MARAVAL Nathan
SHAO Yi

Mini-projet d'Intelligence articielle

Question 1 :

Le cardinal de Π est l'ensemble des permutations de 1 à n, c'est donc tout simplement la factorielle de n :

$$|\Pi| = (n)!$$

Complexité de la formule :

Pour chaque position possible il faut calculer l'aire :

Pour calculer l'aire il faut calculer n fois l'aire d'un triangle.

Pour calculer l'aire d'un triangle il faut faire 2 divisions et 2 multiplications.

Il va falloir faire le calcul de l'aire pour tous les diagrammes. On va considérer ici qu'il y a **m** diagrammes.

On a donc la complexité suivante :

O((n)! * m * n)

Optimisation :

On essaye de voir si des permutations sont inutiles :

exemple : abc
1 - {a,b,c}
2 - {a,c,b} #pareil que la 1

exemple : abcd
1 - {a,b,c,d}
2 - {a,b,d,c}
3 - {a,c,b,d}
4 - {a,c,d,b} #pareil que la 2
5 - {a,d,b,c} #pareil que la 3
6 - {a,d,c,b} #pareil que la 1

Comme il s'agit d'un cercle, la position ne compte pas.
Du coup dans l'exemple on fixe a au début.

De plus comme il s'agit d'un cycle, le même positionnement peut s'écrire dans les 2 sens. Dans l'exemple on observe ainsi des doublons.

Donc on a la formule : (n-1)! / 2
avec n = nombre de dimensions

On a donc une complexité suivante :

O(((n-1)! / 2) * m * n)

Question 2 :

On considère que le meilleure placement des positions des dimensions possible est celui qui maximise l'aire totale de tous les diagrammes.

Signature :

- Paramètres :
    - Liste des Diagrammes : listeDiag = {D1, …, Dm} de taille m
    - Liste des dimensions : listeDim = {d1,...,dn} de taille n
    - Une fonction Max(d) -> ℝ qui associe une valeur maximale à toutes les dimensions
    - Une fonction V(D, d) -> ℝ qui associe une valeur à toutes les dimensions de tous les diagrammes
- Sortie :
    - Une fonction P(ℕ $\in [1;\ n]$) -> d qui associe à chaque position une dimension différente

  Donc listeDiag x listeDim x (d -> ℝ) x (D x d -> ℝ) -> (ℕ -> d)

Pré-conditions :

- La taille de listeDiag peut être 1, 2, ou 3 mais dans dans ce cas le résultat est juste les dimensions dans l'ordre
- La taille de listeDiag peut être 0, mais dans ce cas F est vide.
- Pour toute dimension, on a une taille max associée

  $\forall(d) \ \in \ listeDim,\ Max(d) \ -> \mathbb{R}$
- Pour toute dimension de tout diagramme, on a une valeur associée

  $\forall(D) \ \in \ listeDiag,\ \forall(d) \ \in \ listeDim,\ V(D,\ d) \ -> \mathbb{R}$

Post-conditions :

- Toutes les dimensions ont une position associée

  $|P(\mathbb{N})| \ = \ |listeDim\,|$
- Toutes les dimensions ont une position unique associée

  $\forall(p1, p2) \ \in [1;\ n],\ P(p1) \ \neq \ P(p2)$

- Toutes les positions attribuées vont de 1 à n
  $\forall\ P(e\ \in \mathbb{N}),\ e\ \in [1, n]$
- La solution doit maximiser l'aire

Algorithme non déterministe :

```
P <- VIDE
P U (1, d1)

Dim' <- listeDim / {d1}
Positions = {2,..., n}

tant que Dim' n'est pas vide
    d = choix_nd(Dim')
    p = choix_nd(Positions)
    P = P U (p, d)
    Dim'  = Dim'  / {d}
    Positions = Positions  / {p}
fin tant que

retourne P
```

Question 3 :

(i)

Nous mettons en place une fonction qui va permettre de quantifier la ressemblance entre 2 diagrammes. Nous allons plus exactement avoir une valeur représentant la différence entre 2 valeurs : plus la valeur est haute, plus les les diagrammes sont différents. Pour faire cette valeur, nous allons comparer la valeur des 2 diagrammes sur chaque dimension. La valeur de retour sera alors la somme des différences sur chaque dimension.

Signature :

- Paramètres :
    - 2 diagrammes en entrée : DiagA et DiagB
    - Liste des dimensions : listeDim = {d1,...,dn} de taille n
    - Une fonction Max(d) -> $\mathbb{R}$ qui associe une valeur maximale à toutes les dimensions
    - Une fonction V(D, d) -> $\mathbb{R}$ qui associe une valeur à toutes les dimensions de tous les diagrammes
- Sortie :
    - un réel représentant la différence entre les 2 diagrammes
      la valeur 0 signifie que les diagrammes sont identiques

plus la valeur est élevé, plus les diagrammes sont différents

TODO : D x D -> ($\mathbb{R} \in [0;\ nbDimensions]$)

Pré-conditions :

- Pour toute dimension, on a une taille max associée

  $\forall(d) \in listeDim,\ Max(d) -> \mathbb{R}$
- Pour toute dimension des diagrammes D1 et D2, on a une valeur associée

  $\forall(D) \in \{DiagA\ ,\ DiagB\},\ \forall(d) \in listeDim,\ V(D,\ d) -> \mathbb{R}$

Post-conditions :

- Le réel sortie est compris entre 0 et le nombre de dimensions

  $retour \in [0;\ n]$
- Si la valeur de sortie est 0, alors les 2 diagrammes en entrée sont identiques

  $retour = 0,\ si\ et\ seulement\ si,\ DiagA = DiagB$

On considère que 2 diagrammes sont égaux quand toutes les valeurs de leurs dimensions sont égales.

