2017

Projet RO : Album photo

Présentation :
Projet réalisé dans le cadre du Master Ingénierie du logiciel libre 1ère année. Il consiste à la recherche de la mise en place de la génération automatisée d'un album photo en suivant un problème d'affectation quadratique.

1. Introduction

Le projet de recherche opérationnel proposé a pour objectif de fournir à un album photos généré automatiquement à partir de photos fournies. Plusieurs informations sur ces photos, et sur l'album ont été mise à disposition au format JSON. C'est à partir du traitement de ses données, qu'il nous a été demandé de mettre en place des fonctions objectifs pouvant répondre à ce problème.

Revenons en tout d'abord au problème avant d'énumérer les différentes fonctions objectif proposées. Dans l'exemple fourni avec le sujet du projet, le jeu d'essai comprend 55 photos et peut fournir un album de 9 pages (6x9). Une photo n'est donc pas sélectionnée pour être affichée dans l'album mais cela n'est en soi pas dérangeant. Le fait qu'il y ait 55 photos revient à dire qu'il y a 55! possibilités de permutation de ces photos. Évaluer toutes ces possibilités serait une tâche réalisable par une machine dû moins pour un certain nombre de photos (5, 10 etc…), mais trop coûteuse en terme de ressource et de temps lorsque le nombre de photos engendre un trop grand nombre de possibilités. C'est pour cela que plusieurs algorithmes sont utilisés en se basant sur une heuristique de recherche aléatoire pour lesquels on va fixer une condition d'arrêt.

Pour la réalisation de ce projet, le langage de programmation multi-paradigme Scala a été préféré afin de profiter du paradigme de programmation fonctionnel qu'il incorpore. Étant lui-même une "surcouche" du langage Java, il s'exécute lui aussi au sein d'une JVM et permet l'utilisation directement d'instructions Java au sein de code Scala. Malgré une syntaxe du langage particulière liée à son multi-paradigme, l'adaptation avec le code précédent a été plutôt rapide à mettre en place. L'inconvénient de ce langage de plus haut-niveau que le C notamment est donc son coût cependant la notion de performance en terme de résultat reste identique. Etant donné que le projet ne nécessite pas de calculs trop rudimentaires son choix a été préservé.

2. Définition de l'espace de recherche

Comme évoqué précédemment, le nombre de possibilités d'ordonnancement d'un album constitués de 55 photos est de 55! soit 1,2 x 10^73. Il représente l'ensemble des possibilités de permutations des photos de l'album.

L'espace de recherche est donc défini comme tel pour l'ensemble des fonction objectifs :

S({1,...,n})

3. Représentation du problème

Les fonctions objectif développées, comme celle déjà mise en place en exemple, cherchent à optimiser l'emplacement des photos tel que chaque page soit cohérente. On définit une page cohérente comme une page comprenant des photos dont les critères calculés les définissent comme proches. On ne cherche donc pas uniquement à mettre en place une suite logique à l'ordre des photos mais à chercher un visuel plus appréciable pour l'utilisateur récepteur de l'album photo.

Le problème d'optimisation combinatoire rencontré est pour l'ensemble des fonctions objectifs développées, un problème d'affectation quadratique. En effet, nous essayons de trouver un ordre des photos dont le coût de notre fonction sera à minimiser. Les différents paramètres sont de ce problème d'assignement sont :

- La notion de distance entre chaque photo. Cette notion de distance varie en fonction du critère sélectionné au sein d'une fonction objectif.
- La notion de position au sein de l'album photo. Les valeurs associées à ces positions sont l'inverse des distances spatiales sur l'album.

De manière générale, la formulation mathématique du problème d'affectation quadratique de l'album est la suivante :

Ensembles

$N = \{1, \dots, n\}$

$Sn = \phi : N \rightarrow N$ (représente l'ensemble des permutations)

Paramètres

$D = (Dij)$ est une matrice $n \times n$ où Dij représente la distance entre une photo i et j

$P = (Pij)$ est une matrice $n \times n$ où Pij est la de l'inverse des distances spatiales i et j de l'album.

Formule du problème

$$\min_{\phi \in Sn} \sum_{i=1}^{n} \sum_{j=1}^{n} Dij \cdot P\phi(i)\phi(j)$$

4. Description des fonctions objectif

4.1. Fonctions objectif liées aux empreintes :

Les différents critères de qualités concernant les empreintes sont intégrés au sein d'une méthode commune, l'attribut lié est passé en paramètre tel que la méthode puisse initialiser les données concernant cet attribut. L'ensemble des données sont récupérées au sein du fichier JSON des photos.

