Projet "Binarisation d'images"
Programmation en JAVA 1.7 et en binôme (obligatoirement)

1 Objectifs

En traitement automatique d'images, il est essentiel de savoir identifier les diverses parties d'une image. A priori simple (en tout cas pour l'oeil humain), le problème d'identifier le premier plan d'une image, et son arrière plan, est un problème pas si simple. Le but de ce projet est de définir une méthode pour séparer ces deux plans, basée sur la théorie des flots maximaux.

2 Le problème

L'image est formée de pixels (carrés) rangés en n lignes et m colonnes. Chaque pixel (i, j) a une probabilité $a_{ij} > 0$ pour qu'il appartienne au premier plan et une probabilité $b_{ij} > 0$ pour qu'il appartienne au second plan [1]. Deux pixels (i, j) et (k, l) sont dits *voisins* s'ils partagent un de leur côtés : ils sont voisins horizontalement ssi $k = i$ et $l = j \pm 1$, et ils sont voisins verticalement ssi $l = j$ et $k = i \pm 1$. Pour toute paire de pixels (i, j) et (k, l) qui sont voisins une pénalité $p_{ijkl} \geq 0$ existe lorsqu'on les sépare (l'un est mis dans le premier plan et l'autre dans l'arrière plan). Noter que la fonction pénalité est symmétrique, c'est-à-dire $p_{ijkl} = p_{klij}$.

Les valeurs n, m, a_{ij}, b_{ij} et p_{ijkl} $(1 \leq i, k \leq n, 1 \leq j, l \leq m)$ sont fournies.

Le problème BINARISATION D'IMAGE (abrégé BINIM) consiste à trouver deux ensembles de pixels A et B qui partitionnent l'image en maximisant la fonction :

$$q(A, B) = \sum_{(i,j)\in A} a_{ij} + \sum_{(k,l)\in B} b_{kl} - \sum_{(i,j)\in A,(k,l)\in B \text{ voisins}} p_{ijkl} \qquad (1)$$

3 Vers une solution utilisant flots et coupes

Q1. En notant $Q = \sum_{i=1}^{n} \sum_{j=1}^{m} (a_{ij} + b_{ij})$ et en le faisant intervenir dans les calculs, montrer que maximiser $q(A, B)$ équivaut à minimiser :

$$q'(A, B) = \sum_{(i,j)\in A} b_{ij} + \sum_{(k,l)\in B} a_{kl} + \sum_{(i,j)\in A,(k,l)\in B \text{ voisins}} p_{ijkl} \qquad (2)$$

.

Q2. Construire un réseau de transport R (graphe orienté) représentant les pixels et leurs voisinages. Ajouter d'autres arcs selon les besoins et définir des capacités sur les arcs de sorte que toute coupe dans le réseau ait une capacité donnée par une formule de type (2).

Q3. Justifier l'affirmation : le problème de binarisation d'image a une solution (A, B) si et seulement dans le réseau R la paire $(A \cup \{s\}, B \cup \{t\})$ est une coupe de capacité minimum.

Q4. Ecrire un programme pour résoudre le problème BINIM utilisant l'approche proposée. Pour calculer la coupe de capacité minimum, utiliser le théorème MaxFlow-MinCut et implémenter un algorithme de flot maximum. Attention, la coupe de capacité minimum est facilement calculable si vous avez

1. Ne pas en déduire que $a_{ij} + b_{ij} = 1$. Ces scores sont plutôt des scores de vraisemblance, calculés selon des critères divers, et que l'on appelle des probabilités pour simplifier.

implémenté l'algorithme de Ford-Fulkerson (voir cours), mais l'algorithme de préflots est plus simple à implémenter.

Le programme implémentera, parmi d'autres méthodes, les méthodes suivantes :

1. ConstructionReseau qui construit le réseau de transport R adapté au problème BINIM.
2. CalculFlotMax qui calcule le flot maximum pour un réseau de transport donné.
3. CalculCoupeMin qui calcule la coupe de capacité minimum pour un réseau de transport donné.
4. RésoudreBinIm qui résout le problème BINIM.

