Solar Panel Cleaning Robot Scheduling with Energy Constraints

Introduction

The increasing deployment of solar farms worldwide has made maintaining panel efficiency a critical task. Cleaning solar panels regularly prevents efficiency loss due to dust and dirt accumulation. Autonomous cleaning robots are a sustainable solution, but their operation is constrained by battery capacities and energy management.

In this problem, you are given a solar farm with multiple panels and a fleet of cleaning robots. Each robot's battery is recharged partially by solar energy harvested during the day. The challenge is to schedule the robots' cleaning routes and charging times to maximize the total weighted cleanliness score of the panels cleaned within the limited working hours.

Task

Given a description of solar panels, cleaning robots, and energy constraints, plan the cleaning schedule for each robot — including which panels to clean, in what order, and when — to maximize the total importance score of all cleaned panels within the available working time. The schedule must respect robot battery capacities, energy consumption for travel and cleaning, and allow for energy recharging from solar power during idle or cleaning times.

Problem Description

Solar Panels

There are N solar panels indexed from 0 to N-1. Each panel has an importance score reflecting how critical it is to keep the panel clean to maintain optimal energy production. Cleaning a panel restores its efficiency.

Robots

There are **M** cleaning robots indexed from 0 to M-1. Each robot:

- Has a battery with limited energy capacity.
- Consumes energy both when traveling between panels and when cleaning.
- Starts at a predefined initial position (e.g., charging station).

Time and Energy Constraints

- The robots operate within a fixed total time **T** days (1 day = 24 hours).
- Moving from panel i to panel j takes a known amount of time.
- Cleaning a panel requires a fixed cleaning time.
- Energy consumption rates for moving and cleaning are known for each cell.
- Robots cannot operate if their battery energy is depleted; they must recharge to continue.
- Robots may recharge slowly while idle, but not during cleaning. When a robot's energy is fully depleted, it must stop cleaning and wait to recharge before resuming operations.
- Robots cannot clean the same panel simultaneously.

Cleaning Rules

- Each panel can be cleaned multiple times, but the importance score for that panel is counted only once towards the total score.
- Cleaning of a panel must be completed within the total available time **T**.
- Robots may wait or recharge between tasks but must finish all activities within **T**.

Objective

Schedule routes, cleaning, and charging times for all robots to maximize the total sum of importance scores of all panels cleaned within the working time T.

Example

For **example**, if

Robot 0 requires **2 hours to fully recharge** before starting cleaning,

Robot 1 requires **3 hours to fully recharge**,

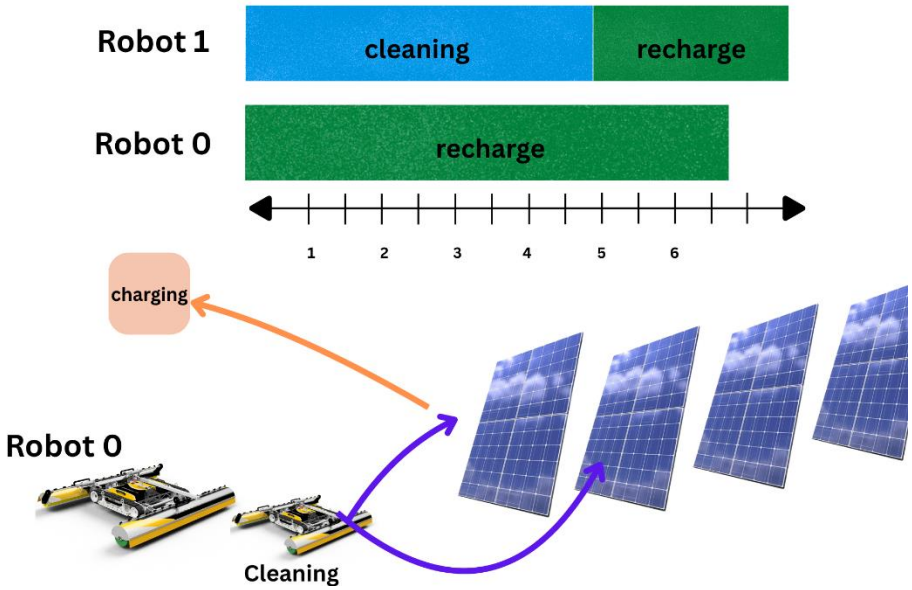
Robot 1 starts recharging **before** Robot 0,then

The recharge process of Robot 1 starts on **hour 0**, and finishes on **hour 3** (3 hours total),

Robot 1 can start cleaning panels beginning at **hour 3** (immediately after recharge completes),

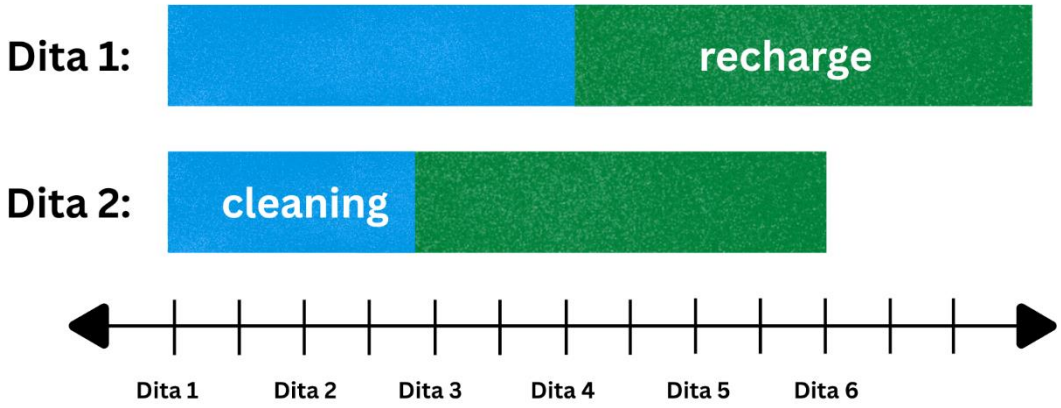
The recharge of Robot 0 starts on **hour 3** (right after Robot 1 finishes) and finishes on **hour 5** (2 hours total),

Robot 0 can start cleaning panels beginning at **hour 5**.



While Robot 1 is cleaning, Robot 0 is recharging. After Robot 0 finishes recharging, it begins cleaning. Robots cannot clean the same panel simultaneously, and each cleaning task consumes battery and time.

Solar Panel Cleaning



The process for solar panel cleaning would look like this:

- All panels are cleaned by the robots in the cleaning facility
- The entire process of moving robots to panels, cleaning them, and returning to charging stations happens over a series of days.

- Each robot can clean a specific number of panels per day, and there's a maximum number of panels that can be cleaned per day by each robot.
- Time constraints: The robots must also manage their energy usage by recharging, which can affect the number of panels they can clean each day.

Example Timeline:

Robot 0 needs to clean 5 panels.

Robot 0 can clean 2 panels per day.

Robot 0 completes its signup/recharge process on day 1.

Then:

Day 2: Robot 0 can clean 2 panels.

Day 3: Robot 0 can clean 2 panels again.

Day 4: The one remaining panel can be cleaned.

Robot 0 has 5 panels to clean and can clean 2 panels per day.