La sélection de l'attribut (ahashdist, phashdist & dhashdist) se fait à l'exécution du programme.

Ainsi la matrice représentée par la variable **photoDist** du fichier Modelisation.scala est créée et comprend l'ensemble des données. Cette matrice est ensuite utilisée comme paramètre du problème d'affectation quadratique pour la notion de distance entre les photos en fonction du choix de l'utilisateur.

4.2. Fonctions objectif liées aux tags :

Trois types de données distances calculées pour les tags ont été mises en place :

- Distance des tags communs : $\sum^{n}_{i=1,\ j=1} \sum^{t}_{k=1,\ m=1} | x_{i,m} - x_{j,k}|$
  - **n** représente le nombre de photos
  - **t** le nombre de tags
  - **x** la valeur du tag

  La valeur est uniquement calculée si le tag est commun aux deux photos puis soustrait à la somme. Dans le cas contraire une valeur par défaut est ajoutée dans le but de pénaliser l'inégalité du tag. Ainsi une photo possédant plus de tags communs sera privilégiée.

- Distance des tags non communs : $\sum^{n}_{i=1,\ j=1} \sum^{t}_{k=1,\ m=1} | x_{i,m} - x_{j,k}|$
  - **n** représente le nombre de photos
  - **t** le nombre de tags
  - **x** la valeur du tag

  La valeur est uniquement calculée si le tag n'est pas commun aux deux photos puis ajouté à la somme. Dans le cas contraire une valeur par défaut est soustrait dans le but de privilégier l'égalité du tag. Ainsi une photo possédant plus de tags non communs sera pénalisée.

- Distance du nombre de tags non communs : chaque photo aura comme critère de distance le nombre de tags non communs avec les autres photos.

Ainsi les matrices sont représentées par les variables **photoDistancesCommonsTags, photoDistancesUncommonsTags, photoDistancesUncommonsNbTags** du fichier Modelisation.scala et comprennent l'ensemble des données. Ces matrices sont ensuite

utilisées comme paramètre du problème d'affectation quadratique pour la notion de distance entre les photos en fonction du choix de l'utilisateur.

4.3. Fonctions objectif liées aux couleurs :

Pour chaque photo deux couleurs ont été spécifiées parmis une palette de 16 couleurs. Ces couleurs sont représentées par les attributs "color1" et "color2" du fichier JSON.

Pour cette fonction objectif, une fonction mathématique permettant d'établir une distance entre deux couleurs a été utilisée :

Soit C1 et C2 des couleurs au format RGB, leur distance est définie tel que :

$$D(C1, C2) = \sqrt{(R_{C1} - R_{C2})^2) * (G_{C1} - G_{C2})^2) * (B_{C1} - B_{C2})^2)}$$

- R : représente la couleur rouge
- G : représente la couleur verte
- B : représente la couleur bleue

La formule mathématique des distances calculées pour la fonction objectif est la suivante :

$$\sum^{n}_{i=1} \sum^{n}_{j=1} D(x_i, x_j) + D(y_i, y_j)$$

Tel que **n** représente le nombre de photos, **x** la première couleur d'une photo et **y** la seconde.

Chaque photo comprendra une valeur de distance de couleur pour chaque autre photo. Ainsi la matrice représentée par la variable **photoDistancesColors** du fichier Modelisation.scala est créée et comprend l'ensemble des données. Cette matrice est ensuite utilisée comme paramètre du problème d'affectation quadratique pour la notion de distance entre les photos en fonction du choix de l'utilisateur.

4.4. Fonctions objectif liées à la moyenne de gris :

Un dernier critère de qualité a été mis en place, concernant l'attribut indiquant la moyenne de couleur grise d'une photo.La matrice comprenant les valeurs du paramètre de distance pour le problème d'affectation quadratique est calculée ainsi :

$$\sum^{n}_{i=1} \sum^{n}_{j=1} | x_i - x_j |$$

Tel que **n** représente le nombre de photos, et **x** la valeur de gris moyen associée à une photo.

Ainsi la matrice représentée par la variable **photoDistancesGreyAVG** du fichier Modelisation.scala est créée et comprend l'ensemble des données. Cette matrice est ensuite utilisée comme paramètre du problème d'affectation quadratique pour la notion de distance entre les photos en fonction du choix de l'utilisateur.

