'un seul essai, et les gardes reviennent dans 2 minutes.

1. Modélisez ce problème en utilisant la programmation par contraintes. Formalisez mathématiquement les données, les variables de décision, la fonction objectif ainsi que les contraintes.
2. Modélisez ce problème sur MiniZinc. Pouvez-vous désamorcer la bombe à coup sûr ? Si non, quelle est votre chance de succès ?
3. Vous réalisez qu'il y a une seconde bombe à désamorcer, et obtenez cette fois-ci les renseignements suivants. Pouvez-vous désamorcer la bombe à coup sûr ? Si non, quelle est votre chance de succès ?
   — il y a un 2 en deuxième position du code.
   — la somme des valeurs des chiffres noirs du code vaut 3.
   — la somme des valeurs des chiffres blancs du code vaut 11.
   — il n'y a pas de 0 noir, de 1 blanc, de 7 noir ni de 9 dans le code.

3 Tournoi de tennis au CEPSUM

Vous êtes en charge d'établir un calendrier de match pour une ligue de tennis au CEPSUM. Le calendrier doit décider pour chaque match de quelles équipes il oppose ainsi que de la semaine à laquelle il a lieu, et doit garantir que :
— chaque équipe rencontre chaque autre.
— une équipe ne peut pas jouer plus de deux fois par semaine.
Le nombre d'équipes n'est pas encore connu. Afin d'être correctement préparé, votre modèle doit générer un programme de matchs valide qui minimise le nombre de semaines requises.

1. Modélisez ce problème en utilisant la programmation par contraintes. Formalisez mathématiquement les données, les variables de décisions, la fonction objectif ainsi que les contraintes.
2. Modélisez ce problème sur MiniZinc, en utilisant au moins une contrainte globale.
3. Résolvez le problème pour 5, 10 et 13 équipes. Comparez les performances de *Chuffed* et *Gecode*. Reportez et commentez brièvement *nodes*, *failures*, *solveTime* et *peakDepth* (des solveurs seront souvent plus adaptés à un problème que d'autres ; les statistiques de résolution sont obtenues avec *Show configuration editor* > *Options* > *Output* > *Output solving statistics*).
4. La résolution du problème était-elle plus dure pour 13 équipes ? Une modélisation basique ne passe pas bien à l'échelle pour ce genre de problème et ceci est partiellement dû à la nature symétrique du problème. Par exemple, pour un calendrier possible, plusieurs autres valides sont obtenus simplement en permutant le numéro des équipes. Quelle contrainte pouvez-vous proposer pour réduire la symétrie du problème ? Étudier son impact sur la résolution pour 13 équipes.