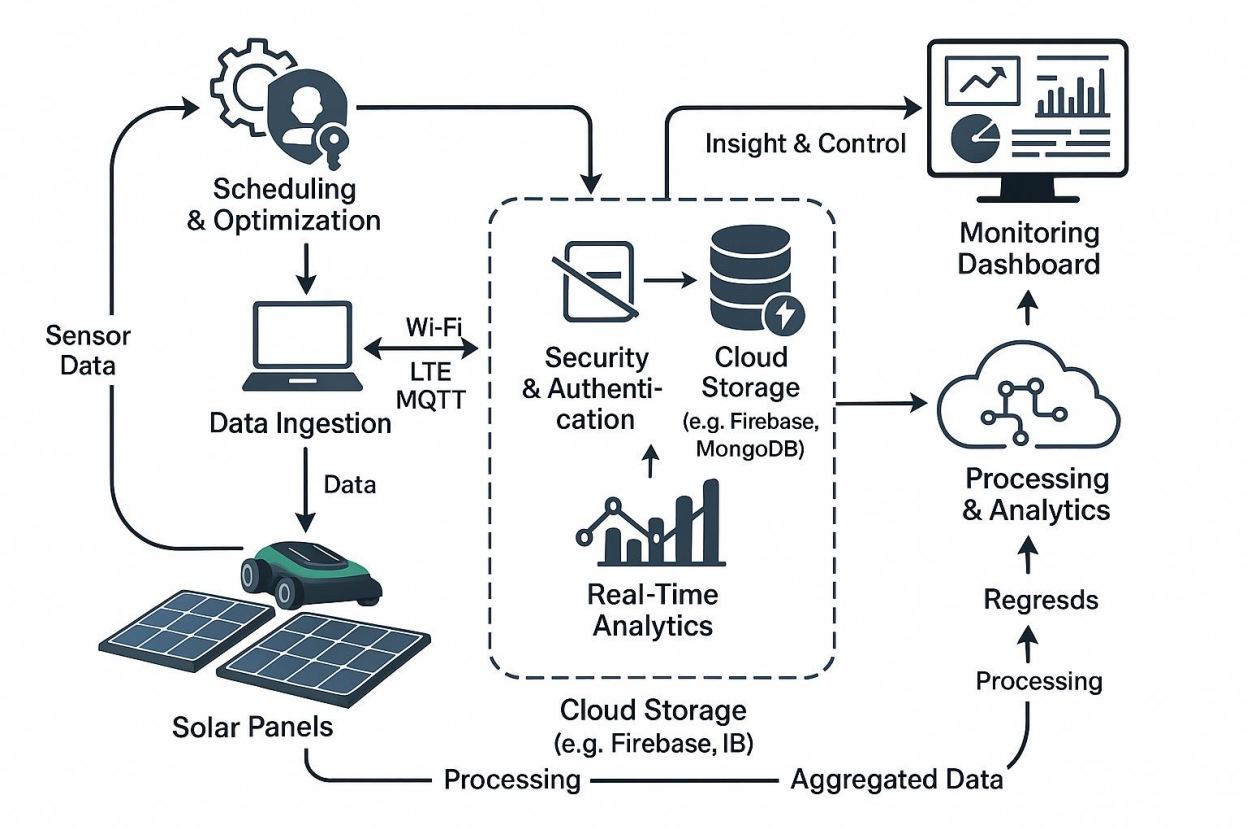
Day 1*: The robot starts its recharge process*

Day 2*: Robot 0 can clean 2 panels.*

Day 3*: Robot 0 can clean 2 more panels.*

Day 4*: The remaining 1 panel is cleaned.*

Cloud Logic and Data Flow (Implementation Plan)



This project is designed with a future-ready architecture that allows for cloud-based scheduling, monitoring, and optimization of solar panel cleaning robots. The following outlines the planned logic for how data will be acquired, transmitted, stored, and processed in a real-world cloud deployment:

Robot to Cloud Communication

Each robot is equipped with sensors and communication modules (Wi-Fi/LTE). After performing cleaning tasks, robots transmit data in real time, including panel ID, cleaning status, energy level, and timestamp.

Secure Transmission

Data is serialized in JSON format and securely sent to a cloud endpoint using HTTPS or MQTT. All communication is authenticated using tokens or API keys to ensure integrity and access control.

Cloud Storage

Received data is validated and stored in a scalable cloud database system, such as Firebase Realtime Database or MongoDB Atlas. Each cleaning log entry is stored with metadata for traceability and analysis.

Data Processing and Analytics

Cloud functions or stream processing engines like Apache Spark are used to analyze the data. This includes identifying frequently cleaned panels, calculating energy efficiency per robot, and detecting idle time or recharge frequency.

Web Dashboard Integration

A user interface built with frameworks like React or Vue displays the system's status, cleaning schedules, panel coverage, and energy metrics in real time. Users can monitor robot performance and get automated insights.

Feedback and Optimization Loop

Based on the analytics, the system can recommend improved schedules or dynamically adjust robot assignments. These adjustments are written back to the database or directly sent to the robots for next-day execution.

Input Data Format

File Format

Each input data set is provided in a plain text file with ASCII characters. The file follows the standard UNIX-style line endings (\n). When multiple numbers are given in one line, they are separated by a single space.

Input Format

The first line of the file contains:

N: The number of solar panels ($1 \le N \le 10^5$)

M: The number of robots ($1 \le M \le 10^5$)

T: The number of total available days ($1 \le T \le 10^5$)

This is followed by one line containing N integers: S_0, S_1, ..., S_(N-1), which represent the importance scores for each panel ($0 \le S_i \le 1000$).