5. Interprétation des résultats pour chaque distance mise en place

5.1. Les algorithmes de résolutions

Les différents algorithmes utilisés, sont des algorithmes à solution unique :

- Hill Climber first improvement
- Iterated Local Search
- L'algorithme évolutionnaire

Le Hill climber first improvement permet de descendre vers un optimum local assez rapidement. Il est intéressant en termes de performance. Il se couple très bien avec la métaheuristique "Iterated local search" qui nous permettra de rebondir une fois un optimum local trouvé vers un autre optimum qui pourrait être l'optimum global.

A ces deux algorithmes s'ajoute l'algorithme évolutionnaire. Basé sur la sélection naturelle évoqué par Darwin, cet algorithme a pour objectif de garder les meilleurs solutions d'une génération avant de former une nouvelle génération à partir de celle-ci. Il utilise le Hill Climber first improvement afin de chercher une meilleure solution au sein des différentes générations dans le but d'améliorer celles qui vont suivre.

L'algorithme évolutionnaire est de manière générale assez lent et coûteux. Pour des raisons de temps, les résultats obtenus par celui-ci ne seront pas expliqués. Les remarques faites quant à ses performances de résultats sont les suivantes :

- Il est nécessaire d'envisager un certains nombres d'itérations afin d'obtenir un résultat proche de l'optimum.
- Il est beaucoup plus lent que les autres algorithmes pour un résultat pratiquement identique.
- L'heuristique de comparaison des solutions est sûrement à améliorer. Ce qui doit certainement entraîner sa lenteur de résolution.
- De façon générale, je pense que l'algorithme évolutionnaire n'est pas l'algorithme de résolution à solution unique le plus adéquat pour un problème de ce genre d'envergure.

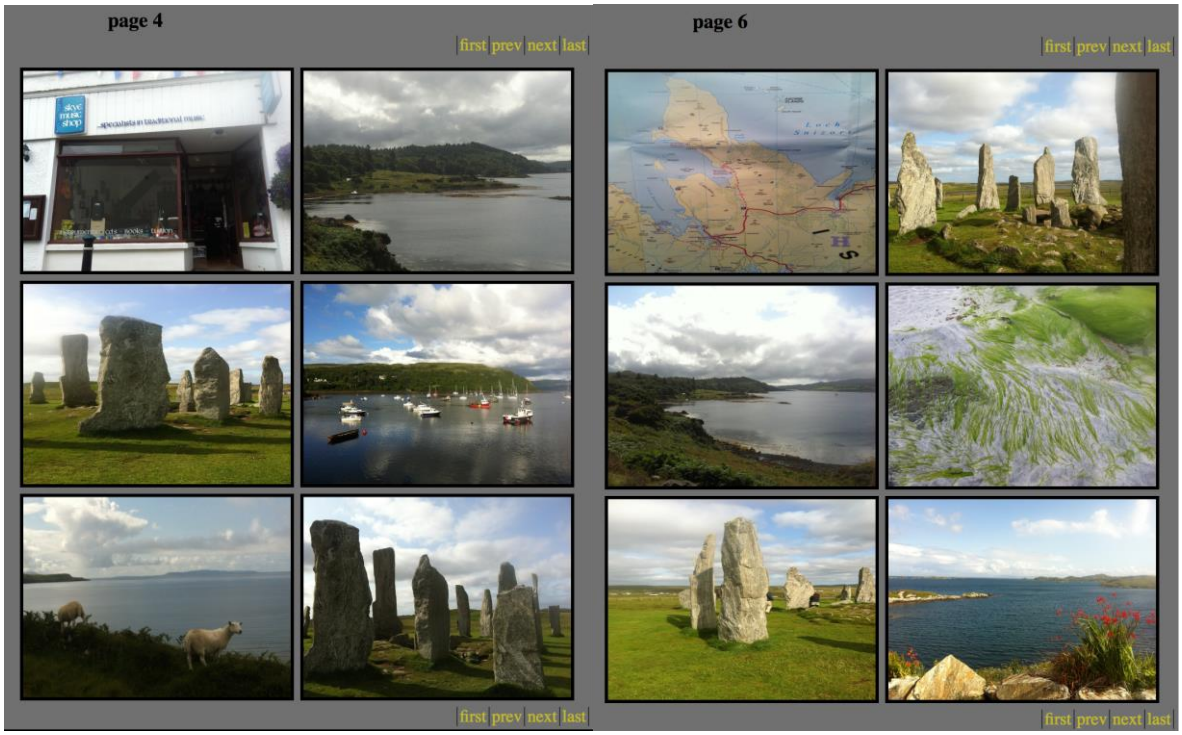
Pour chacune des matrices distances composant une fonction objectif les métaheuristiques **Hill Climber First Improvement** et **Iterated Local Search** ont été utilisé afin de trouver une solution proche de l'optimum. Chaque fonction objectif a deux fichiers comportant une solution proche de l'optimum situé dans le dossier **solutions** du projet. L'un pour le HC et le second pour l'ILS.

Ce sont ces différentes solutions que nous allons utiliser à titre d'exemple dans le rapport. Il est évident que nous ne pouvons pas nous limiter à un exemple pour évaluer la performance d'une fonction objectif. C'est pour cela que le dossiers **scores** a été créé et comprend plusieurs scores trouvés pour chaque fonction objectif. Même si nous nous appuyons sur un seul rendu de l'album (dossier **solutions**), le commentaire associé correspondra à l'ensemble des résultats obtenus (dossier **scores**).

5.2. Hill Climber First Improvement

5.2.1. Critère de qualité avec les distances associées à “Average hash”

De manière générale la solution obtenue comporte des photos ayant des similitudes tout comme des différences. Aucune page ne comporte vraiment des photos avec beaucoup de points communs.



Comme nous le montre les pages de l’exemple les photos ont très peu objectivité entre-elles. La notion de distance de l’average de l’empreinte n’est peut-être pas la meilleure donnée à exploiter pour la conception de l’album.

5.2.2. Critère de qualité avec les distances associées à “Perspective hash”

Par rapport à la donnée de distance précédente, celle-ci comporte des pages ayant des photos plus cohérentes entre elles. Même si toute ne le sont pas, un meilleur résultat est toute de même rendu.

La page 2 notamment comprend des photos avec de la verdure et dont 3 des photos comporte des mégalithes. De même pour la page 4 où l’on peut remarquer que la majorité des images comporte des étendues d’eau.

5.2.3. Critère de qualité avec les distances associées à “Distance hash”

De la même manière que l’average hash, cette notion de distance comporte à la fois des pages avec des photos aux critères communs, ainsi que des pages avec des photos opposées.

On peut imaginer que ce type donnée distance n’est pas adapté à notre problème vu certaines incohérences.

5.2.4. Critère de qualité avec les distances associées aux tags commun

Contrairement aux autres, la notion de distance via les tags communs entre les photos apporte des pages vraiment cohérentes.

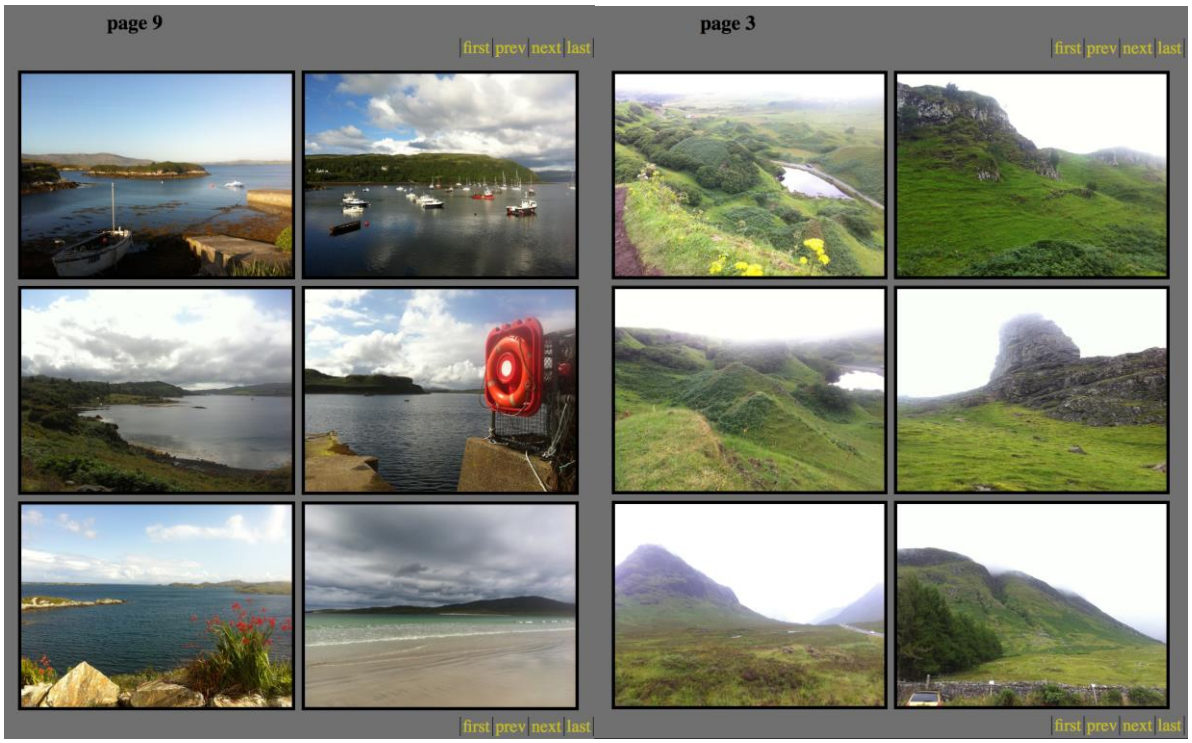
Ce rendu visuel nous montre que ce critère pour être intéressant à traiter dans le cadre de notre objectif.

