OPL - Allocation de Fréquences

Contraintes du problème FAP

- C1 : Les fréquences des différents `Transmitter` d'une même `Cell` doivent être espacées d'au moins 16.
- C2 : Les fréquences des `Cell` adjacentes doivent être espacées d'au moins `distance[ci][cj]`.
- C3 : Minimiser le nombre total de fréquences uniques utilisées, en réduisant autant que possible le nombre de fréquences distinctes attribuées.

Analyse du code OPL

```
int nbCells = ...; # Nombre de Cells
int nbFreqs = ...; # Nombre de fréquences disponibles
range Cells 1..nbCells;
range Freqs 1..nbFreqs;
int nbTrans[Cells] = ..; # Nombre de Transmitter par Cell
int distance[Cells, Cells] = ...; # Espacement minimal des fréquences entre Cells

# Définition du type Transmitter, incluant l'index de la Cell et du Transmitter
struct TransmitterType {Cells c; int t;};
{TransmitterType} Transmits = {<c,t>| c in Cells & t in 1...nbTrans[c]};
var Freqs freq[Transmits]

solve{
    # C1 : Les fréquences des différents `Transmitter` d'une même `Cell` doivent être espacées d'au moins 16
    forall ( c in Cells & ordered t1, t2 in 1...nbTrans[c])
        abs ( freq[<c, t1>] - freq[<c, t2>]) >= 16;

    # C2 : Les fréquences des `Cells` adjacentes doivent être espacées d'au moins `distance[ci][cj]`
    forall ( ordered c1, c2 in Cells: distance[c1, c2] > 0)
        forall ( t1 in 1...nbTrans[c1] & t2 in 1...nbTrans[c2])
            abs ( freq[<c1, t1>] - freq[<c2, t2>]) >= distance[c1,c2];
};

search{
    forall ( t in Transmits ordered by increasing <dsize(freq[t]), nbTrans[t.c]>)
        tryall ( f in Freqs ordered by decreasing nbOccur(f, freq))
            freq[t] = f;
};
```

Analyse du code Python

Variables

```
# Index
Cells = range(nbCells)
Freqs = range(1, nbFreqs + 1)
Transmits = [(c, t) for c in Cells for t in range(nbTrans[c])]
# Variables
freq = VarArray(size=len(Transmits), dom=Freqs)
```

Définition du tableau de variables `freq`, dont la taille correspond à la longueur de `Transmits`, et dont chaque élément a pour domaine `Freqs` (plage de valeurs possibles pour les fréquences).
Chaque `freq[i]` représente la fréquence attribuée au `Transmitter[i]`.

Contraintes

```
satisfy(
    # C1 : Les fréquences des différents `Transmitter` d'une même `Cell` doivent être espacées d'au moins 16
    [abs(freq[i] - freq[j]) >= 16
     for c in Cells
     for i, (c1, t1) in enumerate(Transmits) if c1 == c
     for j, (c2, t2) in enumerate(Transmits) if c2 == c and t1 < t2],

    # C2 : Les fréquences des `Cells` adjacentes doivent être espacées d'au moins `distance[ci][cj]`
    [abs(freq[i] - freq[j]) >= distance[c1][c2]
     for i, (c1, t1) in enumerate(Transmits)
     for j, (c2, t2) in enumerate(Transmits)
     if c1 < c2 and distance[c1][c2] > 0]
)
# C3 : Minimiser le nombre total de fréquences uniques utilisées
minimize(NValues(freq))
```

Comparaison des stratégies

Sans minimisation

option	temps	résolu
FrbaOnDom	5.65s	Oui
FirstFail	3.82s	Non
MaxDegree	4.03s	Non
MinDomain	5.11s	Non
DomOverDeg	4.28s	Non
Random	3.79s	Non
Min	4.98s	Non
Max	4.25s	Non
Random	4.07s	Non
OccurMost	6.30s	Non
OccurLeast	4.00s	Non

Avec minimisation

option	temps	résolu
FrbaOnDom	+10Mins	Non
FirstFail	4.25 s	Non
MaxDegree	4.10 s	Non
MinDomain	5.32 s	Non
DomOverDeg	4.17 s	Non
Random	4.12 s	Non
Min	5.19 s	Non
Max	4.51 s	Non
Random	4.22 s	Non
OccurMost	5.34 s	Non
OccurLeast	3.95 s	Non

FrbaOnDom n'a pas trouvé de solution en dix minutes et a retourné la même solution que sans minimisation après un arrêt forcé.

Problèmes rencontrés

1. Impossible d'expliquer pourquoi, sans options ou avec `options="-varh=FrbaOnDom"`, une solution est trouvée, alors que les autres options échouent.
2. Après l'ajout de `Minimise`, sans options ou avec `options="-varh=FrbaOnDom"`, le temps de calcul devient excessif sans trouver de solution, et l'arrêt forcé retourne la même solution que sans minimisation.
3. Dans le cas (2), on observe que l'utilisation de `varh` nécessite un peu plus de temps pour indiquer l'absence de solution, tandis que `valh` est légèrement plus rapide pour conclure à l'absence de solution.