NRECA GRIP Final Report

By David.Pinney@NRECA.coop in September 2020

NRECA's Role in GRIP

- Platform partner: adapt capabilities from OMF.coop to GRIP use cases to provide a comprehensive grid intelligence platform for simulation, data capture, and machine learning.
- Analytics developer: Identify and develop key machine intelligence and resilience analytics.
- Provide test data for validation of said analytics.
- The following slides provide a rough overview of work completed by NRECA under GRIP; additional edits are forthcoming.

Table of Contents

Platform - Weather Data Enhancements

- DNI/DHI solar irradiance estimator
  - Goal: generate DNI/DHI data for years and locations that only have GHI data available
  - Accurately estimate Diffuse Horizontal Irradiance, given a Global Horizontal Irradiance measurement and a cloud cover measurement from darkSky
  - Current approach involves trying to fit numerous statistical models including OLS regression, polynomial regression, and GLS regression with various data transformations
  - Most accurate fits now have MAPE error of c. 30%
- NDFD weather and wind data
  - API to request and parse forecast data from the NDFD
  - Can request by coordinate, coordinate-grid, and zip code
  - Methods support other models on the GRIP platform
- Front end integration in weatherPull
  - Goal: user-friendly interface for weather analytics available on the platform
  - Completed integration for darkSky, planned: tmy3_pull, get_nrsdb_data, get_radiation_data

Platform - Enhanced Weather Data Import, Solar

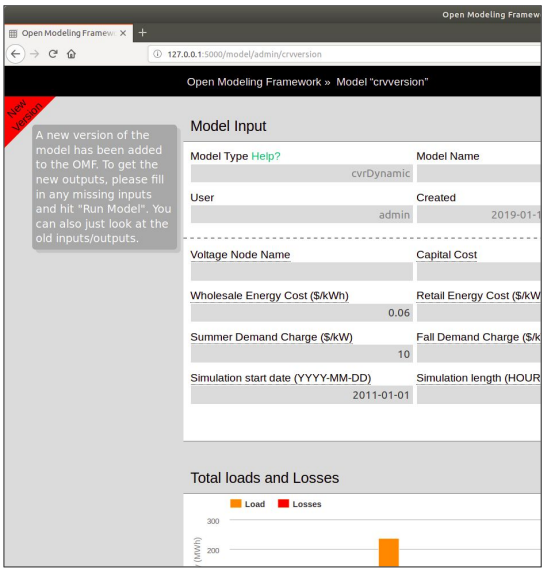
- Functionality to pull solar data from a variety of sources into the OMF.
- Options to download raw data in .csv format or use in memory.
- Sources:
  - NSRDB: physical solar model, physical solar model (tmy3), SUNY international data, spectral (tmy3).
  - Surfrad/Solrad: aggregate either surface or solar radiation data over a year.
  - TMY3: data sets of hourly values of solar radiation and meteorological elements for a 1-year period.

```
class NSRDB():
    def __init__(self, data_set, longitude, latitude, year, api_key, utc='true', leap_day='false'
        self.base_url = 'https://developer.nrel.gov'
        self.data_set = data_set
        self.params = {}
        self.params['api_key'] = api_key
        #wkt will be one point to use csv option - may need another call to get correct wkt value
        self.params['wkt'] = self.latlon_to_wkt(longitude, latitude)
        #names will be one value to use csv option
        self.params['names'] = str(year)
        #note utc must be either 'true' or
        self.params['utc'] = utc
        self.params['leap_day'] = leap_day
        self.params['email'] = email
        self.interval = interval

    def latlon_to_wkt(self, longitude, lati
        if latitude < -90 or latitude > 90:
            raise('invalid latitude')
        elif longitude < -180 or longitude
            raise('invalid longitude')
        return 'POINT({} {})'.format(longitu

    def create_url(self, route):
        return os.path.join(self.base_url,

    #physical solar model
    def psm(self):
        self.params['interval'] = self.inte
        route = 'api/solar/nsrdb_psm3_downlo
        self.request_url = self.create_url(

    #physical solar model v3 tsm
    def psm_tmy(self):
        route = 'api/nsrdb_api/solar/nsrdb_
        self.request_url = self.create_url(

    #SUNY international
    def suny(self):
        route = 'api/solar/suny_india_downlo
        self.request_url = self.create_url(
```

