Code genetic algorithm

Trong folder evrp:

* Evrp\_instance.py:
  + Class EvrpInstance: Nhiệm vụ là lưu cách đọc file data và trả về khoảng cách euclide của 2 điểm bất kì trên map
  + priority\_queue.py: Code 1 hàng đợi Queue để phục vụ cho giải thuật GA
  + utils.py:

1. is\_capacity\_feasible(): Hàm kiểm tra hợp lệ của 1 cá thể (khách hàng): Nếu tổng khối lượng của tất cả các khách hàng trong 1 chặng đường lớn hơn tổng khối lượng cho phép thì False, else True
2. reconstruct\_individual(): Đầu vào hàm này sẽ là tuyến đường đã được làm phẳng và độ dài tuyến đường đó. tái tạo lại định dạng ban đầu của cá nhân với hai tuyến đường. Ví dụ:  
    Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, số

   Mô tả được tạo tự động
3. def deduplicate\_population(): Loại bỏ phần tử trùng lặp của 1 quần thể
4. def find\_nearest\_station(): tìm trạm sạc gần nhất
5. def fitness\_evaluation(): tính toán tổng khoảng cách của các tuyến đường trong cá nhân, mục đích chính là đánh giá chất lượng của các tuyến đường đã được tạo ra. \*CT trong paper thầy gửi
6. các hàm plot và tạo file csv: tự đọc
   * ga.py:
7. generate\_individual\_evenly() : Tạo 1 danh sách các cá thể dựa theo số lượng khách hàng và số xe (không ngẫu nhiên)
8. generate\_individual\_randomly(): Tạo 1 danh sách các cá thể dựa theo số lượng khách hàng và số xe (ngẫu nhiên)
9. cx\_partially\_matched(): Lai tạo chéo theo phương pháp : Partially Matched Crossover (PMX)
10. two\_opt(): thực hiện tìm kiếm cục bộ 2 opt để tối ưu hóa tuyến đường và dựa trên tuyến đường và khoảng cách -> kết quả sẽ thu về đc 1 tuyến đường đc tối ưu hóa về mặt khoảng cách
11. calculate\_total\_distance(): trả về tổng khoảng cách của 1 tuyến đường dựa trên ma trận khoảng cách toàn cục
12. simple\_repair(): fix lại từng route sao cho phù hợp với các ràng buộc về năng lượng và khối lượng
13. local\_search(): sử dụng để thực hiện tìm kiếm cục bộ trên một cá thể (individual) để tối ưu hóa:
    1. 2 – opt: tối ưu hía giữa các tuyến đường dựa vào các cá thể và khoảng các và tìm tìm ra tuyến đường được tối ưu hóa
    2. Simple\_repair() #ZGA: sửa chữa các tuyến đường được tối ưu hóa và lưu vào repaired\_invidual
    3. Fitness\_evaluation(): tính toán chi phí
    4. Kết quả trả về: ược sử dụng để thực hiện tìm kiếm cục bộ trên một cá thể (individual) trong bài toán tối ưu hóa
14. Run\_ga():