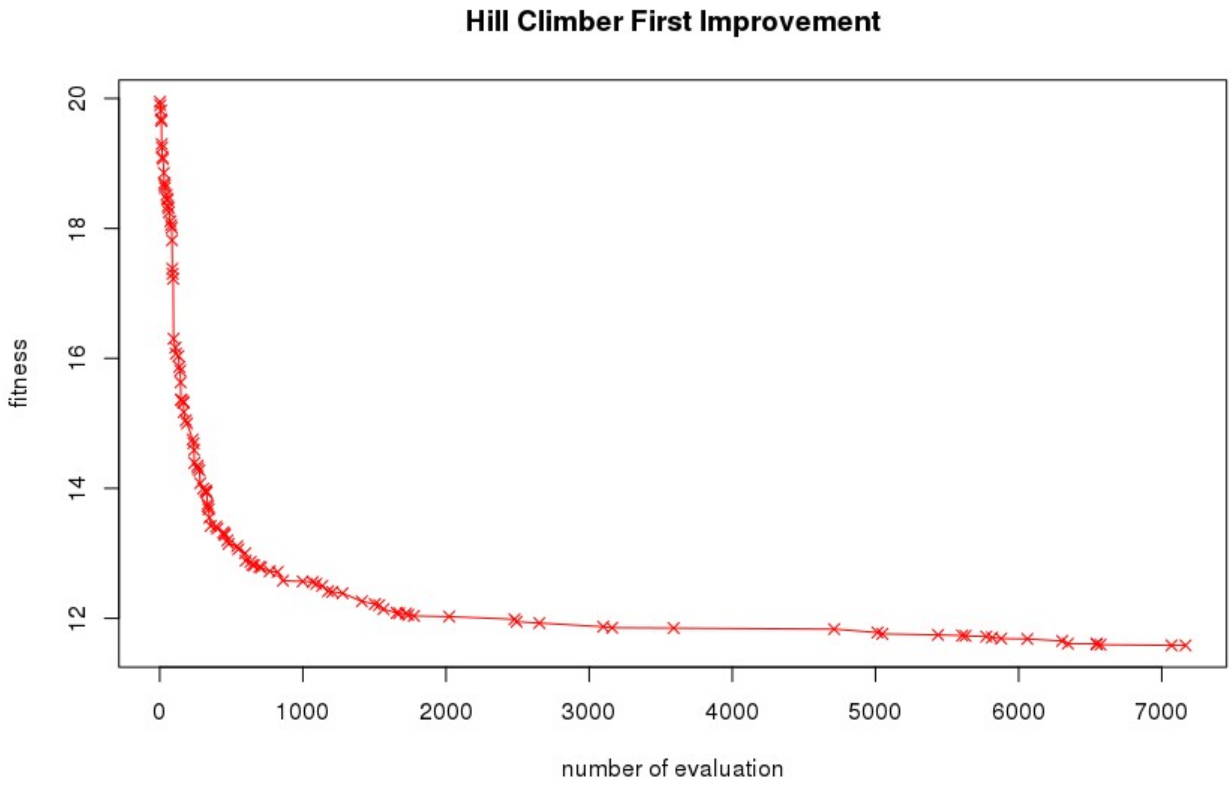
I. LES ALGORITHMES

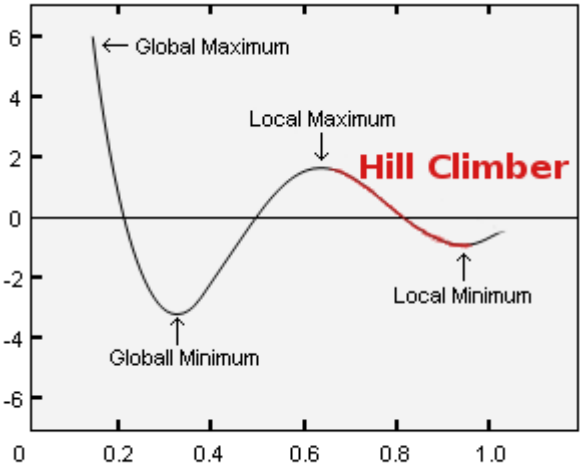
Les résultats présentés ont été générés depuis la fonction d'évaluation montrée comme exemple que je nomme fonction ahash. La distance de hamming entre deux voisins est 1, la taille du voisinage est de (54*55)/2. Une solution a donc 1485 voisins.

