Clustering Based Optimization and Automation of Utility Scale Solar Site Design

Kurt Rhee
EDF Renewables
North America
San Diego, USA
kurt.rhee@edf-re.com

***Abstract*— EDFR has developed a series of methods for quickly drafting a set of utility scale photovoltaic plant layouts and choosing an optimized plant design from that set. The automated drafting methodology utilizes a standard clustering technique and a novel cluster equalizing post-processing algorithm to solve a problem in existing automated drafting software, which is that existing techniques cannot assign dc power in the form of trackers to inverters without human intervention. This step is crucial to create end to end automation of a PV plant layout. Without it, it is impossible to accurately determine the layout of dc wiring and associated electrical equipment. The work nearly eliminates the need for developer drafting of utility scale photovoltaic plant layouts and provides a foundation for reducing levelized cost of energy by allowing EDFR to select the most financially optimal project design without investing large amounts of time creating the feasibility space under which optimization can occur. (*Abstract*)**

Keywords—spectral clustering, layout, levelized cost of energy, automation, optimization (*key words****)***

I. INTRODUCTION

Over the years, competition for power purchase agreements in the utility scale solar marketplace has increased, driving the price of utility scale solar down to their lowest levels in history. In light of this competition, it is becoming more and more necessary to explore as many possible land scenarios and site designs as possible in order to choose which combination of land control and plant design ultimately gives the most competitive PPA price.

Before the advent of the tool mentioned in this paper, exploring land scenarios and site designs at EDFR was a manual process involving a project developer responsible for possible project boundaries and parcels, a GIS analyst responsible for removing unsuitable land sections from within the project boundary, and a Solar Engineer responsible for creating a design, implementing it in AutoCAD, and creating an energy assessment in PVSyst. Due to the time-cost associated with implementing a design in AutoCAD, often times, only one layout would be generated for each land scenario and possible alternative layouts that might fit in the same land scenario would be determined by ratio of dc/ac ratio or sizing ratio (SR), ground coverage ratio (GCR) and module efficiency.

The ratio method involves setting up inequalities of those three factors which affect space constrained plant design and solving for the desired variable. For example if an original plant design is known, and a variant of that design is desired with an increased sizing ratio with the same module efficiency, then a new increased GCR could be solved for. There are obvious problems with this model including the fact that it does not take into account irregularities in the

$$\frac{GCR\ 1}{GCR\ 2} \times \frac{SR\ 2}{SR\ 1} \times \frac{Efficiency\ 1}{Efficiency\ 2} = 1$$

The financial model, consisting of capital expenditures, operational expenditures and energy assessment among other factors, when reading from this old process, could not then contain information about layout specifics such as lengths of DC cable required, lengths of AC cable, and there was no guarantee that the optimal design was actually buildable due to the ratio method being used to determine the land feasibility space.

Along with the problem of generating many layouts in a short period of time, solar engineers in charge of designing a plant also face the possibly more interesting problem of which design is most optimal, as for any given possible combination of ground coverage ratio and sizing ratio there are are an infinite amount of ways to place the a given design on a given land parcel, with some initializations being more optimal than others with many different ways to connect rack objects to inverters and so on and so forth. The choice of most optimal plant design also relies on the need for a similarly automated financial model in order to process the amount of data created from all of the possible plant designs. In terms of time commitment it is not feasible for an engineer to devote many hours to designing a solar plant in many different configurations in order to figure out which of those configurations it the best. In practice, this lack of time means that decisions around many plant level design decisions are made by rules of thumb rather than by data driven processes and models.

Many alternatives to AutoCAD exist which assist engineers with automating parts of solar plant design. EDFR has explored

many of these options and found an impressive set of software, each with the potential to save time in the design phase, but what we did not find from any existing software at the time of writing is one which could group horizontal single axis trackers to inverters without human intervention. Without this step, we found it would be too time consuming in existing software to weigh the advantages and disadvantages of non-uniform land areas which may increase the DC line cost or know exactly which trackers correspond to which inverters and where those inverters should sit within the project boundary. This is an especially large issue when considering many different (100+) sizing ratio's and designs at once, as human intervention would be required each time that trackers needed to be assigned to inverters.

EDFR's clustering approach addresses the design problem listed above, and while not perfect, gives EDFR a headstart on creating a feasibility space of possible solar designs for a given land boundary and a framework for choosing a optimal design from that space.