USAF	Site Name	State	Latitude	Longitude	TZ	Elev	Class	Pool
690150	TWENTYNINE PALMS	CA	34.3	-116.167	-8	626	II	12
690190	ABILENE DYESS AFB	TX	32.433	-99.85	-6	545	II	12
690230	WHIDBEY ISLAND NAS	WA	48.35	-122.667	-8	10	II	12
699604	YUMA MCAS	AZ	32.65	-114.617	-7	65	II	12
700197	SELAWIK	AK	66.6	-160	-9	8	III	11
700260	BARROW W POST-W ROGERS ARPT [NSA - ARM]	AK	71.32	-156.62	-9	10	II	24
700637	DEADHORSE	AK	70.2	-148.483	-9	23	II	12
701043	POINT HOPE (AWOS)	AK	68.35	-166.8	-9	4	II	12
701195	SHISHMAREF (AWOS)	AK	66.267	-166.05	-9	2	II	12
701330	KOTZEBUE RALPH WEIN MEMORIAL	AK	66.883	-162.6	-9	3	II	24
701625	ANAKTUVUK PASS	AK	68.133	-151.733	-9	657	III	11
701718	AMBLER	AK	67.1	-157.85	-9	88	II	12
701740	BETTLES FIELD	AK	66.917	-151.517	-9	196	II	24
701780	TANANA RALPH M CALHOUN MEM AP	AK	65.167	-152.1	-9	71	II	12
701940	FORT YUKON	AK	66.567	-145.267	-9	136	III	10
702000	NOME MUNICIPAL ARPT	AK	64.517	-165.45	-9	4	II	24
702005	SAINT MARY`S (AWOS)	AK	62.067	-163.3	-9	95	II	12
702035	SAVOONGA	AK	63.683	-170.5	-9	17	III	12
702040	GAMBELL	AK	63.783	-171.75	-9	8	III	10
702070	UNALAKLEET FIELD	AK	63.883	-160.8	-9	5	III	11
702075	ANVIK	AK	62.65	-160.183	-9	99	III	10
702084	EMMONAK	AK	62.783	-164.5	-9	4	II	12
702185	MEKORYUK	AK	60.367	-166.267	-9	15	II	12
702186	HOOPER BAY	AK	61.517	-166.15	-9	6	II	12
702190	BETHEL AIRPORT	AK	60.783	-161.833	-9	38	II	24
702225	HUSLIA	AK	65.7	-156.383	-9	55	III	10
702310	MCGRATH ARPT	AK	62.95	-155.6	-9	105	II	24
702320	ANIAK AIRPORT	AK	61.583	-159.533	-9	26	III	11
702460	MINCHUMINA	AK	63.883	-152.283	-9	213	II	12
702495	HAYES RIVER	AK	61.983	-152.083	-9	305	II	12
702510	TALKEETNA STATE ARPT	AK	62.317	-150.1	-9	105	II	24
702590	KENAI MUNICIPAL AP	AK	60.583	-151.233	-9	26	II	12
702595	SOLDOTNA	AK	60.467	-151.033	-9	34	III	12
702600	NENANA MUNICIPAL AP	AK	64.55	-149.1	-9	109	II	12
702606	CHULITNA	AK	62.883	-149.833	-9	381	II	12
702607	HOONAH	AK	58.083	-135.45	-9	6	II	12

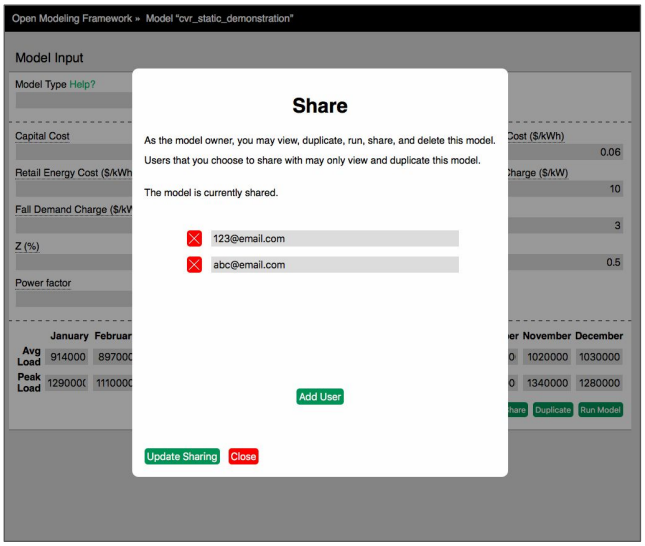
Platform - OMF Application Model Versioning

