Projet Prolog

April 7, 2022 11:04 AM

Partition: Groupe de clusters

Relation partition: Decrit un point d'une partition

The input to the algorithm is a knowledge base describing the current clustering over the partitions. It is made of facts using the partition predicate:

```
partition(PARTITION_ID, POINT_ID, X, Y, CLUSTER_ID).
```

- PARTITION_ID : ID de la partition a laquelle ce point appartient
- POINT_ID : ID du point en question
- CLUSTER_ID : ID du cluster auquel ce point appartient

A partir de ce prédicat, nous voulons vérifier si les groupes d'une partition intersectent avec les groupes des partitions adjacentes. Ceci est réalisé à partir de l'algorithme présenté ci-dessous. La sortie de cet algorithme sera une liste de groupes (fusionnés) représentée ainsi :

```
[[POINT_ID, X,Y,CLUSTER_ID], ...]
```

Relation representant un point d'un cluster
On peut peut-etre l'appeler point(POINT_ID, X,Y,CLUSTER_ID)

ClusterList

ClusterList contient plusieurs points de differents Clusters
Il peut donc avoir des elements avec le meme cluster_ID, mais jamais le meme Point_ID

```
ClusterList := 0     // list of clusters to be produced
for each partition P in knowledge base
   for each cluster C in P {
      for each cluster C in P {
         I := C ∩ ClusterList // List of points in ClusterList intersecting with points in C
         for each label L in I {
            change label L in ClusterList to Label(C)
         }

         ClusterList := C ∪ ClusterList
      }
   }
}
```

Knowledge base: Contient toutes les relations partitions

Indice: Puisque la boucle principale se fait à travers toutes les partitions et tous les groupes, il est aussi possible de fonctionner à partir d'une liste globale de groupes (retirant l'information sur les partitions). Cette liste globale s'obtient ainsi :

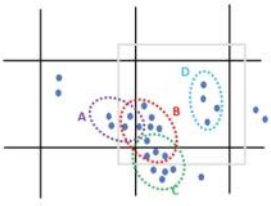
```
?- findall([D,X,Y,C],partition(_,D,X,Y,C),L).
L = [[1345, 40.750304, -73.952031, 65000001], [6017, 40.760146, -73.957873, 65000002], [17457, 40.760213, -73.955471, 65000003], [18582, 40.750299, -73.952027, 65000001], [20050, 40.750365, -73.952127, 65000001], [25351, 40.760153, -73.955467|...], [34767, 40.758621|...], [36487|...], [...|...]|...].
```

Knowledge base sans Partition_ID

-> Partition_ID est-il vraiment necessaire???

```
For each partition P | for each element (if partition_ID = P)
        For each cluster C in P |for each element  (if cluster_ID = C)
            For each cluster C in P |for each element  (if cluster_ID = C)
                I:= C INTERSECTION ClusterList | I : elements de ClusterList qui ont le meme point_ID qu'un element de C
                      For each label L in I | for each Cluster_ID in I
                          Change label L in ClusterList to Label ( C ) | Cluster_ID = a son equivalent en C
                ClusterList:= C U ClusterList | ClusterList.append(C )
```

Illustrons le fonctionnement de cet algorithme avec la figure suivante.



Supposons que nous avons les 5 points du groupe A et les 8 points du groupe C dans la liste courante `ClusterList`. Nous sommes à traiter la partition contenant les groupes B et D. Considérons d'abord le groupe D, celui-ci n'a pas d'intersection avec les points de `ClusterList`, alors ses 4 points sont simplement insérés dans cette liste. Maintenant si nous considérons le groupe B, celui-ci a 3 points en intersection avec le groupe A et 4 points en intersection avec le groupe C. Ces 7 points forment l'ensemble d'intersection I et les étiquette de I sont A et C. Il s'ensuit donc que tous les points dans `ClusterList` ayant l'étiquette A et C seront changés en B. Finalement les points restants de B sont insérés dans `ClusterList`.

Notons que cet algorithme ne vérifie si les partitions sont adjacentes avant de calculer les intersections. Ceci n'est pas très efficace car nous savons que les groupes provenant de partitions non-adjacentes ne peuvent pas avoir d'intersection. Toutefois, afin de simplifier notre problème, nous allons procéder ainsi et accepter ces quelques inefficacités. Il serait aussi possible d'initialiser la `ClusterList` avec tous les groupes provenant d'une partition initiale (au lieu de l'ensemble vide) mais encore, notre souci n'est pas l'efficacité. Notons enfin que pour le calcul des intersections, il est préférable d'utiliser le POINT_ID afin de comparer les points et non leur coordonnée X,Y.

Screen clipping taken: 2022-04-07 11:19 AM

```
?- partition(P,7030,40.749987, -73.94406,C).
P = 74,
C = 74000003 ;
P = 75,
C = 75000001 ;
P = 84,
C = 84000001 ;
P = 85,
C = 85000003 .
```

Exemple de point present dans plusieurs clusters

Le point 7030 est present dans les partitions 74,75 , 84 et 85
Il appartient aux clusters 74000003, 75000001, 84000001 et 85000003

A la fin de l'algorithme il ne devrait n'avoir qu'une UNIQUE relation (7030, 40.749987, -73,94406, C)

Avec C ∈ {74000003, 75000001, 84000001 ,85000003}