II. Methodology

Solving the generalized problem of fitting objects inside of a given boundary, optimizing for number of objects has been studied in the fields of computational geometry and operations research for decades [1] and is considered NP-hard. The generalized problem specified to the solar engineering application is the fitting of racks inside of a buildable land object. The buildable land can be defined as a set of polygons which describes the areas feasible for laying down trackers that lies within the boundary of land controlled by EDFR. For example, all of the leased area of a project minus streams, transmissions easements and flood zones may make up a buildable land.

The general solar engineering problem can be broken down into a a set of steps that approximate the global maximum while also taking into account factors that affect the design such as cable length, continuitiy of DC blocks, etc. In other words, this paper will describe a set of algorithms that solve in polynomial time and will not discuss a general solution. The program was written in python using only open source libraries. The choice to use open source libraries was mostly due to cost and convenience. The relative merits of open source vs. closed source software is beyond the scope of this paper.

1. Pre-Processing

As a given project boundary can contain many discontiguous parcels and buildable areas, it is necessary to divide each of these polygons into their own separate optimization problem. It is also necessary to assign an appropriate projection to convert spherical latitude and longitude to a flat coordinate system. This can all be done in python relatively easily via the open source package geopandas [2].

2. Fill Buildable Land with Horizontal Single Axis Trackers

After creating a polygon (rectangle) object which represents the length and width of a standard tracker, the program then finds the farthest west longitude and the farthest south latitude point on the buildable polygon. Starting from the coordinate that combines these two minimums, the program pastes these tracker objects from west to east, south to north until it surpasses the farthest north latitude point and farthest east longitude point. While pasting these trackers, the program considers the pitch between trackers and necessary roads. The program then attempts *n* different starting positions and selects whichever iteration gives the largest number of tracker objects.

At this point in the program there exists a buildable land object and a set of tracker objects arranged in an array which overlays the buildable land from its southwestern most corner to the northeastern most corner. After this step is complete, trackers that fall outside of or on the border of the buildable land polygon are deleted so that only trackers with feasible building locations remain.

3. Remove Trackers in Excess of specified DC/AC Ratio

The trackers placed by the program at this point represent all possible tracker positions. Given that EDFR generally designs solar plants so that each inverter has roughly the same amount of DC capacity electrically tied to it, some of these possible tracker positions will not be used. For example, if there are 5 blocks that fit within a given boundary and each block contains 100 trackers, but there are 543 possible tracker positions, then 43 potential tracker positions would need to be removed.

In order to remove DC capacity in excess of what is needed to hit the desired DC/AC ratio for each inverter on the buildable polygon, each tracker object is given a continuity score from low to high with high numbers representing a high likelihood that the tracker will be saved. The scores are generated by first converting each tracker object into a point object in a point cloud, and then creating separate adjacency matrices in the x and y direction. The python library networkx was instrumental in creating these adjacency matrices [3]. Each neighbor in the x direction and each neighbor in the y direction adds to the score. Additional scoring rules are also added to account for undesirable shapes in the resulting site design. In this way, the least useful trackers per polygon are removed.

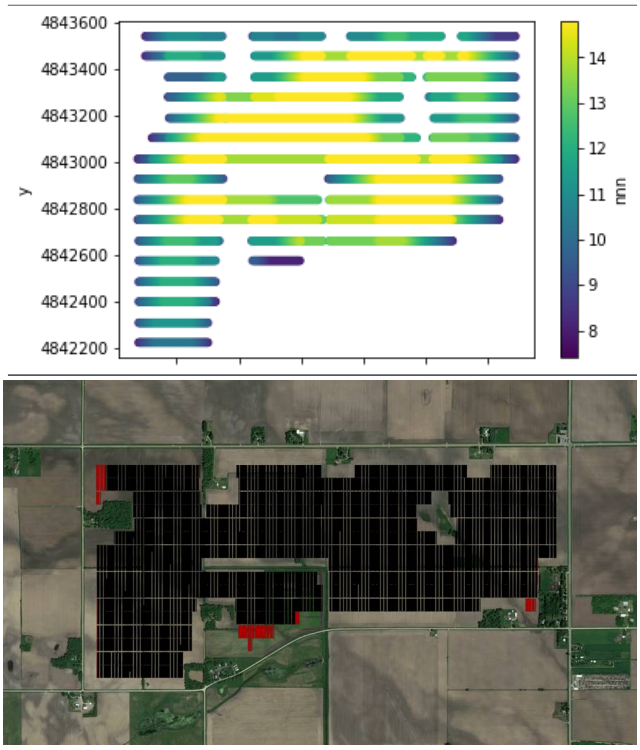

Fig. 1. (Top) Scoring of individual tracker points in the site design point cloud. X and Y axes represent the coordinate plane, nnn represents the number of nearest neighbors within the scoring system. (Bottom) Tracker objects that are saved by the tool in black and tracker objects removed by the tool in red.

4. Use Clustering Algorithm to Find Possible Blocks

The most difficult portion of the solving tool is determining out of all saved trackers, which ones should be grouped into which AC blocks and in what fashion. In order to complete this task, first each contiguous grouping of trackers is separated from the rest to be optimized on its own. This can be done by once again utilizing the point cloud of tracker position. From the corresponding adjacency matrix one can split the undirected graph into several disconnected subgraphs in places where no adjacency is found. The algorithm of splitting subgraphs via the spectrum of the graph Laplacian is outside of the scope of this paper but is a well-researched topic in the field of applied mathematics [4].

Each point cloud then is separated into k separate, internally contiguous clusters based on how many blocks could theoretically fit in the point cloud. For example, if there are 1000 trackers on the buildable land and we plan to assign 200 trackers to a given block, then $k = 5$. Clustering algorithms such as K Means clustering and Spectral clustering which take the number of clusters as a parameter are useful in this step. We at EDFR have found that spectral clustering outperforms K means clustering because it allows for the creation of non-convex cluster groups. We chose to utilize scikit-learn's spectral clustering algorithm for this first pass grouping of trackers. [5]

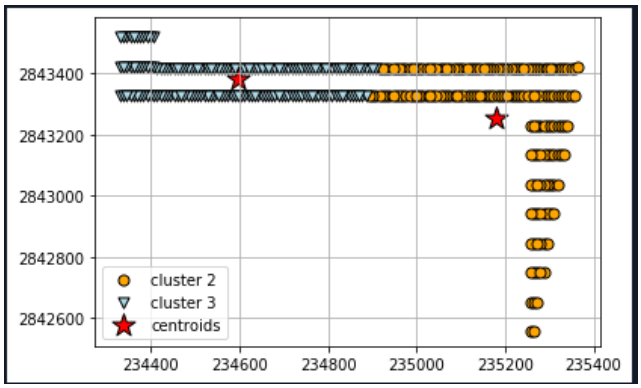

Fig. 2. Example of an initial spectral clustering of one contiguous section of block trackers

The problem with each of these clustering algorithms is that the clusters produced by each algorithm do not produce equal sized clusters. When clusters are not equal sized, the program will assign too many trackers (DC capacity) to some inverters while leaving others relatively sparse. In order to equalize the number of points in each cluster point cloud it is necessary to post-process the clustering algorithm output.

5. Post Process Clustering Algorithm Output

After creating clusters via Spectral clustering, an adjacency matrix of clusters is generated. The adjacency matrix contains information not only of which clusters are contiguous with other clusters, but also the size of each cluster. The algorithm for equalizing clusters involves adjacency pathways between the largest and smallest clusters:

- Find the adjacency chain linking the largest "overfull" cluster and the smallest cluster.
- Pass the number of trackers needed to equalize the entire cluster group, from largest "overfull" cluster, to adjacent clusters, to smallest cluster. Djikstra's shortest path algorithm is useful for finding efficient chains. [6]
- At each step, the adjacent cluster should gain ownership of trackers from the previous cluster in the chain based off tracker distance to cluster centroid.
- Cluster sizes and adjacencies are recomputed.
- Once every cluster has an equal number of trackers assigned, the algorithm can stop.

