TEXAS A&M UNIVERSITY
Engineering

Auto-Layout Techniques for Displaying Large-Scale, Geographically-Embedded Power System Data

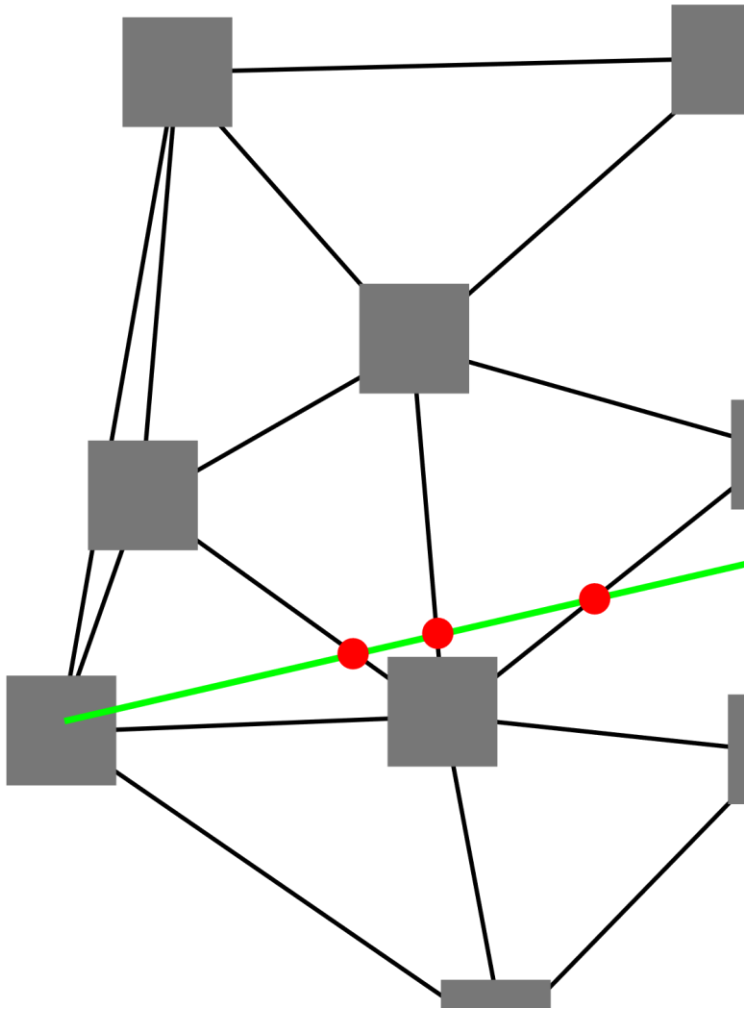
Adam Birchfield
Dept. of Electrical and Computer Engineering
Texas A&M University

Texas A&M Smart Grid Center Webinar
January 19, 2022

abirchfield@tamu.edu

Presentation Overview

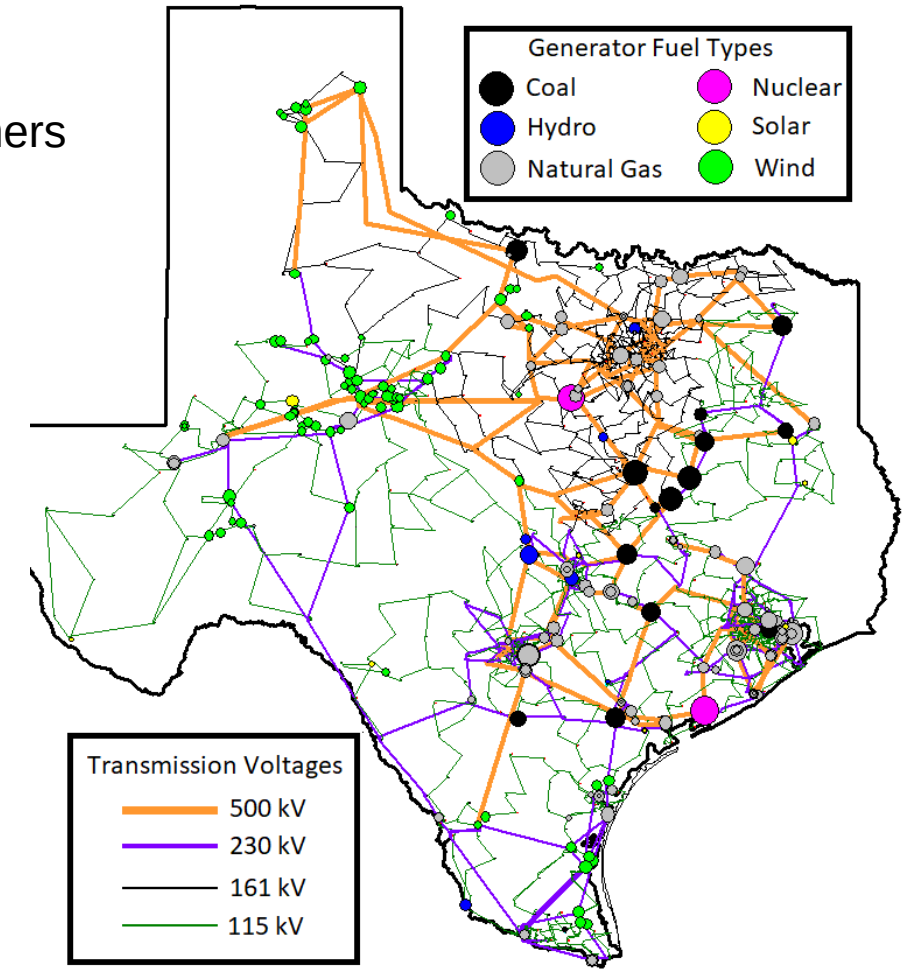
- Power system visualization techniques help engineers and others understand and analyze these large datasets with new insights
- This presentation covers two auto-layout algorithms used to build data visualizations for power systems
  - Automatic line routing for clearer onelines
  - Mosaic diagrams to show geographically-embedded variables



Support for this work from ARPA-E, DOE, and NSF is greatly acknowledged, as well as contributions from TAMU colleagues and students

Background: Creating Synthetic Grids

- Substation Planning
  - Start with public data for generation, load
  - Cluster substations, add buses, transformers
- Transmission Planning
  - Place lines and transformers
  - Iterative dc power flow algorithm
  - Match topological, geographic metrics
  - Contingency overload sensitivity
- Reactive Power Planning
  - Power flow solution (ac)
  - Voltage control devices
- Extensions
  - Transient stability
  - Geomagnetic disturbances
  - Single-line diagrams
  - Optimal power flow (OPF), time series scenarios, interactive simulations, …

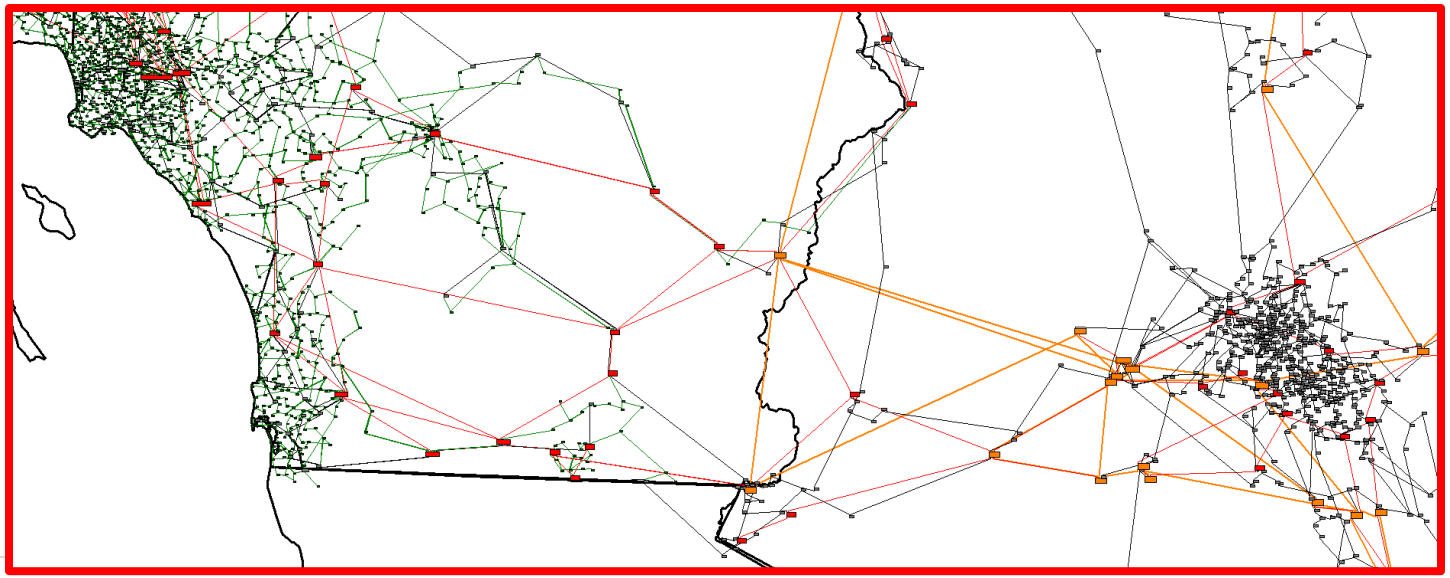


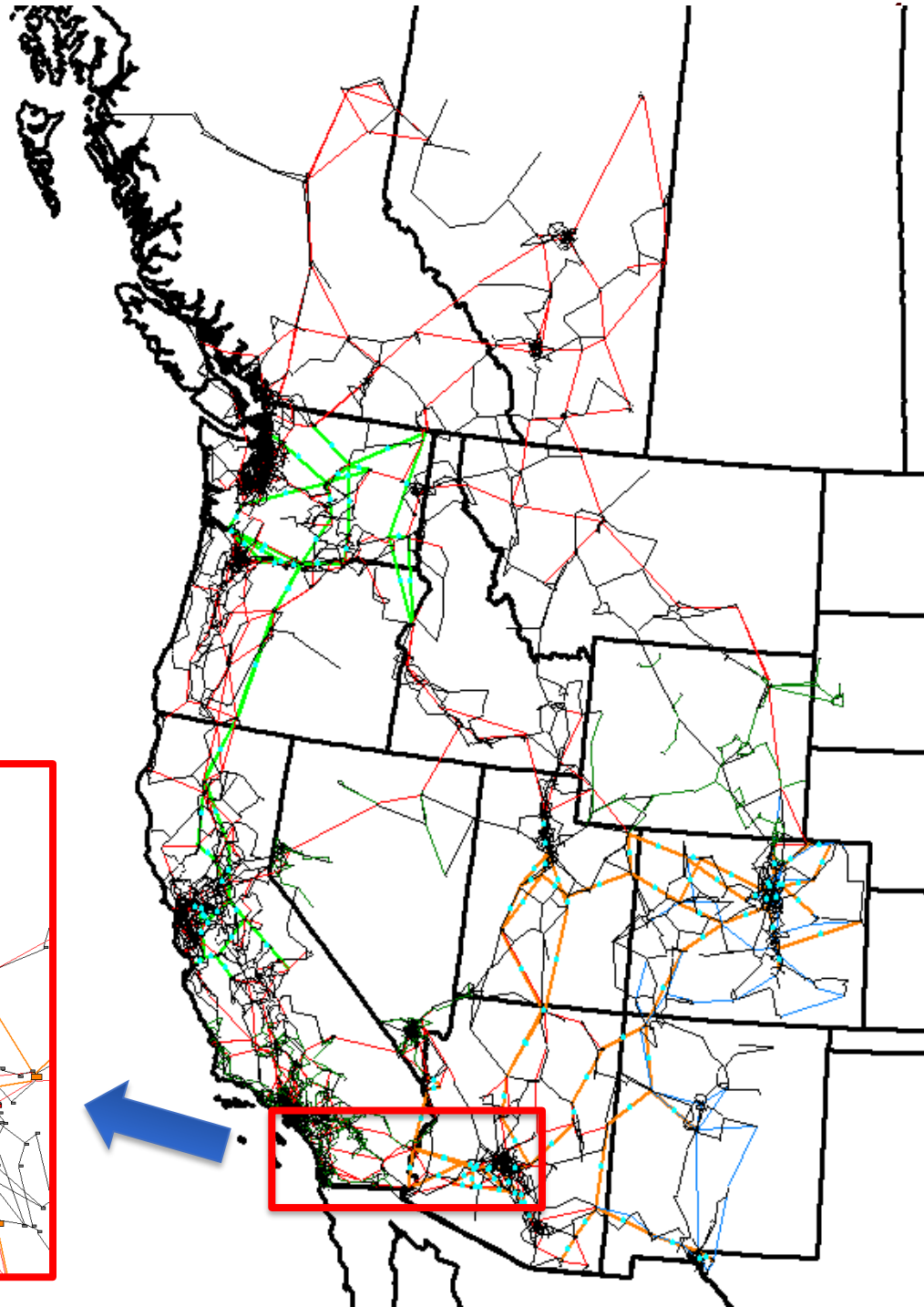
2000-bus synthetic grid on the Texas footprint

