ΙΩΑΝΝΗΣ ΧΑΡΑΛΑΜΠΟΥΣ
ΑΜ:1059685

Β1. Σχεδιασμός ΓΑ *[30 μονάδες]*

α) Κωδικοποίηση:

Τα άτομα του αρχικού πληθυσμού αποτελούνται από διανύσματα μεγέθους 784 και κάθε σημείο του διανύσματος περιέχει τυχαία 0 ή 1.

β) Αρχικός πληθυσμός

Τα άτομα του αρχικού πληθυσμού αποτελούνται από διανύσματα μεγέθους 784 και κάθε σημείο του διανύσματος περιέχει τυχαία 0 ή 1.

γ) Συνάρτηση καταλληλότητας:

Ως συνάρτηση καταλληλότητας χρησιμοποιώ την αρνητική συνάρτηση κόστους ,όπου από την δημιουργία του δικτύου είναι η cross entropy loss.Συγκεκριμένα ελέγχονται τα αποτελέσματα από το αρνητικό κόστος έχοντας ως εκτίμηση τα δεδομένα εκπαίδευσης (train_x,train_y).Με αυτό τον τρόπο επιτυχαίνεται μία πολύ αποτελεσματική και γρήγορη αξιολόγηση.

δ) Γενετικοί Τελεστές:

i) Ως τελεστής επιλογής επέλεξα την χρήση ρουλέτας με βάση το κόστος καθώς προσφέρει καλά αποτελέσματα και είναι γρήγορη στην εκτέλεση .Η μέθοδος αυτή παρόλο τα θετικά της εμφανίζει κίνδυνο να συγκλίνει πρόωρα σε τοπικό βέλτιστο λόγω κυρίαρχης βέλτιστης τιμής όταν αυξάνουμε την πιθανότητα του.Παρόλαυτα η επιλογή με βάση το κόστος προσφέρει καλά αποτελέσματα και γρήγορη εκτέλεση .Η επιλογή με τουρνουά μπορεί να προσφέρει καλύτερα αποτελέσματα καθώς χρησιμοποιεί ποικιλία ατόμων σε έναν πληθυσμό ,αλλά ο μέσος όρος γενεών είναι πολύ μεγαλύτερος και τα αποτελέσματα αλλάζουν πολύ κάθε φορά που τον τρέχουμε.

ii) Για την διασταύρωση επιλέγουμε την διασταύρωση μονού σημείου μετά από πειραματισμό .Με την διασταύρωση μονού σημείου έχουμε καλύτερα αποτελέσματα και μεγαλύτερη ποικιλομορφία στις γενεές καθώς το σημείο αυτό είναι τυχαίο.

iii) Με την χρήση μετάλλαξης αυξάνεται η ποικιλομορφία του κάθε πληθυσμού και αποφεύγεται το τοπικό βέλτιστο που θα αποτρέψει τον αλγόριθμο να κάνει καλή βελτιστοποίηση.Για τον ελιτισμό χρησιμοποιώ τα τρία καλύτερα αποτελέσματα ,τα οποία δεν μπορούν να υποστούν μετάλλαξη ή διασταύρωση .Με αυτό τον τρόπο ο αλγόριθμος συγκλίνει πιο γρήγορα και τα αποτελέσματα είναι πάντα αρκετά καλά καθώς οι πρώτες τυχαίες τιμές του πληθυσμού έχουν ποικιλομορφία και πλησιάζουν το βέλτιστο.Όμως σε περίπτωση που η πιθανότητα μετάλλαξης είναι μεγάλη τότε είναι σχεδόν σίγουρο ότι θα προκύψει τοπικό βέλτιστο λόγω του ελιτισμού όπως φαίνεται και στη συνέχεια.

Β2. Υλοποίηση ΓΑ *[30 μονάδες]*
Να γράψετε ένα πρόγραμμα, σε οποιοδήποτε περιβάλλον ή γλώσσα προγραμματισμού, που να υλοποιεί τον γενετικό αλγόριθμο που σχεδιάσετε.

Link για τον κώδικα: https://github.com/cgiannos/2h-ergasia-ypologistikhs-noymosynhs.git

Β3. Αξιολόγηση και Επίδραση Παραμέτρων *[30 μονάδες]*
α) Να τρέξετε τον αλγόριθμο για τις τιμές των παραμέτρων που φαίνονται στον παρακάτω πίνακα και να τον συμπληρώσετε. Ο αλγόριθμος θα τερματίζει όταν πληρούνται ένα ή περισσότερα από τα *κριτήρια τερματισμού*, δηλαδή όταν:
i i. το καλύτερο άτομο της κάθε γενιάς πάψει να βελτιώνεται για ορισμένο αριθμό γενεών ή
ii ii. βελτιώνεται κάτω από ένα ποσοστό (<1%) ή
iii iii. έχει ξεπεραστεί ένας προκαθορισμένος αριθμός γενεών (π.χ. 1000)

