ARTIFICIAL INTELLIGENCE

ENCS3340

Anas Dawoud 1220523

Baraa said 1220280

Section 1

Instructor: Dr. Yazan Abu Farha

2025

Abstract :

In this project, you will simulate and optimize the operations of a local package delivery shop. The goal is to design algorithms and strategies to efficiently manage delivery routes and resource allocation, minimizing operational costs. Also we use for that two algorithm to solve the problem, Genetec Algorithm , and Simulating Annealing Algorithm.

Contents

Simulated Annealing: 4

Overview: 4

Problem formulation: 4

Genetic algorithm: 5

Overview 5

Problem formulation: 5

Data Structure: 5

Population : 5

Crossover : 6

Mutation : 6

Cases: 7

Case 1: 7

Package: 7

Vehicles: 7

GA result: 7

Plot GA result: 8

SA result: 8

Plot SA result: 8

Case 2: 9

Package: 9

Vehicles: 9

GA result: 9

Plot GA result: 9

SA result: 10

Plot SA result: 10

Simulated Annealing:

Overview:

Simulated Annealing is a local search algorithm used in Artificial Intelligence for optimization problems. It explores the state space by randomly selecting the next possible state. It keeps track of the best solution found so far. Unlike greedy algorithms, Simulated Annealing allows *downhill moves*—transitions to worse states—to avoid getting stuck in local optima.

The decision to accept a worse state is based on two parameters:

- **ΔE**: The change in energy (or cost) between the current and the new state.
- **T**: The temperature, which gradually decreases over time.

At high temperatures, the algorithm is more likely to accept worse moves, allowing broader exploration. As the temperature decreases, it becomes more conservative, favoring only better or equal states. This gradual cooling increases the chances of finding a global optimum.

Problem formulation:

We have a set of packages and vehicles stored in separate files. Each package includes an ID, priority, weight, and (x, y) coordinates, while each vehicle has an ID and a maximum capacity. The goal is to assign packages to vehicles without exceeding their capacity. Packages are appended to vehicles based on a chosen strategy. The final structure after distribution will be like this:

[[ID_{v1},remaining capacity ,[ID_{px},priority,weight,x_co,y_co]..][ID_{v2},remaining capacity ,[ID_{py},priority,weight,x_co,y_co]..]..] .

The algorithm begins by generating an initial state, randomly selecting packages from the package list and assigning them to available vehicles. To explore neighboring solutions, the next state is derived from the current one by swapping two packages between randomly selected vehicles. Additionally, every 20 iterations, the algorithm resets the state using the same function used to generate the initial state, adding variability to the search.

The objective function evaluates each state based on two key factors: **path cost** and **priority cost**. A *lower path cost* indicates a more efficient delivery route, while a *higher priority cost* reflects better alignment with package importance. If the new state's performance (i.e., lower path cost and higher priority cost) surpasses the current best, it replaces it as the new best state. Otherwise, the algorithm may still accept a worse state based on the energy difference (ΔE) and the current temperature (T), enabling exploration of suboptimal solutions in earlier stages.

Initially, the temperature **T** is set to 1000 and is reduced by multiplying it by 0.9 every 100 iterations to gradually decrease the likelihood of accepting worse states. The energy function is defined as:

$$E=\text{path cost} + \text{priority cost}$$

and the change in energy is:

$$\Delta E= E_{next} - E_{current}$$

The acceptance probability for a worse state is given by:

$$P= e^{\Delta E /T}$$

with the goal of minimizing ΔE over time.

Genetic algorithm:

Overview

A genetic algorithm maintains a population of candidate solutions for the problem at hand, and makes it evolve by iteratively applying a set of stochastic operators. successor states are generated as variations of two parent states, not only one. Genetic Algorithms follow the idea of SURVIVAL OF THE FITTEST - Better and better solutions evolve from previous generations until a near optimal solution is obtained. mutation provides an additional random element.

Problem formulation:

Data Structure:

We have a set of packages and vehicles stored in separate files. Each package includes an ID, priority, weight, and (x, y) coordinates, while each vehicle has an ID and a maximum capacity. The goal is to assign packages to vehicles without exceeding their capacity. Packages are appended to vehicles based on a chosen strategy. The final structure after distribution will be like this:

[[ID_{v1},remaining capacity ,[ID_{px},priority,weight,x_co,y_co]..][ID_{v2},remaining capacity ,[ID_{py},priority,weight,x_co,y_co]..]..] .

Population :

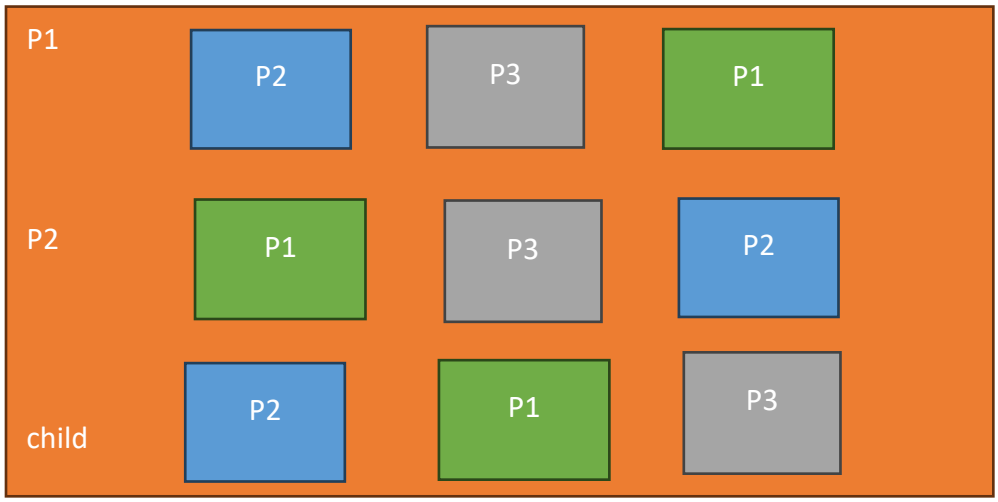
In our project the population size = 100 that mean there is 100 state generated randomly where each state has that format that we show above.

Fitness and selection function :

We evaluate the cost of each state using the following objective function: **Cost = (Priority Cost × 1000) – Path Cost**, where a higher cost indicates a better solution. To guide the selection process, we randomly sample five states from the population and choose the one with the highest cost, representing the best fit among the sample. This sampling and selection procedure is repeated 50 times to maintain diversity and increase the likelihood of identifying an optimal or near-optimal solution.

Crossover :

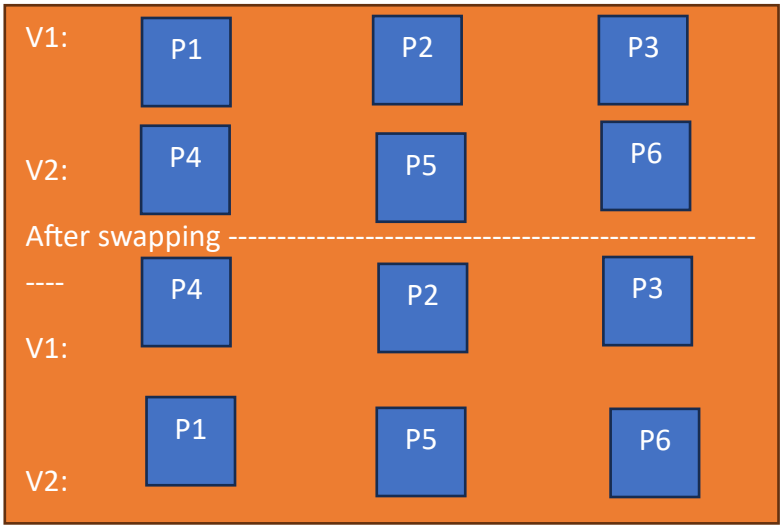
From the selected states, we randomly choose two parents. We then extract the list of packages from both parents and begin creating a child by alternately taking packages from each parent—starting with the first package from parent 1, then the first from parent 2, and so on. If a package has already been added to the child, it is skipped in future iterations to avoid duplication. For example, if parent 1 has the sequence [P2, P3, P1] and parent 2 has [P3, P2, P1], the resulting child will include packages in the following order: P2 → P3 (remove duplicate) → P1 (remove duplicate), resulting in [P2, P3, P1].



Mutation :

This operation is performed at a mutation rate of 0.01, meaning that for every 100 generated children, one mutation is applied. The mutation process involves randomly selecting two vehicles and swapping one package from each, introducing variability into the population to help explore new potential solutions.

Ex:



Cases:

Case 1:

Package:

cases.txt

```
1#1#5#34#46
2#1#5#20#54
3#1#5#67#63
```

Vehicles:

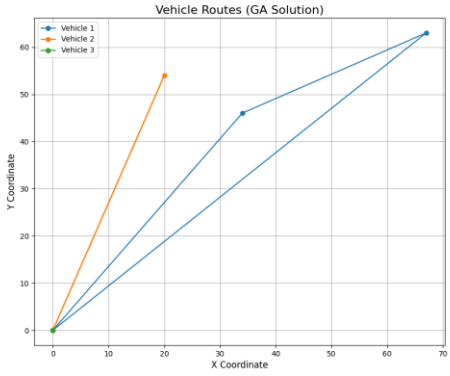
vehicles.txt

```
1#10
2#5
3#2
```

GA result:

```
best solution that Genetic solved : [[1, 0, [1, 1, 5, 34, 46], [3, 1, 5, 67, 63]], [2, 0, [2, 1, 5, 20, 54]], [3, 2]]
path cost =  301.4596468852926
priority cost = 12.5
```

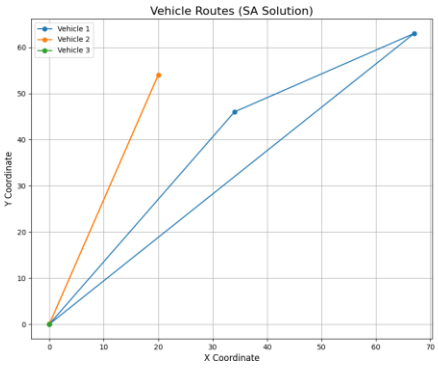
Plot GA result:



SA result:

```
best solution that Simulating Aneling solved : [[1, 0, [3, 1, 5, 67, 63], [1, 1, 5, 34, 46]], [2, 0, [2, 1, 5, 20, 54]], [3, 2]]
path cost = 301.4596468852926
priority cost = 12.5
```

Plot SA result:



Case 2:

Package:

cases.txt

```
1#1#5#3#3
2#2#5#3#3
3#3#5#3#3
```

Vehicles:

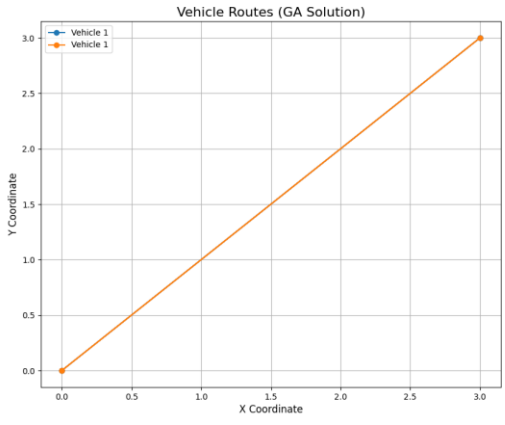
vehicles.txt

```
1#10
```

GA result:

```
best solution that Genetic solved : [[1, 0, [1, 1, 5, 3, 3], [2, 2, 5, 3, 3]], [1, 5, [3, 3, 5, 3, 3]]]
path cost =  16.97056274847714
priority cost = 8.0
```

Plot GA result:



SA result:

```
best solution that Simulating Aneling solved : [[1, 0, [1, 1, 5, 3, 3], [2, 2, 5, 3, 3]], [1, 5, [3, 3, 5, 3, 3]]]
path cost = 16.97056274847714
priority cost = 8.0
```

Plot SA result: