

Traveling Salesman Problem με γενετικό αλγόριθμο.

Η συγκεκριμένη υλοποίηση ξεκινά με το initialization των κόμβων (points) και των μονοπατιών που τα συνδέουν (paths).

Από την βασική function (run_sim) καλούνται οι εξής βοηθητικές functions:

- **get_path_cost(a,b)**: Επιστρέφει το κόστος μετάβασης από τον κόμβο a στον κόμβο b.
- **eval_sample(sample)**: Επιστρέφει το συνολικό score ενός sample.
- **crossover(parent)**: Επιστρέφει ένα sample-offspring το οποίο δημιουργείται από v ($0 \leq v \leq \text{len}(\text{parent}) - 1$) όμοια στοιχεία, στο τέλος των οποίων εντάσσει τυχαία όσα στοιχεία δεν υπάρχουν στο sample . Π.χ. parent \rightarrow c,a,d,b,e $v \rightarrow 2$ offspring \rightarrow c,a (from parent) b,e,d (random)
- **mutate(s, rate)**: Με πιθανότητα rate πραγματοποιεί mutation στο δοθέν sample (s), αλλάζοντας του τυχαία 2 στοιχεία.
- **check_end(lst, threshold)**: Επιστρέφει True αν η λίστα lst έχει τα τελευταία #threshold στοιχεία τις όμοια. Χρησιμοποιείται για early stopping.

Οι βασικές functions είναι οι εξής:

run_sim(epochs, pop_len, elitism_no, points, paths, mut_rate, early_stopping=False)

Πραγματοποιεί ένα simulation δοθέντων των παραπάνω παραμέτρων. Αρχικά δημιουργεί έναν πληθυσμό μεγέθους pop_len με τυχαία samples. Έπειτα ξεκινά τις επαναλήψεις (#epochs) στις οποίες:

1. Αξιολογεί τον πληθυσμό.
2. Ελέγχει για το ενδεχόμενο early stopping την λίστα (scores) που αποθηκεύει το καλύτερο score κάθε επανάληψης.
3. Αρχικοποιεί την λίστα population με #elitism_no στοιχεία από την sorted λίστα ratings.
4. Ολοκληρώνει το νέο population εντάσσοντας pop_len - elitism_no παιδιά στην λίστα new_pop, τα οποία μεταλλάσσονται με πιθανότητα mut_rate.
5. Καθαρίζει την λίστα population, ώστε να μπορεί να χρησιμοποιηθεί στην επόμενη επανάληψη, και εντάσσει σε αυτήν τα παιδιά (new_pop).
6. Αποθηκεύει το καλύτερο score της γενιάς που πέρασε και επαναλαμβάνει.

run_multiple_sims(no_of_sims, epochs, pop_len, elitism_no, points, paths, mut_rate, early_stopping=True)

Η συγκεκριμένη συνάρτηση επιτρέπει την πραγματοποίηση πολλών simulations, ώστε να παραχθεί μια καλύτερη εικόνα για τις εν λόγω παραμέτρους.

find_best_params(no_of_sims=20)

Η συνάρτηση αυτή πραγματοποιεί #no_of_sims simulations για τον κάθε συνδυασμό παραμέτρων εντός του δοθέντος εύρους. Σκοπός της είναι η εύρεση ενός συνόλου από παραμέτρους που επιστρέφουν την βέλτιστη λύση.