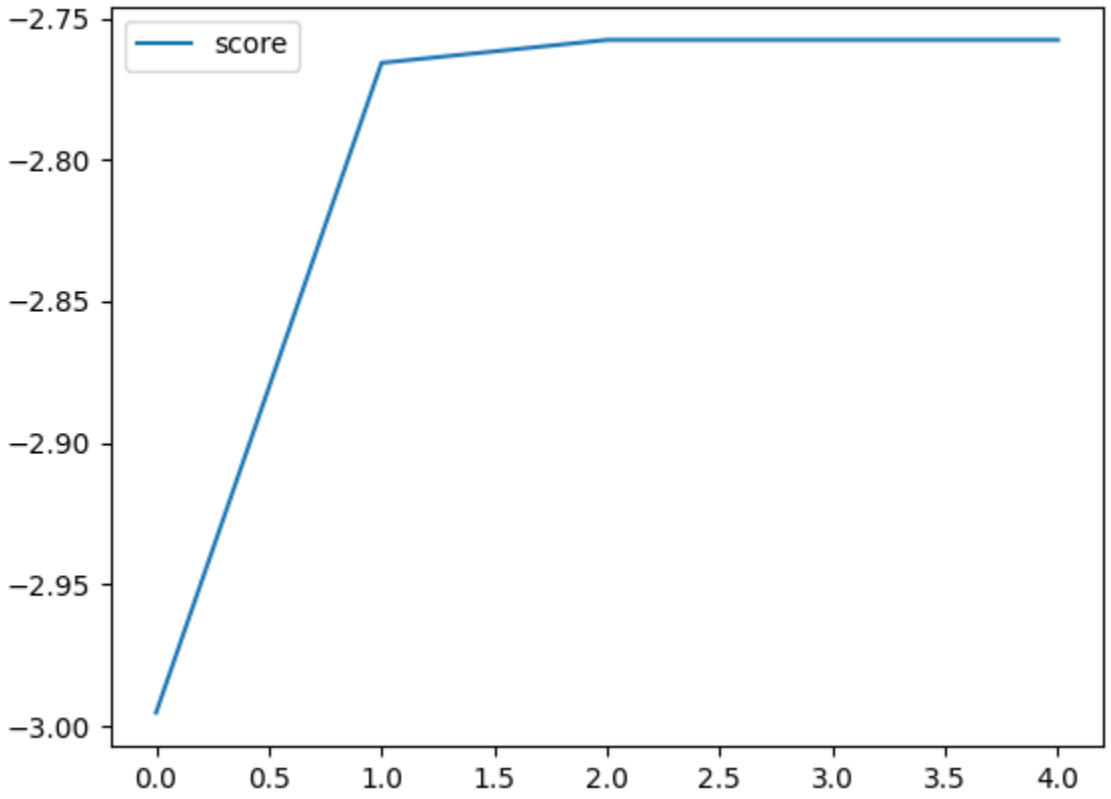
Α/Α	Μέγεθος πληθυσμού	Πιθανότητα διασταύρωσης	Πιθανότητα Μετάλλαξης	Μέση τιμή βέλτιστου	Μέσος αριθμός γενεών
1	20	0.6	0.00	2.7577	5
2	20	0.6	0.01	2.4227	5
3	20	0.6	0.10	2.4116	1
4	20	0.9	0.01	2.4686	5
5	20	0.1	0.01	2.3837	5
6	200	0.6	0.00	2.3247	5
7	200	0.6	0.01	2.4263	5
8	200	0.6	0.10	2.3691	1
9	200	0.9	0.01	2.3171	5
10	200	0.1	0.01	2.3954	5

β) Για κάθε περίπτωση του παραπάνω πίνακα να σχεδιάστε την καμπύλη εξέλιξης (απόδοση/αριθμό γενιών) της καλύτερης λύσης (της μέσης τιμής αυτής, σε κάθε τρέξιμο).

Σε κάθε σχεδιάγραμμα βλέπουμε την αρνητική συνάρτηση κόστους .Ο αλγόριθμος τρέχει για πέντε γενιές καθώς χρειάζεται πολύ χρόνο για να εκτελεστεί με περισσότερες γενιές.

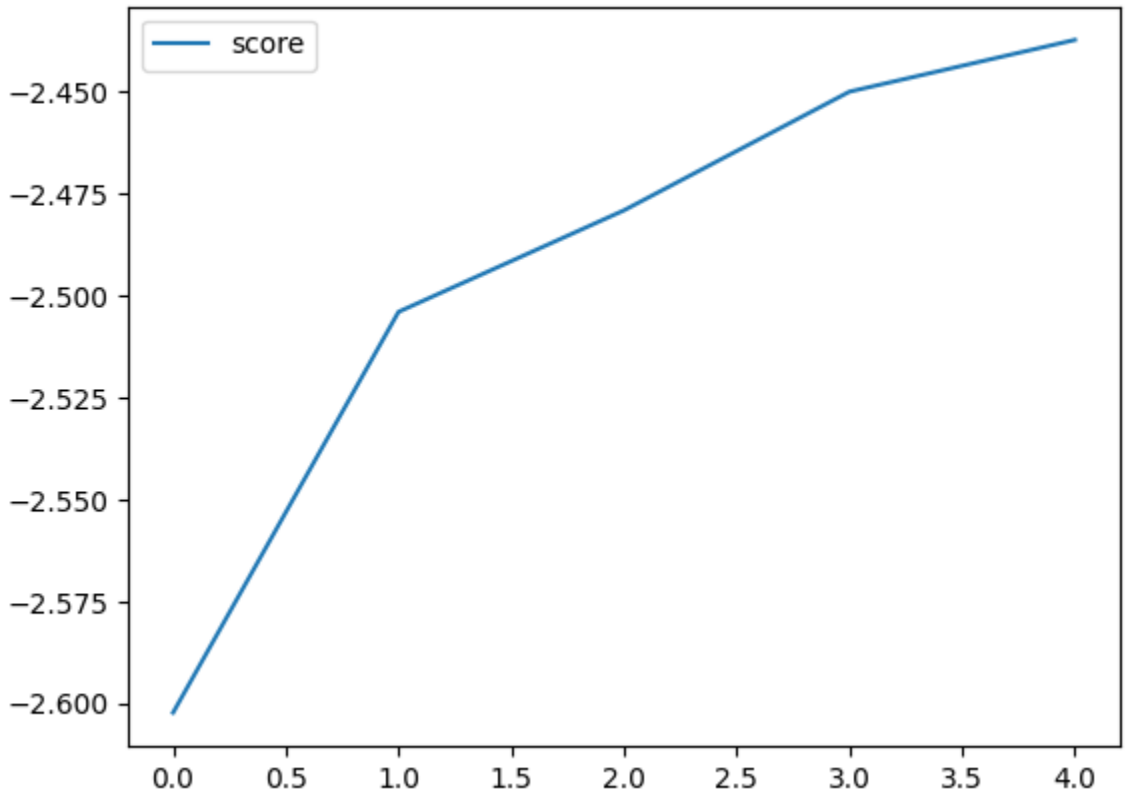
1	20	0.6	0.00	2.7577	5

```
[Generation 4 / 5] score: -2.7577 (best: -2.7577; std: 0.0)
[Generation 5 / 5] score: -2.7577 (best: -2.7577; std: 0.0)
1667/1667 [==============================] - 12s 7ms/step - loss: 2.7577 - accuracy: 0.0834
Model accuracy: 0.08340000361204147
```



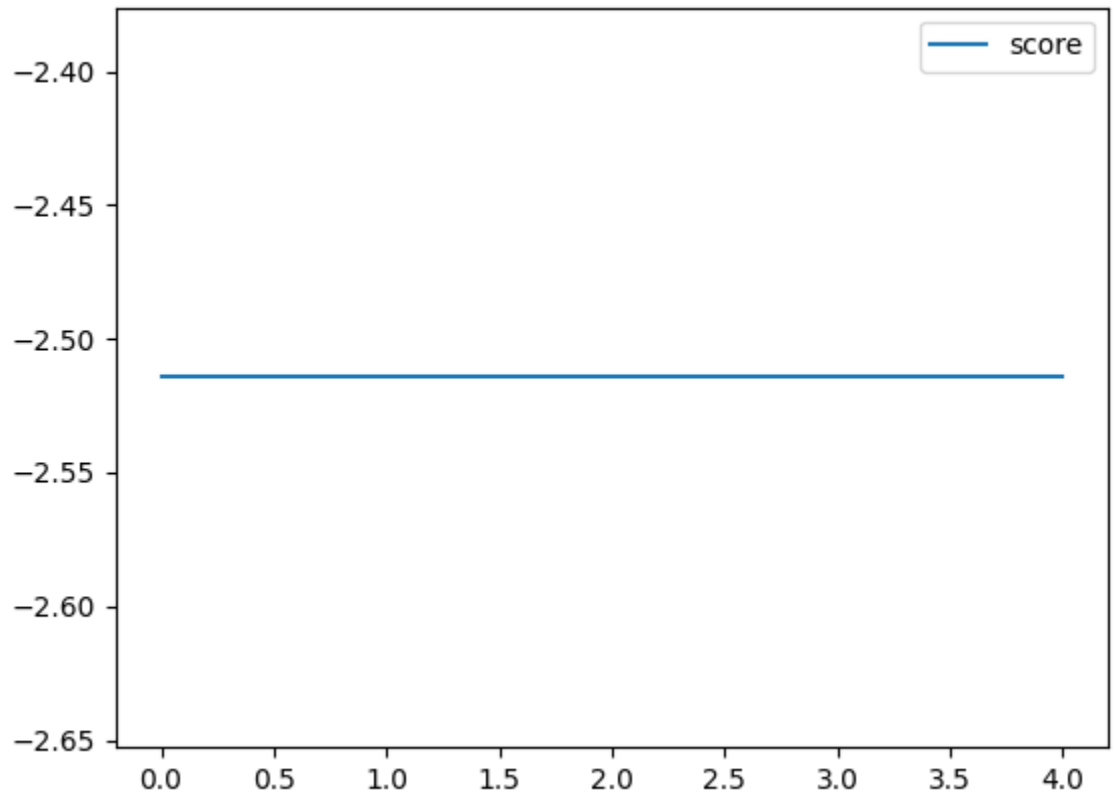
2	20	0.6	0.01	2.4227	5

```
2021-06-18 18:28:41.972925: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] Non
[Generation 1 / 5] score: -2.6022 (best: -2.5353; std: 0.0819)
[Generation 2 / 5] score: -2.5039 (best: -2.4457; std: 0.0504)
[Generation 3 / 5] score: -2.479 (best: -2.4457; std: 0.0452)
[Generation 4 / 5] score: -2.4499 (best: -2.4431; std: 0.0095)
[Generation 5 / 5] score: -2.4372 (best: -2.4227; std: 0.0126)
1667/1667 [==============================] - 14s 8ms/step - loss: 2.4227 - accuracy: 0.1125
Model accuracy: 0.1124500036239624
```