Le programme doit prendre en entrée un fichier texte (.txt) avec un nom arbitraire, et de la forme suivante :

$n\ m$

$a_{11}\ a_{12}\ \dots\ a_{1m}$
$a_{21}\ a_{22}\ \dots\ a_{2m}$
$\dots$
$a_{n1}\ a_{n2}\ \dots\ a_{nm}$

$b_{11}\ b_{12}\ \dots\ b_{1m}$
$b_{21}\ b_{22}\ \dots\ b_{2m}$
$\dots$
$b_{n1}\ b_{n2}\ \dots\ b_{nm}$

$p_{1112}\ p_{1213}\ p_{1314}\ \dots\ p_{1(m-2)1(m-1)}\ p_{1(m-1)1m}$
$p_{2122}\ p_{2223}\ p_{2324}\ \dots\ p_{2(m-2)2(m-1)}\ p_{2(m-1)2m}$
$\dots$
$p_{n1n2}\ p_{n2n3}\ p_{n3n4}\ \dots\ p_{n(m-2)n(m-1)}\ p_{n(m-1)nm}$

$p_{1121}\ p_{1222}\ p_{1323}\ \dots\ p_{1(m-1)2(m-1)}\ p_{1m2m}$
$p_{2131}\ p_{2232}\ p_{2333}\ \dots\ p_{2(m-1)3(m-1)}\ p_{2m3m}$
$\dots$
$p_{(n-1)1n1}\ p_{(n-1)2n2}\ p_{(n-1)3n3}\ \dots\ p_{(n-1)(m-1)n(m-1)}\ p_{(n-1)mnm}$

Il faut bien respecter les lignes vides et les espaces entre les valeurs. Le programme doit retourner les ensembles A et B de la partition optimale avec un affichage en console de l'image sur n lignes et m colonnes, les pixels de A étant indiqués par un A, et les pixels de B par un B.

4 Rendu de TP

Le programme doit être écrit *en intégralité* par le binôme qui le présente. Il sera lancé en ligne de commande et pourra être utilisé indépendemment d'Eclipse ou autre IDE. Il doit fonctionner parfaitement sur les machines du CIE, sous Linux. Tout programme qui ne compile pas, ne démarre pas, ne récupère pas le fichier sous la forme indiquée, pose des questions que l'utilisateur ne comprend pas etc. ou qui a besoin qu'on entre dans le code pour une raison ou pour une autre sera considéré comme non-évaluable, et votre travail aura été inutile. Assurez-vous que vous fournissez un programme en état de marche et que l'utilisateur a toutes les informations nécessaires pour l'exécuter dès le premier essai.

Un rapport d'au maximum 10 pages sera fourni, comprenant (entre autres) :

— les commandes de compilation et exécution en mode console
— les réponses aux questions Q1, Q2 et Q3

— une vue globale de votre approche, sous la forme d'un algorithme (c'est-à-dire en pseudo-code) dont les instructions sont des appels à des procédures/fonctions ; la spécification et les paramètres de chaque procédure/fonction sont précisément décrits
— pour chaque méthode parmi celles de la liste demandée, sauf celle implémentant l'algorithme de calcul de flot :
  * le détail de la procédure/fonction, sous forme algorithmique (c'est-à-dire en pseudo-code) ;
  * l'explication de son fonctionnement ;
  * sa complexité
— des jeux de données commentés

Important

1. Les fichiers du programme, ainsi que les jeux de données, seront mis dans un répertoire Nom1Nom2 (où Nom1 et Nom2 sont les noms des deux étudiants du binôme). Ce répertoire sera ensuite archivé sous le nom Nom1Nom2.tar.gz ou Nom1Nom2.zip. L'archive sera déposée sur Madoc, au plus tard le **vendredi 6/12/2018 à 23h55** (Madoc n'acceptera pas de retard).
2. La dernière séance est consacrée à une démo de 10 minutes par projet, pour laquelle vous aurez à saisir les données en entrée de la manière décrite ci-dessus (et sans entrer dans le code). Cette dernière séance **n'est donc pas une séance de travail sur le projet**.
3. *Tout* est important pour la notation. En particulier, il sera accordé beaucoup d'attention au respect des consignes et à la recherche d'une complexité minimum - garantie d'une efficacité maximum - pour votre méthode, via la mise en place des structures de données les plus adaptées.