Following this, there are M sections that describe each robot:

Each section contains:

1. The first line:

C: The cleaning capacity of the robot (maximum number of panels cleaned per day)

R: The recharge time for the robot (time in hours required to recharge after cleaning)

E: The energy capacity of the robot (maximum energy the robot can carry)

2. The second line:

The initial position of the robot (charging station ID) (0 ≤ position ≤ N-1)

After the robots' descriptions, each panel's cleaning time is provided on a separate line.

6 2 10 N = 6: There are 6 panels.

10 5 8 6 4 7 M = 2: There are 2 robots.

2 3 10 T = 10: The total available time is 10days.

0 4 Panels' importance scores are [10, 5, 8, 6, 4, 7].

2 3 12

1 5

2 3 1 4 2 3

Input Structure

- First line:
  - N: Number of solar panels (1 ≤ N ≤ 10^5)
  - M: Number of robots (1 ≤ M ≤ 10^5)
  - T: Total time in days (1 ≤ T ≤ 10^5)
- Second line: N integers for importance scores: S_0, S_1, ..., S_(N-1)
- Next M robot descriptions:
  - Line 1 (per robot):
    - C: Cleaning capacity (panels/day)
    - R: Recharge time (in hours)
    - E: Energy capacity
  - Line 2: Initial position (charging station ID, 0 ≤ position ≤ N-1)
- Next N lines: Cleaning time per panel (in hours), one per line:

2 3 1 4 2 3

This means:
- Panel 0: 2 hours
- Panel 1: 3 hours
- ...
- Panel 5: 3 hours

Score Calculation

The goal is to maximize the total sum of importance scores from all solar panels that are cleaned at least once during the allowed time **T**. If a panel is cleaned multiple times, its score is still counted only once.

For example, if:
- Panel 0 has an importance score of 10,
- Panel 1 has a score of 5,
- Panel 2 has a score of 8,

And only panels 0 and 2 are cleaned during time **T**, then the total score is:
> **10 (Panel 0) + 8 (Panel 2) = 18**

Panels cleaned more than once do **not** add additional score.

Definitions:

- Let P={0,1,…,N−1} be the set of panel indices, where N is total number of panels.
- Let I(p) be the importance score of panel p, i.e. panel_importance[p].
- Let there be R robots, each with a schedule $Sr = [t1, t2, \dots, tkr]$ where each task t_i has:
    - an action (like "clean")
    - a panel cleaned p_i

- Let C⊆PC be the set of all cleaned panels by all robots:
  $C = \{p_i \mid \exists r, \exists t_i \in S_r, action(t_i) = \text{"}clean\text{"}\}$

  Base Score:

  The base score is the sum of importance values of all cleaned panels:

$$\text{BaseScore} = \sum_{p \in C} I(p)$$

Cluster Bonus:

For each robot r, sort the tasks s_r. Then, for consecutive tasks (t_{i-1}, t_i), check if the cleaned panels are adjacent:

$$\text{Adjacency}(p_{i-1}, p_i) = \begin{cases} 1 & \text{if } |p_i - p_{i-1}| = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{ClusterBonus} = B \times \sum_{r=1}^{R} \sum_{i=2}^{k_r} \text{Adjacency}(p_{i-1}, p_i)$$

Total Score:

$$\boxed{\text{TotalScore} = \text{BaseScore} + \text{ClusterBonus}}$$

Robot types	Move Energy	Clean Energy	Total Energy	Score
Clean Grid Gama	25	6	31	78
Helio Clean Omicron	11	88	99	158
Panel Patrol Beta	3	10	13	27
Robot Shine	7	53	60	120
Solar Sweep Alpha	7,157	88,700	95,857	6389599
Sun Trackers	5	36	41	45

Capacity and Energy Considerations

- Cleaning Capacity: Limits how many panels a robot can clean per day.
- Energy Use: Robots consume energy during movement and cleaning.
- Recharge Time: Robots must recharge when depleted, during which no cleaning can occur.

- Duplicate Cleaning: Only the **first cleaning** of a panel counts toward the score.