- Goal: never show mismatch between output data and code/interface.
- OMF automatically checks if a model's code or interface has been updated via the following:
  - Using SHA-256 algorithm, hashes of a model's .py and .html files are saved in the model output data
  - When viewing a saved model, the hashes in the output data are compared against the current hashes of the model .py and .html files
  - If there is a mismatch, the user sees an unobtrusive flag explaining that the model code or interface has been changed
- Users have the option to rerun a model to view any additional outputs that the new version of the model produces
- Users may also continue to use their existing outputs if necessary for their work



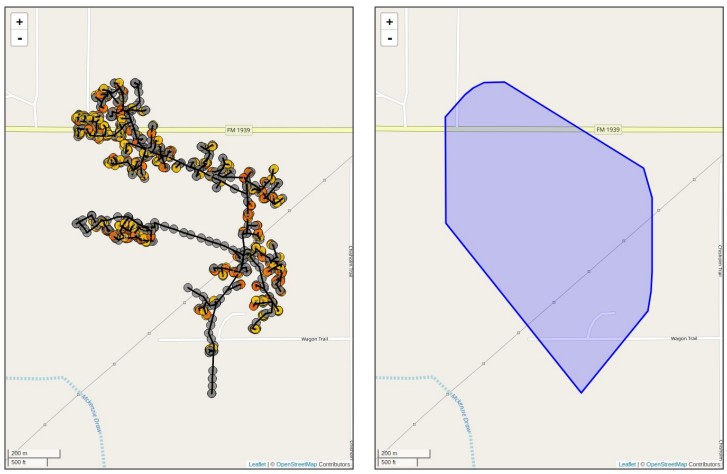
Platform - OMF Access, Sharing Enhancements

- The OMF now features a new granular sharing mechanism that gives model owners more control over who they share with.
- Up to this point, a model owner could only share their model with another user by publishing the model so that it could be viewed by all users as a public model.
- The publish feature has now been replaced with a model-sharing feature that allows a model owner to share a read-only view of their model with specified users.
- The new sharing mechanism is supported by more strict user authorization checks across all of the OMF web application routes.



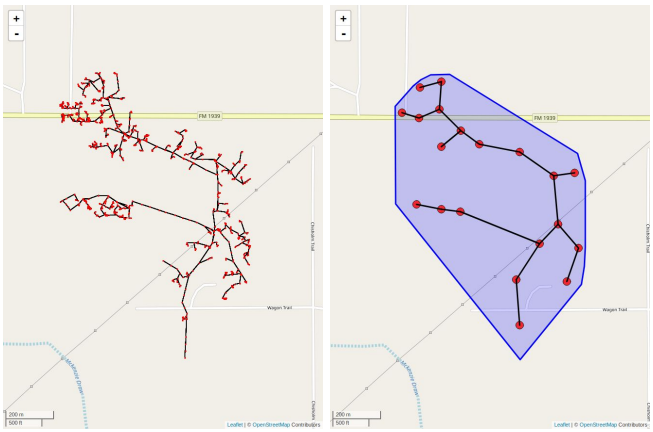
Platform - Enhanced Geospatial Capabilities

- Goal: allow circuit data to be visualized along with a map. Road and building locations are critical information for many utility workflows.
- OMF supports .glm, Milsoft Windmil, .omd, and Cymdist circuit formats through our toolchain.
- Specific geospatial capabilities developed:
  - Convert circuit coordinates to geoJSON compliant format
  - Display a vector circuit diagram on an interactive leaflet map
  - Calculate the convex hull of a large circuit, represented as a geoJSON polygon which can be displayed on a leaflet map (for visualizing multiple large circuits via partial loading)