1. Hill Climber First Improvement

La méta-heuristique Hill Climber First Improvement permet d'avoir de bons résultats rapidement comme le montre ce graphique généré avec Rstudio depuis les données générées avec le mode debug de ma méthode HillClimber.



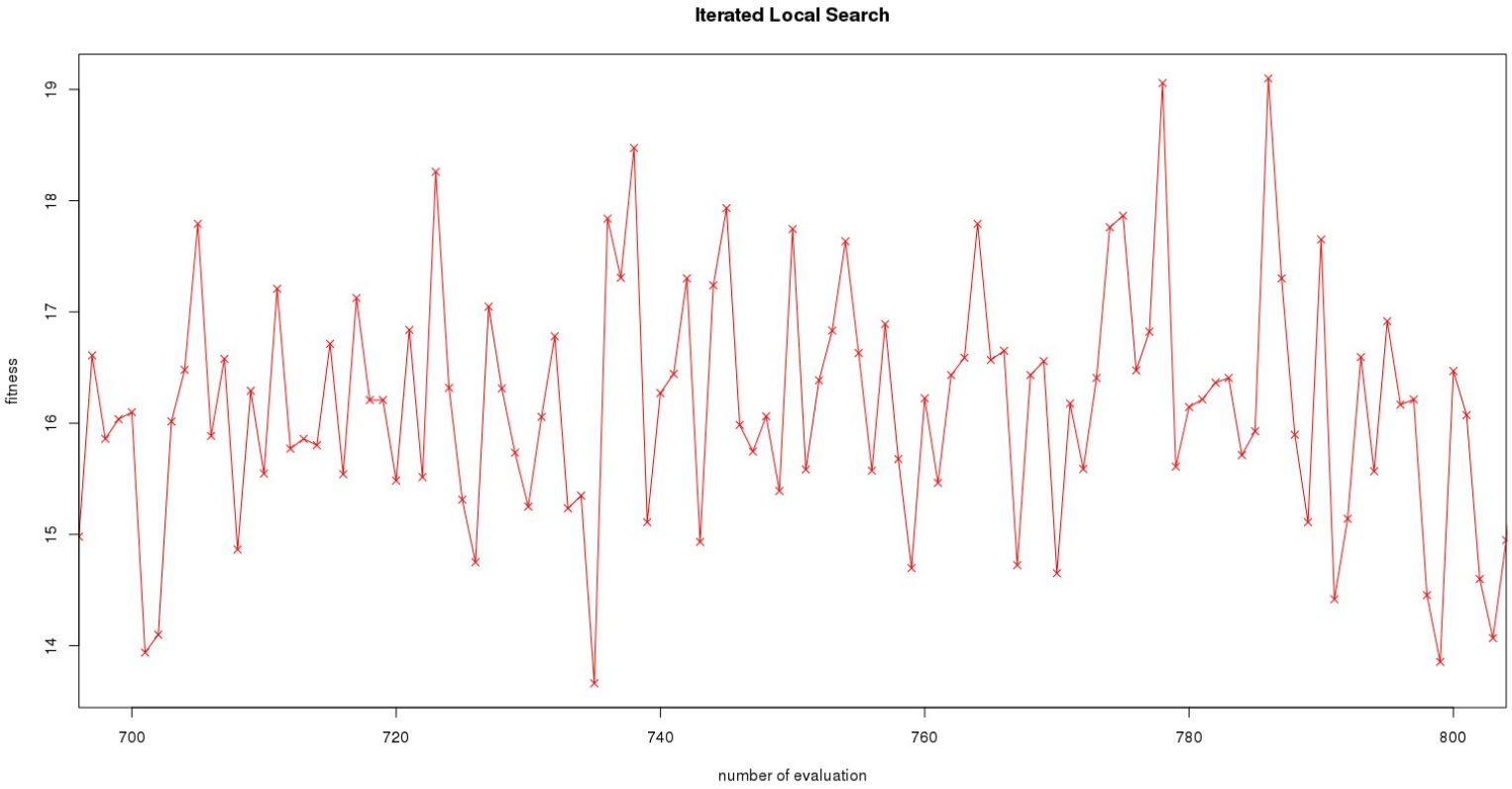
L'inconvénient de cet algorithme est qu'il se dirige et stagne vers des minima. On peut imager cet inconvénient sur le graphique ci-contre. Les résultats du Hill Climber sont représentés par la partie rouge.



Pour avoir de meilleurs résultats il faut pouvoir contrer cette stagnation pour repartir vers un nouveau minimum local qui sera peut-être mieux.

2. Iterated Local Search

L'algorithme ILS ajoute une perturbation à la solution trouvé par l'algorithme de recherche local (ici un hill climber first improvement), pour remonter sur la courbe et chercher de nouveaux minima locaux comme le montre ce graphique généré avec Rstudio depuis les données de l'algorithme ILS en mode debug.



Cet algorithme est plus long car il exécute une succession de HC. Après plusieurs tests de paramétrage, les meilleurs résultats (10.843) ont été trouvés après 2000 répétitions en moyenne ce qui dure environ 3 minutes sur mon ordinateur.

II. LES FONCTIONS D'ÉVALUATION

Les fonctions d'évaluation que j'ai développé ont été inspirées de la fonction ahash (la fonction de démonstration) qui multiplie la distance ahash entre deux images avec la distance physique des images sur l'album.

1. Ahash & Tags

Présentation

Cette fonction d'évaluation est basée sur la fonction de démonstration mais ajoute un effet d'harmonie par thèmes (nature, mer, …).

Pour ce faire on commence par récupérer les meilleurs tags des photos (ceux avec une probabilité supérieure à 97%, ce qui permet d'avoir au moins un tag par image).

Il faut ensuite traduire ces données sous forme numérique pour les intégrer au calcul de notre fonction d'évaluation. Pour ce faire on calcule un coefficient qui sera déterminé en fonction du nombre de tags commun entre deux photos. Le coefficient sera bon (inférieur à 1) quand plusieurs tags seront en commun et inversement il sera mauvais (égale à 2) quand aucun tag ne sera en commun.

Il suffit ensuite d'intégrer ce coefficient à la fonction d'évaluation.

Performance

Algorithme	Nombre d'éxécution de l'algorithme	Nombre d'itérations maximum	Temps total	Meilleur résultat
HillClimber FI	40	10000 (jamais atteint)	126 secondes	1.99
ILS	4	150	43 minutes	1.83

On remarque que le temps d'exécution du HillClimber First Improvement est de 3 secondes pour une exécution, ce qui est beaucoup plus que pour la fonction de démonstration qui était pratiquement instantanée sur mon ordinateur. Ceci impacte la durée de l'algorithme ILS qui exécute une succession de hill climber. Le temps moyen pour une exécution de 150 itérations est environ 10 minutes alors que pour la fonction de démonstration on effectue 2000 itérations en un peu plus de 3 minutes.

Le temps d'éxecution assez élevé vient du calcul du coefficient qui calcule les différences entre deux photos selon les tags des photos. Ce calcul utilise des objets ArrayList de chaîne de caractères, beaucoup plus consommateur en ram et en cpu que les tableaux primitifs.

2. Grey Average

Présentation

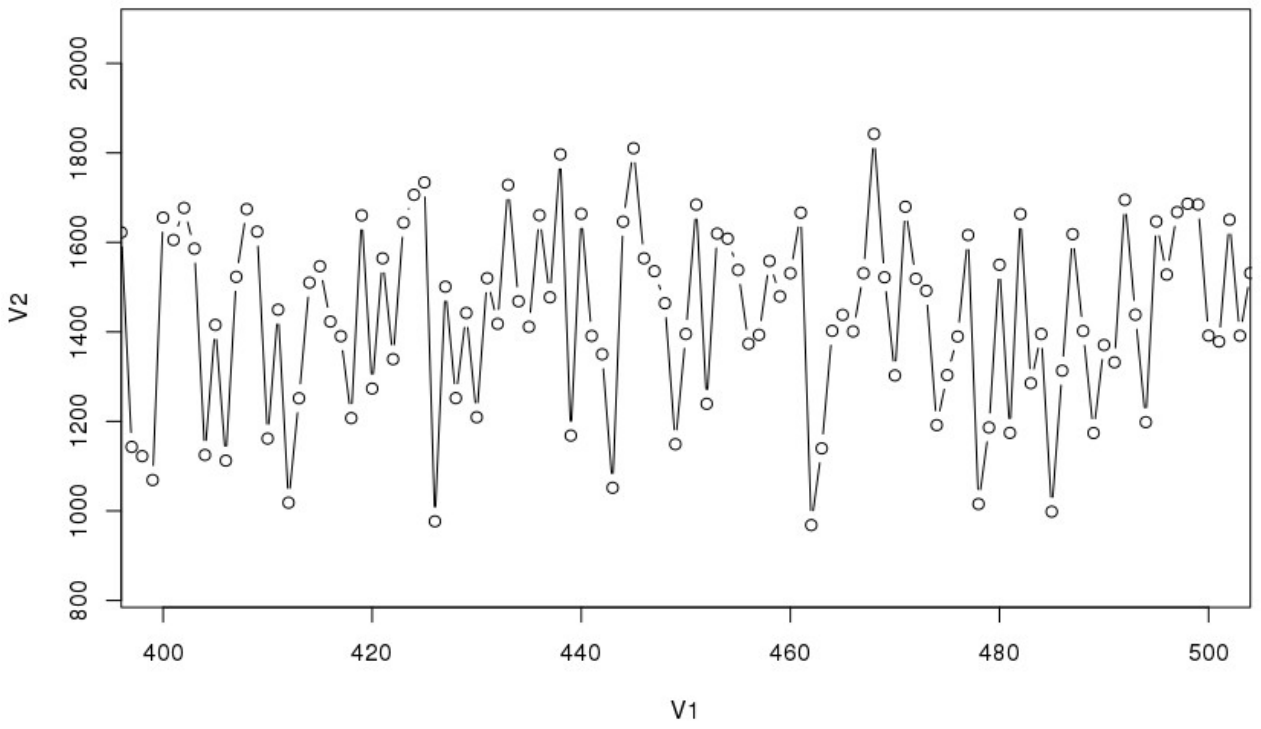
Cette fonction d'évaluation utilise la caractéristique greyavg qui est le niveau de gris moyen lorqu'elle est convertie en noir et blanc. Le niveau de gris d'une photo sert à identifier le niveau de luminosité de la photo. On va donc avec cette fonction d'évaluation trier les photos pour que sur chaque page, le niveau de luminosité moyen soit globalement le même.