Detailed Timeline Example

- Robot 0:
- Cleaning capacity: 2 panels/day
- Energy capacity: 10 (enough for 2 panels)
- Recharge time: 3 hours

- Robot 1:
- Cleaning capacity: 2 panels/day
- Energy capacity: 12 (enough for 3 panels)
- Recharge time: 3 hours

Timeline:

- Day 1: Robot 0 completes recharge.
- Day 2: Robot 1 cleans panel 0 and panel 4.
- Day 3: Robot 1 cleans panel 1.
- Day 4: Robot 0 cleans panel 2 and panel 3.
- Day 5: Robot 0 cleans panel 5.

- Robot 0 has a cleaning capacity of 2 panels per day, recharge time of 3 hours, and energy capacity of 10.
- Robot 1 has a cleaning capacity of 2 panels per day, recharge time of 3 hours, and energy capacity of 12.

- ## Instance Logic and Results
-
- Below are the descriptions and logic for each input instance in the code, along with the solution scores obtained from the latest run.
-
- ### 1. SolarSweep_Alpha_1.json
- **Logic:**
- A very large instance (your "big data" test) with a high number of panels and robots. Designed to stress-test the algorithm’s scalability and efficiency.

- To generate all this big data are generated by 'generate_large_instances.py'

-
- ---
-
- ### 2. PanelPatrol_Beta_2.json
- **Logic:**
- A diagonal line of 7 panels with increasing importance. Two robots start at opposite ends, each with different capacities and energy. The scenario is designed to test how robots can efficiently meet in the middle and maximize the score with minimal overlap.

- ### 3. CleanGrid_Gamma_3.json
- **Logic:**
- A compact grid of 12 panels arranged in a 2-column layout. Four robots start at different positions, each with unique cleaning capacities, recharge times, and energy capacities. The scenario tests efficient coverage of a small, dense grid with limited time and energy.

- ### 4. SunTrackers_Mu_4.json
- **Logic:**
- A smaller, focused scenario with a moderate number of panels and robots. The setup is intended to test the algorithm's performance on mid-sized, realistic solar farms.

- ### 5. RoboShine_Nu_5.json
- **Logic:**
- A 30-panel instance with panels arranged in a zig-zag pattern. Five robots with varied capacities, recharge times, and energy levels start at different strategic positions. The scenario tests coordination among robots with different strengths and the ability to cover a large area efficiently.
- ### 6. HelioClean_Omicron_6.json
- **Logic:**
- A large and diverse instance with 50 panels, each represented as an object with position, importance, and type (standard, fragile, high-efficiency). Six robots of different types (fast, standard, heavy, eco, mini) start at various positions, each with unique capacities and recharge profiles. This scenario encourages strategic assignment of robots to panels based on panel type and robot specialization.

Energy Model and Scheduling Formulas

To ensure robots operate efficiently within energy and time limits, we define the following model based on movement and cleaning operations:

1. Distance Between Panels

We assume the panels are placed linearly or in a grid layout. The distance between two panels i and j is calculated using the **Manhattan Distance**:

$$\text{Distance}(i, j) = |x_i - x_j| + |y_i - y_j|$$

2. Movement Energy

Each robot consumes energy when moving between two panels:

$$E_{\text{move}} = \text{move\_energy\_cost} \times \text{Distance}(i, j)$$

3. CleaningEnergy

Cleaning a panel uses a fixed amount of energy:

$$E_{\text{clean}} = \text{clean\_energy\_cost}$$

You may optionally define it as:

$$E_{\text{clean}} = \text{cleaning\_time}(p) \times \text{energy\_rate}$$

4. Total Energy Used

The total energy used by a robot during a session is:

$$E_{\text{total}} = \sum_{\text{moves}} E_{\text{move}} + \sum_{\text{cleans}} E_{\text{clean}}$$

5. Energy Recharge Rule

If the robot does not have enough remaining energy to move and clean:

$$E_{\text{left}} < E_{\text{move}} + E_{\text{clean}} \Rightarrow \text{Recharge required}$$

The robot recharges for recharge_time hours and resets its energy.

6. Time Accounting

The total time per robot includes:

- Travel time between panels: equal to the distance,
- Cleaning time per panel (from input),
- Recharge duration when energy is depleted.

7. Scheduling Output

Each robot maintains a log of:

- Cleaning actions,
- Panels visited,
- Cleaning start and end times,
- Positions at each action.

8. Energy Usage Report

For each robot, the system tracks:

- total_energy_used
- move_energy_used
- clean_energy_used