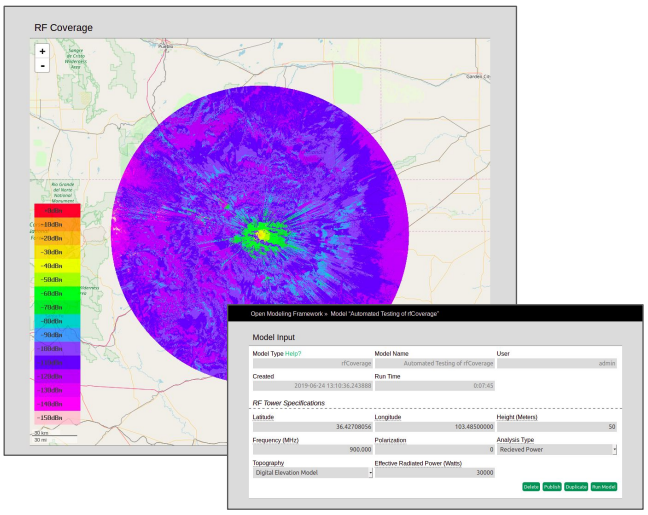
Platform - Enhanced Geospatial Capabilities pt. 2

- Specific geospatial capabilities (continued):
  - Create and serve raster tiles to view extra-large circuits down to 50 meter resolution
  - Simplification an ultra-large circuit represented as interconnected cluster centers calculated via k-means clustering method
  - Trace the shortest path between two nodes on a circuit e.g. from a component to substation, to enable fault analysis applications
  - Static (.png) and interactive (.html) options for users to save circuit visualizations on a map
- Next steps: integrate with GRIP HTTP API.



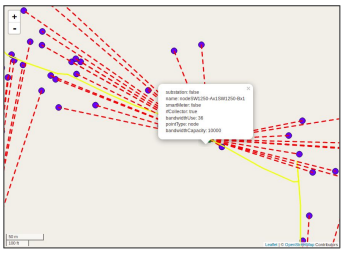
Platform - Communications Modeling

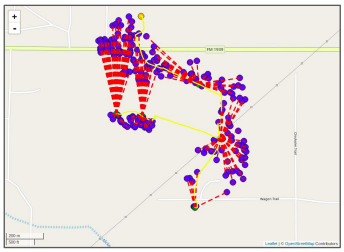
- rfCoverage model provides an interactive visualization of radio frequency coverage from a tower to assist with communications planning
- User defined inputs include latitude, longitude, tower height, frequency, polarization, effective radiated power
- Model leverages SPLAT! cli tool to generate an image of radio frequency coverage
- Multiple analysis types from basic line of sight to advanced analysis with scales for path loss, field strength, and received power
- Model takes into interference from varying elevations, but can also be run with a ‘sea level’ elevation assumption for a quick estimate
- The image is overlaid on an interactive leaflet map

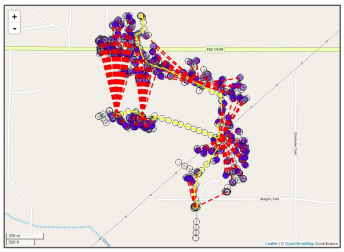


Platform - Communications Modeling pt. 2

- Comms.py tools to model a communications network on top of an existing feeder:
  - Geojson based schema (.omc).
  - Path tracing for bandwidth and cost calculations for RF and fiber components.
  - Comms objects are linked by name to circuit objects to be displayed on same leaflet map.
- Initial communications network defined with fiber from substation to switches and RF collectors with RF to meters.