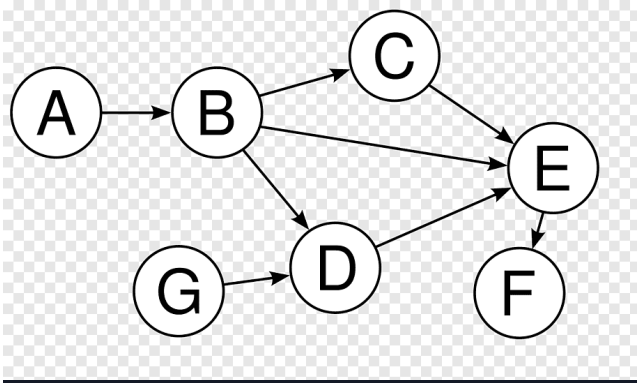

Fig. 3. Example of an adjacency chain from pngwave.com. Overfull cluster A donates tracker objects to overfull cluster B which donates to cluster D, E and eventually F until F becomes exactly full.

6. Removing Unnecessary Blocks

Once the algorithm has looped through every discontiguous polygon the program may have created more clusters (blocks) than necessary. In some cases, it is simplest to choose the farthest clusters from the substation and remove them from consideration as these polygons often bear the higher cost of additional wire quantities or losses in transporting energy to the project substation. In other cases, another strategy might be used to improve project economics.

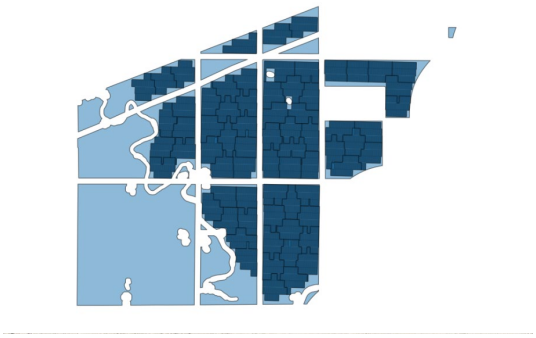

Fig. 4. Example of a theoretical layout generated from the program in which the amount of land available exceeds the amount of land needed to fit a specified system.

7. Identifying Bill of Quantities

Now that the layout is complete, the quantities of several capex and opex parameters can be calculated. For example, the number of piles required per tracker, the distance of travel for ac cable to the substation, the amount of dc cable needed, etc. Once this is complete each design can be compared and the most optimal design in can be chosen to be entered into the competitive bid process.

III. Results and Discussion

The resulting program, without optimizing for speed, returns a design for moderately complex buildable land polygon in approximately 20 minutes and can be parallelized across multiple cores. In this way, EDFR can run many possible designs in parallel with no human interaction and minimum time commitment compared to the manual process or even software assisted processes. In case manual changes need to be made, deliverable files can be exported to .kml format for viewing in Google Earth or exported into .dxf for editing in AutoCAD.

IV. Conclusion

Determining which trackers are assigned to which blocks plays a crucial role in the effectiveness of automatic site design methods and is necessary when trying to design many sites at scale. Overall, this clustering based approach to utility scale solar power plant design fills some of the gaps within the existing cadre of solar site design tools and increases the speed at which EDFR can generate scenarios and compare their relative advantages and disadvantages. We believe that it represents an incremental step towards increasing our competitiveness against natural gas and a major step in reducing the amount of engineer time spent doing manual design work.

Acknowledgments

This research was supported by EDFR. We would like to thank James Alfi and James Christopherson for the time and ability to work on optimization topics. We would also like to express our gratitude towards Cameron Nielsen, Marine Bila and Angel Velasco for their helpful conversations on the topic of designing utility scale photovoltaic power plants.

References

[1] Wäscher, G.; Haußner, H.; Schumann, H. An Improved Typology of Cutting and Packing Problems. European Journal of Operational Research Volume 183, Issue 3, 1109-1130

[2] Kelsey Jordahl; Joris Van den Bossche; Jacob Wasserman; James McBride; Martin Fleischmann; Jeffrey Gerard; Jeff Tratner; Matthew Perry; Carson Farmer; Geir Arne Hjelle; Sean Gillies; Micah Cochran; Matt Bartos; Lucas Culbertson; Nick Eubank; Aleksey Bilogur; maxalbert (2020). Geopandas. https://geopandas.org/

[3] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, "Exploring network structure, dynamics, and function using NetworkX", in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008

[4] Chung, Fan (1997) [1992]. Spectral Graph Theory. American Mathematical Society. ISBN 978-0821803158.

[5] Pedregosa et al. Scikit-learn: Machine Learning in Python., JMLR 12, pp. 2825-2830, 2011.

[6] E. W. Dijkstra. (1959) A Note on Two Problems in Connection with Graphs. Numerische Mathematik, 1. 269-271..