Developing these new datasets has highlighted research questions about how best to show and visualize large-scale power system information

Recent Progress in Building Synthetic Electric Grids

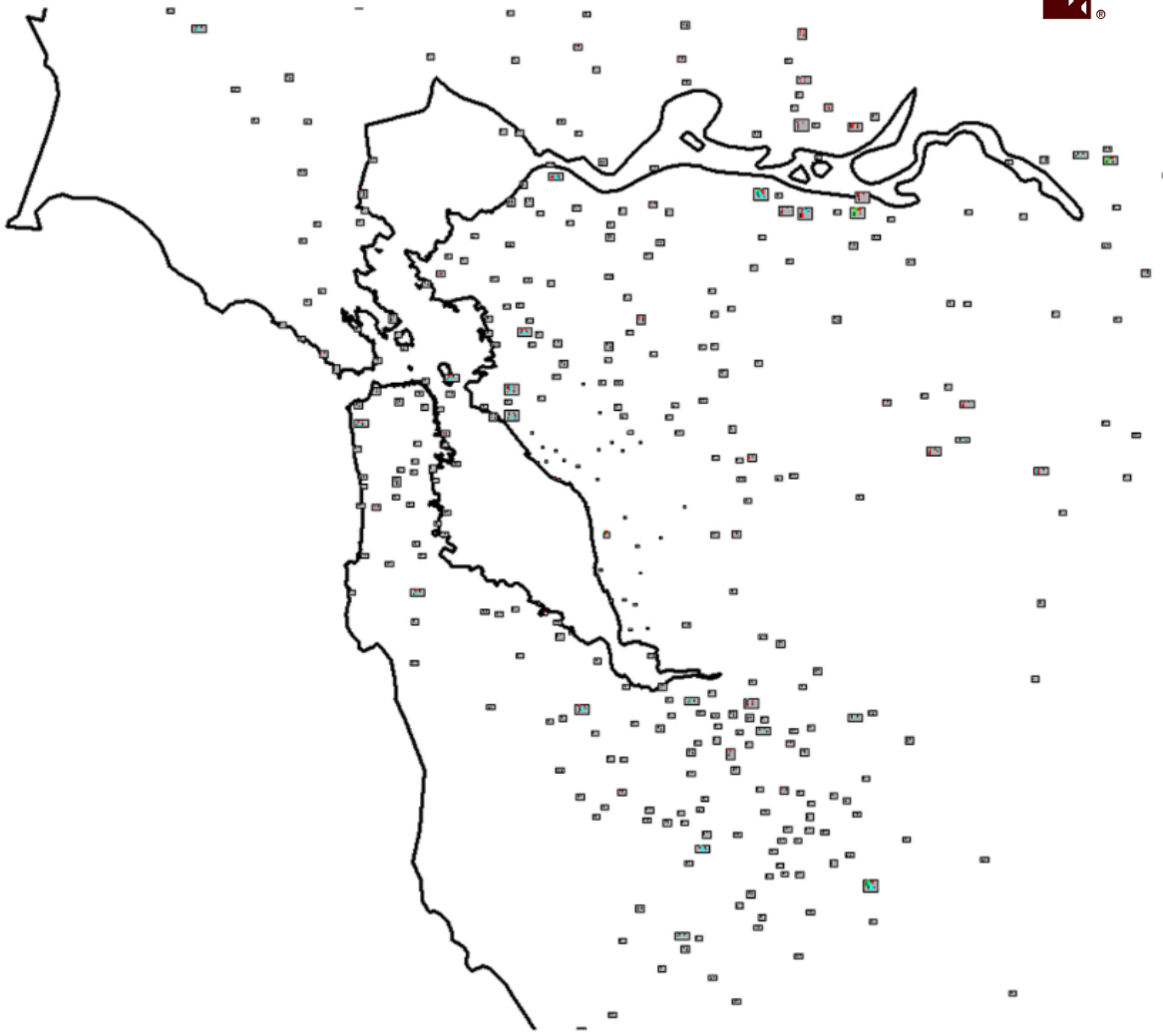
- **Large**: This case is 20,000 buses, similar to the actual Western Interconnect (WECC)
- **Complex**: Multiple interacting voltage levels, remote regulation, capacitors, taps
- **Realistic**: Matching a large suite of validation metrics against actual systems
- **Fully public**: It does not correspond to any actual grid or contain any confidential information





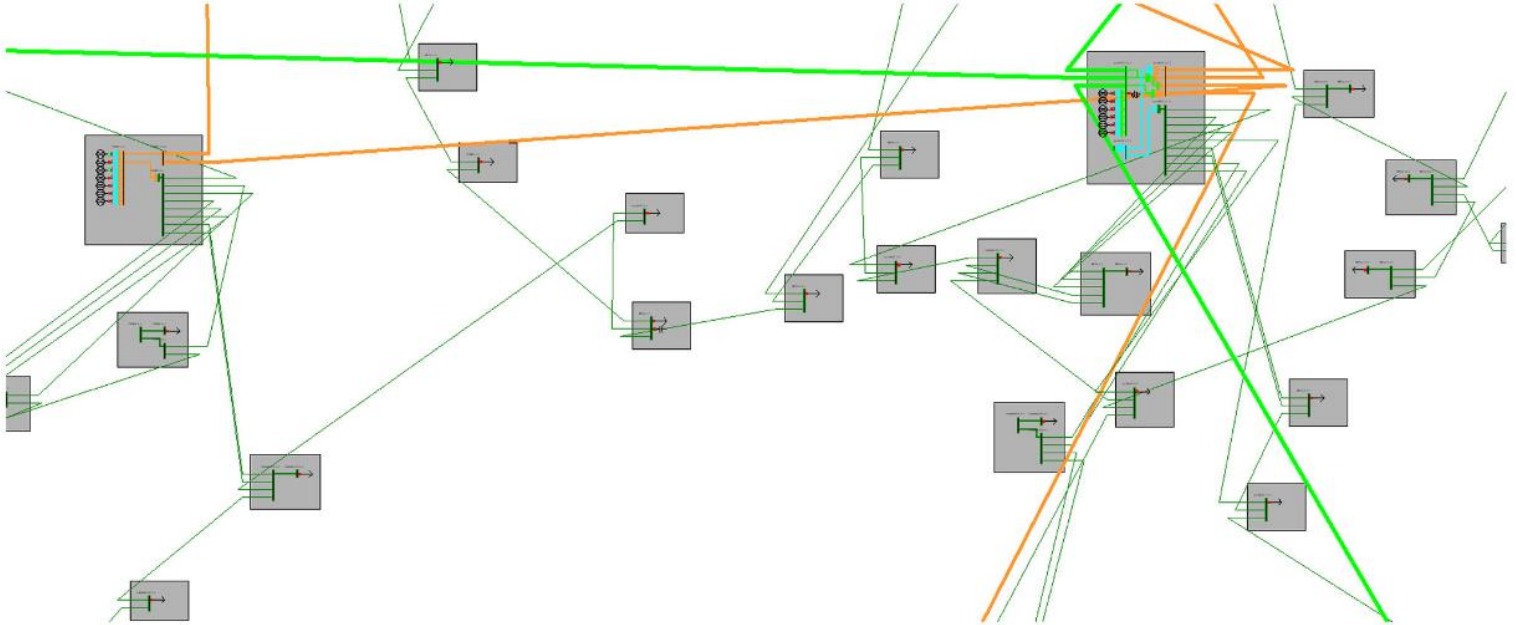
Line Layout Algorithm

- This problem came to the fore when building and publishing synthetic grid datasets: how can we make a useful oneline diagram for them?
  - They are geographically embedded
- The first step is to place the substations
  - This method is not presented here in detail
  - Purely putting them geographically means they will have significant overlap to not be extremely small
  - A force-directed approach models substations as particles with mutually repulsive forces and springs attached to the geographic center. It has been used with some success. It can be slow, and it can distort the substation positions more that desired.
  - A priority-queue greedy approach works quite well
- Next we need to connect them by drawing the transmission lines



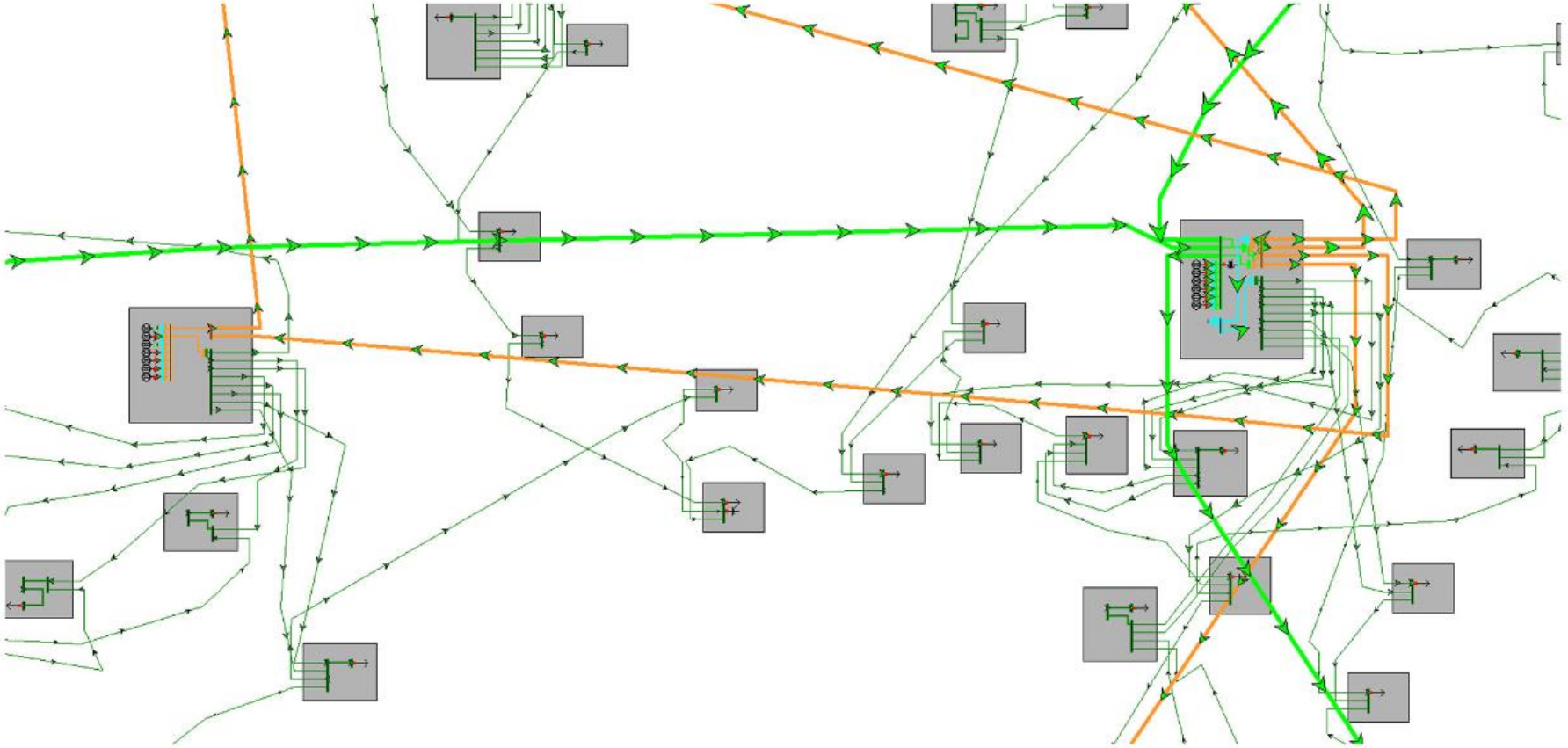
The Basic Line Routing Problem

- Substations have been adequately spaced, with information shown inside the gray boxes.
- Initially, lines are drawn with a straight line from substation to substation
- Line routing is adding a number of waypoints (bends) in the lines
- Objective is to avoid overlapping of lines with substations and with each other, while minimizing the number of bends and total length of the lines



