ΑΣΚΗΣΗ ΓΕΝΕΤΙΚΩΝ ΑΛΓΟΡΙΘΜΩΝ

ΠΑΝΑΓΙΩΤΗΣ ΠΑΠΑΝΙΚΟΛΑΟΥ ΑΜ 1067431

ΚΩΔΙΚΟΠΟΙΗΣΗ

Λαμβάνοντας υπόψη πως πρέπει να μπορεί να αναπαρασταθεί κάθε πιθανό υποσύνολο λέξεων θα αναπαραστήσουμε κάθε άτομο ως bag-of-words, δηλαδή ένα bit αντιστοιχεί στην επιλογή ή όχι μιας λέξης.

ΑΡΧΙΚΟΠΟΙΗΣΗ

Για την αρχικοποίηση του πληθυσμού επιλέγουμε για κάθε άτομο ένα τυχαίο ποσοστό και έπειτα επιλέγουμε τυχαία αυτό το ποσοστό λέξεων από τις 8520. Έτσι ο πληθυσμός έχει μεγαλύτερη ποικιλία στον αριθμό λέξεων και μπορεί πιο εύκολα να βρει λύσεις με αριθμούς λέξεων που απέχουν από το 50%.

ΕΠΙΔΙΟΡΘΩΣΗ

Επειδή οι καλύτερες λύσεις βρίσκονται στο όριο με τις παράνομες, αν ο πληθυσμός φτάσει κοντά σε τέτοια λύση ένα μεγάλο ποσοστό των νέων μελών να μην είναι έγκυρες λύσεις. Αν επιλέξουμε απόρριψη ή αντιστοίχιση είναι πιθανό να χάνουμε πληροφορία από λύσεις πολύ κοντά στη βέλτιστη. Για αυτό το λόγο χρησιμοποιήθηκε εφαρμογή ποινής. Η ποινή που εφαρμόστηκε είναι ανάλογη των λέξεων κάτω από το όριο που έχει το άτομο.

ΣΥΝΑΡΤΗΣΗ ΚΑΤΑΛΗΛΟΤΗΤΑΣ

Η συνάρτηση καταλληλότητας που εφαρμόστηκε είναι το μέσο tf-idf των λέξεων που επιλέγει ως σημαντικές το άτομο κατά μέσο όρο ανά κείμενο με ποινή την ποινή παρανομίας που περιγράφεται παραπάνω αν το άτομο έχει λιγότερες από 1000 λέξεις και μια σταθερά επί των λέξεων που έχει πάνω από 1000 αλλιώς. Η σταθερά επιλέχτηκε ώστε η συνάρτηση καταλληλότητας να έχει αποτέλεσμα 0 σε ένα άτομο που επιλέγει κάθε λέξη. Η μέγιστη τιμή που μπορεί να πάρει είναι 0.00266

ΔΙΑΣΤΑΥΡΩΣΗ

Καθώς τα γονίδια αποτελούνται από 1 bit το καθένα και δεν υπάρχει κάποια συσχέτιση μεταξύ γειτονικών γονιδίων δεν υπάρχει λόγος να επιλέξουμε διασταύρωση μονού ή πολλαπλού σημείου από την ομοιόμορφη.

ΕΠΙΛΟΓΗ

Η ρουλέτα με βάση τη κατάταξη δείχνει να υπερισχύει της ρουλέτας με βάση το κόστος σε αυτό το πρόβλημα καθώς ο πληθυσμός φτάνει συχνά σε κατάσταση που τα κόστη όλων των ατόμων είναι παρόμοια και σε τέτοιες περιπτώσεις η ρουλέτα κόστους δεν δίνει αρκετή ώθηση στα καλύτερα άτομα. Η επιλογή τουρνουά επιλέγει περισσότερα άτομα χαμηλότερης απόδοσης από την ρουλέτα κατάταξης που οδηγεί σε πιο αργή σύγκλιση. Επιλέχτηκε ρουλέτα κατάταξης.

ΕΛΙΤΙΣΜΟΣ

Χρησιμοποιούμε σχετικά μικρούς πληθυσμούς, και ο ελιτισμός μειώνει περαιτέρω την ποικιλομορφία και οδηγεί σε πιο γρήγορη σύγκλιση, που συνήθως φαίνεται να δίνει χαμηλότερη απόδοση. Οπότε δεν χρησιμοποιήθηκε.

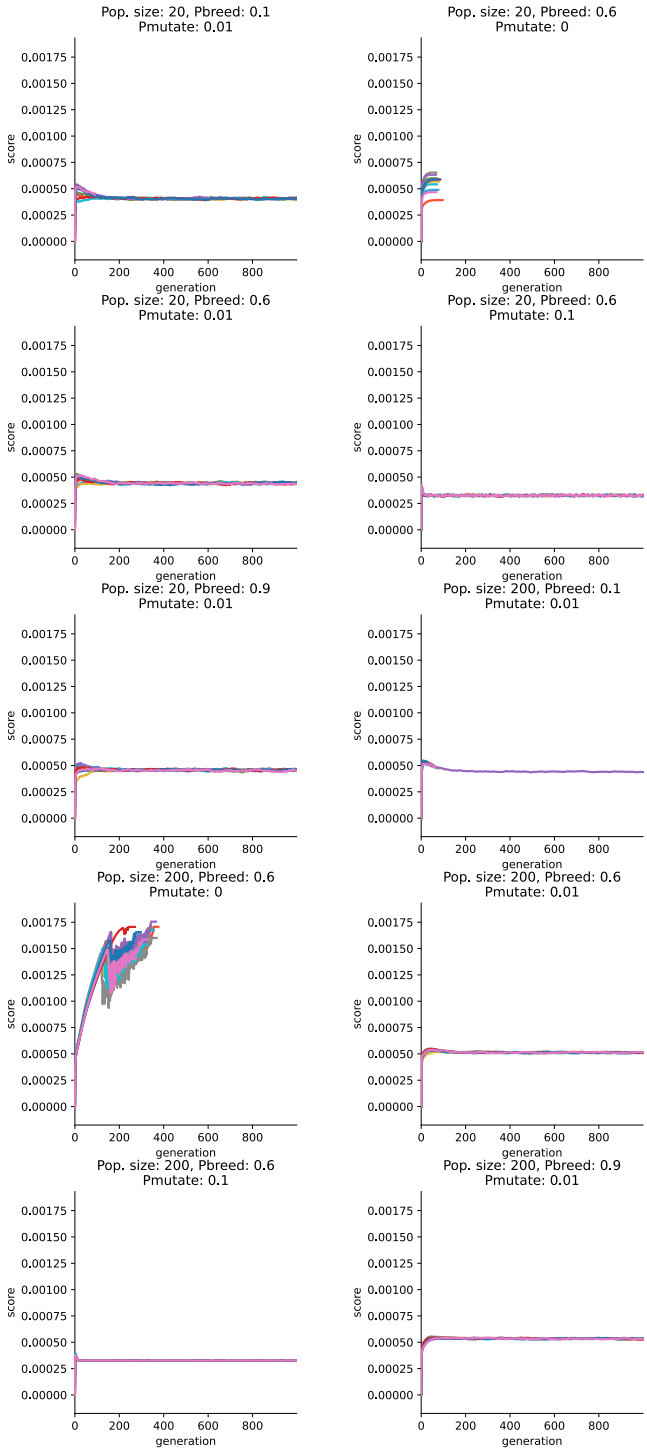
ΠΙΝΑΚΑΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

	Population Size	Crossover Propability	Mutation Propability	Average Score	Average Duration
0	20	0.6	0	0.000552119	77.1
1	20	0.6	0.01	0.00044265	1000
2	20	0.6	0.1	0.000324911	1000
3	20	0.9	0.01	0.000457648	1000
4	20	0.1	0.01	0.00041669	814.1
5	200	0.6	0	0.00165705	338.7
6	200	0.6	0.01	0.000511447	1000
7	200	0.6	0.1	0.000327446	1000
8	200	0.9	0.01	0.000531264	1000
9	200	0.1	0.01	0.000490356	147.8

ΣΥΜΠΕΡΑΣΜΑΤΑ

Παρατηρούμε ότι η αύξηση πληθυσμού από 20 σε 200 βελτιώνει την απόδοση, καθώς δειγματοληπτείται μεγαλύτερο εύρος σε κάθε γενιά. Η μετάλλαξη έχει αρκετά αρνητικό αποτέλεσμα σε αυτό τον αλγόριθμο, επειδή καθώς μειώνεται ο αριθμός λέξεων στα άτομα προσθέτει λέξεις πιο συχνά από ότι αφαιρεί, έτσι ο αριθμός λέξεων τείνει ~ στις 4000. Με μηδενική μετάλλαξη το αποτέλεσμα είναι πολύ καλύτερο και συγκλίνει σε λύση πιο γρήγορα. Η πιθανότητα διασταύρωσης συνδέεται θετικά με την απόδοση, αλλά δεν προκαλεί τόση διαφορά όσο το μέγεθος πληθυσμού ή η μετάλλαξη. Χαμηλή πιθανότητα διασταύρωσης προκαλεί πρόωρη σύγκλιση, καθώς τα ίδια καλύτερα άτομα επιλέγονται κάθε γενιά χωρίς ιδιαίτερες αλλαγές.

ГРАФНМАТА


Pop. size: 20, Pbreed: 0.1
Pmutate: 0.01
Pop. size: 20, Pbreed: 0.6
Pmutate: 0
Pop. size: 20, Pbreed: 0.6
Pmutate: 0.01
Pop. size: 20, Pbreed: 0.6
Pmutate: 0.1
Pop. size: 20, Pbreed: 0.9
Pmutate: 0.01
Pop. size: 200, Pbreed: 0.1
Pmutate: 0.01
Pop. size: 200, Pbreed: 0.6
Pmutate: 0
Pop. size: 200, Pbreed: 0.6
Pmutate: 0.01
Pop. size: 200, Pbreed: 0.6
Pmutate: 0.1
Pop. size: 200, Pbreed: 0.9
Pmutate: 0.01
score
generation
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
0
200
400
600
800

ΥΛΟΠΟΙΗΣΗ

Repo: https://github.com/cyberseihis/Geneword/tree/master

Ο υπολογισμός των tf-idf των λέξεων γίνεται στο tf_idf.py, ο γενετικός αλγόριθμος υλοποιείται στο Jean.py και στο TestJean.py καλείται ο αλγόριθμος για και αποθηκεύονται τα αποτελέσματα.