Algorithme :

```
ressemblance()
        difference = 0
        pour toutes les dimensions dx:
                dxA = V(diagA, dx)
                dxB = V(diagB, dx)
                difference += ( | (dxA / max(dx) - dxB / max(dx) | )

        fin pour
        retourne difference
```

(ii)

Nous avons décider de disposer les diagrammes en ligne, en mettant côte à côte les plus ressemblants. Ainsi chaque diagramme aura à gauche et à sa droite des diagrammes semblables, afin de rendre le tout plus facile à lire. Cependant les diagrammes sur les extrémités gauche et droite ne seront comparés qu'à 1 diagramme.

Fonction permettant de quantifier l'optimisation des placements.

Signature :

- Paramètres :
  - Liste des Diagrammes : listeDiag = {D1, …, Dm} de taille m

- Liste des dimensions : listeDim = {d1,...,dn} de taille n
    - Une fonction Max(d) -> ℝ qui associe une valeur maximale à toutes les dimensions
    - Une fonction V(D, d) -> ℝ qui associe une valeur à toutes les dimensions de tous les diagrammes
- Sortie :
    - un réel représentant l'optimisation du placement
      la valeur 0 signifie que tous les diagrammes sont identiques
      plus la valeur faible, plus le placement est intéressant

  TODO : D x D -> ($\mathbb{R} \in [0;\ nbDimensions]$)

Pré-conditions :

- La taille de listeDiag peut être 0, 1 ou 2 mais dans dans ce cas le résultat est 0
- Pour toute dimension, on a une taille max associée

  $\forall(d) \in listeDim,\ Max(d) \rightarrow \mathbb{R}$
- Pour toute dimension de tout diagramme, on a une valeur associée

  $\forall(D) \in listeDiag,\ \forall(d) \in listeDim,\ V(D,\ d) \rightarrow \mathbb{R}$

Post-conditions :

- Le réel sortie est compris entre 0 et (le nombre de diagrammes - 1)* le nombre de dimensions

  $retour \in [0;\ (m-1) * n]$
- Si la valeur de sortie est 0, alors tous les diagrammes en entrée sont identiques

  $retour = 0,\ si\ et\ seulement\ si,\ \forall(D1)\ ET\ (D2) \in listeDiag,\ D1 = D2$

Algorithme :

```
OptimisationPlacementGrille()
        difference = 0
        pour tous les diagrammes Dx de (listeDiag \ Dm) :
                difference +=  ressemblance(Dx, (Dx+1), ...)
        fin pour
        retourne difference
```

(iii)

Signature :

- Paramètres :
    - Liste des Diagrammes : listeDiag = {D1, …, Dm} de taille m
    - Liste des dimensions : listeDim = {d1,...,dn} de taille n
    - Une fonction Max(d) -> ℝ qui associe une valeur maximale à toutes les dimensions

- Une fonction V(D, d) -> ℝ qui associe une valeur à toutes les dimensions de tous les diagrammes
- Sortie :
  - Une fonction G($\mathbb{N} \in [1;\ m]$) -> D qui associe à chaque position sur la grille un diagramme différent

TODO : D x D -> ($\mathbb{R} \in [0;\ nbDimensions]$)

Pré-conditions :
- La taille de listeDiag peut être 0, mais dans ce cas G est vide.
- La taille de listeDiag peut être 1 ou 2, mais dans dans ce cas le résultat est juste les diagrammes dans l'ordre
- Pour toute dimension, on a une taille max associée
  $\forall(d) \in listeDim,\ Max(d) \ {->} \ \mathbb{R}$
- Pour toute dimension de tout diagramme, on a une valeur associée
  $\forall(D) \in listeDiag,\ \forall(d) \in listeDim,\ V(D,\ d) \ {->} \ \mathbb{R}$

Post-conditions :
- Tous les diagrammes ont une position associée
  $|G(\mathbb{N})| = |listeDiag|$
- Tous les diagrammes ont une position unique associée
  $\forall(p1, p2) \in [1;\ m],\ G(p1) \neq G(p2)$
- Toutes les positions attribuées vont de 1 à m
  $\forall\ P(e \in \mathbb{N}),\ e \in [1, m]$
- La solution doit maximiser *OptimisationPlacementGrille()*

Algorithme non déterministe :

```
G <- VIDE

Diag' <- listeDiag
Positions = {1,..., m}

tant que Diag' n'est pas vide
        D = choix_nd(Diag')
        p = choix_nd(Positions)
        G = G U (p, D)
        Diag' = Diag' / {d}
        Positions = Positions  / {p}
fin tant que

retourne G
```