Master de Bio-Informatique
de l'Université de Nantes
2018 – 2019

Advanced algorithmics and programming for biologists /
Models, methods and algorithms for bioinformatics

Sujet de Travaux Pratiques
Encadrement C. Sinoquet

Distribué le jeudi 18 octobre 2018.
Date de contrôle intermédiaire: lundi 19 novembre 2018.
Date limite de remise : lundi 10 décembre 2018 (démonstration en salle sur matériel sur lequel votre TP a été réalisé (portable admis)).
Pénalité de 1 point par jour de retard, ouvré ou non.
Aucun devoir rendu en janvier 2019 ne sera évalué (notation 0).

Sujet à traiter en langage C++ (et R ou Python pour les scripts), sous Linux.
Sauvegardes nécessaires en cours de travail, sur divers supports.
Sauvegardes nécessaires de versions partielles en cours de travail, sur divers supports.
(Aucun délai ne sera accordé si le projet a été perdu, faute de sauvegarde).

**Détection de patterns d'épistasie dans des données génétiques.**

Chaque binôme d'étudiants devra implémenter

- une méthode parmi la liste suivante (méthode attribuée aléatoirement lors du premier TP, de façon que les méthodes soient distribuées équitablement au sein de la promotion) :
    - VNS
    - GRASP
    - algorithme génétique
    - algorithme mémétique
    - EDA
- une méthode parmi la liste suivante (attribution aléatoire lors du premier TP) :
    - la méthode SMMB-ACO avec choix de modalité « one-pass » ou « two-pass »
    - la méthode multi-objectif vue en cours et/ou en TP,
    - la méthode de « path-relinking » vue en cours et/ou en TP.

- le protocole d'évaluation commun à ces deux méthodes (en script R ou Python, commenté en anglais).

Chaque binôme d'étudiants alternants devra implémenter :

- une méthode parmi la liste suivante (attribution aléatoire lors du premier TP) :
    - la méthode SMMB-ACO avec choix de modalité « one-pass » ou « two-pass »

- la méthode multi-objectif vue en cours et/ou en TP,
- la méthode de « path-relinking » vue en cours et/ou en TP.

Ceci, sans protocole d'évaluation avancé comme décrit ci-dessous.

Plusieurs formats d'entrée des fichiers seront considérés :
- format SMMB-ACO issu de simulations :
génotypes : p individus, n SNP
1 ligne d'identifiants de SNP
puis p lignes comportant chacune n valeurs dans {0,1,2}
exemple :
N0,N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15,N16,N17,N18,N19,N20,N21,N22,N23,N24,N41,M0P7,M0P8
1,2,0,1,0,1,0,0,0,0,1,0,1,0,0,1,1,0,1,0,0,1,0,1,0,1,2,1
1,0,0,1,0,0,1,0,1,1,2,1,0,1,0,0,0,1,1,0,1,0,0,0,0,2,1,1
...
1,0,0,1,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,1,1,0
1,1,0,1,0,1,0,1,1,1,1,0,1,1,2,1,0,0,0,1,0,0,0,1,0,1,1,1
1,2,1,1,1,0,0,1,0,0,0,0,0,0,1,1,0,0,1,0,0,0,0,1,0,1,1,0

phénotypes : p états malade / non malade
première ligne : identifiant du phénotype
puis p lignes comportant chacune une valeur dans {0,1} (0 : non malade ; 1 : malade)
exemple :
Class
1
1
...
0
0

Attention : les p lignes du fichier de phénotypes et les p lignes du fichier de génotypes se correspondent une à une.

- format WTCCC (données réelles) (description à venir)

- fichier de paramètres

exemple :
fichier de paramètres pour SMMB-ACO (obligatoire si méthode attribuée):

```
# number of header lines in the genotype file
header 1

# separation between fields in the genotype file
separator ,

# alpha type I error rate
```

alpha 0.05

# number of consecutive runs of SMMB-ACO
n_smmb_aco_runs 1 # 1 : one pass, 2 : two-passes

# number of ACO iterations
n_it_t 100

# number of ants
n_ants 100

# size of the subset of variables sampled by each ant
k_large 10

# size of a combination of variables sampled amongst the k_large above variables, in nested function learnMB (k_small < k_large)
k_small 3

# maximal number of iterations to coerce the exploration of the search space, in nested function learnMB
n_it_n 30

# global type I error threshold
# alpha 0.05

# parameters for ACO optimization

# constant to initiate the pheromone rates
tau_0 100

# rho and lambda: two constants used in evaporation rate updates
# evaporation rate (from 0.01 to 0.1 (see antEpiSeeker))
rho 0.05
lambda 0.1

# modality used to initiate prior knowledge eta
mod_init_prior_knowledge 1 # 1 : eta_0 is used; 2 : a file is used (set alpha and beta appropriately in this case)

# constant to initiate (uniform) prior knowledge for each variable
eta_0 1

# alpha and beta: two constants used to ajust the relative importance between pheromone rate and a priori knowledge (by default, both parameters are initialized to 1).
alpha 1 # to be set appropriately if mod_init_prior_knowledge is set to 2
beta 1 # to be set appropriately if mod_init_prior_knowledge is set to 2

---

Convention code :
n : nombre de SNP
p : nombre d'observations

Vous fournirez un fichier readme.md pour chacune des deux méthodes (installation, librairie, compilation, contenant la référence au répertoire de l'exemple jouet (toy_example) (incluant fichier de paramètres modèle)).
En plus du readme.md, vous fournirez un script Python launch_<method>_toy_example.py d'exécution de <method> sur l' exemple jouet (à inclure dans le sous-répertoire toy_example).

Pour chaque méthode, le fichier de sortie aura le format illustré à l'aide de l'exemple suivant :

{var1, var2, var3} <score>
{var1, var45, var2000, var5000} <score>
…
{var67, var340} <score>

---

Chacune des deux méthodes sera évaluée à l'aide d'un protocole d'évaluation commun, à implémenter (scripting en langage R ou en langage Python).

Trois jeux de données vous seront fournis (100 fichiers simulés sous la même condition pour chaque jeu de données) (simulation « réaliste » par logiciel GAMETES).
Il vous sera demandé de simuler vous-même trois autres jeux de données (simulation naïve).
Votre logiciel de simulation simu_naive.xxx prendra en entrée :
un nom de répertoire de sortie
le nombre de fichiers à générer
le préfixe commun aux identifiants des fichiers
le nombre de variables à simuler
le nombre de cas à simuler
le nombre de contrôles à simuler

Vous fournirez un script Python launch_simu_naive_toy_example.py de simulation modèle sur exemple jouet qui utilisera eval_simu.xxx.

Le script d'évaluation eval_simu.xxx (R ou Python) prendra en entrée :
un nom de répertoire d'entrée
un nom de répertoire de sortie
le nombre d'exécutions de la méthode à réaliser sur chaque fichier du jeu de données simulées (nommé n_runs ci après)
Pour chaque fichier <identifiant_fichier_i.txt> du jeu de données simulées, un fichier <identifiant_fichier_i>_results.txt de n_runs mots dans {TP, FP, FN} sera généré.

Exemple de fichier <identifiant_fichier_i>_results.txt :
TP
FN

TP
FP
…
FN

Pour chaque jeu de données (comportant par exemple n_files fichiers), un fichier f_measures.txt sera généré. Il comportera les n_files f-measures calculées à partir des n_runs fichiers <identifiant_fichier_i>_results.txt générés pour chacun des fichiers <identifiant_fichier_i.txt> ($1 \leq i \leq$ n_files) :

recall = #TP/(#TP + #FN)
precision = #TP/(#TP + #FP).

f-measure =2 / (1/recall +1/precision)

Pour chaque jeu de données (comportant par exemple n_files fichiers), un fichier powers.txt sera généré. Il comportera les n_files f-measures calculées à partir des n_runs fichiers <identifiant_fichier_i>_results.txt générés pour chacun des fichiers <identifiant_fichier_i.txt> ($1 \leq i \leq$ n_files).

power = #TP / n_runs

La notation prendra en compte les points suivants :

- correction des algorithmes,
- lisibilité du code produit,
- modularité du code et des tests,
- preuve de réalisation de tests, production des jeux d'essais et résultats (copies d'écran, fichiers texte – correctement présentés),
- facilité de maintenance du code produit,
- séparation claire des définitions de fonctions par des lignes du type ********** (OBLIGATOIRE),
- présence de commentaires en anglais correct dans les programmes et/ou choix d'identificateurs « parlants »,
- *si nécessaire*, mention des préconditions et postconditions, en commentaire, immédiatement après le prototype de la fonction définie,

• documents à rendre:

- un rapport, correctement relié, d'au plus 4 pages, présentant
  - les structures de données utilisées,
  - un bilan concis récapitulant la réalisation effective par rapport au texte du devoir, et récapitulant ce qui reste à faire (par rapport au texte du devoir), ou au regard des améliorations que vous apporteriez si vous aviez plus de temps,
  - éventuellement, un bilan concis des erreurs que vous n'avez pas réussi à corriger (il vaut mieux le placer ici, plutôt que nous le découvrions nous-mêmes…)

Le rapport devra comporter en annexe les résultats d'exécution de votre algorithme sur plusieurs jeux d'essai significatifs.

- l'édition papier de tous les codes sources, en un document correctement relié et portant explicitement le nom des étudiants concernés.

Lors de l'évaluation, le code source et le rapport seront remis sous format papier, obligatoirement reliés (pas de pochette en plastique). Un document non relié sera noté 0. Un document sans identification sera noté 0.

- une archive exhaustive des codes source <nom1_nom2.tar.gz> sera déposée sur MADOC (ne remplace pas l'édition papier).

• Un rapport redécrivant les fonctionnalités des diverses fonctions est totalement inutile: tout doit être lisible dans le code source.

• Le nombre de trinômes est limité à 1.

• Une importante modulation selon la part réelle effective apportée au travail en binôme sera éventuellement appliquée.

• Les étudiants seront évalués selon les critères détaillés ci-dessus, ainsi que sur la vérification effective, le lundi 10 décembre 2018, que le logiciel élaboré fonctionne totalement ou partiellement.