5.2.5. Critère de qualité avec les distances associées aux tags non commun

Par rapport au précédent critère le visuel apporté est un peu moins bon. Il reste toutefois intéressant, et de mon point de vu meilleur que ceux associés à l'empreinte.

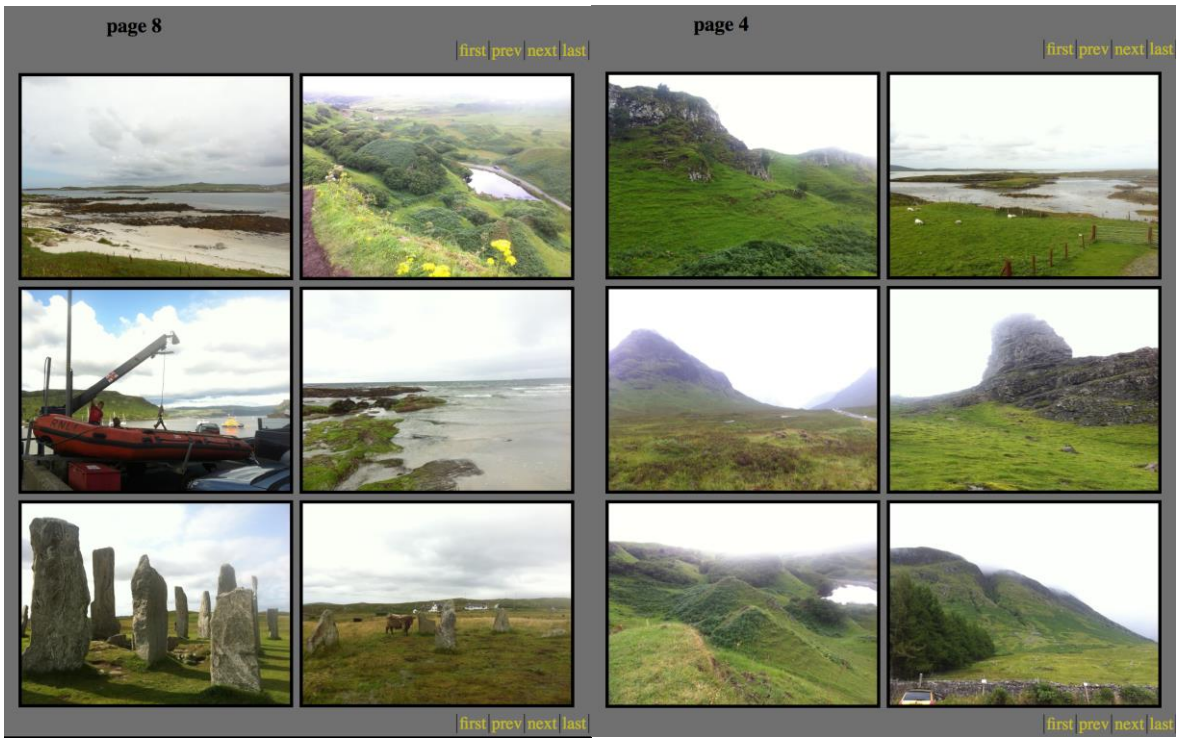
5.2.6. Distances associées aux nombres de tags non commun

Comme les deux précédents, la génération des pages a amené un visuel agréable. Une certaine logique d'ergonomie est mise en place au sein de chaque page.



