PROPOSED SYSTEM

MODULE-1

TRAIN TIMETABLE OPTIMIZATION USING GENETIC ALGORITHM

The train timetable optimization aims to enhance efficiency, safety, and service quality within the Indian railway network. The approach involves encoding train schedules and routes into genetic algorithm populations, defining a fitness function to evaluate solutions based on objectives like minimizing delays, optimizing throughput, and ensuring safety. Genetic operators such as selection, crossover, and mutation are applied to generate improved solutions.

The key objectives include the development of a robust encoding scheme to represent train schedules, routes, and relevant factors as individuals in the genetic algorithm population. This encoding will capture essential information such as departure times, routes, and speeds, laying the foundation for comprehensive optimization. A crucial aspect of the system is the formulation of a fitness function that meticulously evaluates solutions based on defined objectives. These objectives encompass minimizing delays, maximizing train throughput, optimizing travel time, and ensuring safety by avoiding collisions. The assignment of higher fitness values to solutions aligning with these objectives will drive the evolution of more efficient and secure railway systems.

The genetic operators, including selection, crossover, and mutation, play a pivotal role in generating new candidate solutions. Through these mechanisms, the system favours the survival of the fittest solutions, combining genetic information from successful parents to create potentially improved offspring. The introduction of mutation injects an element of exploration, facilitating the discovery of novel solutions.

The Genetic algorithm implemented as follows

**1. Population:** The `Population` class represents a collection of schedules. Each schedule within the population is an individual solution to the optimization problem. The size of the population is specified by the `POPULATION\_SIZE` constant.

**2. Schedule**: The `Schedule` class represents a single timetable solution. It contains a list of classes, each representing a scheduled train journey. The fitness of each schedule is calculated based on conflicts such as platform capacity violations and overlapping schedules.

**3. Genetic Algorithm:**

Initialization: The initial population is created with random schedules generated using the `initialize` method of the `Schedule` class.

Selection: The `Tournament Selection` method is used to select schedules from the population for crossover. This method randomly selects a subset of schedules (determined by `TOURNAMENT\_SELECTION\_SIZE`) and selects the best schedule from this subset based on fitness.

Crossover: The `crossover population` method selects elite schedules directly, then iteratively selects two parent schedules from the population using tournament selection. These selected parent schedules undergo crossover to produce new schedules with characteristics from both parents.

Mutation: After crossover, the resulting population undergoes mutation with a probability determined by the `MUTATION\_RATE`. Mutation introduces random changes to individual schedules to maintain diversity and explore new regions of the search space.

**4. Fitness Function**

The fitness function is defined within the `Schedule` class. It evaluates each schedule based on conflicts such as platform capacity violations and overlapping schedules. The fitness is calculated as the reciprocal of the total number of conflicts plus one, ensuring that higher fitness values correspond to better schedules.

**5. Elitism**

The top performing schedules (determined by `NUMB\_OF\_ELITE\_SCHEDULES`) are preserved in each generation without undergoing crossover or mutation. This ensures that the best solutions found so far are retained in subsequent generations.

Overall, the genetic algorithm iteratively evolves a population of schedules over multiple generations, with the aim of finding optimal or near-optimal solutions to the train timetable optimization problem.

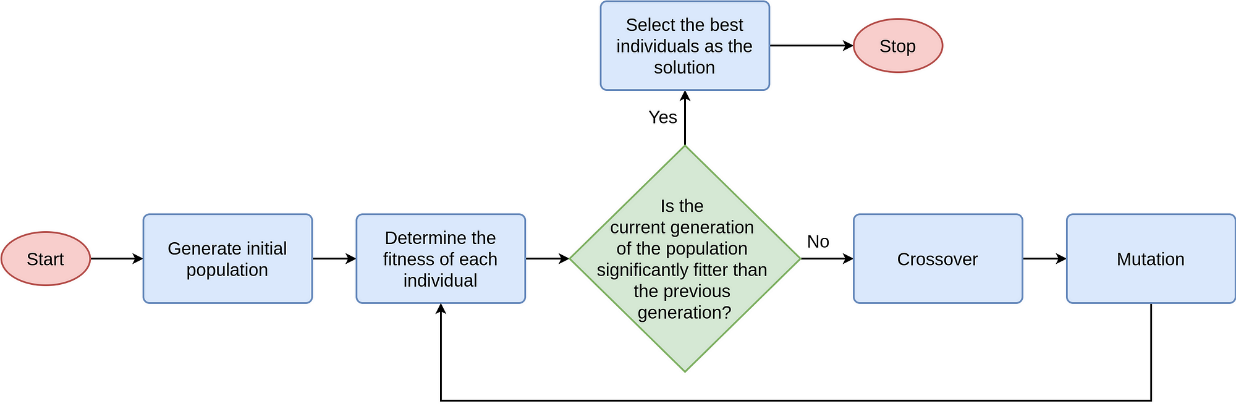


Fig1: Genetic algorithm