Psueudo code for complete_notebook.ipynb cell wise

Cell 1:

- We create a session object for making HTTP requests.
- We read patient and MSU locations from CSV files.
- We extract relevant columns (latitude and longitude).
- We build KD-trees for efficient spatial queries.

Cell 2:

- Define a fallback function to return a high travel time in case of retries exhaustion.
- Define a function for calculating cached travel time between two coordinates.
- OSRM API endpoint for routing.
- Format coordinates for the API request.
- Construct the full URL for the API request.
- Use the global session object for making the request.
- Adding a timeout for good measure.
- Check if the response contains valid routing data.
- Extract travel time from the first (and only) route.
- Convert travel time to minutes.
- Return a high value or handle the error appropriately if no route is found.
- Calculate travel time based on distance and speed assumption.
- Define a list to store indices of rendezvous locations within 30 minutes from MSU.
- Loop through rendezvous_locations and calculate travel time to MSU.
- Check if travel time is within 30 minutes and not equal to 0.
- Add index to min30_indices list.
- Create a DataFrame with rendezvous locations within 30 minutes from MSU and save to CSV.

Cell 3:

Here's the provided code snippet with comments:

- Set the size of the population for the genetic algorithm.
- Define the number of sites.
- Specify the number of generations for the genetic algorithm.
- Determine the size of the tournament selection.
- Set the rate of mutation for the genetic algorithm.
- Set the rate of crossover for the genetic algorithm.
- Specify the output path for saving files.
- Check if the output directory exists, and create it if it doesn't.
- Create a DataFrame with the specified parameters.
- Save the DataFrame to a CSV file with the best parameters.

Cell 4:

• Initialize population
function initialize_population():
    return a list of randomly sampled chromosomes from the range of rendezvous locations

• Optimize the fitness calculation to sum of only nearby 10 patients
function calculate_population_fitness(population):
    create an empty list population_weights
    for each chromosome in population:
        create a KDTree for the chromosome
        find the nearest rendezvous for each patient
        find the nearest MSU for each patient
        find the distance from each rendezvous to its nearest MSU
        calculate the weight of the chromosome
        add the weight to population_weights
    return population_weights

• Crossover two parents when sampling is less than crossover_rate
function crossOver(parent1, parent2, crossover_rate):
    if a random number is less than crossover_rate:
        select a crossover point
        perform crossover at the selected point
        return the resulting children
    else:
        return the parents unchanged

• Mutate each gene of chromosome when sampling for that gene is < mutation_rate
function mutate(chromosome, mutation_rate):
    create a new_chromosome
    for each gene in chromosome:
        if a random number is less than mutation_rate:
            replace the gene with a randomly sampled gene within 30 minutes
        else:
            keep the gene unchanged
    return the mutated chromosome

• Perform tournament selection to create the next generation
function tournament_selection(population):
    create an empty list one_gen
    repeat until the size of one_gen is equal to half the size of population:
        randomly select individuals for a tournament from the population
        calculate the fitness of the tournament participants
        select the two best individuals from the tournament
        perform crossover and mutation on the selected individuals
        add the resulting children to one_gen

```
    return one_gen

function main():
    initialize the initial population
    create empty lists to store fitness scores and best chromosomes for each generation
    repeat for a fixed number of generations:
        calculate the fitness of the current population
        store the current population's fitness score
        select the best chromosome from the current population
        store the best chromosome and its fitness score
        create the next generation using tournament selection
        if the fitness of the next generation is worse than the current:
            replace the current population with the next generation
    return the final population, fitness scores, best chromosomes, and their fitness scores

function handle_results(last_gen):
    create a DataFrame of the best rendezvous sites
    save the DataFrame to a CSV file
    calculate nearest rendezvous for each patient and nearest MSU for each rendezvous
    calculate travel time between rendezvous and MSU
    save the results to CSV files
```

Cell 5:

- Run the Genetic algorithm

Cell 6:

- Generate the results and save them to CSV files

Cell 7:

• Plotting each generation's fitness score

```
function plot_fitness_scores(fitness_scores):
    plot the fitness scores using a line plot
    set the title of the plot as 'Each_gen_fitness_score'
    set the label for the x-axis as 'generation_no:'
    set the label for the y-axis as 'fitness score'
```

Cell 8:

• Create scatter plot with patient locations
• Specify latitude and longitude from patient_locations array
• Set marker color sequence to green
• Store in variable fig

• Add scatter plot for rendezvous locations on the same figure
• Specify latitude and longitude from rendezvous_locations array
• Set marker color sequence to blue
• Extract the first trace and add it to the figure

• Add scatter plot for MSU location on the same figure
• Specify latitude and longitude from msu_location array
• Set marker color sequence to yellow
• Extract the first trace and add it to the figure

• Add scatter plot for the last rendezvous location visited
• Specify latitude and longitude from rendezvous_locations array, using last index from c
• Set marker color sequence to red
• Extract the first trace and add it to the figure

• Update layout settings
• Set autosize to False
• Set width to 1100 and height to 800
• Set background color to LightSteelBlue

• Display the figure
• Show the figure