Conception automatique d'album photo par une technique d'optimisation

Master 1 informatique, année 2016 / 2017

1 Description et but

Beaucoup d'outils d'aide à la conception d'album photo existent. Ces outils proposent principalement quelques patrons génériques d'album et des outils d'amélioration de ces patrons : choix des couleurs, dimension des photos, dimension et position des emplacements des photos, etc. Seulement, ces applications ne proposent pas d'outils intelligents et automatiques de répartition des photos sur l'album hormis quelques répartitions basiques comme l'ordre chronologique.

Il s'agit dans ce projet d'automatiser la phase de disposition des photos sur un album à l'aide de techniques d'optimisation. Tout d'abord, il s'agit de modéliser ce problème de conception en un problème d'optimisation combinatoire, puis d'utiliser des algorithmes d'optimisation efficaces pour trouver les meilleures solutions à ce problème.

2 Modélisation du problème

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'interpréte comme une distance, ou de maximiser si le score s'interprète comme la qualité de l'album.

La fonction score doit être construite sur les informations relatives aux photos et à l'album. Les informations de l'album sont contenues dans le fichier au format json `data/info-album.json`. Les informations données sont :

- page : nombre de page de l'album,
- pagesize : liste du nombre de photo sur chaque page,
- basename : le nom de base des pages web de l'album (`page_0.html`, etc.)
- pages : indique pour chaque page, les dimensions de celle-ci, puis la position et les dimensions de chaque photo sur la page

Globalement, la composition de l'album proposé est simple. Il est composé de 9 pages contenant chacune 6 photos disposées en grille (voir figure 1).

Les informations sur chacune des photos sont contenus dans le fichier au format json `data/info-photo.json`. Les informations données sont :

- index : index (commençant à 0) utilisé pour identifier la photo lors de la construction de l'album,
- name : nom du fichier contenant la photo dans le dossier `html/img`,
- id : index défini par l'appareil photo. Il donne aussi l'ordre chronologique des prises de vue,
- size : dimensions en pixel de la photo originale,
- date : date de la prise vue (année, mois, jour, heure, minute et seconde),

FIGURE 1 – Exemple d'une page de l'album.

— color1 : en réduisant la palette des couleurs à 16 couleurs, couleur la plus fréquente de la photo,
— color2 : en réduisant la palette des couleurs à 16 couleurs, deuxième couleur la plus fréquente de la photo,
— greyavg : niveau de gris moyen de la photo lorsqu'elle est convertie en noir et blanc,
— note : note entre 0 et 100 attribuée par des personnes (0 est la note la plus basse et 100 la plus haute),
— ahash, phash et dhash : empreintes (hash) calculées sur les photos. Les empreintes sont des chaines binaires de 64 bits écrites en hexadécimal. A partir des caractéristiques de la photo, une fonction de hachage calcule l'empreinte. Lorsque les empreintes sont identiques, les photos sont proches voire identiques elles aussi. 3 types de hachage sont calculées [1] : average, perceptive [2] [3], et difference hash [4].
— ahashdistance, phashdistance et dhashdistance : distances de Hamming (normalisées entre 0 et 1) entre les photos basées sur les empreintes précédentes qui mesurent de différentes manières la similarité des images. Lorsqu'une distance est nulle, les photos sont censées être identiques modulo quelques transformations ou détails. La liste des distances aux autres photos est donnée dans l'ordre des index.
— tags : mots tag associés à l'image. En utilisant les outils de la société clarifai [5], les tags sont calculés automatiquement à partir d'algorithmes d'apprentissage de type réseau de neurones profond. Le champs `classes` donne la liste des mots tag et le champ `probs` donne la liste des probabilités que chaque mot soit associé à l'image.

Les 55 photos au format jpeg sont dans le dossier `html/img`. A noter qu'il existe une photo supplémentaire

1. Le code python utilisé est celui de Jens Segers `https://github.com/jenssegers/imagehash`
2. Zauner, Christoph : Implementation and Benchmarking of Perceptual Image Hash Functions. Master's thesis, Upper Austria University of Applied Sciences, Hagenberg Campus, 2010.
3. Voir aussi le billet du blog du Dr. Neal Krawetz `http://www.hackerfactor.com/blog/?/archives/432-Looks-Like-It.html`
4. Issu également d'un billet du blog du Dr. Neal Krawetz `http://www.hackerfactor.com/blog/index.php?/archives/529-Kind-of-Like-That.html`
5. `http://www.clarifai.com/`

par rapport aux 54 emplacements disponibles sur l'album. Les codes java et c++ montrent comment lire et utiliser ces données au format json.

De nombreuses fonctions score peuvent être imaginée. Globalement, leur but est de faire en sorte que les photos sur le même thème, les plus similaires, etc. soient les plus proches sur l'album (même page et/ou proximité sur la même page). De plus, on pourra introduire une notion de qualité de la page (harmonie, cohérence, etc.) ou de l'album dans son ensemble. Imaginons par exemple que l'on définisse à partir des données photo une distance abstraite d_p entre les photos et que la fonction d_a donne la distance entre 2 emplacements de photo sur l'album. Par ailleurs, imaginons que l'on arrive à définir la fonction q mesurant la qualité d'une page. La fonction score pourra être définie comme $f_\alpha(\sigma) = \alpha \sum_i \sum_j d_p(i,j) \frac{1}{d_a(\sigma(i),\sigma(j))} + (1-\alpha) \sum_k \frac{1}{q(\sigma_k)}$ où $\sigma(i)$ et $\sigma(j)$ sont les emplacements des photos i et j sur l'album, et où σ_k est la liste des photos sur la page k. Les "bons" albums seront données par les minimums de f_α. Les codes java et c++ montrent une exemple très simple de fonction score.

3 Construction de l'album

L'album photo se présente sous forme de pages web : chaque page de l'album est une page web. Le code python `code/buildAlbum.py` permet d'éditer cet album par la commande :
`python code/buildAlbum.py my.sol`
où `my.sol` est un fichier indiquant l'emplacement de chaque photo. Les emplacements sur l'album sont représentés par des entiers entre 0 et 53. Le fichier `my.sol` doit être constitué de 55 nombres entiers, le premier entier indique l'emplacement de la première photo et ainsi de suite. La photo indiquée par le dernier entier du fichier n'est pas sur l'album. `data/chronologic-order.sol` est un exemple d'un tel fichier. Par défaut, l'album est construit dans le dossier `html` (voir code source python pour le changer).

4 Travail Demandé

Les documents à rendre sont :
— un court document écrit (au format pdf) répondant aux questions suivantes,
— l'ensemble du code produit sous forme d'une archive (au format zip ou tar.gz) avec votre nom comme verel.tar.gz
Une courte soutenance orale (entre 5 et 10 mins) sera organisée pour présenter vos projets.

Questions :
a - Définir plusieurs fonctions score en expliquant leur construction. Par exemple, vous pouvez même créer des jeux de photos artificiels montrant les propriétés de votre fonction score et qui pourront servir dans l'analyse des performances des algorithmes.
b - Donner la taille de l'espace de recherche et la taille du voisinage pour chaque opérateur que vous aurez choisi.
c - Coder la métaheuristique hill-climbing first-improvement (HC).
d - Analyser les performances de la métaheuristique HC ainsi que les dynamiques de recherche.
e - Coder une recherche "Iterated Local Search" (ILS).
f - Analyser les performances de la métaheuristique ILS ainsi que les dynamiques de recherche.
g - Analyser les résultats obtenus en terme de qualité visuelle de l'album pour chaque fonction score proposée.
h - Et si vous voulez (et que vous avez le temps !), apportez des améliorations à vos algorithmes.