Pour ce faire on commence par récupérer le niveau de gris moyen pour chaque photo. La fonction d'évaluation va multiplier la distance physique de deux photos sur l'album avec la distance (l'écart) de niveau de gris de ces deux photos.

Performance

Algorithme	Nombre d'exécution de l'algorithme	Nombre d'itérations maximum	Temps total	Meilleur résultat
HillClimber FI	40	10000 (jamais atteint)	4 secondes	304.33
ILS	5	2000	8 minutes	304.33

On remarque que l'algorithme ILS ne fait pas mieux que le Hill Climber First Improvement sur 2000 itérations (et même sur 10000). On peut penser qu'il s'agit d'un bug de l'algorithme ILS et qu'il doit stagner quelque part mais en regardant le graphique du mode debug ci dessous on remarque que tout semble normal. On peut donc conjecturer que soit les algorithmes atteignent le minimum global, soit que l'ILS a besoin de beaucoup plus d'itérations pour faire mieux que le Hill Climber.



ILS sur la fonction grey-average entre 400 et 500 itérations avec RStudio

III. RÉSULTATS VISUELS

1. Grey Average

Cette fonction a pour but de rassembler sur une même page les photos ayant un niveau de luminosité semblable grâce au niveau de gris. Les résultats sont plutôt concluant les bandes rouges représentent l'écart de niveau de gris entre les deux images :

On voit bien que sur la page de gauche toutes les photos ont une très bonne luminosité alors que sur la page de droite la luminosité est moins bonne, les photos sont plus foncées. Les photos sont placées avec l'objectif que l'écart de niveau de gris entre deux photos soit le plus petit possible (c'est une fonction à minimiser).

On voit un intrus sur la dernière photo de la deuxième page, la photo est prise dans un endroit lumineux (le contour du magasin est bien visible) mais l'intérieur du magasin est noir à cause d'un manque de luminosité dans le magasin, le niveau de gris moyen est donc sombre car l'intérieur du magasin prend une grande place sur la photo.

2. Ahash & Tags

Cette fonction utilise la fonction de démonstration qui rassemble sur une même page les photos ayant un ahash semblable. J'ai ajouté à cette fonction une amélioration pour que sur une même page se retrouve les photos ayant le plus de tags en commun.

On remarque que les résultats sont concluant, les images ayant un thème similaire sont regroupés. On remarque néanmoins que des intrus peuvent apparaître sur les pages. Ces intrus peuvent venir du fait que les photos ont quand même plus de tags en commun que les autres photos (l'intrus est isolé des autres photos). J'ai essayé de combler ce problème en utilisant un système de coefficient qui minimise d'autant plus le résultat qu'il y a de tags en commun entre les deux photos.

IV. ANNEXES

Graphiques générés avec Rstudio : https://www.rstudio.com/

Code source et documentation : https://github.com/geoffreyp/OptimisationAlbumPhoto

Environnements de développement : https://eclipse.org/ http://www.vim.org/