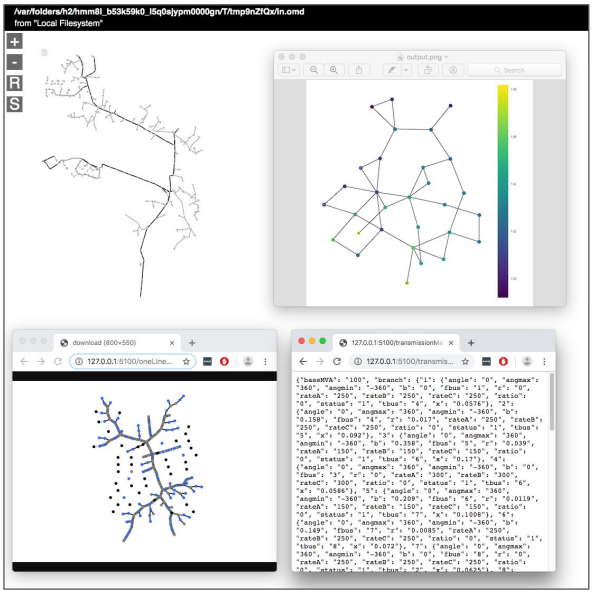


Platform - OMF-GRIP API

- Use Docker to serve a standalone web API of the OMF in any configuration, such as on a local network or on the internet.
  - The API can be consumed through any client interface, not just through a web browser.
  - Clients can start, cancel, and download the results of an operation.
  - Clients can monitor the progress of an operation through JSON responses.
  - The API returns the results of an operation as JSON, a PNG, a ZIP, or plain text.
- Docker and the underlying Gunicorn web server ensure that the API is reliable and simple to host.

Platform - OMF-GRIP API pt. 2

- The API allows access to 11 different functions of the OMF:
  - Import and convert Milsoft WindMil CYMDIST, and MATPOWER files to run in GridLAB-D.
  - Run and visualize MATPOWER transmission and generation simulations.
  - Run NREL's system advisor model.
  - Run LANL's General Fragility Model
  - View and edit transmission and distribution feeders as local files

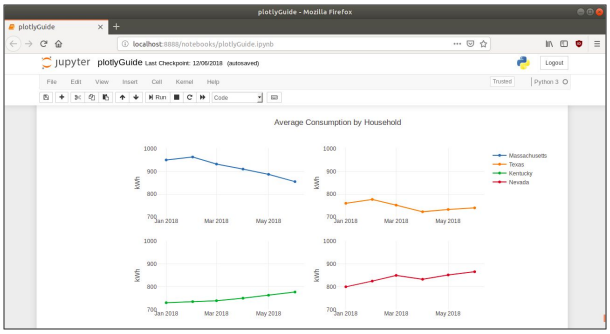


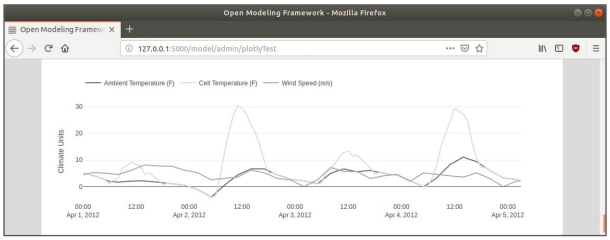
Platform - OMF-GRIP API pt. 3

- Built HTTP APIs for key OMF functionality
- Containerized OMF deployment enhanced to allow GRIP platform to control dependencies and performance
- Full test suite to insure API integrity
- PresencePG confirmed successful integration of the OMF container in to the GRIP platform in December 2018
- Route functionality enabled:
  - Conversion of Cymdist and Windmil models to GridLAB-D
  - Visualization and running of GridLAB-D models
  - Translation and running of GridLAB-D models in the micot-gfm damage modeling framework
  - Interface to NREL System Advisor Model for renewable modeling
- Future work:
  - Expand API routes to include new functionality (from this and future quarters)
  - Widget-based API routes for embedding large chunks of OMF functionality in to GRIP (e.g. cost analysis of an interconnection, dispatch analysis of a battery)
- Code location: https://github.com/dpinney/omf/tree/2f6207eeb9ed2d3710d9a98dde9a5906fff903e9/omf/scratch/GRIP
- Test procedure: see "grip.Dockerfile" and "test.py" at above code repository.

Platform - Charting Enhancement (Plotly Integration)

