Les règles de Golomb

Sujet proposé par Philippe Baptiste

Philippe.Baptiste@polytechnique.fr

1 Définition des règles de Golomb

Une règle de Golomb est définie par n taquets positionnés à des valeurs entières de la règle $m_1 < m_2 \cdots < m_n$ telles que $\forall i, j, k, l \in \{1, ..., n\}$ avec $i < k$,

$$|m_j - m_i| \neq |m_l - m_k|.$$

Une règle de Golomb est donc un ensemble d'entiers naturels dans lequel les distances entre les éléments sont deux à deux distinctes.

Pour un entier n donné, une règle de Golomb est "optimale" si sa longueur $(m_n - m_1)$ est minimum parmi toutes les règles de Golomb possibles. Dans la suite on supposera toujours que $m_1 = 0$. Les règles de Golomb sont utiles pour les radaristes, les astronomes, les cristallographes, etc.

Voici quelques règles de Golomb optimales.

```
Règle      0    1    2    3    4    5    6    7    8    9    10   11
___________|____|____|____|____|____|____|____|____|____|____|____|__
n = 2      ^    ^
n = 3      ^    ^         ^
n = 4      ^    ^              ^         ^
n = 5      ^         ^                        ^    ^              ^
```

Le but de ce projet est de construire des règles de Golomb optimales pour $n \leq 10$. Nous nous intéresserons aussi à quelques variantes des règles de Golomb.

Notons que les règles de Golomb optimales sont connues jusqu'à $n = 24$. Il a fallu des dizaines d'années de CPU pour résoudre ce dernier problème.

2 Une recherche arborescente

Construire une procédure récursive qui, pour un entier n donné, énumère toutes les règles possibles et renvoie la règle optimale. Les règles seront construites en plaçant les taquets de gauche à droite. On notera qu'une telle procédure correspond à un arbre de recherche dont les noeuds sont exactement les appels récursifs.

En supposant que la recherche est limitée à 10 mn de CPU par règle, donner, pour chaque règle optimale qu'il a été possible de calculer, sa longueur, le temps CPU utilisé et le nombre de backtracks dans la procédure arborescente. Ces résultats seront présentés dans un tableau de synthèse.

3 Borne inférieure

Remarquons que la longueur de la règle optimale est au moins

$$LB_0(n) = \frac{n(n-1)}{2}.$$

Supposons maintenant que les positions $m_1, ..., m_q$ des q premiers taquets sont connues et qu'il reste donc à placer les $n-q$ derniers taquets. Proposer un algorithme qui permet de calculer une borne inférieur plus efficace que $m_q + LB(n-q)$. On pourra utiliser le fait que les distances $m_j - m_i$ pour $i \leq j \leq q$ sont interdites pour les taquets à positionner aprés m_q. On notera LB_1 cette borne inférieure.

Si au cours de la recherche arborescente, la borne inférieure (LB_0 ou LB_1) dépasse la longueur de la meilleure règle connue, il est alors possible de bactracker immédiatement. Tester l'efficacité de ces deux bornes (on donnera à chaque fois des tableaux complets de résultats).

4 Propriété de dominance

Pour $n \geq 3$, il existe une règle optimale dans laquelle $m_2 - m_1 \geq 2$. Pourquoi ? On dit alors que les règles de Golomb qui satisfont $m_2 - m_1 \geq 2$ sont dominantes.

Comment tenir compte de cette propriété dans la recherche arborescente ? Donner un tableau complet des résultats obtenus en combinant cette dominance avec LB_1.

5 Utilisation de règles précédemment calculées

Soit l_u la longueur optimale d'une règle de Golomb de u taquets. Dans une une règle de Golomb de n taquets, la postion du $u$ème taquet ($u \leq n$) est au moins l_u. Pourquoi ? Comment exploiter cette propriété dans la recherche ?

Donner un tableau complet des résultats obtenus en combinant cette règle avec la dominance et avec LB_1.

6 Cercles de Golomb

On chercher maintenant à construire une règle de Golomb en positionnant n marques sur un cercle. Les distances entre deux marques sont définies par la longueur du plus petit arc de cercle permettant de cheminer de l'une à l'autre. Sur un cercle de Golomb, les distances entre les marques sont toutes entières et, comme pour la règle de Golomb, les distances entre les marques sont toutes distinctes.

Proposer une méthode permettant de calculer un cercle de Golomb de périmètre minimal.

En supposant que la recherche est limitée à 10 mn de CPU par cercle, donner, pour chaque cercle optimal qu'il a été possible de calculer, son périmètre, le temps CPU utilisé et le nombre de backtracks de la procédure arborescente. Ces résultats seront synthétisés dans un tableau. Les cercles optimaux seront inclus dans le rapport (de préférence sous forme graphique).