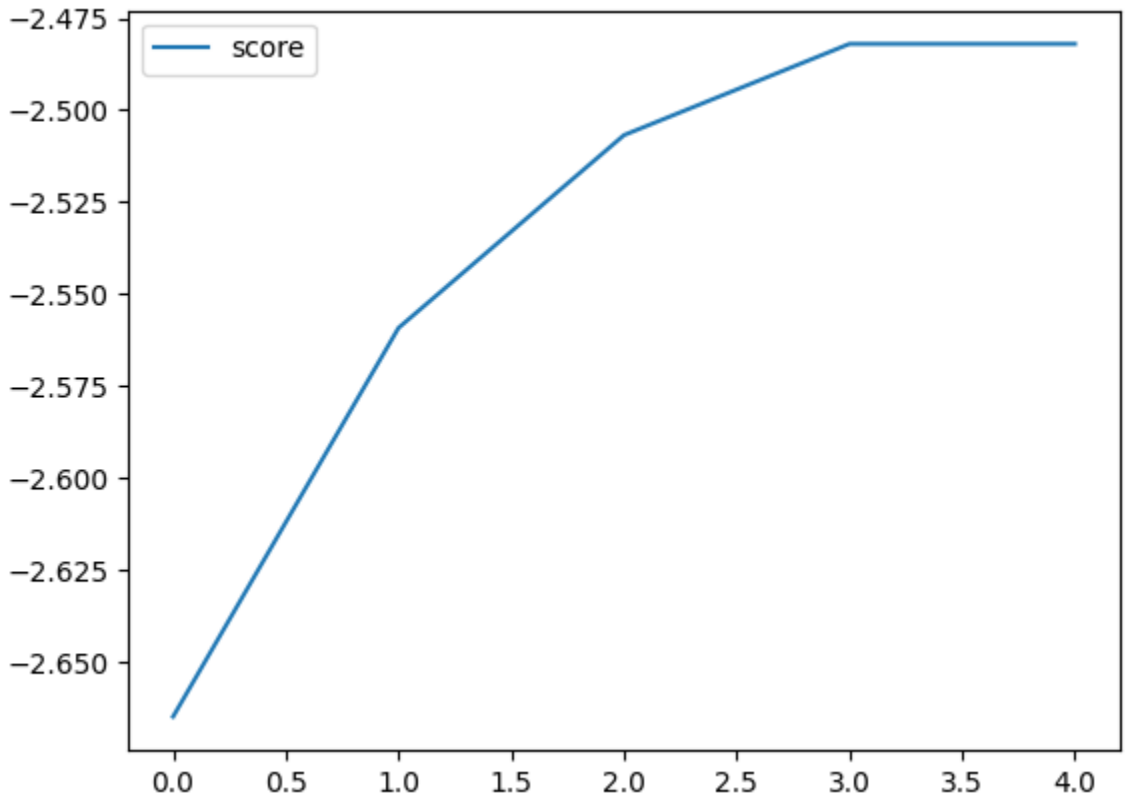
3	20	0.6	0.10	2.4116	1

```
2021-06-18 18:53:37.633929: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] No
[Generation 1 / 5] score: -2.5144 (best: -2.4116; std: 0.0909)
[Generation 2 / 5] score: -2.5144 (best: -2.4116; std: 0.0909)
[Generation 3 / 5] score: -2.5144 (best: -2.4116; std: 0.0909)
[Generation 4 / 5] score: -2.5144 (best: -2.4116; std: 0.0909)
[Generation 5 / 5] score: -2.5144 (best: -2.4116; std: 0.0909)
1667/1667 [==============================] - 12s 7ms/step - loss: 2.4116 - accuracy: 0.0894
Model accuracy: 0.08935000002384186
```



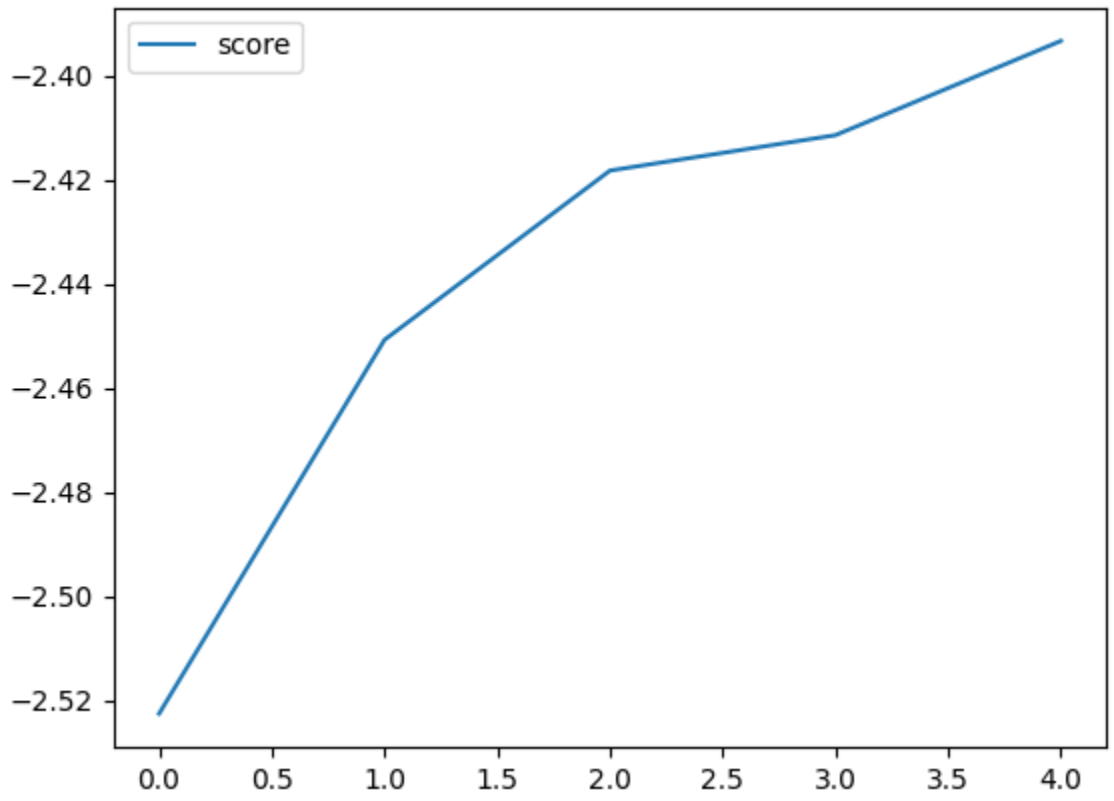
4	20	0.9	0.01	2.4686	5

```
[Generation 1 / 5] score: -2.6649 (best: -2.4961; std: 0.1467)
[Generation 2 / 5] score: -2.5592 (best: -2.4961; std: 0.0562)
[Generation 3 / 5] score: -2.5068 (best: -2.4814; std: 0.0321)
[Generation 4 / 5] score: -2.482 (best: -2.4686; std: 0.0137)
[Generation 5 / 5] score: -2.482 (best: -2.4686; std: 0.0137)
1667/1667 [==============================] - 9s 5ms/step - loss: 2.4686 - accuracy: 0.1162
Model accuracy: 0.11623333394527435
```



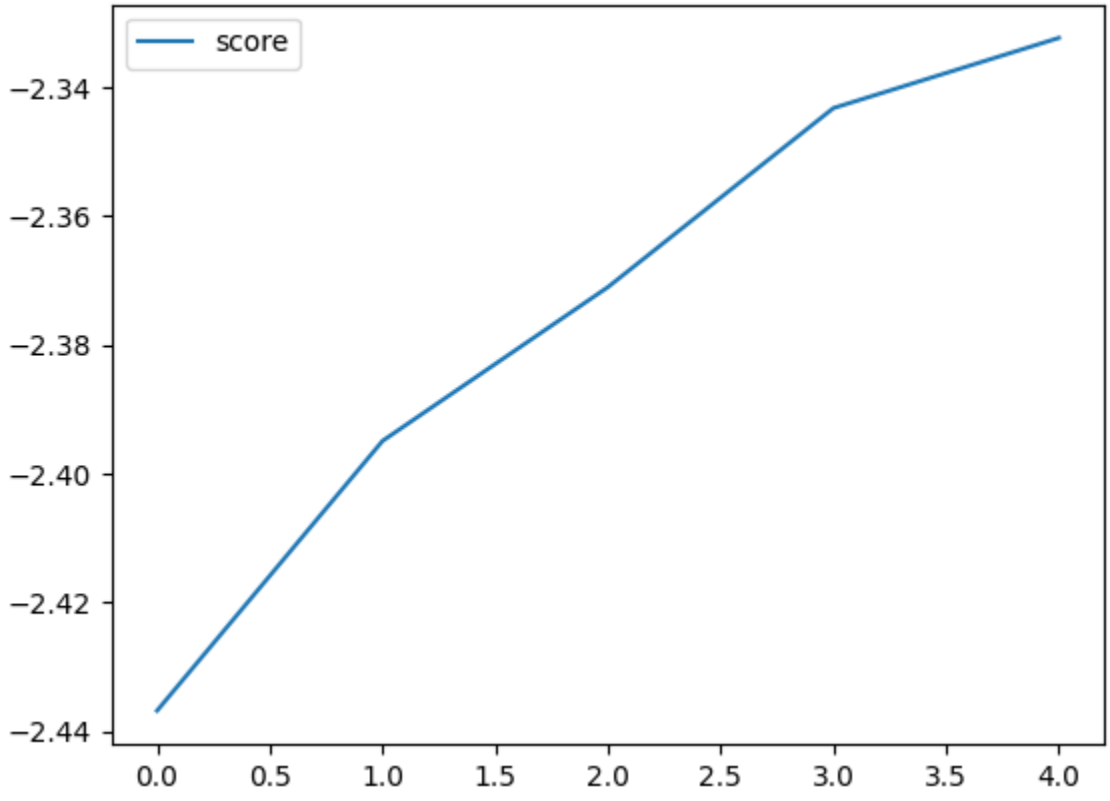
5	20	0.1	0.01	2.3837	5

```
[Generation 1 / 5] score: -2.5225 (best: -2.422; std: 0.1171)
[Generation 2 / 5] score: -2.4507 (best: -2.422; std: 0.0384)
[Generation 3 / 5] score: -2.4182 (best: -2.4132; std: 0.0045)
[Generation 4 / 5] score: -2.4114 (best: -2.4015; std: 0.0091)
[Generation 5 / 5] score: -2.3933 (best: -2.3837; std: 0.009)
1667/1667 [==============================] - 9s 6ms/step - loss: 2.3837 - accuracy: 0.1070
Model accuracy: 0.10703333467245102
```