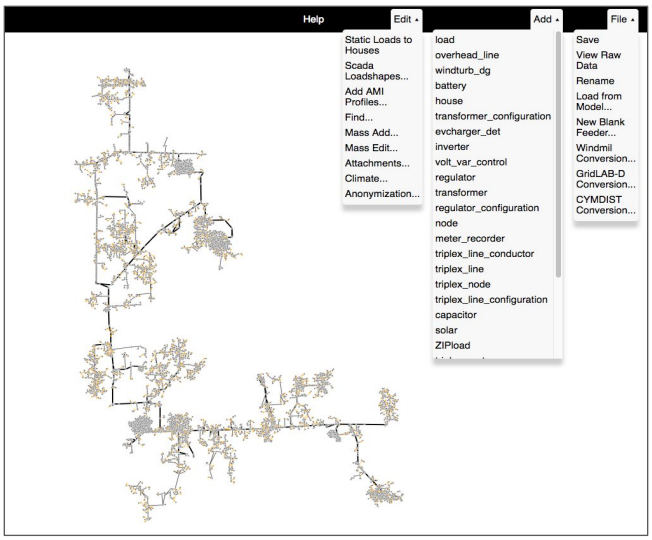
- Key visualization issue: existing platform charting libraries (matplotlib, highcharts) lack interactivity, programmer usability, or have inflexible licenses.
- Prototyped plotly replacements for all OMF charting with plotly.
- Superior geospatial charting capabilities when compared to current D3 code.
- Replacements have better licensing terms.
- Enables easier integration between frontend and backend with supported JavaScript interface.
- Also allows for enhanced offline ease of use and functionality.
- Code location: https://github.com/dpinney/omf/blob/2f6207eeb9ed2d3710d9a98dde9a5906fff903e9/omf/scratch/plotlyInfo/plotlyGuide.ipynb





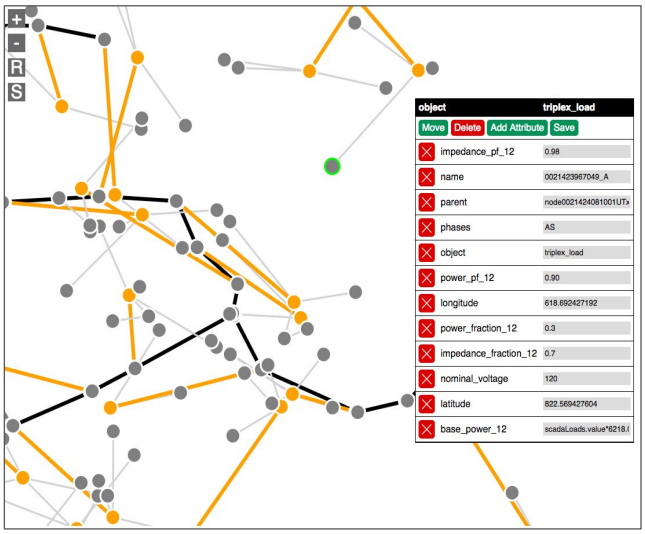
Platform - Distribution Circuit Viz. and Editing

- Circuit visualization and editing key to most distribution engineering analysis tasks.
- Developed editor with support for circuits with >= 10k nodes.
- Added support for very large coordinate values >= 15m
- Feature parity with legacy interface:
  - Edit/add/move objects in the circuit, rescale, and zoom in and out.
  - Apply climate models, scada calibration, ami calibration, and anonymization.
  - Import data from Milsoft Windmil, GridLAB-D, and CYMDIST formats



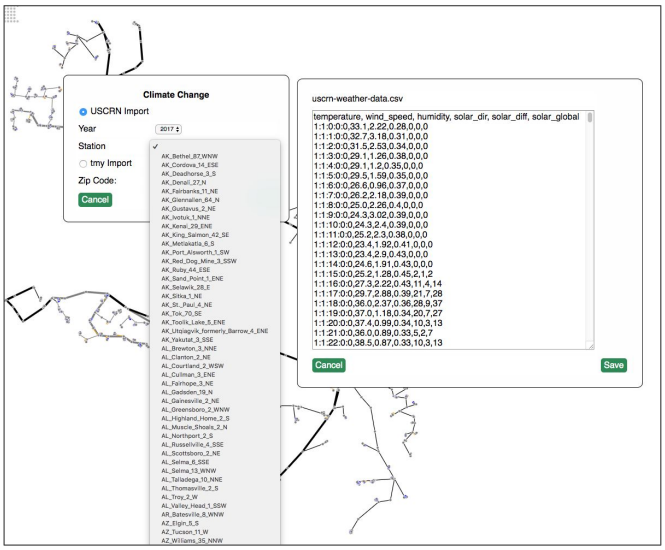
Platform - Distribution Circuit Viz. and Editing Pt. 2

