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:

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

- 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:
- $v_i = w \cdot v_i + c1 \cdot r1 \cdot (pBest - x_i) + c2 \cdot r2 \cdot (gBest - x_i)$
- $x_i = x_i + v_i$
- 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:

Solar Power Plant Optimizer

Optimize placement using Particle Swarm Optimization (PSO)

Upload Custom Dataset

Choose File | No file chosen | Upload

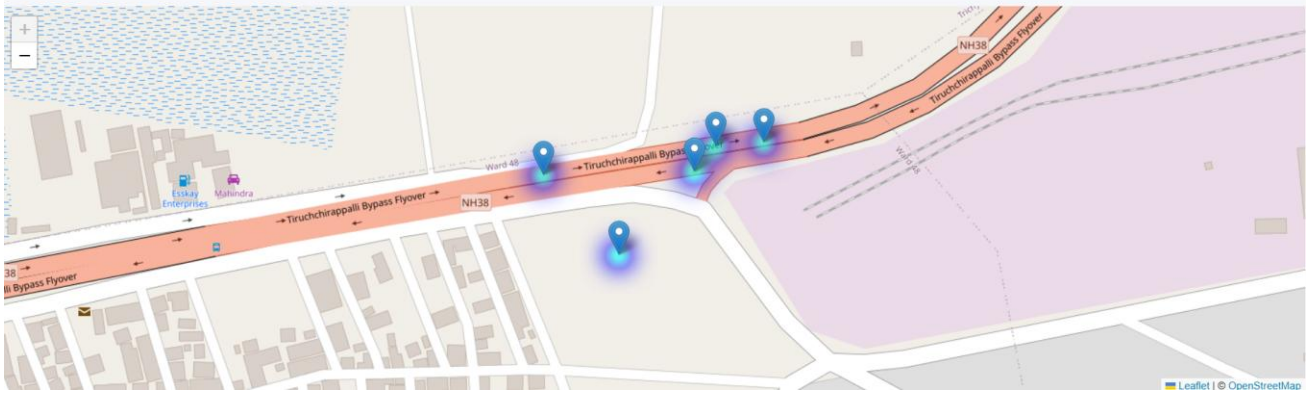
Optimize Solar Plant Sites

Number of Sites: 5
Weight - Solar Irradiance: 0.5
Weight - Land Cost: 0.3
Weight - Proximity to Infra: 0.2
Use uploaded dataset | Run Optimization | Download Results

Optimized Site Results

#	Latitude	Longitude	Irradiance	Land Cost	Infra Distance	Fitness
1	10.7904	78.7051	5.56	₹53.56	0.04 km	0.845
2	10.7906	78.7052	5.56	₹53.73	0.06 km	0.844
3	10.7904	78.7041	5.56	₹53.82	0.06 km	0.844
4	10.7899	78.7046	5.56	₹53.84	0.07 km	0.844
5	10.7906	78.7055	5.55	₹54.1	0.09 km	0.842

Optimized Site Locations



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