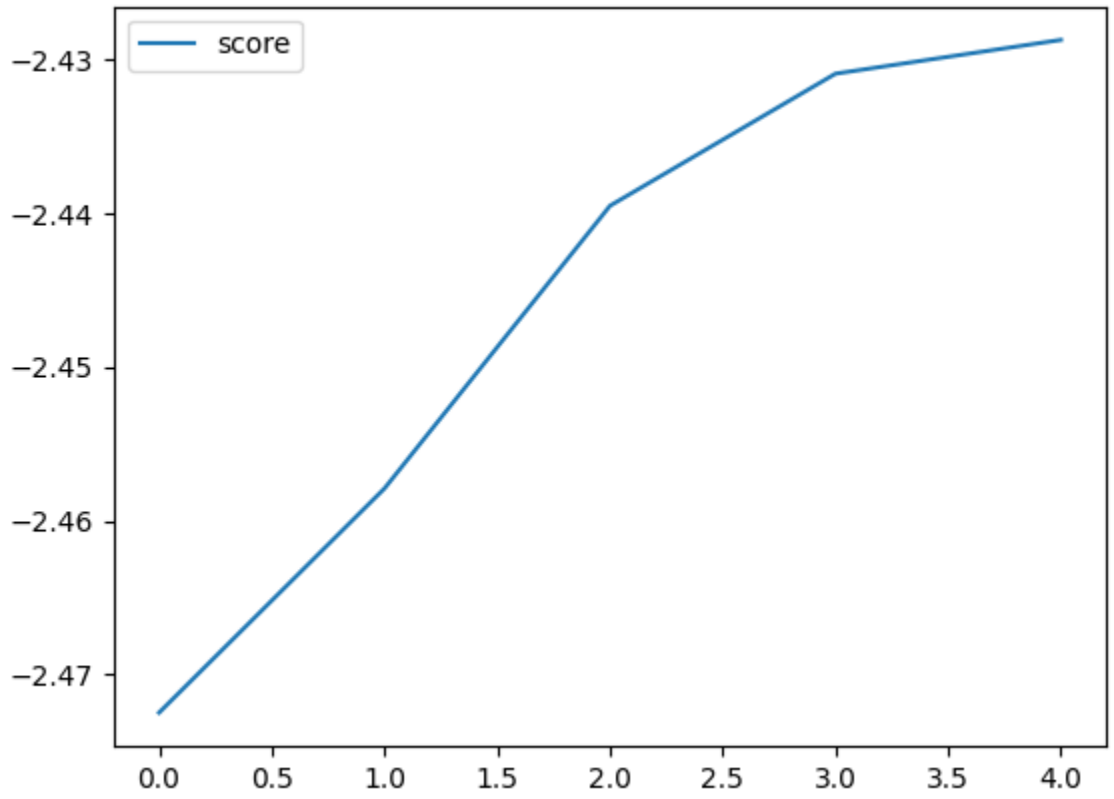
6	200	0.6	0.00	2.3247	5

```
[Generation 1 / 5] score: -2.4368 (best: -2.4137; std: 0.0397)
[Generation 2 / 5] score: -2.3949 (best: -2.3574; std: 0.0325)
[Generation 3 / 5] score: -2.371 (best: -2.3574; std: 0.0235)
[Generation 4 / 5] score: -2.3432 (best: -2.3247; std: 0.0168)
[Generation 5 / 5] score: -2.3323 (best: -2.3247; std: 0.0132)
1667/1667 [==============================] - 9s 5ms/step - loss: 2.3247 - accuracy: 0.1594
Model accuracy: 0.15940000116825104
```



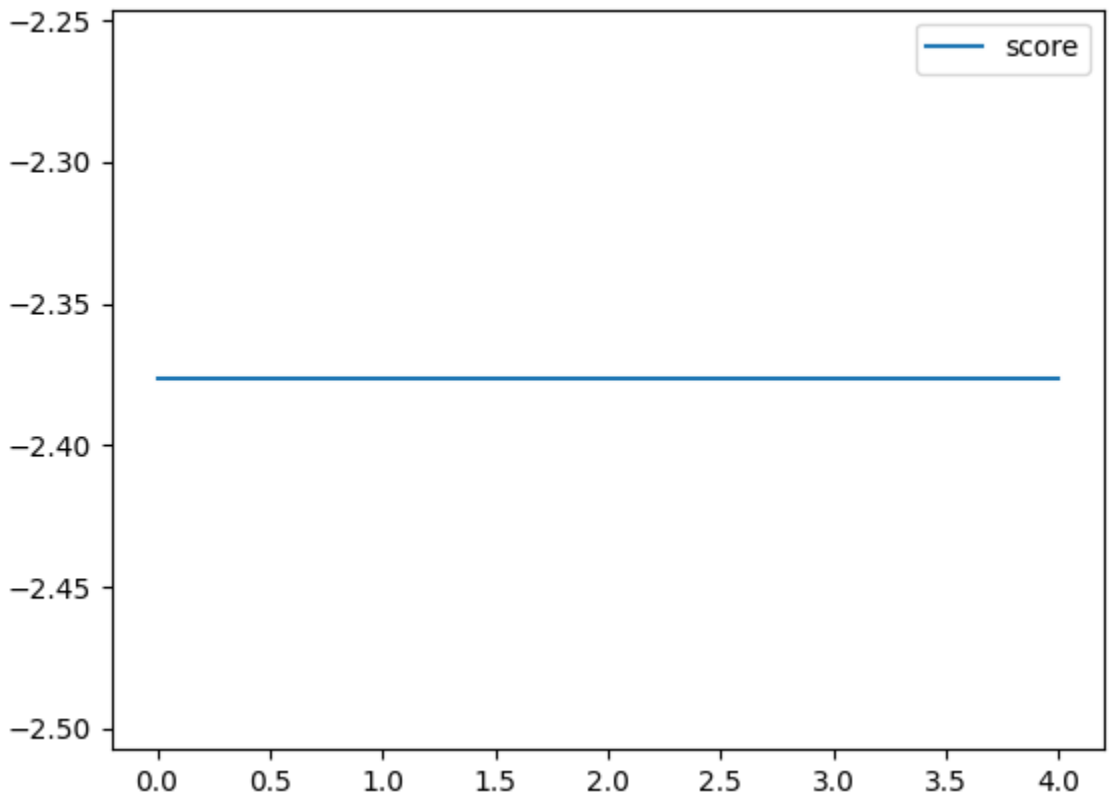
7	200	0.6	0.01	2.4263	5

```
2021-06-19 19:14:44.649218: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116]
[Generation 1 / 5] score: -2.4725 (best: -2.4522; std: 0.0325)
[Generation 2 / 5] score: -2.4579 (best: -2.4522; std: 0.0073)
[Generation 3 / 5] score: -2.4395 (best: -2.4302; std: 0.0113)
[Generation 4 / 5] score: -2.4309 (best: -2.4263; std: 0.005)
[Generation 5 / 5] score: -2.4287 (best: -2.4263; std: 0.0021)
1667/1667 [==============================] - 9s 5ms/step - loss: 2.4263 - accuracy: 0.1057
Model accuracy: 0.10570000112056732
```



