Name: **Alec Durgheu** User-name: **pwtn63**

Algorithm A: **Genetic Algorithm**

Algorithm B: **Particle Swarm Optimization**

Description of enhancement of Algorithm A:

The concept of elitism was introduced, which first determines the elite group within a population at a given time. This group consists of the *k* highest fitness chromosomes, and parents are always selected to reproduce the next generation from this elite group only. The size of this elite group *k* can be manipulated via the hyperparameter ***elite_size*** and is set at 4, but largely depends on the population size. Elite parents are still selected in a roulette wheel fashion.

A new mutation operator was used, referred to as the *reverse mutation*, which picks two breakpoints in a chromosome (excluding the first and last city in the tour) and reverses the gene sequence in between to create a variation. This method breaks and reforms only 2 edges in the Hamiltonian cycle, whereas the basic implementation of a mutation usually changes 4. The *reverse mutation* operator is usually implemented, but there is a probability that if a mutation occurs, it will be a *swap mutation* (as seen in the basic implementation and in lectures). This probability can be controlled by the hyperparameter ***harsh_mute*** and is typically set to 0.1.

If the algorithm has not discovered a new best tour after a certain period of "in-script" time (time modelled by the parameter t in the main loop of the Genetic algorithm), the population is recreated entirely with a devolved variant of the current best tour. If no new best tour is found within a certain time limit (*iters* + *cook_t*), we repeat this procedure. Each repetition increases the strength to which we devolve the chromosomes (the global variable *devol*), but only up to a certain limit (typically, *num_cities* // 2). If we reach this limit, we return to the lowest degree of *devol* (typically, 2) before increasing similarly in a cyclical manner. *devol* specifies the number of times a chromosome undergoes the *swap mutation* in the devolve() function, which is responsible for devolving individuals. *devol* is also reset when a new best tour is found, and the time limit for a devolved search increases appropriately as well. The intuition behind this procedure is to expand the search to undiscovered, potentially superior, regions and pull the algorithm out of a local optimum.

Description of enhancement of Algorithm B:

First, the positions of the particles are initialised to be that of the greedy tour, or some variation of it. The greedy tour refers to selecting the closest (shortest distance) city repeatedly, starting with city 0. Since this idea only produces one tour (and ideally the particles should be spread out to begin with), the remaining *N*-1 particles will be some variation of this greedy tour. The *reverse mutation* technique is utilised here as well since it maintains the properties of the greedy tour relatively well compared to other methods. Please see the above section for more details.

The basic algorithm suffers from particles getting stuck in local optima frequently, and this is partly due to the velocity vectors becoming redundant vectors such as: [[1, 5], [5, 1]] or [ ]. These vectors undesirably keep the particles stationary for a prolonged period of computation, so the particles' velocity was recreated randomly whenever it remains in the same position after the prior velocity vector was applied. The magnitude of this vector can be controlled via the hyperparameter ***confusion*** and is typically set to 1.

The final enhancement uses ideas from the genetic algorithm to mutate a particle when it is stationary (due to a redundant velocity vector), AND if it is at the best position found thus far. The particles are mutated into a random tour to allow for exploration of completely different regions of the search space, before they gradually make their way back to the current best position (hopefully discovering new and improved tours along the way). The probability of a mutation occurring is controlled by the hyperparameter ***mutation_prob*** and is typically set to 0.2.

DESCRIPTION OF ALGORITHM ONLY IF THE ALGORITHM IS NOT COVERED IN LECTURES

Description of *non-standard* Algorithm A:

Description of *non-standard* Algorithm B: