User Manual: Photovoltaic Panel Grouping Optimization and Reporting

Sirun Tang

1. Overview

This project consists of two Python scripts that streamline the process of organizing photovoltaic (PV) panels into optimized groups and generating human-readable reports:

* GA\_500\_panels.py

Applies a genetic algorithm (GA) to partition panels into series–parallel strings for maximum power output.

* print\_PV\_panel\_group\_string\_data.py

Reads the GA output and produces a concise text report summarizing each group and string.

1. Environment & Dependencies

* Python Version:

Tested with Python 3.7 or higher.

* Required Libraries

**Install via pip**: pip install numpy pandas

**numpy**: numerical computations

**pandas**: CSV reading/writing and data tables

1. Input Data: dataset1.csv

**Requirement:**

To ensure successful operation of the genetic algorithm for photovoltaic panel grouping, a properly formatted input dataset is required. The algorithm expects a comma-separated values (CSV) file named dataset1.csv, placed in the same directory as the main Python script.

**Format**: CSV, header row included.

**Required Columns** (case-sensitive):

The file must include a header row containing exactly the following four case-sensitive column names. Each row thereafter represents one panel’s key electrical characteristics.

UOC – Open-circuit voltage (Voc) in volts

ISC – Short-circuit current (Isc) in amperes

UMPP – Voltage at maximum power point (Vmp) in volts

IMPP – Current at maximum power point (Imp) in amperes

You may prepare the CSV file using either Excel or a text editor. Below is a minimal valid example:

Examples:

UOC, ISC, UMPP, IMPP

51.99,13.84,43.09,12.50

49.67,13.45,41.72,11.90

To ensure that the photovoltaic panel data can be correctly interpreted by the algorithm, the input values in dataset1.csv must follow strict formatting rules. Improper formatting may lead to runtime errors or incorrect optimization results.

Specifically, the following requirements must be satisfied:

All values must be numeric, and floating-point numbers should use a dot (.) as the decimal separator. Commas or other symbols are not permitted. The CSV file should not contain any extra spaces, invisible control characters, or non-standard delimiters between values. The content must not include any Chinese characters, units (such as “V” or “A”), or symbols copied from Excel formatting. Only plain numerical values are accepted.

These rules ensure compatibility with the program’s parser and facilitate seamless data preprocessing. Violations of these rules will result in the corresponding row being discarded automatically by the system during data loading. By adhering to these quality recommendations, users can significantly enhance the robustness and efficiency of the panel grouping algorithm.

1. Genetic Algorithm Script: GA\_500\_panels.py

* Functionality

**Purpose**: Partition dataset1.csv panels into G groups.

**Topology**:

The execution of the script follows a structured sequence, which includes data loading, population initialization, evolutionary optimization, and result output. A typical run completes the following steps:

Loading input data: The script begins by reading dataset1.csv, parsing each row into a panel object with attributes such as Voc, Isc, Vmp, and Imp. Estimating group count: Based on the number of panels, the algorithm estimates how many groups (each consisting of 5 strings) can be formed, assuming around 10 panels per string. Generating initial population: A set of valid individuals is generated. Everyone represents a full configuration of panels arranged in groups and strings. Running genetic evolution: Through multiple generations, individuals are selected, crossed over, and mutated to explore better layouts. Each generation outputs the best current fitness score. Saving output: Upon convergence, the best individual is exported to best\_individual.csv, with a power summary saved for comparison.

**Constraints**:

1. String voltage between 360 V and 400 V
2. String current consistency: max Imp ≤ min Imp × (1 + tol)

Objective: Maximize total array power at maximum power point.

* Key Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Default Value | Description |
| generations | 30 | Number of evolutionary iterations |
| G | 10 | Number of groups (can be computed automatically) |
| L | 6-12 | Panels per string |
| population\_size | 20 | Number of individuals per generation |
| mutation\_rate | 0.1 | Probability of mutation per individual |
| imp\_tol | 0.10 | Allowed current difference within each string (±10%) |
| voltage\_limit | 400/360 | Maximum/min allowed voltage for any string |
| voltage\_tol | 0.10 | Max allowed deviation between parallel string voltages |

* How to use it

To execute the optimization algorithm, simply open PyCharm and run the python code: GA\_500\_panels.py. And after getting two CSV documents, then run python code: print\_PV\_panel\_group\_string\_data.py.

* Outputs

**best\_individual.csv**:

Columns: Group, String, Position, ID, Voc, Isc, Vmp, Imp

**final\_voltage\_current\_summary.csv**:

Columns: Group, String, Voltage (V), Current (A), Power (W)

* Execution

**python GA\_500\_panels.py**:

Upon completion, the two CSV files appear in the script’s folder (or in output\_results/ if configured).

1. Reporting Script: print\_PV\_panel\_group\_string\_data.py

* Purpose and Functionality

This auxiliary script reads the output file best\_individual.csv generated by the main optimization program and produces a human-readable summary report. The report, saved as summary.txt by default, details the composition and performance of each group and string within the optimized PV layout.

Each string is evaluated on:

Number of panels

Minimum and maximum Imp

Total string voltage (sum of Vmp)

String power = (sum of Vmp) × min Imp

This report facilitates technical communication by providing a structured and concise breakdown of the optimization result.

* Usage

To run the reporting script, use the following command in your terminal:

python print\_PV\_panel\_group\_string\_data.py [INPUT\_CSV] [OUTPUT\_TXT]

**INPUT\_CSV**: (optional) path to GA output CSV, default best\_individual.csv

**OUTPUT\_TXT**: (optional) report filename, default summary.txt

**Example:**

# Default usage

python print\_PV\_panel\_group\_string\_data.py

# Custom filenames

python print\_PV\_panel\_group\_string\_data.py final\_best.csv report.txt

* Outputs

**Sample Output Format**:

Panel Group Summary

===================

Group 0, String 0:

Number of panels: 10

Min Imp: 8.50 A, Max Imp: 8.60 A

Total Vmp: 380.00 V

Power: 3230.00 W

Group 0, String 1:

...

1. Recommended Workflow

To use the full optimization and reporting pipeline, follow these steps:

* Prepare Data & Scripts

Ensure that **dataset1.csv**, **GA\_500\_panels.py**,

and **print\_PV\_panel\_group\_string\_data.py** are placed in the same working directory.

* Run Grouping Optimization

Execute the optimization script: bash python **GA\_500\_panels.py** This generates **best\_individual.csv** and **final\_voltage\_current\_summary.csv.**

* Generate Readable Report

Run the reporting script: bash python print\_PV\_panel\_group\_string\_data.py This produces summary.txt, detailing each group and string.

1. Conclusion

By following this workflow, users can transform raw PV panel specifications into a fully optimized series–parallel layout, and generate readable, stakeholder-friendly reports. This toolchain enables both quantitative performance evaluation and transparent communication of grouping outcomes.