Projet de recherche opérationnel

Florian VANSTEENE

06/11/2017

1 Présentation générale

L'Objectif de ce projet est l'automatisation du placement de photos dans un album grâce à l'utilisation d'algorithme de recherche opérationnel. Le principe étant d'optimiser le placement d'une photo en fonction de différents critères tels que son empreinte (hash), les différents tags liés aux formes et objets présents sur la photo avec le taux de probabilité d'exactitude de la présence de ce tag dans l'image, les différentes couleurs de la photo ... La modélisation du problème de conception d'album en problème d'optimisation combinatoire associe à chaque répartition de photo sur l'album un score qu'il s'agit de minimiser s'il s'interprète comme une distance, ou de maximiser si le score s'interprète comme la qualité de l'album. L'espace de recherche est donc l'ensemble des solutions possible aillant pour taille (nbPhotos * (nbPhotos - 1))/2. Dans notre cas : (55*54)/2 soit 1485 solutions possibles.

2 Les différents outils

Pour mener à bien ce projet, les différents outils utilisés sont :

- Un fichier d'information sur l'album sous le format JSON.
- Un fichier d'information sur les photos sous le format JSON également.
- Un fcher au format SOL pour la chronologie d'apparition des photos (c'est ce fichier qui sera modifié en fonction de la solution déterminée par l'algorithme).
- Un dossier regroupant les différentes photos à classer.
- Un fichier de génération des pages HTML en fonction du fichier de chronologie et des différentes photos.

3 Les algorithmes utilisés

- Hill-Climbing : L'avantage de l'utilisation de cet algorithme est qu'il est peut couteux en terme de développement et d'exécution, on est sur de la maximisation ou minimisation, on s'attend alors à obtenir un optimum local assez rapidement. Cet algorithme est donc parfaitement adapté à notre problème combinatoire.
- Iterated Local Search en utilisant le Hill-Climbing comme algorithme de recherche local.

4 Les performances

L'exécution de l'algorithme confirme ce choix, la fitness chute très rapidement jusque 1000 evaluations puis diminue plus lentement cette fois, on obtient alors un optimum local assez rapidement et pour peu d'évaluations.

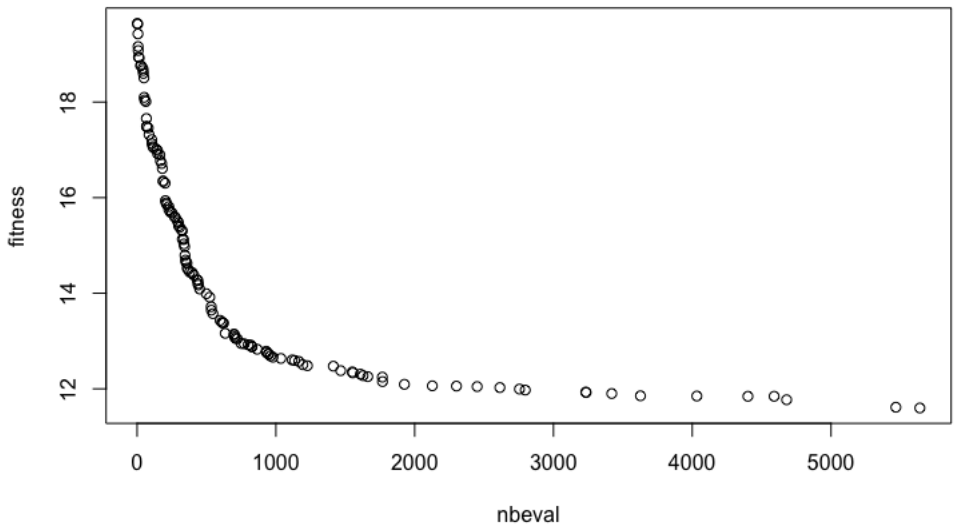

Figure 1: Exécution du Hill-Climbing

Pour éviter de stagner autour d'un optimum local et ainsi reprendre une phase d'exploration, on utilise l'Iterated Local Search. Ce qui va permettre, grâce à une perturbation d'une partie de la solution de repartir sur un autre point plus loin sur le graphique et ainsi augmenter la probabilité d'obtenir un optimum global : Il nous reste alors à "jouer" avec le pourcentage de perturbation de la solution afin d'augmenter au diminuer la distance du "saut".