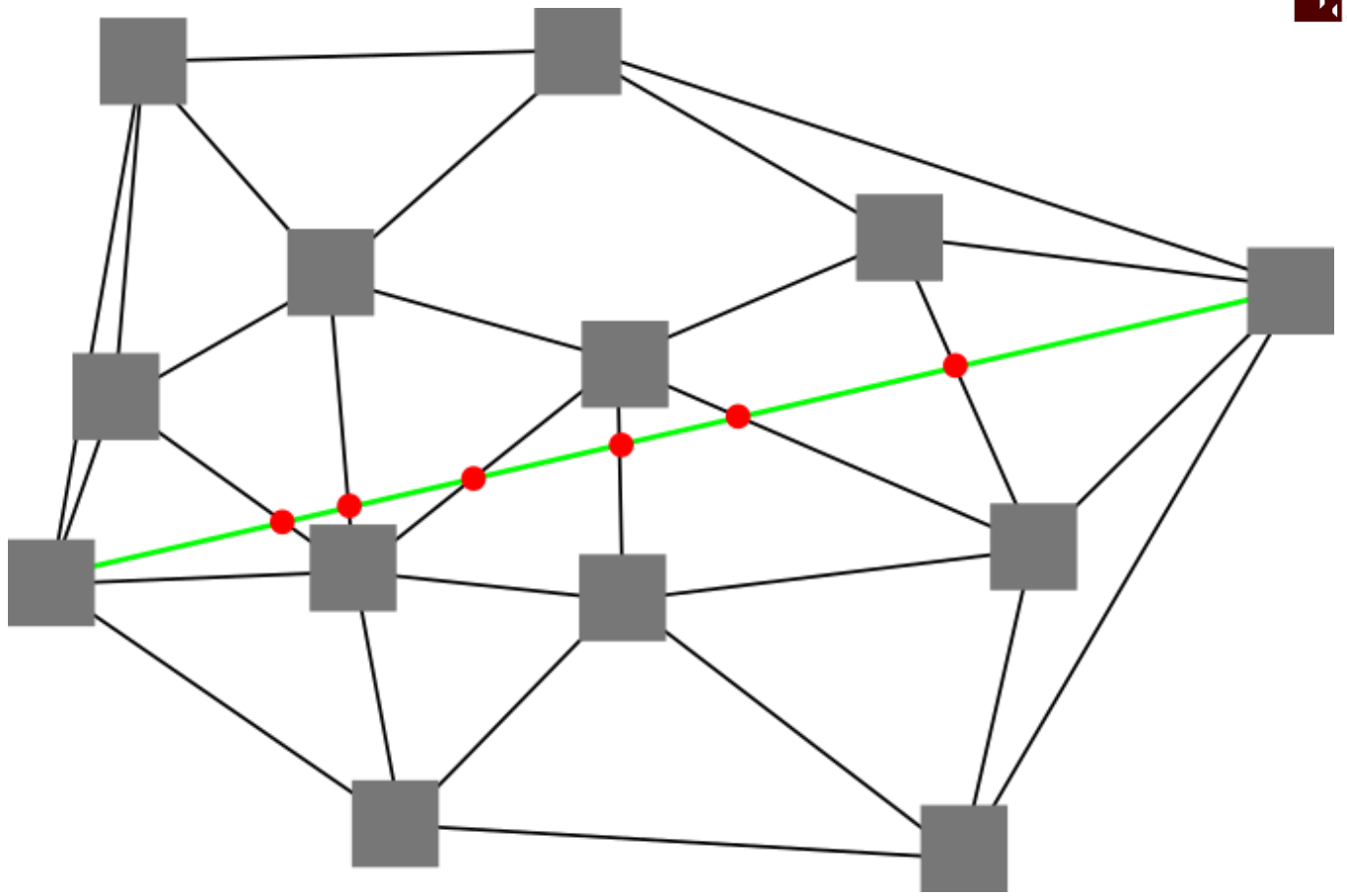
Example Results for Delaunay Approach

- The Delaunay approach for line routing involves navigating the transmission line through the Delaunay triangulation graph
- Much fewer intersections with substations and other lines
- Note: EHV lines are allowed to intersect lower-voltage substations



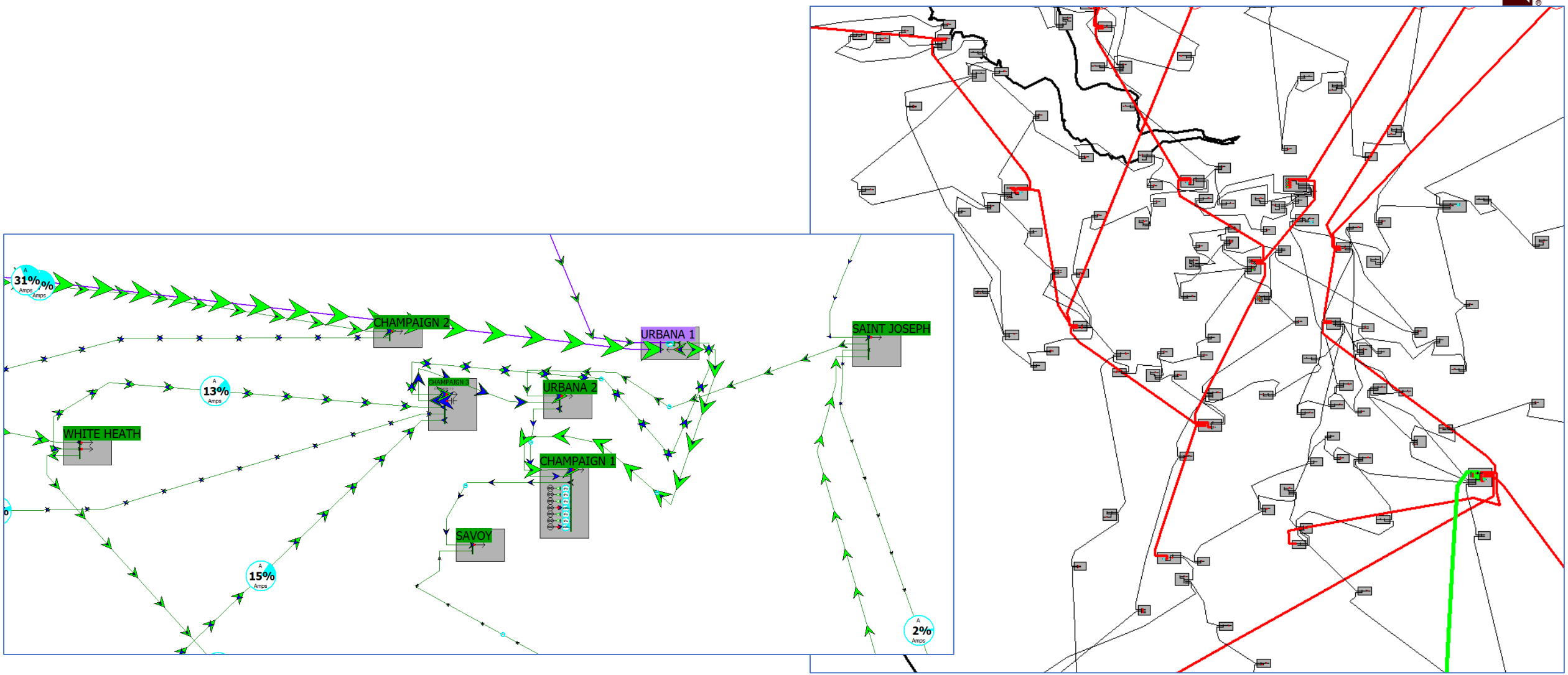
Delaunay Line Routing Algorithm

1. Set up Delaunay triangulation between all substations (this is O(n-log-n)). Each edge of this graph is now a routing channel.
2. Route lines straight through the Delaunay triangulation. As shown in picture, this green line crosses six routing channels.
3. Adjust waypoints to ensure good spacing. Each routing channel is analyzed to make sure waypoints (red dots) are far enough from substations and waypoints of other.
4. Iterate step 3 at least 10 times, allowing coordination of neighboring routing channels.



Gray boxes are substations; black lines are Delaunay routing channels; green line is transmission line; red dots are waypoints

More Examples



Mosaic Diagram

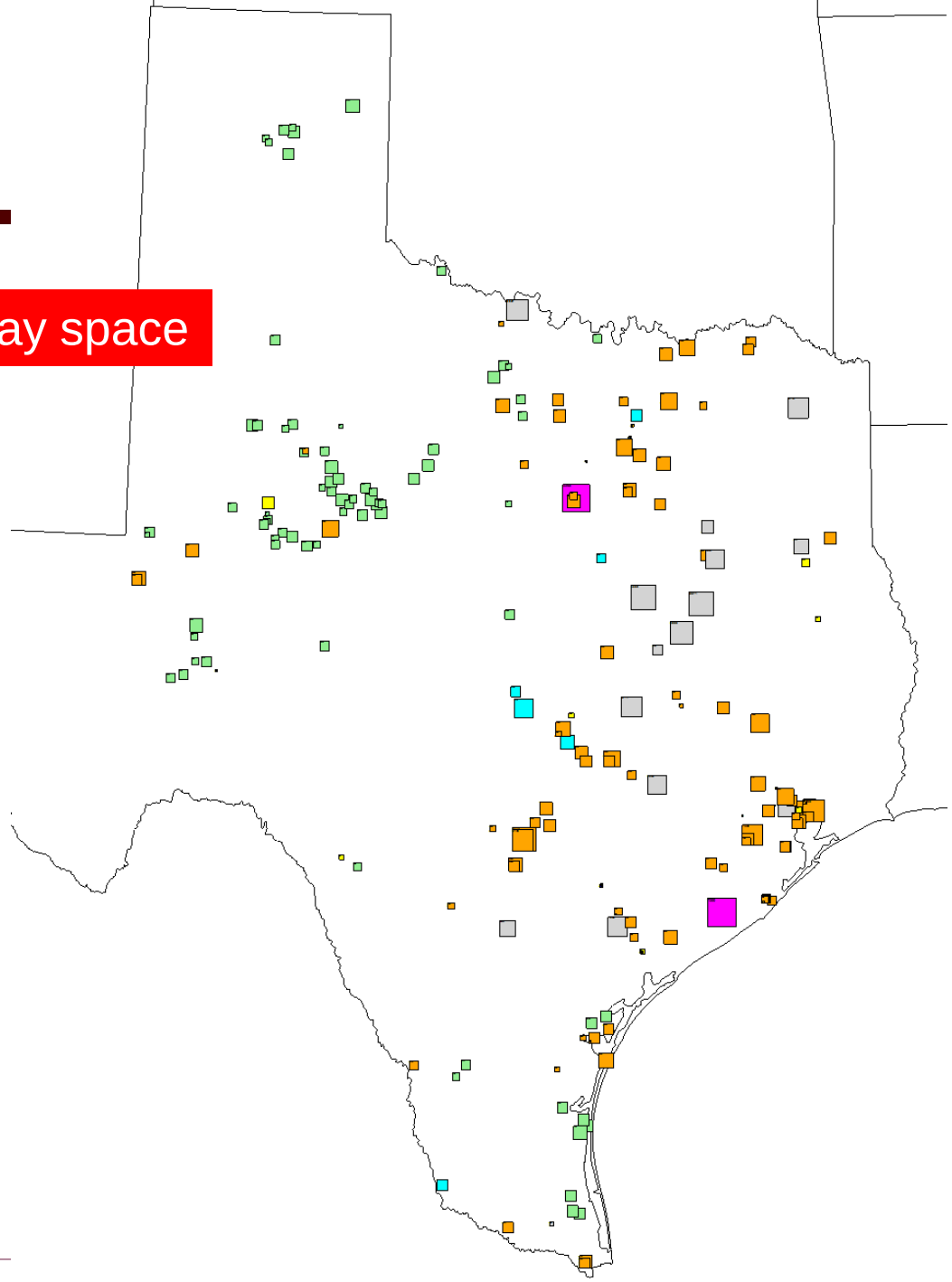
- To improve situational awareness, we want to show power system data in a way that is most useful and informative to people, e.g. engineers
- One approach is using relative size on a display to show relative importance (larger generators bigger)
- Geographic context is key to situational awareness
- However, power system data is geographically distributed in a highly non-uniform way: lots of clusters
- Can we make a useful display that also maintains geography?

A mosaic is a picture formed by arranging small colored tiles to give an overall impression to the viewer!

Example

- This diagram shows the system generators for synthetic 2000 bus case, colored by fuel type.
- Units at the same or nearby location overlap! And they are small and hard to see.

Using 5% of display space



Showing generators, with size proportional to max MW

Magenta – Nuclear

Gray – Coal

Orange – Natural Gas

Blue – Hydro

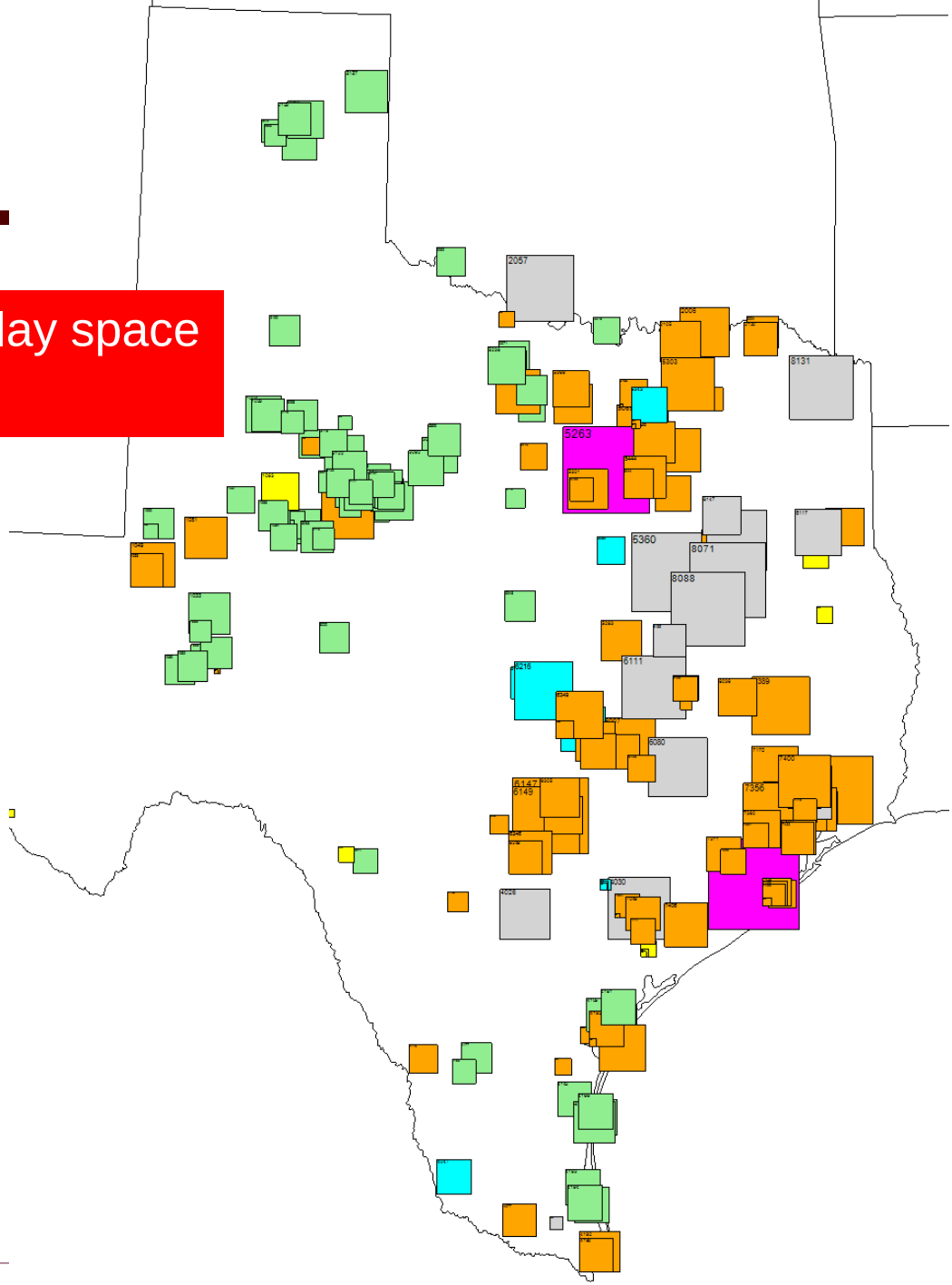
Green – Wind

Yellow – Solar

Example, Cont.

- We can make them bigger, but overlap is even worse. (There is a lot more in the Houston area than can be seen here.)

Using 50% of display space (counting overlap)



Showing generators, with size proportional to max MW

Magenta – Nuclear

Gray – Coal

Orange – Natural Gas

Blue – Hydro

Green – Wind

Yellow – Solar

Problem Statement

- Find an optimal location within a rectangular screen for a number of rectangular objects of different sizes
  - We are calling this a “mosaic display”
  - Objects or “mosaic tiles” could represent substations, buses, loads, generators, shunts, lines, transformers
  - Size could represent a static or dynamic quality
  - Doesn’t need to be square, but shouldn’t be extremely thin
  - Each mosaic tile has a preferred location: “optimal” refers to minimizing the displacement norm
  - But no overlapping is allowed: this is the crucial constraint
- Because an interactive display environment is preferred, computational speed is important

The definition of a mosaic is a larger picture formed by small colored tiles!

Mathematically

For N tiles to fit inside bounding box $[x_{\text{min}}, x_{\text{Max}}]$ and $[y_{\text{min}}, y_{\text{Max}}]$. Each rectangle i has area A_i and preferred position (x_{i0}, y_{i0}), with decision variables position (x_i, y_i) and dimensions (w_i, h_i).

$$\text{Min}_{\bar{x},\bar{y}} \sum_{i \in N} ((x_i - x_{i0})^2 + (y_i - y_{i0})^2)$$

$$s.t. \quad x_{\text{min}} \leq x_i \leq x_{\text{Max}}$$

$$y_{\text{min}} \leq y_i \leq y_{\text{Max}}$$

$$w_i \cdot h_i = A_i \qquad r_{\text{m}} \leq \frac{w_i}{h_i} \leq r_M \quad \text{(Aspect ratio bounds)}$$

For each $i, j \in N, i \neq j$ one of the following must be true (non−overlapping constraint):

$$x_i + w_i \leq x_j \quad \text{OR} \quad x_i \geq x_j + w_j$$

$$\text{OR} \quad y_i + h_i \leq y_j \quad \text{OR} \quad y_i \geq y_j + h_j \ \square$$

- Find an optimal location within a rectangular screen for a number of rectangular objects of different sizes
- Doesn't need to be square, but shouldn't be extremely thin
- Each mosaic tile has a preferred location: "optimal" refers to minimizing the displacement norm
- But no overlapping is allowed: this is the crucial constraint

Approaches and Issues

- The main trouble is with the last constraint, which makes the feasible space disjoint, non-convex
- One option is to relax this constraint, but we want to strictly avoid all overlaps.
- We could take a combinatorial, integer approach
  - Discretize $x_i, y_i,$ bounds, and w_i, h_i to be integer variables. Basically, we make a grid with resolution as appropriate for screen, accuracy, and speed requirements.
  - Now we have a constrained assignment problem to place the N rectangles (split into N' sub-squares) on a $x_{\mathrm{M}} \times y_{\mathrm{M}}$ grid.
  - However, we need additional complicated constraints so that the sub-squares are all adjacent to each other in the form of a rectangle, so we cannot use traditional algorithms for the assignment problem (slow anyway)
- The number of combinatorial possibilities is > $N!$
- A N^2 algorithm is probably too slow, anything larger is definitely too slow
- Branch-and-bound is too slow, but this approach might get us an optimal solution in e^N time (that doesn't work either)

Greedy Approach

- Add each rectangle once to the locally-optimal spot
- Search M locations, but greatly reduce this by radiating from preferred position until feasible spot found and others can be ruled out
- Computational upper bound is O(N*M), but in practice it is usually more like O(N)
- Does not reach the optimal solution: a key issue is that the last objects added tend to be very far from initial spot, depending on how constrained the problem is.
- Hence it is highly dependent on the placement order, with no good way to tell how they should be ordered.
- Can we do better than this and still be fast?

Horizontal Packing Algorithm

- This approach is O($N^{1.5}$) in practice and gives much better results than the greedy approach
- Discrete in the vertical direction in powers of 2, while continuous in the horizontal direction
- Starting from the median, alternate adding tiles to the left and right ends, then pushing existing ones inward towards the center
- Imagine packing the screen from left and right
- Before going through the details, let's take a look at example results with increasing the size of the generators…

Example, Horizontal Packing

- Starting out, most generators' geographic locations are easily visible.
- There is still enough distortion that each unit is visible. Before we couldn't see that each nuclear plant is actually two units.

Showing generators, with size proportional to max MW

Magenta – Nuclear

Gray – Coal

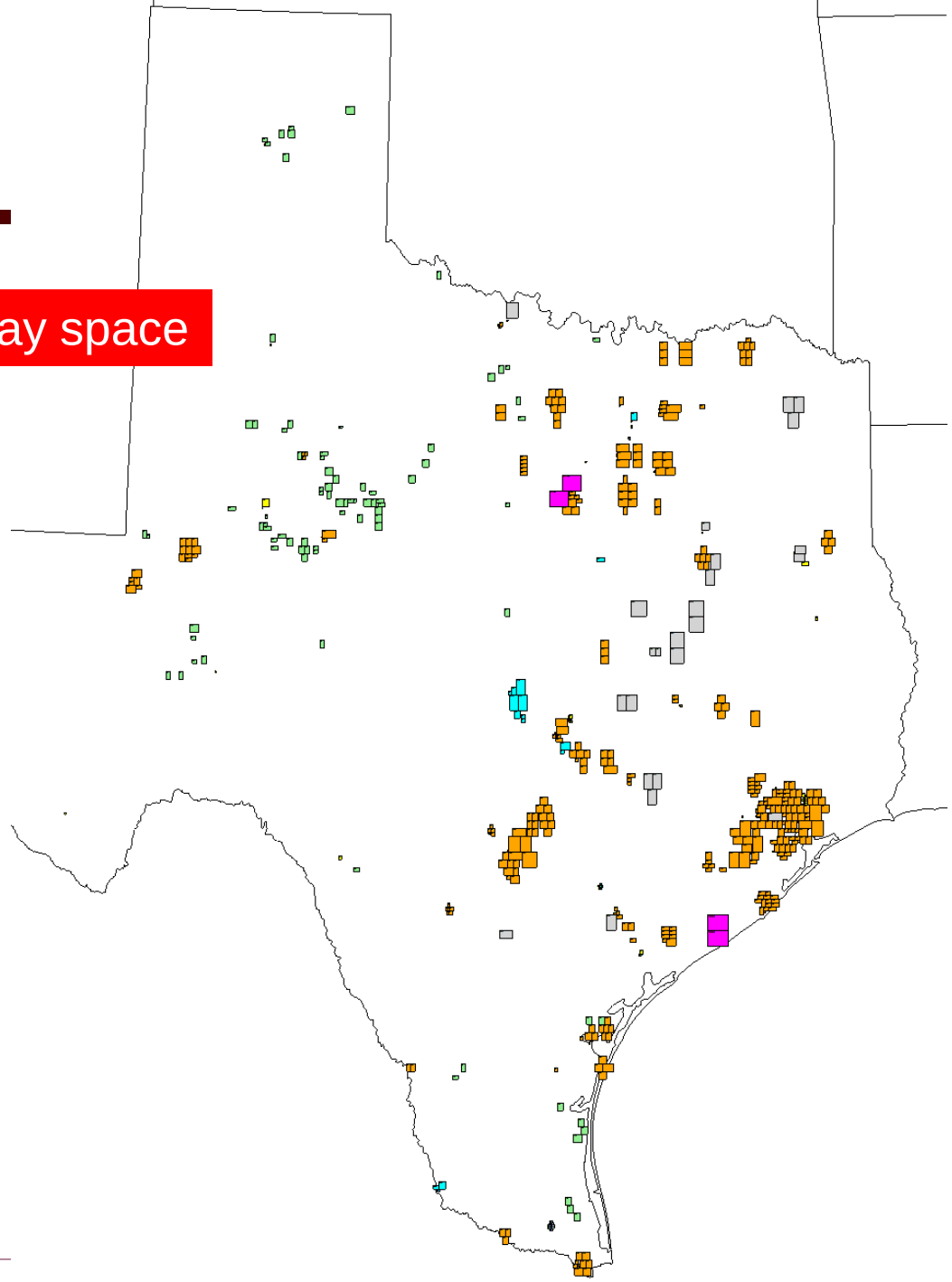
Orange – Natural Gas

Blue – Hydro

Green – Wind

Yellow – Solar

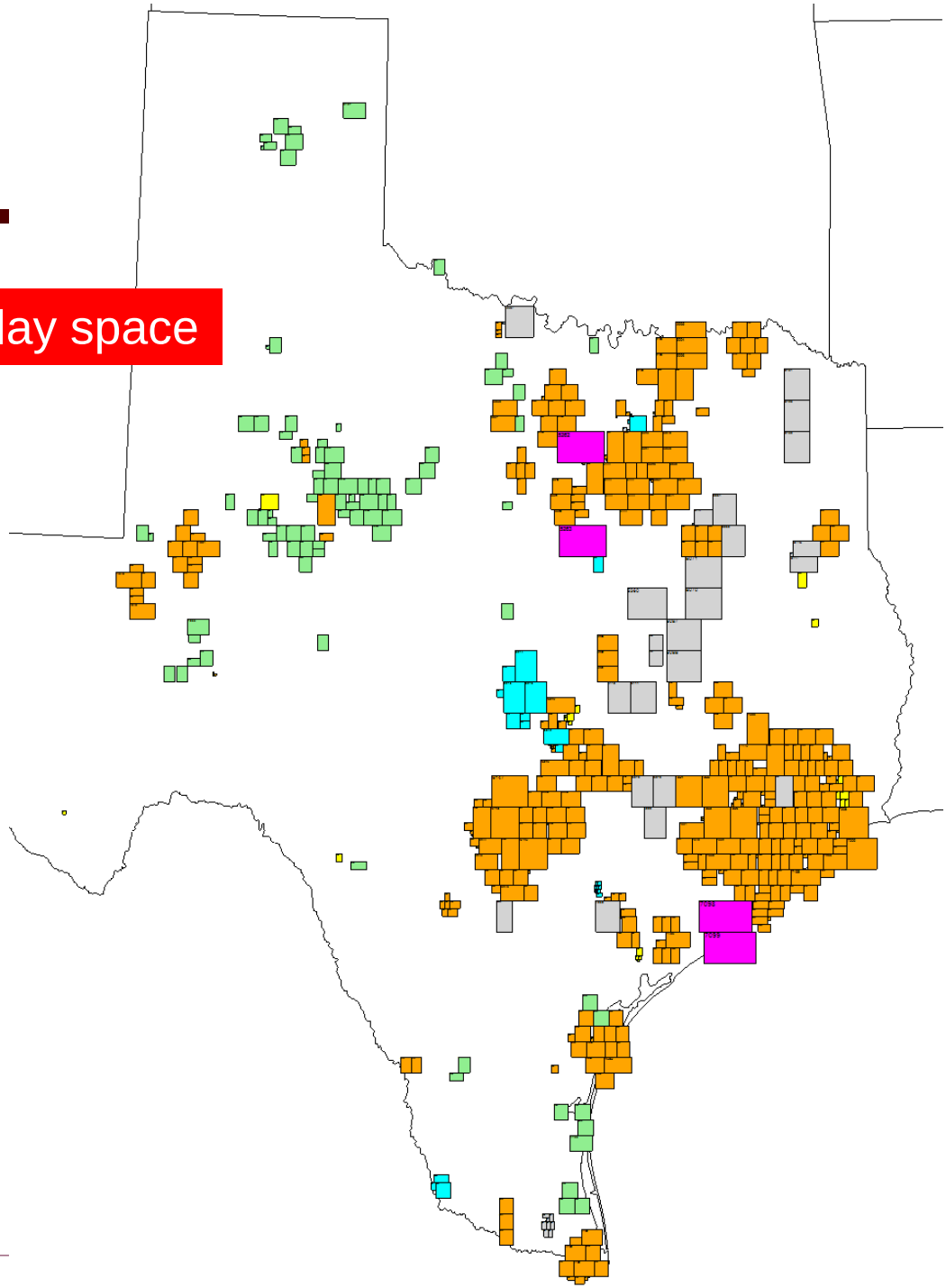
Using 2% of display space



Example, Horizontal Packing

- As the rectangle size increases, the more crowded areas become distorted so you can see everything.

Using 10% of display space



Showing generators, with size proportional to max MW

Magenta – Nuclear

Gray – Coal

Orange – Natural Gas

Blue – Hydro

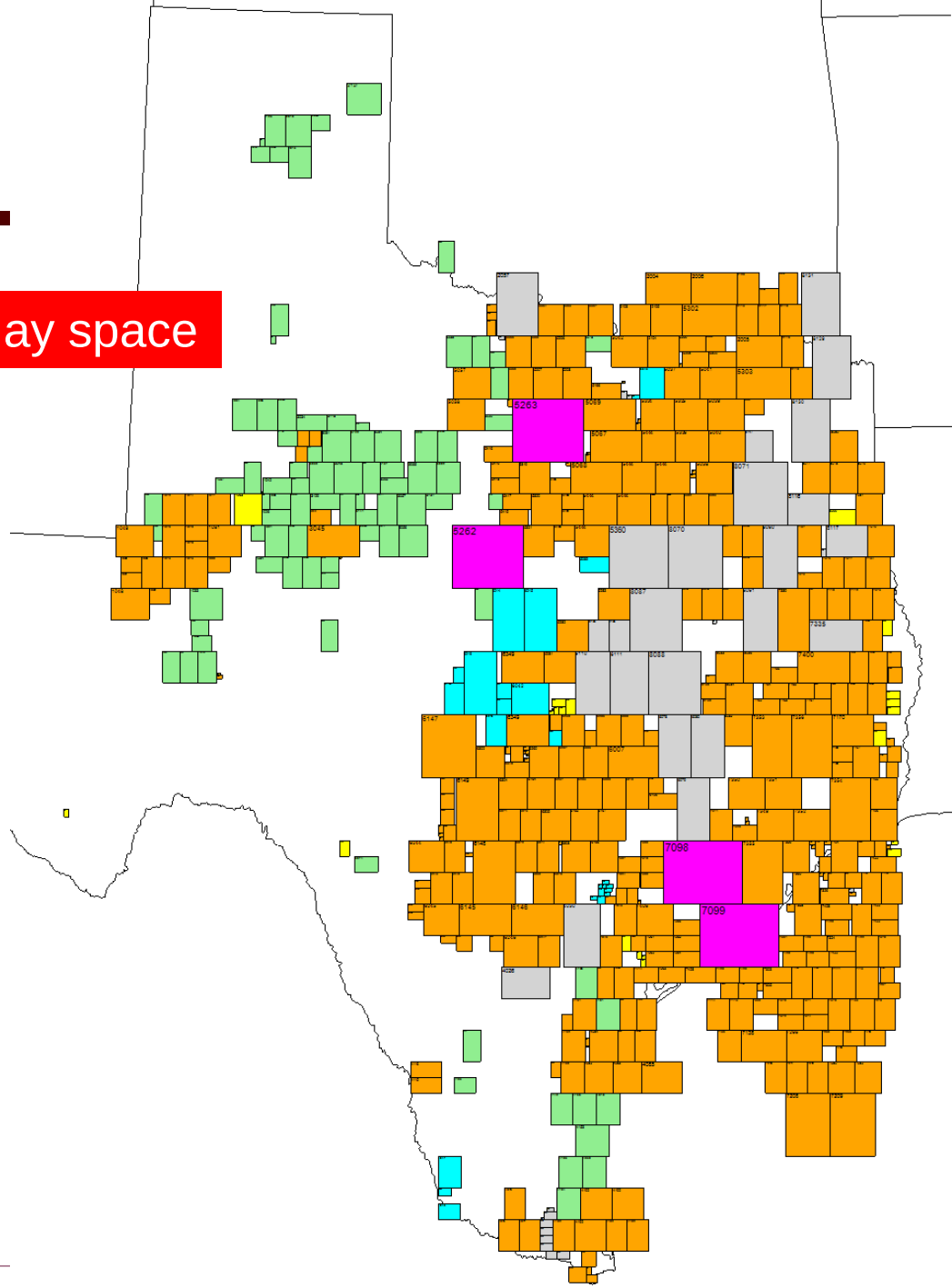
Green – Wind

Yellow – Solar

Example, Horizontal Packing

- Now the clusters merge into one big cluster, which is starting to push against the boundary box on the right side

Using 30% of display space



Showing generators, with size proportional to max MW

Magenta – Nuclear

Gray – Coal

Orange – Natural Gas

Blue – Hydro

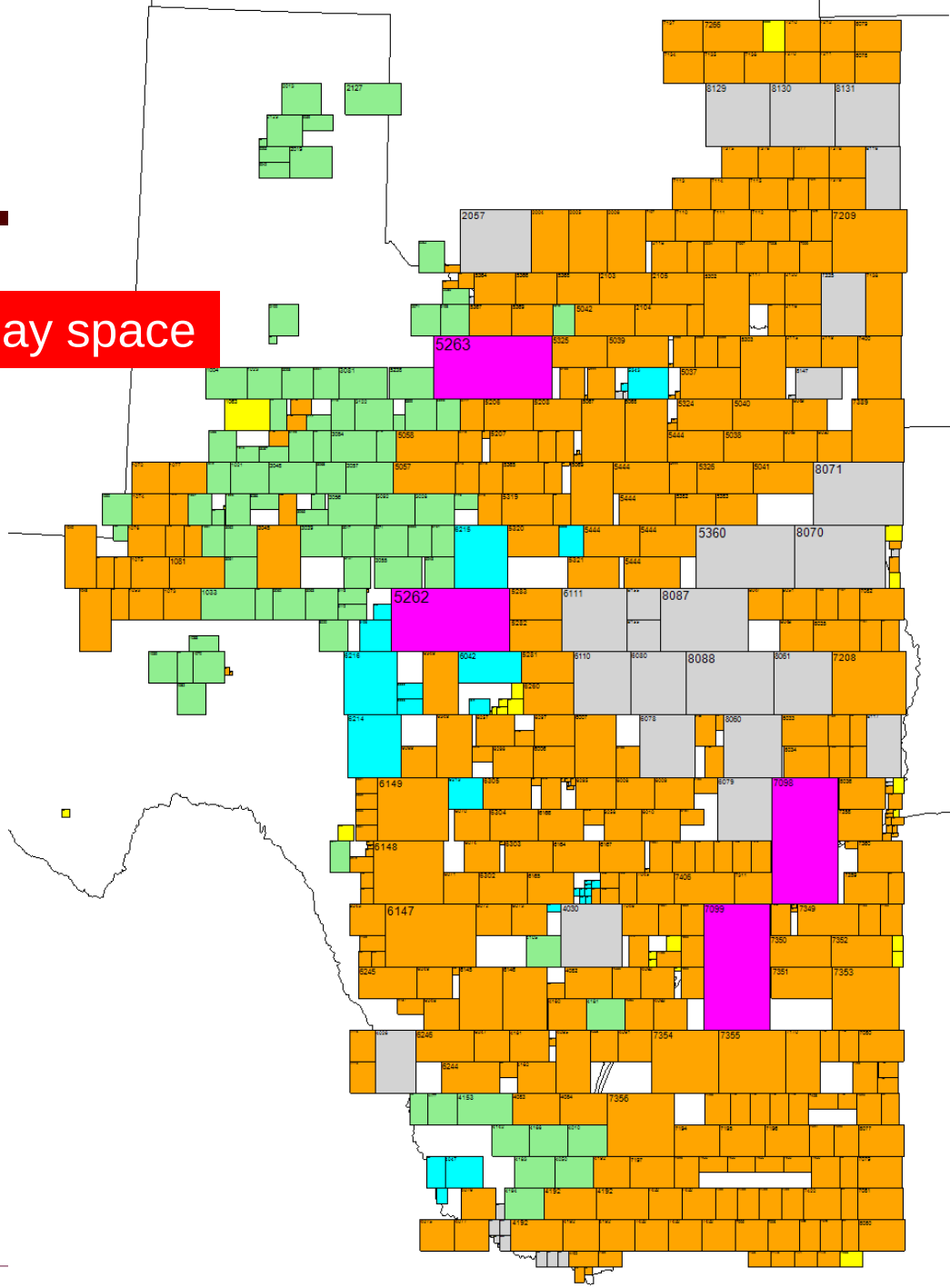
Green – Wind

Yellow – Solar

Example, Horizontal Packing

- Here most of the screen appears filled up, with Houston pushing against the boundary box right, top, and bottom

Using 50% of display space



Showing generators, with size proportional to max MW

Magenta – Nuclear

Gray – Coal

Orange – Natural Gas

Blue – Hydro

Green – Wind

Yellow – Solar

Example, Horizontal Packing

- Make the tiles much larger and the algorithm cannot fit everything on the screen. Only vague geographic relationships are still noticeable (wind west, coal east)

Using 80% of display space

Showing generators, with size proportional to max MW

Magenta – Nuclear

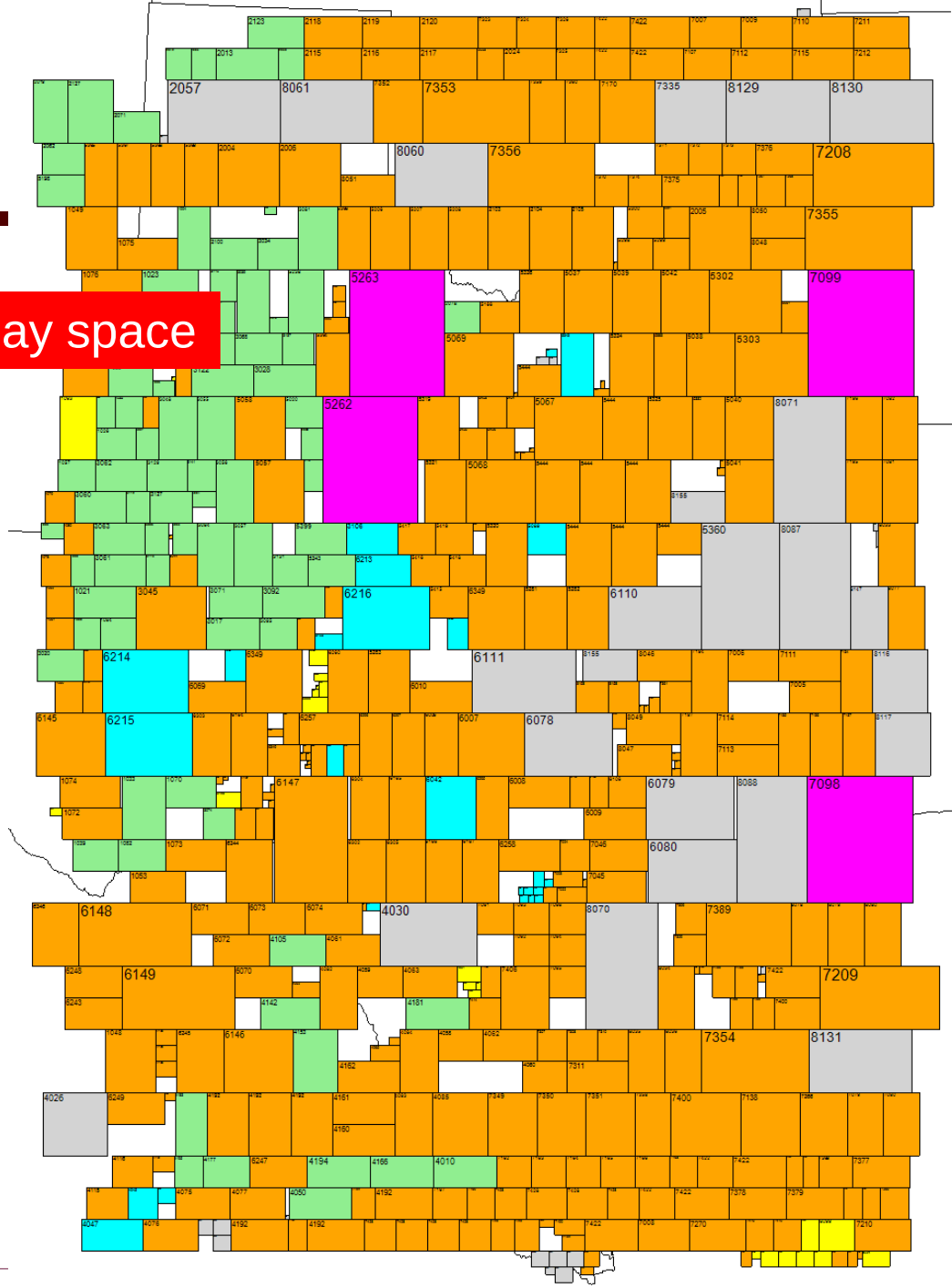
Gray – Coal

Orange – Natural Gas

Blue – Hydro

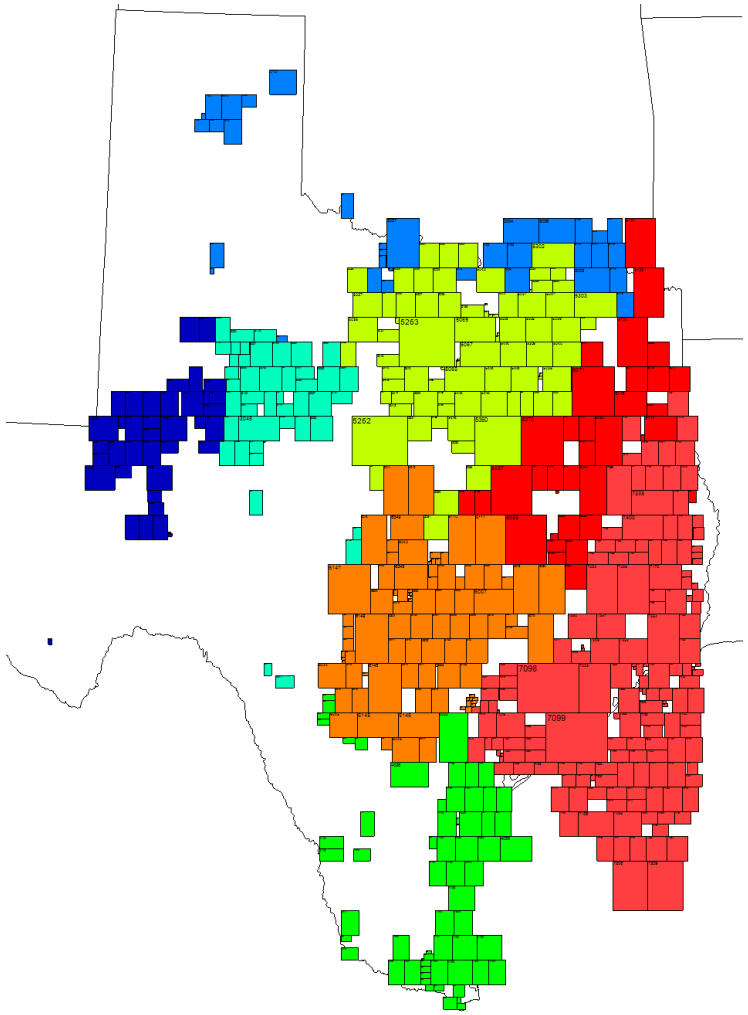
Green – Wind

Yellow – Solar

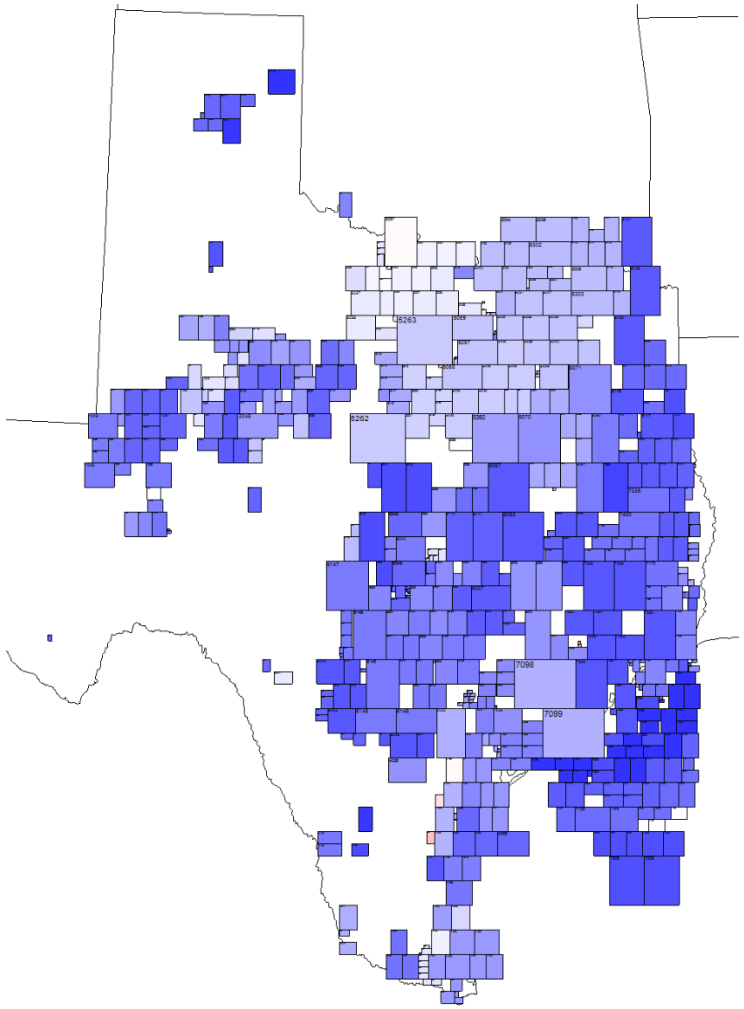


Variations – Data Views

Areas

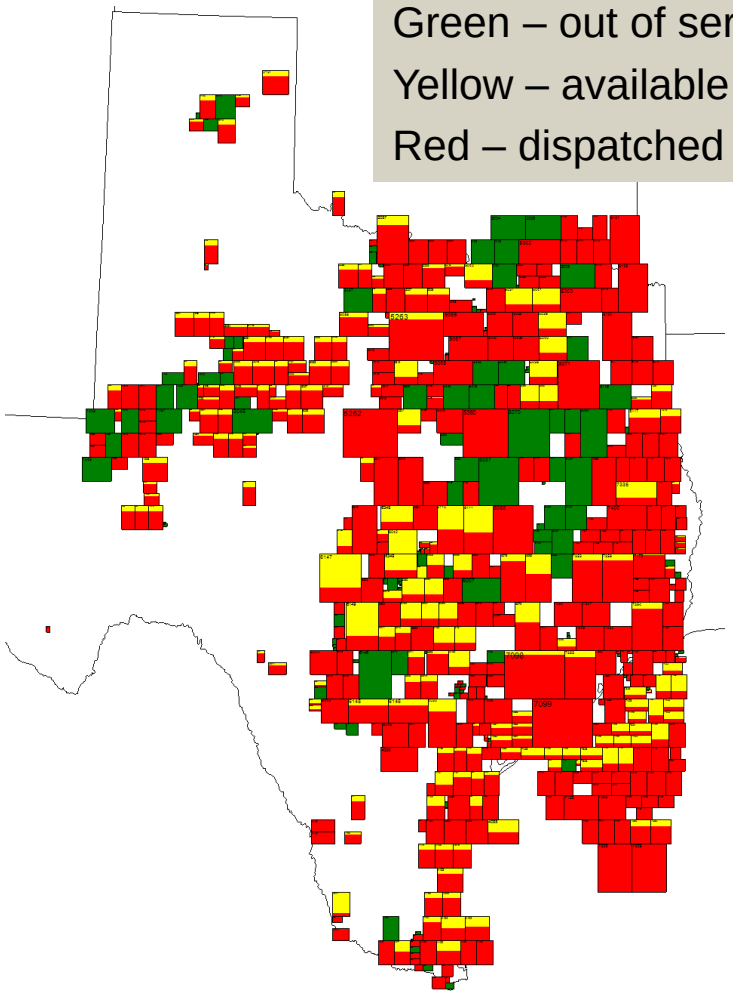


Voltage



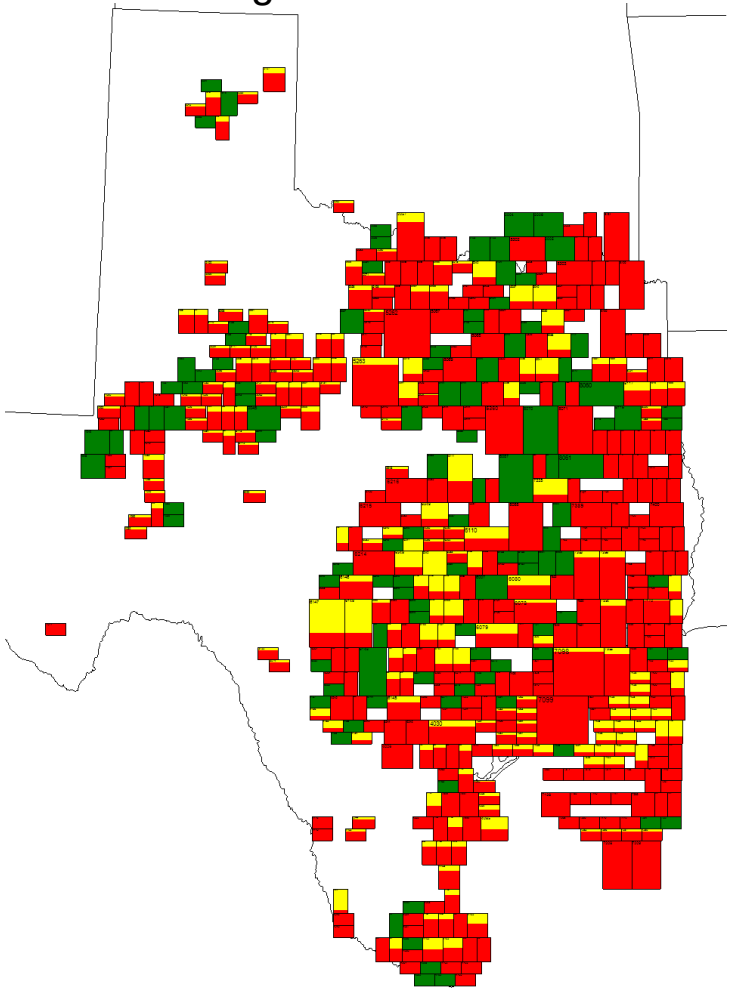
Status + Dispatch

Green – out of service
Yellow – available MW
Red – dispatched MW

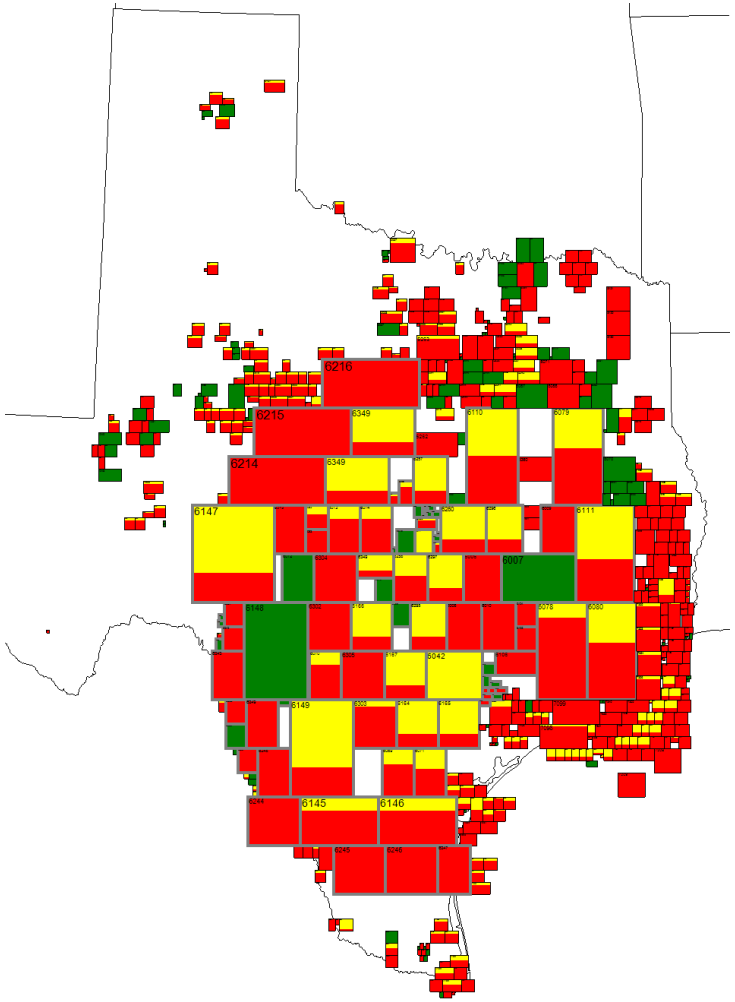


Variations — Presentation

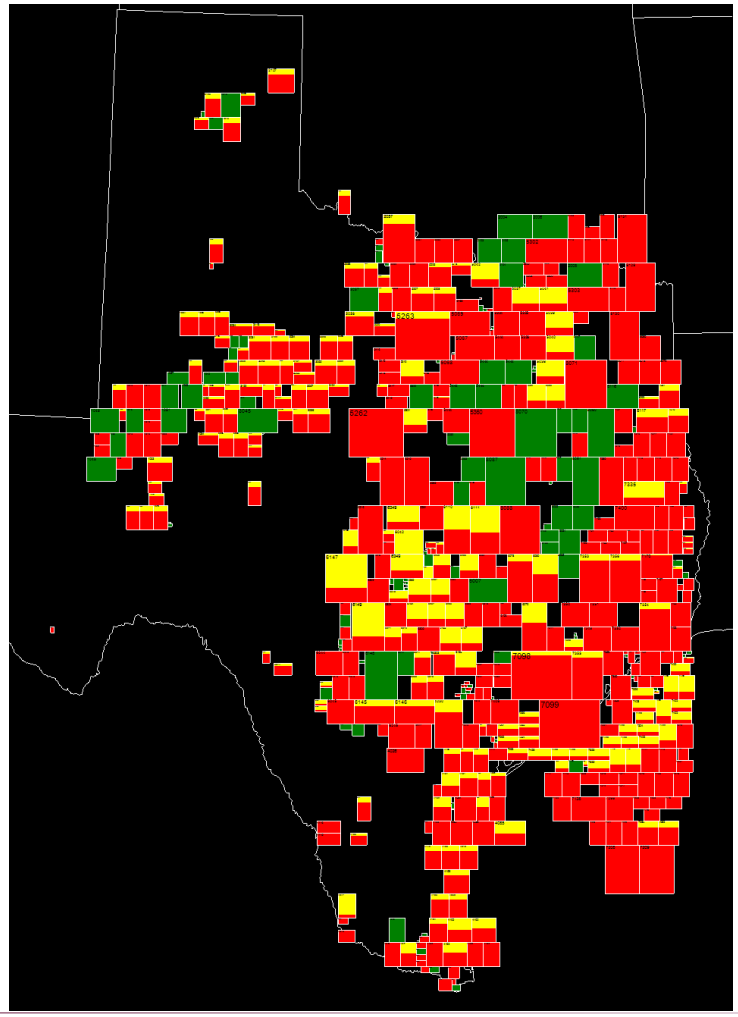
We can enforce a minimum tile size so the little generators can be seen



We can emphasize one area with size and borders



A darker background can be used for control room applications



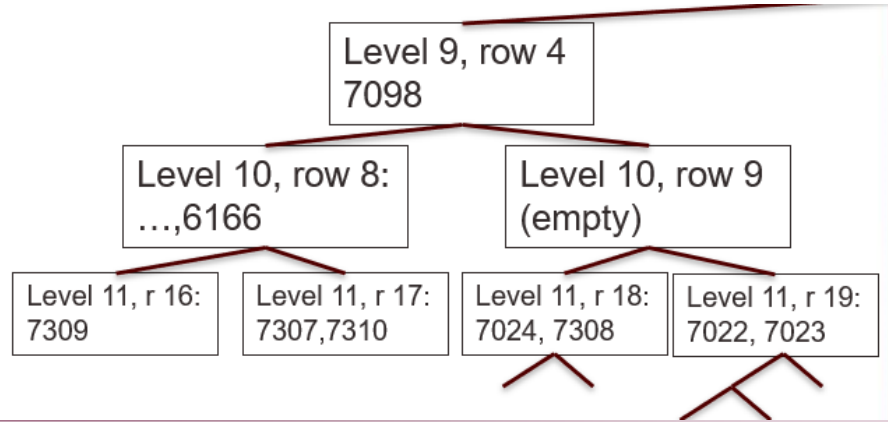
Algorithm Steps in More Detail

- The intuition behind the approach is to set the tiles up to slide easily left and right

Heights and Y values are standardized in powers of 2; widths are variable to make the areas correct and X can be tightly packed

Each tile can be given a width and height that guarantees the aspect ratio between 1:2 and 2:1

Stored on a binary tree as shown below

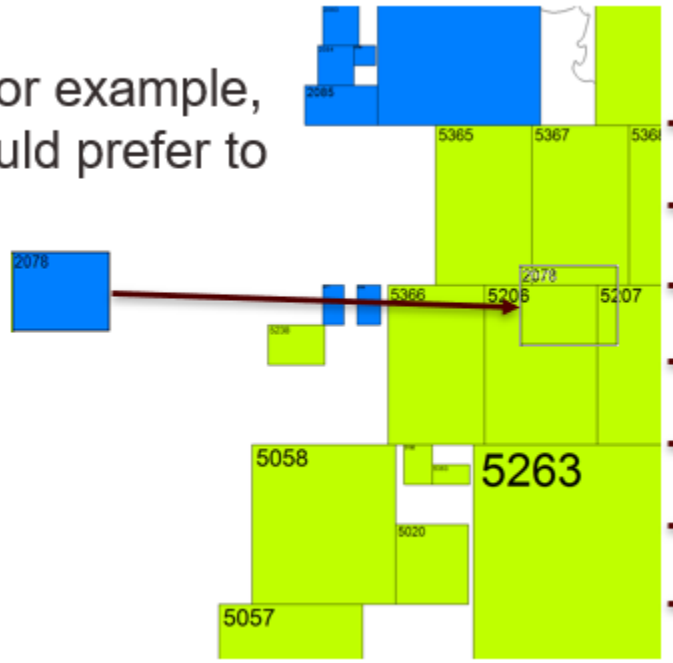


- Each tile is added a two-step process:
  1. Add to the outside (left or right ends)
  2. Push towards the center

Adding a New Tile

- We are packing the display from the center out! (Alternate between left-packing and right)
- Let's take an iteration of adding a tile to the left (process is symmetric for adding to the right)
- The tile to be added will only be added to the left of already-placed tiles

Let's say, for example, this tile would prefer to be here:



There are only a discrete number of rows to check, based on the standard tile height and Y.

Search radially from home (preferred) row.

Radius 2

Radius 1

Home row

Radius 1

Radius 2

Radius 3

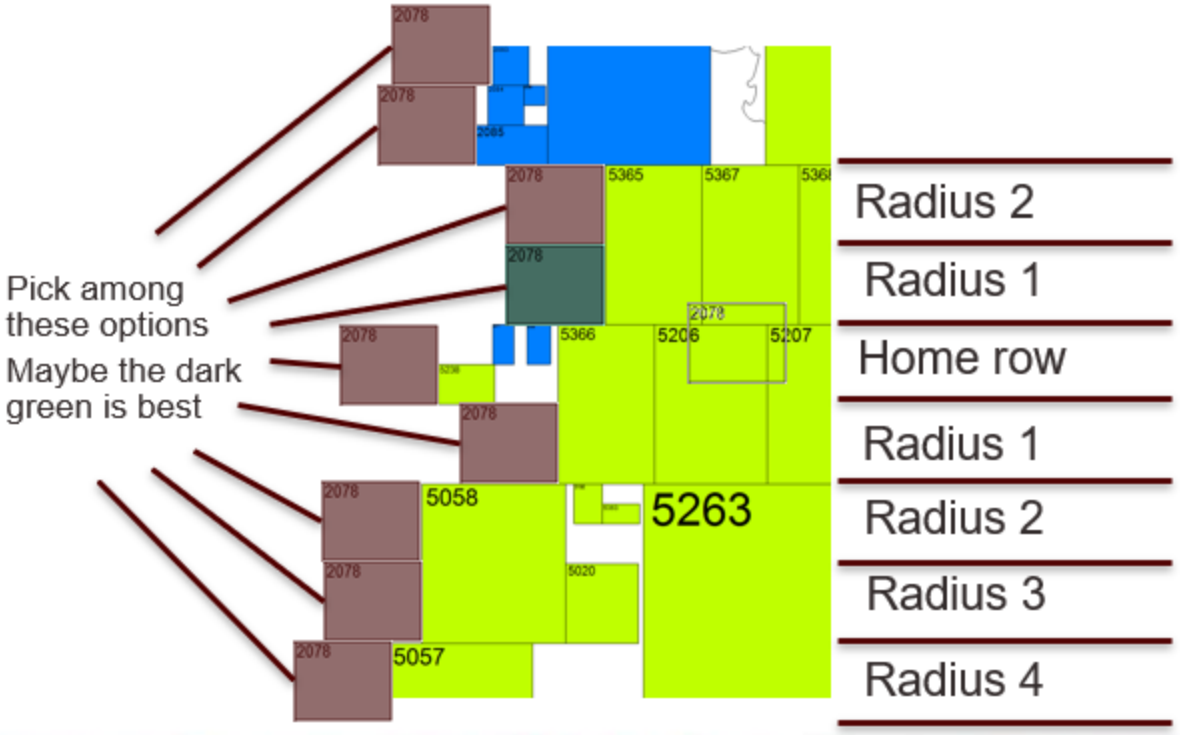
Radius 4

At each row, find what the closest X to preferred location would be, fully to the left of already-placed tiles.

We can stop when we know we won't find anything better by radiating further.

Choosing the Row

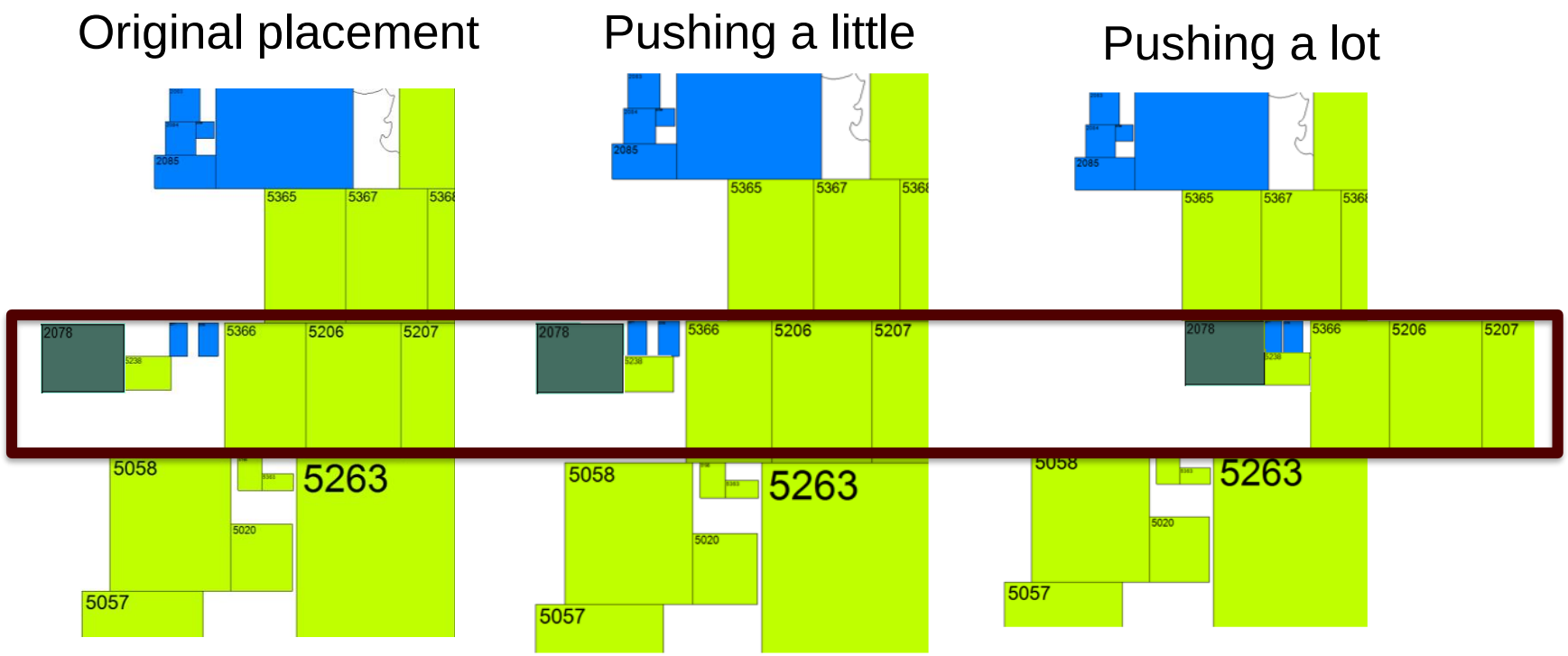
- Choose the row that minimizes the tile's displacement (x-x0)^2 + (y-y0)^2
  - We can usually stop after searching just a few rows
- Add it to the binary tree node for that row



In finding the X value for each tested row, just search the binary tree up and down from the node for that row, only looking at the first tile in each node. Choose the minimum x of these values, minus the width of the tile we are adding.

The Pushing Step

- Once we have placed a new tile on the left side of the diagram (again, right is symmetric), we push it to the right a certain amount.
- We use a golden section line search to find the pushing distance that minimizes the total displacement



"Pushing group"; other tiles are not affected

Properties of These Mosaics

- Tiles can be any area and any preferred location (multiple tiles can prefer same location)
- No tiles will be overlapping
- Tile area will match input exactly
- Width:height ratio will be between 1:2 and 2:1
- Location will be a close as practical to preferred
- If desired, tiles forced within boundary box
- Drawing the mosaic is rapid, and once it is created, any color scheme can be used, hence panning and zooming is fast
  - Theoretical computational order is $n^{1.5}$, experimental scaling matches
  - Detailed algorithm analysis in paper
- Feasible in originally defined problem, sub-optimal

Summary and Concluding Thoughts

- NAE has recognized the importance of new visualization techniques
- Automatic creation of oneline diagrams has a number of applications
  - Synthetic grids
  - Getting a quick look at a case
  - Making a starting point for a high-quality diagram
  - Handling modifications to a case rapidly
- Mosaic diagrams can be used to display several different types of power system data
- Lots of ongoing work in the area!

A. B. Birchfield and T. J. Overbye, "Mosaic packing to visualize large-scale electric grid data," in *IEEE Open Access Journal of Power and Energy*, vol. 7, pp. 212-221, 2020.

A. B. Birchfield and T. J. Overbye, "Techniques for drawing geographic one-line diagrams: Substation spacing and line routing," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 7269-7276, Nov. 2018.

T. J. Overbye, J. Wert, A. B. Birchfield, and J. D. Weber, "Wide-Area Electric Grid Visualization Using Pseudo-Geographic Mosaic Displays," *2019 North American Power Symposium (NAPS)*, Wichita, KS, USA, Oct. 2019, pp. 1-6.