META HEURISTIC OPTIMIZATION TECHNIQUES

**Name: MITHUNSANKAR S**

**Course: MSc AIML, 4th Year**

**Institution: Coimbatore Institute of Technology**

**Subject Code: 19MAM83**

Title: Solar Power Plant Placement Optimization using Particle Swarm Optimization (PSO)

**Problem Statement:**An organization aims to identify optimal locations to set up solar power plants across Tamil Nadu. The goal is to maximize solar energy generation while minimizing land and infrastructure costs. To solve this, we apply Particle Swarm Optimization (PSO) to score and select the best sites based on:

* Solar Irradiance
* Land Cost
* Proximity to Infrastructure

**Tech Stack Used:**  
Frontend: React.js + Tailwind CSS  
Backend: FastAPI (Python)  
Mapping: Leaflet.js + React Leaflet  
Algorithm: Particle Swarm Optimization (PSO)  
Data Format: CSV

**Functionalities Implemented:**

* Upload custom dataset (CSV with location, irradiance, cost, infra)
* Set number of sites and weight preferences for each factor
* Run optimization via PSO
* Display optimized sites in a styled table
* Visualize selected sites on an interactive map with heatmaps
* Download optimization results as CSV.

**Algorithm**:  
 **Input Parameters:**

* Number of sites to select (N)
* Weights:
  + Solar Irradiance (W1)
  + Land Cost (W2)
  + Proximity to Infrastructure (W3)
* Dataset with fields: latitude, longitude, solar\_irradiance, land\_cost, distance\_to\_infra

 **Normalize Features:**

* Normalize each value to a range of 0–1:
  + norm\_irr = (irradiance - min) / (max - min)
  + norm\_land = 1 - ((cost - min) / (max - min)) *(inverse because lower cost is better)*
  + norm\_infra = 1 - ((distance - min) / (max - min)) *(inverse because closer is better)*

 **Compute Fitness Function:**

fitness=(norm\_irr×W1)+(norm\_land×W2)+(norm\_infra×W3)

* Returns a score between 0 and 1
* Higher score = better site

 **Initialize Swarm:**

* Generate particles (candidate sites) randomly from dataset
* Assign random velocity vectors
* Initialize:
  + pBest = personal best for each particle
  + gBest = global best across all particles

 **Repeat for multiple iterations:**

* For each particle:
  + Calculate fitness
  + Update pBest if current fitness is better
* Update gBest with best fitness from swarm

 **Update Position and Velocity:**

* For each particle, update:
* vi=w⋅vi+c1⋅r1⋅(pBest−xi)+c2⋅r2⋅(gBest−xi)
* xi​=xi​+vi​
* Where:
  + w = inertia weight
  + c1, c2 = cognitive and social acceleration coefficients
  + r1, r2 = random values between 0 and 1

 **Convergence:**

* Repeat until stopping criteria (e.g., fixed iterations or minimal change in gBest)
* Swarm stabilizes around optimal regions

 **Select Best Sites:**

* Sort all particles by fitness score
* Select top N sites as the final output

 **Output:**

* Optimized site details in a results table
* Locations visualized on a map with markers and heatmaps
* Option to download results as CSV

**Main Code Snippet**:  
# Normalize values between 0 and 1

def normalize(value, min\_val, max\_val):

return (value - min\_val) / (max\_val - min\_val) if max\_val != min\_val else 0.0

# Compute normalized fitness

def compute\_fitness(site, weights, min\_max):

norm\_irr = normalize(site['solar\_irradiance'], \*min\_max['irradiance'])

norm\_cost = normalize(site['land\_cost'], \*min\_max['land\_cost'])

norm\_infra = normalize(site['distance\_to\_infra'], \*min\_max['infra'])

fitness = (

norm\_irr \* weights['irradiance'] +

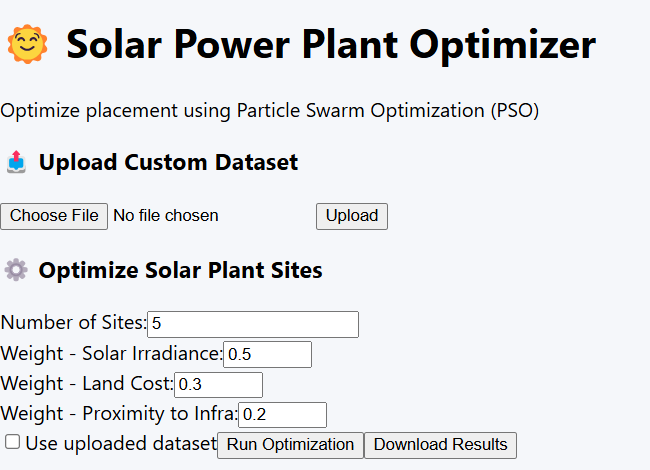
(1 - norm\_cost) \* weights['land\_cost'] +

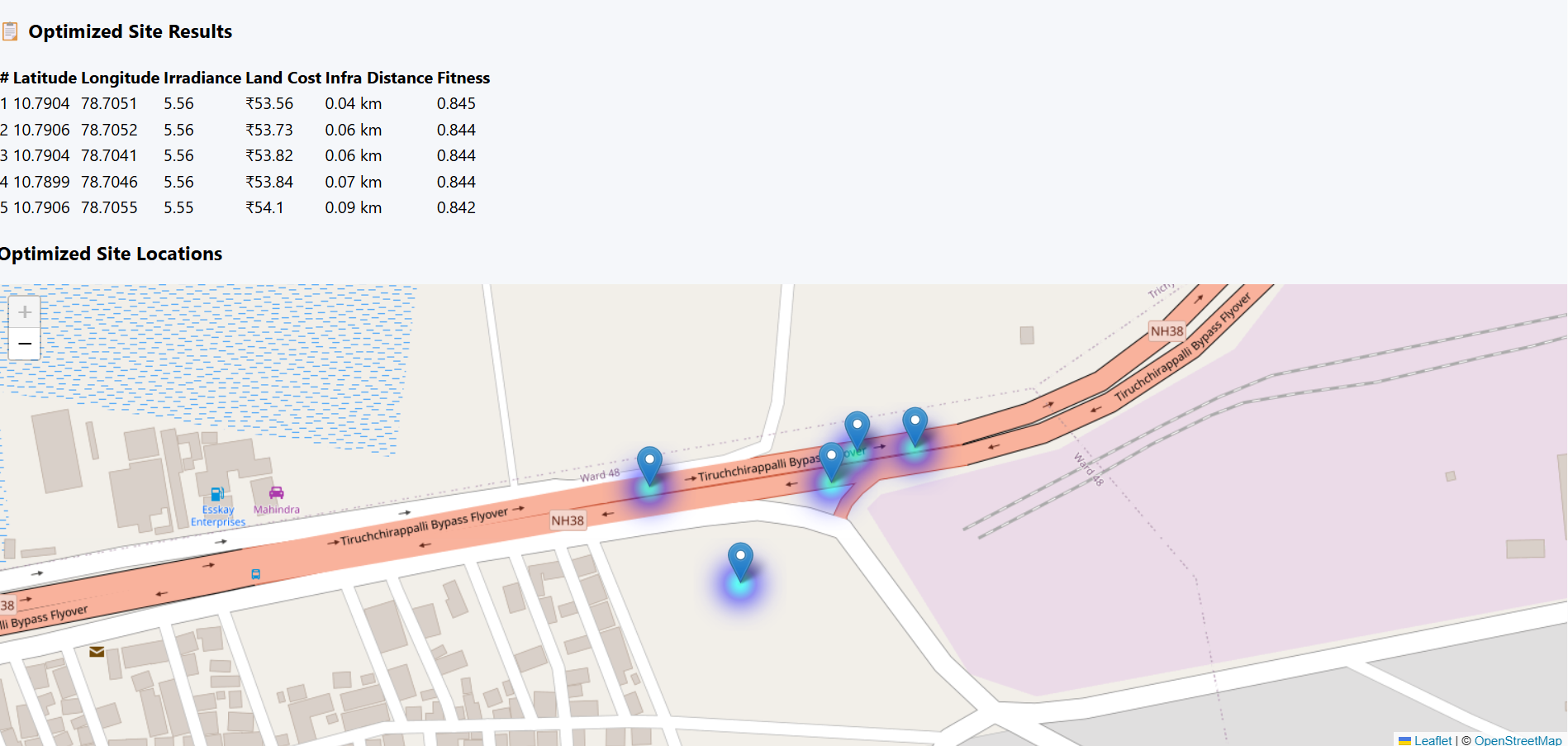
(1 - norm\_infra) \* weights['proximity']

)

return fitness

**Output Screenshot:**

****

****

**User Interface / Interactions:**

* Clean dashboard-style layout
* Forms for inputting weights and uploading CSV
* Result table with zebra-striping and fitness highlights
* Leaflet map with markers, heatmap, and popups

**Inference / Insights:**

* PSO effectively prioritizes sites based on solar suitability and cost factors
* Users can dynamically control weight preferences and regenerate optimal results
* Visual interface enhances interpretability of model decisions