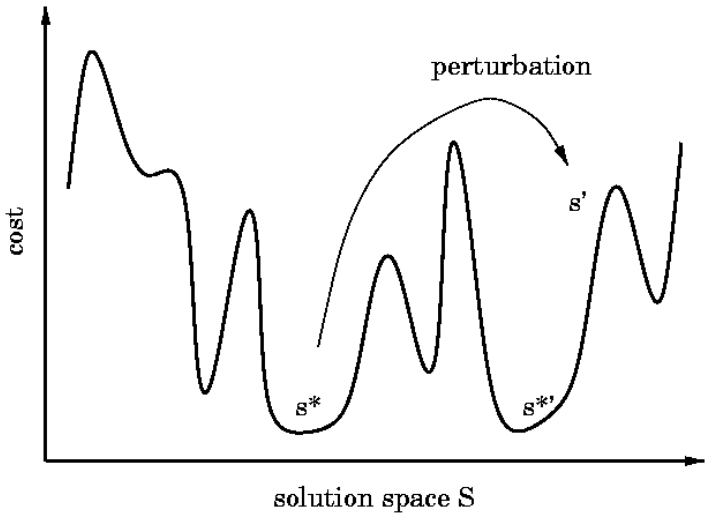

Figure 2: Schéma d'exécution de l'Iterated Local Search

5 Le choix des critères de selection

- Distance des empreintes(hash) : Dans le fichier JSON regroupant les informations des différentes photos, il y a trois type d'empreinte différentes : ahash (average), phash(perspective) et dhast(difference). Ainsi que leur distance avec chaque photo normalisée entre 0 et 1. On utilise alors la fonction d'evaluation suivante :

```
// Empreinte.java
for(int i = 0; i < Distances.getAlbumInvDist().length;
    i++) {
      for(int j = i + 1; j <
          Distances.getAlbumInvDist().length; j++) {
     sum += photoDist[ solution[i] ][ solution[j] ] *
         Distances.getAlbumInvDist()[i][j] ;
     }
}
```

  Ici l'objectif consiste à minimiser la distance entre les photos similaires par leur empreinte.

- Niveau de gris (greyavg) Pour ce nouveau critère de selection, mon choix est de garder la partie de minimisation des distance en fonction de l'empreinte et d'y ajouter le niveau de gris de chaque photo afin d'amélioré le tri effectué avec la méthode précédente. La fonction d'evaluation est la suivante :

```
// Grey.java
for(int i = 0; i < Distances.getAlbumInvDist().length;
    i++) {
     for(int j = i + 1; j <
         Distances.getAlbumInvDist().length; j++) {
     sum += photoDist[ solution[i] ][ solution[j] ] *
         Distances.getAlbumInvDist()[i][j] *
         photoGrey[i];
     }
}
```

- Tags Pour ce nouveau critère de selection, mon choix est de garder la partie de minimisation des distance en fonction de l'empreinte et de regrouper les photos en fonction de leur tags. Pour cela plusieurs opérations sont nécessaire :
  - Tout d'abord, pour chaque photo on récupère les 3 premiers tags (les 3 tags dont la probabilité est la plus élevée).
  - On obtient alors une liste de tags, on calcule ensuite le nombre d'apparition de chaque tag afin d'obtenir un classement des 3 tags les plus utilisés parmi les 3 tags les plus probables de chaque photo.
  - On decide alors de mettre un coefficient sur chacun de ses 3 tags : 0.10 pour le premier, 0.20 pour le deuxième, 0.40 pour le troisième et enfin 0.40 pour les autres.
  - On parcours ensuite à nouveau la liste des photos. Pour chaque photo, si l'un des 3 premiers tags fait partir des 3 plus utilisé, on affecte à la photo le coefficient correspondant.
  - On peut enfin utiliser ce coefficient pour l'evaluation d'une solution

  La fonction d'evaluation est la suivante :

```
// Tags.java
for(int i = 0; i < Distances.getAlbumInvDist().length;
    i++) {
     for(int j = i + 1; j <
         Distances.getAlbumInvDist().length; j++) {
     sum += photoDist[ solution[i] ][ solution[j] ] *
         Distances.getAlbumInvDist()[i][j] *
         (1-getClassement()[i]);
     }
}
```

- Combnaisons On décide ensuite de combiner la distance de l'empreinte de similitude des photos avec leurs niveau de gris et leur regroupement par tags:

```
// Grey_tags.java
for(int i = 0; i < Distances.getAlbumInvDist().length;
    i++) {
     for(int j = i + 1; j <
         Distances.getAlbumInvDist().length; j++) {
     sum += sum += photoDist[ solution[i] ][
         solution[j] ] *
         Distances.getAlbumInvDist()[i][j] *
         photoGrey[i] * (1-getClassement()[i]);
     }
}
```

Pour chacune de ces evaluations, l'objectif est de minimiser la distance entre les photos dans le but de regrouper les photos les plus semblables.

6 Résultat visuel

Après avoir demandé l'avis de plusieurs personnes, et par mon avis personnel, l'utilisation des critères : average hash distance + grey avg + regroupement par tags avec l'algorithme Iterated Local Search semble être la solution dont le placement des photos est le plus harmonieux.

On remarque par exemple sur ces 3 pages que les images semblables se suivent, les niveaux de gris sont assez proche et les catégories semblent être respectées.

Figure 3: Page 2 de l'album

Figure 4: Page 5 de l'album

Figure 5: Page 5 de l'album