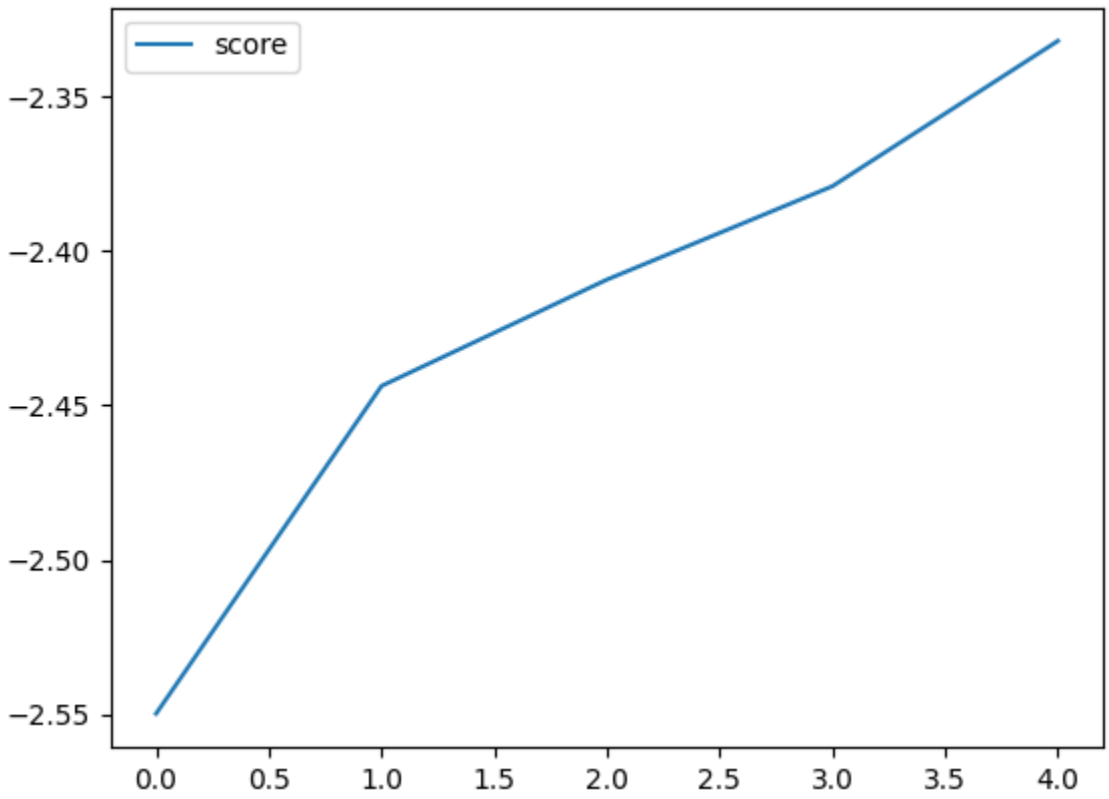
8	200	0.6	0.10	2.3691	1

```
------------------------------------------------------------------
2021-06-19 20:25:03.299762: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None
[Generation 1 / 5] score: -2.3767 (best: -2.3691; std: 0.0107)
[Generation 2 / 5] score: -2.3767 (best: -2.3691; std: 0.0107)
[Generation 3 / 5] score: -2.3767 (best: -2.3691; std: 0.0107)
[Generation 4 / 5] score: -2.3767 (best: -2.3691; std: 0.0107)
[Generation 5 / 5] score: -2.3767 (best: -2.3691; std: 0.0107)
1667/1667 [==============================] - 10s 6ms/step - loss: 2.3691 - accuracy: 0.1554
Model accuracy: 0.15541666746139526
```



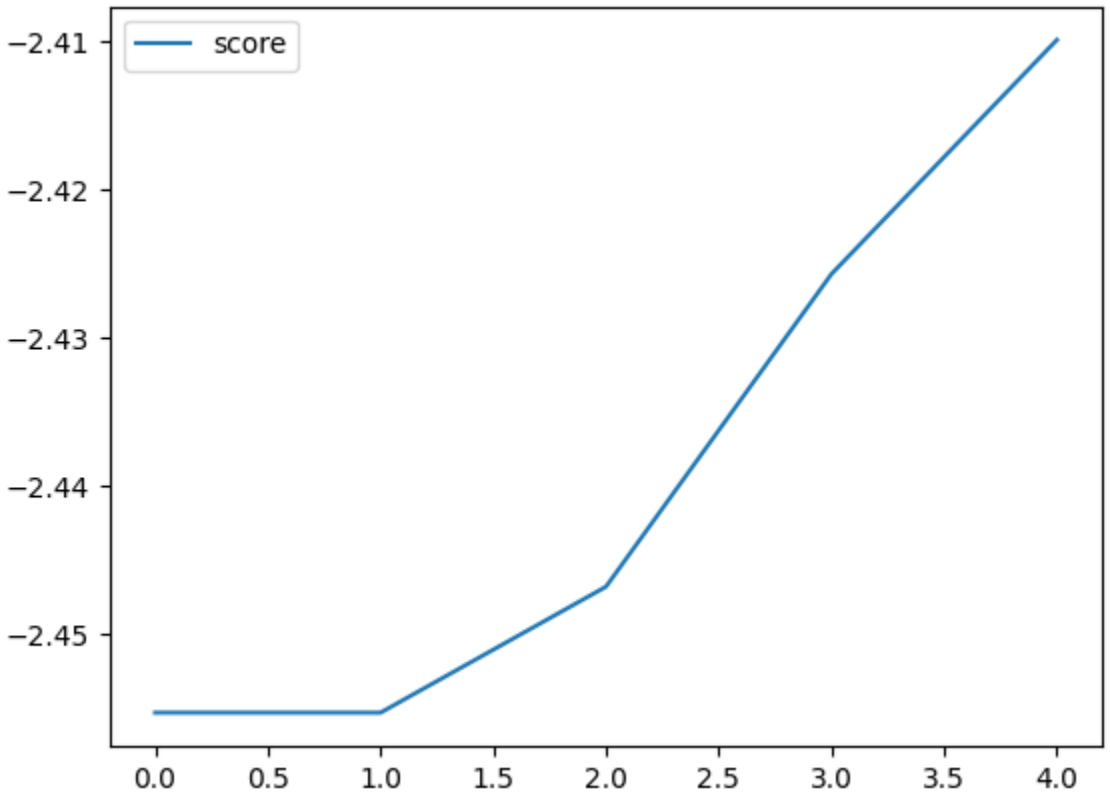
9	200	0.9	0.01	2.3171	5

```
2021-06-19 21:01:59.442109: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of th
[Generation 1 / 5] score: -2.5497 (best: -2.4421; std: 0.0932)
[Generation 2 / 5] score: -2.4437 (best: -2.405; std: 0.0396)
[Generation 3 / 5] score: -2.4093 (best: -2.3809; std: 0.0308)
[Generation 4 / 5] score: -2.3791 (best: -2.3513; std: 0.0269)
[Generation 5 / 5] score: -2.3321 (best: -2.3171; std: 0.0175)
1667/1667 [==============================] - 10s 6ms/step - loss: 2.3171 - accuracy: 0.1847
Model accuracy: 0.1846500039100647
```



10	200	0.1	0.01	2.3954	5

```
[Generation 1 / 5] score: -2.4553 (best: -2.4463; std: 0.0084)
[Generation 2 / 5] score: -2.4553 (best: -2.4463; std: 0.0084)
[Generation 3 / 5] score: -2.4468 (best: -2.4422; std: 0.0049)
[Generation 4 / 5] score: -2.4257 (best: -2.403; std: 0.0203)
[Generation 5 / 5] score: -2.4099 (best: -2.3954; std: 0.0189)
1667/1667 [==============================] - 9s 5ms/step - loss: 2.3954 - accuracy: 0.1639
```



γ) Με βάση αυτές τις καμπύλες, αλλά και τα αποτελέσματα του παραπάνω πίνακα, να διατυπώσετε αναλυτικά τα συμπεράσματά σας σχετικά με την επίδραση της κάθε παραμέτρου (μέγεθος πληθυσμού, πιθανότητα διασταύρωσης, πιθανότητα μετάλλαξης) στη σύγκλιση του αλγορίθμου.

Σύμφωνα με τις καμπύλες και τα αποτελέσματα καταλήγουμε στο συμπέρασμα ότι τα αποτελέσματα για κάθε μέγεθος επηρεάζονται από τις πιθανότητες μετάλλαξης και διασταύρωσης.Όσων αφορά την πιθανότητα διασταύρωσης η ιδανική τιμή φαίνεται να είναι κάτω από 50% για μικρό μέγεθος πληθυσμού ενώ χρειαζόμαστε μεγάλη πιθανότητα διασταύρωσης για μεγάλο μέγεθος πληθυσμού καθώς όσο αυξάνονται τα άτομα χρειαζόμαστε περισσότερες διασταυρώσης για να διαφοροποιούνται τα άτομα και να έχουμε καλά αποτελέσματα.Τέλος όσως αφορά την πιθανότητα μετάλλαξης για κάθε μέγεθος πληθυσμού η ιδανική τιμή είναι κοντά στο 1% .Αν δεν βάλουμε πιθανότητα μετάλλαξης δεν έχουμε καλά αποτελέσματα ενώ αν βάλουμε μεγαλύτερη πιθανότητα έχουμε ελάχιστα καλύτερα αποτελέσματα ,όμως έχουμε και τοπική βέλτιστη τιμή η οποία εμποδίζει τον αλγόριθμο να τρέξει για καινούργιες γενεές.