Hot-TSP

Travelling salesman problem, python implementation of genetic solution

Authors:

- Romain Claret
- Steve Visinand

Implementation

Strategy

Below a short description of our algorithm strategy:

- First we initialise the starting population
  - Create chromosomes from shuffled genes list
- For each generation:
  - Check time limit, stop if reached.
  - Do the elitism to reduce the population and keep the *best*
  - Check if the previous king is a clone of the current king if the clone limit is passed, we stop.
  - If the limit is not passed, add the other elites with king to the noble population
  - Then do the evolution
  - Selection (tournament, ranked, roulette) on each chromosome of the couples

  - Crossover (breed)
  - Mutation (X-Men)
  - Population replacement (Darwin Awards)
  - Calculate the fitness of the new population
  - And sort the new population

Claret_Visinand.py is fully commented, please read them for more information. PVC-tester.py has been updated for python 3.5.

Optimisation

After multiples tests (brute force) with search_params_bruteforce.py on Professeur Bilat's CUDA Server we have concluded that:

- The **ranked** selection on both chromosomes of the couple gives the best results.
- This brute force utility helped us to optimise the coefficients of the algorithm, indeed, our genetic algorithm is fully customisable.

With this project we joined a sample of the results generated by search_params_bruteforce.py (csv files) for the following commands:

- python search_params_bruteforce.py 2 9 1 2 9 1 1 91 10 1 51 10
- python search_params_bruteforce.py 10 19 1 2 9 1 1 91 10 1 51 10
- python search_params_bruteforce.py 20 29 1 2 9 1 1 91 10 1 51 10
- python search_params_bruteforce.py 30 39 1 2 9 1 1 91 10 1 51 10
- python search_params_bruteforce.py 40 49 1 2 9 1 1 91 10 1 51 10
- python search_params_bruteforce.py 50 59 1 2 9 1 1 91 10 1 51 10
- python search_params_bruteforce.py 60 69 1 2 9 1 1 91 10 1 51 10
- python search_params_bruteforce.py 70 79 1 2 9 1 1 91 10 1 51 10
- python search_params_bruteforce.py 80 89 1 2 9 1 1 91 10 1 51 10
- python search_params_bruteforce.py 90 99 1 2 9 1 1 91 10 1 51 10
- python search_params_bruteforce.py 100 150 50 2 9 1 10 90 10 10 90 10

The arguments of search_params_bruteforce.py are defined, in order:

- population size min
- population size max
- population size unit
- tournaments min
- tournaments max
- tournaments unit
- elitism rate min
- elitism rate max
- elitism rate unit
- mutation rate min
- mutation rate max
- mutation rate unit

Annexes

- Papers in *papers* folder
- Sample of brute-force results in *test_sample_results*
- Data of genes in *data* folder
- Updated *PVC-tester.py* for python 3.5 at root
- Our genetic algorithm *Claret_Visinand.py* at root
- The brute-force file *search_params_bruteforce.py* at root