5.2.7. Critère de qualité avec les distances associées aux couleurs des photos

La distance associée aux couleurs des photos donne un visuel agréable avec quelques petites incohérences toutefois. On remarque une vraiment ressemble en termes de teintes de couleurs pour ces photos.



5.2.8. Critère de qualité avec les distances associées à la moyenne de couleur grise

Ce critère est quant à lui un peu moins bon que le précédent, appuyé également sur une notion de couleur, la moyenne de gris implique quelques différences des photos triées. Notamment la première photo en à gauche de la page 8, qui comprend de gros contraste avec une certaine luminosité et l'ombre des mégalithes. La moyenne d'une couleur n'est peut-être un très bon critère de résultat.

5.2.9. Conclusion Hill Climber First Improvement

Globalement, le Hill Climber First Improvement nous permet d'obtenir un optimum local sans possibilité de rebondir. Etant donné que nous sommes sur une recherche aléatoire il sera très difficile de tomber sur l'optimum global. La génération de pages comportant des photos avec peu de cohérence entre-elles pourrait être expliqué par ce fait. Toutefois comme évoqué précédemment certains types de données utilisées dans le problème QAP ne sont peut-être pas les plus pertinentes. Cette notion qui pourrait également donner une explication à ce manque de résultat.

Certaines données se distingue déjà un peu en termes de qualité de rendu. Notamment les tags associés aux photos qui ont pour le moment donné un bon résultat. Reste à comparer avec les résultats de l'ILS qui devraient peut-être permettre une meilleure cohérence pour certains critères et une amélioration pour d'autres.

5.3. Iterated Local Search

L'Iterated Local Search permettra dans notre cas de confirmer nos remarques ou nos doutes sur une solution.

5.3.1. Critère de qualité avec les distances associées à "Average hash"

Les incohérences obtenues lors du passage de cette distance au sein d'HC sont moins présente. On remarque de façon générale des couleurs et paysages cohérents entre les photos malgré certaines qui ne le sont pas.

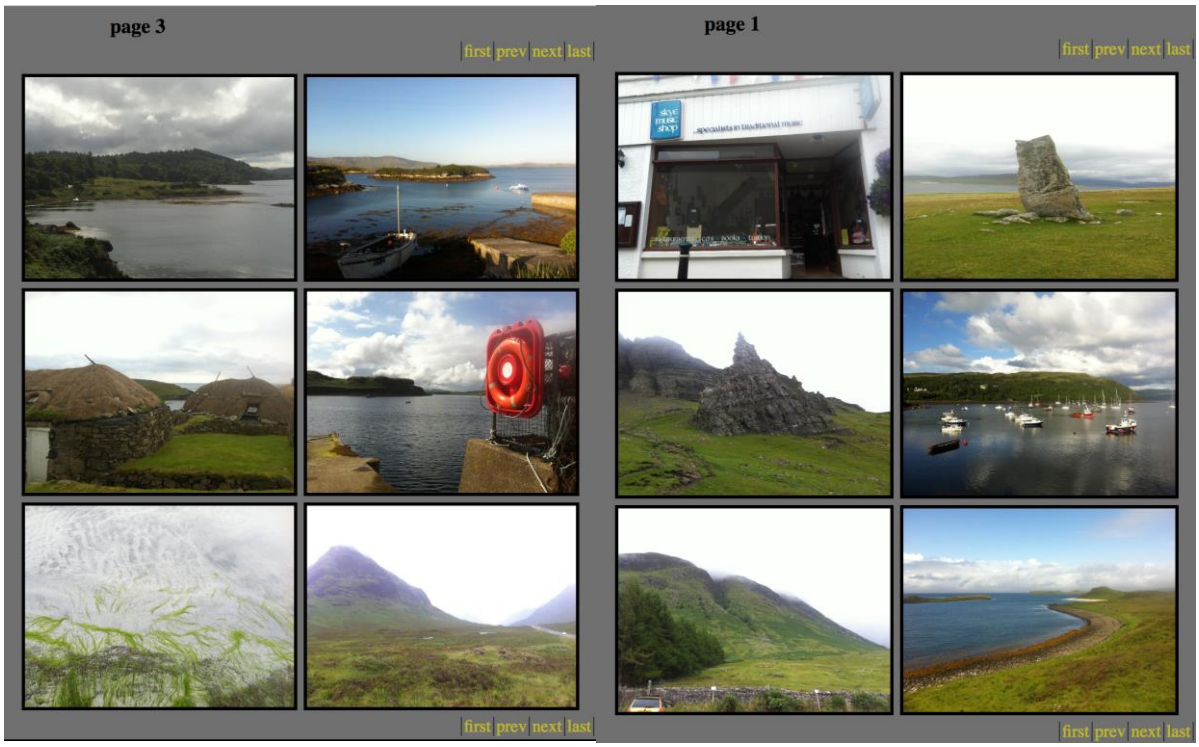
Cette valeur n'est peut-être pas le bon critère pour la génération de l'album du moins pour une solution avec un résultat vraiment flagrant.

5.3.2. Critère de qualité avec les distances associées à “Perspective hash”

Cette valeur qui paraissait avoir un résultat plus pertinent que l’average hash ne l’est pas une fois utilisée au sein de l’ILS. Beaucoup trop de différences sont présentes entre les photos pour qu’elle soit une valeur utilisable par la suite.

5.3.3. Critère de qualité avec les distances associées à “Distance hash”

De même que pour les autres valeurs liées à l’empreinte, les pages contiennent également des photos sans trop de similitudes. De manière générale les valeurs liées aux empreintes ne sont peut-être pas adaptés pour le rendu visuel de l’album de façon automatisé.



5.3.4. Critère de qualité avec les distances associées aux tags commun

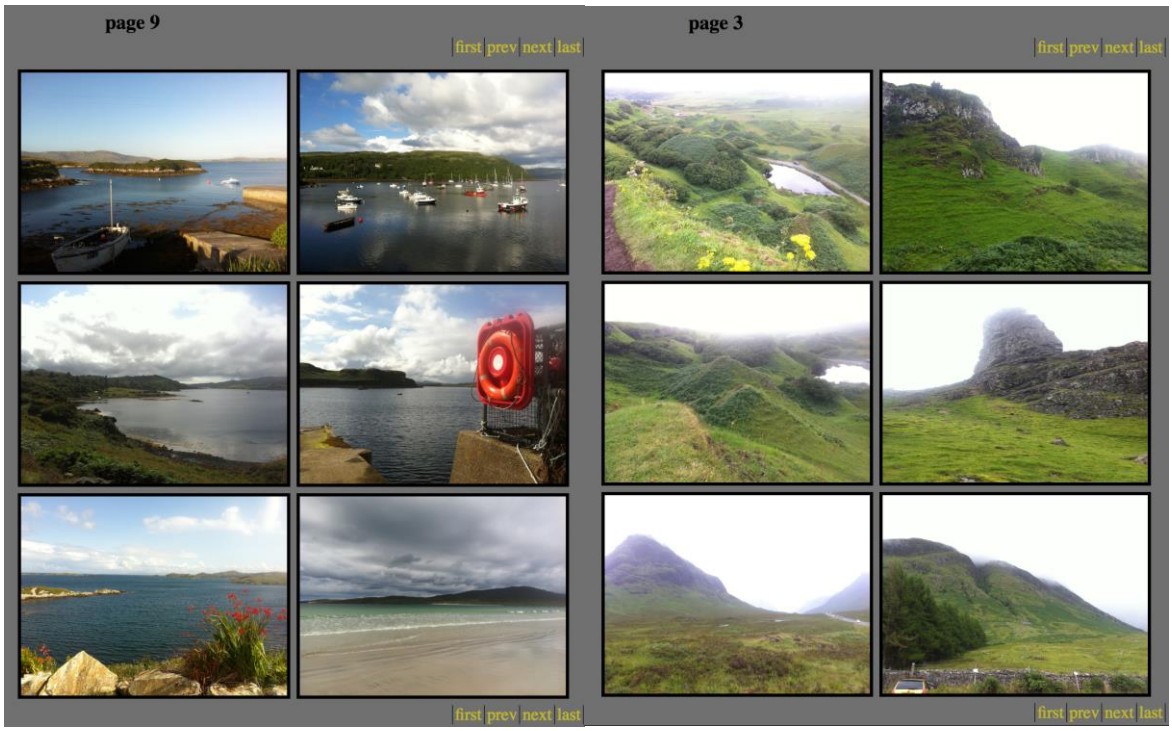
Comme pour l'HC, la solution obtenu est cohérente ce qui peut nous indiquer qu'en plus de générer une solution proche d'un rendu demandé, il peut être générée assez rapidement.

5.3.5. Critère de qualité avec les distances associées aux tags non commun

La remarque peut être identique que pour la distance liée aux tags communs. Le visuel est lui aussi agréable et cohérent.

5.3.6. Critère de qualité avec les distances associées aux nombre tags non commun

Cette distance moins portée sur les valeurs associée aux tags permet également d'affirmer sa performance en termes de résultat. Les tags sont généralement un très bon indicateur pour la génération de l'album.



5.3.7. Critère de qualité avec les distances associées aux couleurs des photos

Avec l'ILS, les couleurs apportent elles aussi un bon rendu. Quelques petites différenciations des photos sont remarquables mais dans l'ensemble le résultat semble satisfaisant.

5.3.8. Critère de qualité avec les distances associées à la moyenne de couleur grise

Pour la valeur associée à la moyenne de gris des photos, les résultats ne semblent pas trop meilleurs qu'avec l'HC. Se baser sur une couleur n'est peut-être pas la meilleure façon de faire compte rendu du résultat précédent.

5.3.9. Conclusion de l'Iterated Local Search

Globalement l'ILS apporte vraiment des réponses plus claires sur les résultats des différentes valeurs de distances utilisées. Le fait de chercher plusieurs optimaux locaux et de garder le meilleur apporte un meilleur résultat.

Dans la plupart des valeurs distances, il a été remarqué que certains scores obtenus étaient redondant après plusieurs essaies. On peut imaginer que l'optimum global correspond à cette valeur et qu'en fonction du résultat on peut affirmer ou non si une fonction objectif est vraiment cohérente.

Ce cas a été remarqué pour la valeur Average hash qui après être utilisé par l'ILS et l'EA, le meilleur résultat obtenu était 10.84. Etant donné le rendu visuel trop aléatoire de l'Average hash on peut laisser de côté ce critère et continuer nos recherches sur les autres.

6. Conclusion et améliorations à envisager

Les distances générées par les tags et les couleurs se démarques des autres et offre un résultat qui semble correcte. Les similitudes entre les photos sont bien présentes sur les pages, seules quelques incohérences apparaissent sur certaines.

Il aurait été intéressant d'utiliser un autre jeu de photo afin de comparer à nouveau les résultats sur ces critères et confirmer ou non leur vraisemblable performance. Le fait de mettre en place des pénalités pour les tags non commun ou commun en fonction du critère, a sûrement permis un meilleur rendu. Il pourrait être envisager de continuer à chercher la pénalité qui amènerait le meilleur rendu possible. Malheureusement il est également possible qu'il ne soit cohérent que pour un jeu de photos particuliers.

Il est également envisageable d'essayer d'autres algorithmes à solution unique qui amènent plus rapidement à la solution globale ou très proche de celle-ci tels que le recuit simulé ou encore la recherche taboue.

Une autre alternative serait une fois certains paramètres de qualités connus pour leurs résultats, de passer par un problème de satisfaction de contraintes voir de satisfaction de contraintes pondérées (SCP / WSCP). Basés sur la programmation par contrainte, il permettrait via un solver déjà présent ou développé d'apporter une solution souhaitée. Il faut que le problème soit bien modélisé et les contraintes connues.

Je tiens à dire que le projet a été très intéressant, tant bien par sa confection pour son adaptation avec le langage Scala que par l'analyse qui en découlé. A la suite de ses analyses plusieurs autres idées de critères me sont venus à l'esprit.

Jouer davantage avec les pénalités pour les tags. On pourrait amener plus de tags associés aux photos et y ajouter des contraintes de pénalité plus fortes afin d'avoir toujours des résultats cohérents. Il est vrai que si trop de tags sont présents il serait plus dur de déterminer les photos proches et les résultats seraient eux moins bon.

A cela s'ajoute la notion des couleurs, étant l'un des meilleurs rendus avec les tags, on pourrait envisager si cela n'est pas déjà existant de ressortir le taux de présence des couleurs primaires d'une photo et de l'associer sous forme de distances au problème QAP. Ou bien une API qui permettrait de trouver les couleurs les plus communes aux photos ce qui nous permettrait de les traiter de la même façon que celle imaginé précédemment.