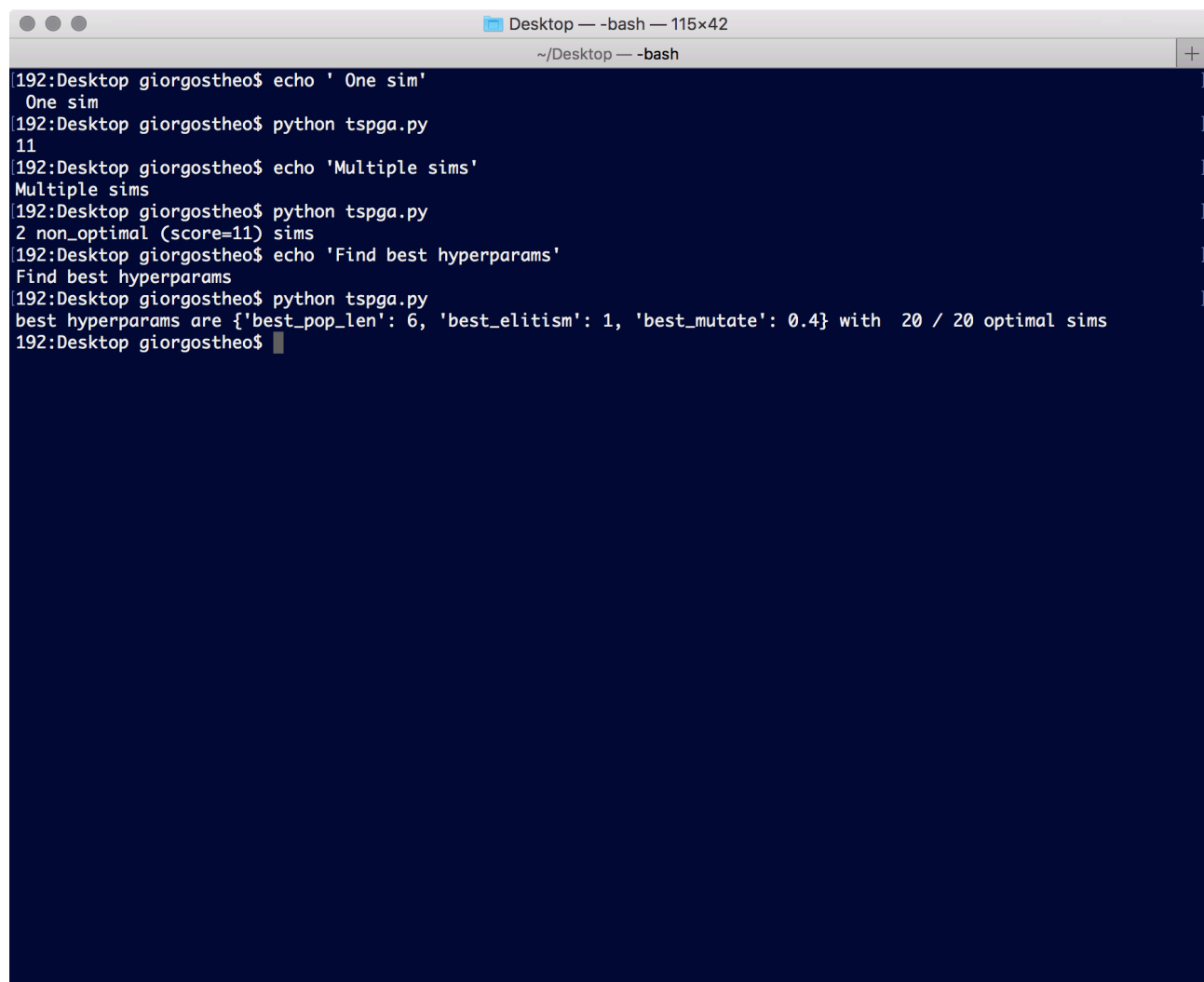
Κάνοντας uncomment το κάθε μέρος της __main__ , πραγματοποιείται το αντίστοιχο σενάριο.

Παραδείγματα εκτέλεσης:

run_sim(epochs=20, pop_len=7, elitism_no=1, mut_rate=0.3).

run_multiple_sims(no_of_sims=20, epochs=20, pop_len=7, elitism_no=1, mut_rate=0.3).

find_best_params()



```
192:Desktop giorgostheo$ echo ' One sim'
One sim
192:Desktop giorgostheo$ python tspga.py
11
192:Desktop giorgostheo$ echo 'Multiple sims'
Multiple sims
192:Desktop giorgostheo$ python tspga.py
2 non_optimal (score=11) sims
192:Desktop giorgostheo$ echo 'Find best hyperparams'
Find best hyperparams
192:Desktop giorgostheo$ python tspga.py
best hyperparams are {'best_pop_len': 6, 'best_elitism': 1, 'best_mutate': 0.4} with 20 / 20 optimal sims
192:Desktop giorgostheo$
```