- Render to single file option for offline debugging and file sharing.
- Responsive interface.
  - Simultaneously edit ~25k rendered svg nodes with acceptable performance in the browser.
- Decreased rendering time.
  - Depending on the circuit, rendering time has decreased to between 5% to 25% of the original time, which saves several seconds.
- Code location:
  - Frontend: https://github.com/dpinney/omf/blob/2f6207eeb9ed2d3710d9a98dde9a5906fff903e9/omf/templates/distNetViz.html
  - Backend: https://github.com/dpinney/omf/blob/2f6207eeb9ed2d3710d9a98dde9a5906fff903e9/omf/distNetViz.py
- Test it in production:
  - https://omf.coop/distribution/admin/q10techReport/1



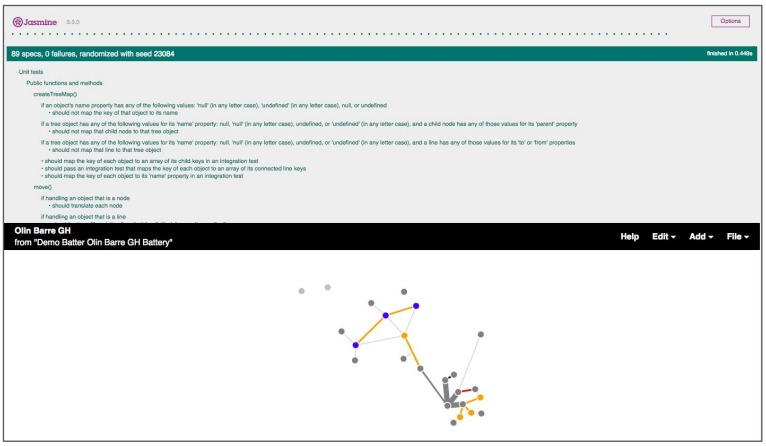
Platform - Distribution Circuit Viz. and Editing Pt. 3

- Import USCRN weather data:
  - Import quality controlled weather data generated by 156 United States Climate Reference Network stations from 2000 - present.
  - Run GridLAB-D with a feeder and specific weather data from any chosen year and station.
- Improved user interface:
  - Long-running server operations can be cancelled through the interface instead of having to wait until an operation completes.
  - More robust process and file control on the server to accommodate process cancellation without server instability.



Platform - Distribution Circuit Viz. and Editing Pt. 4

- Full feature parity with existing editor:
  - Import WindMil, CYMDIST, Grid-LAB-D files into the interface.
  - Pseudonymize and anonymize imported data.
  - Apply SCADA loadshapes and AMI profiles to the data.
- Refactor of core:
  - Used Jasmine unit testing library in-browser.
  - Interface is more stable.

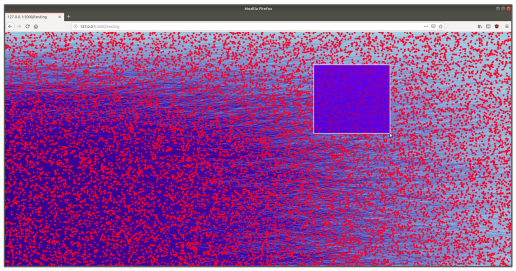


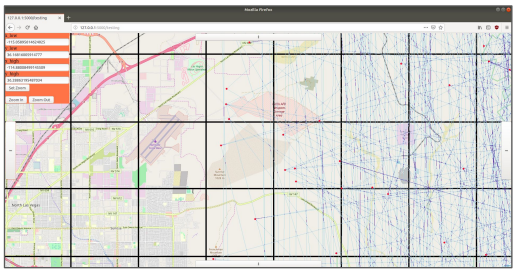
Platform - Distribution Circuit Viz. and Editing Pt. 5

- Component Database:
  - Added detailed hardware models from real specs.
  - Includes 12 houses, 113 overhead line conductors, 256 transformer configurations, and more.
  - Database will continue to expand.
- Enhancements:
  - Intuitive UI for adding objects.
  - Improved validation of user-entered data.
  - Changing opacity of objects for search
  - Coloring based on voltage, current, etc.

Platform - Ultra-Large Scale Circuit Viz. (Datashader Integration)

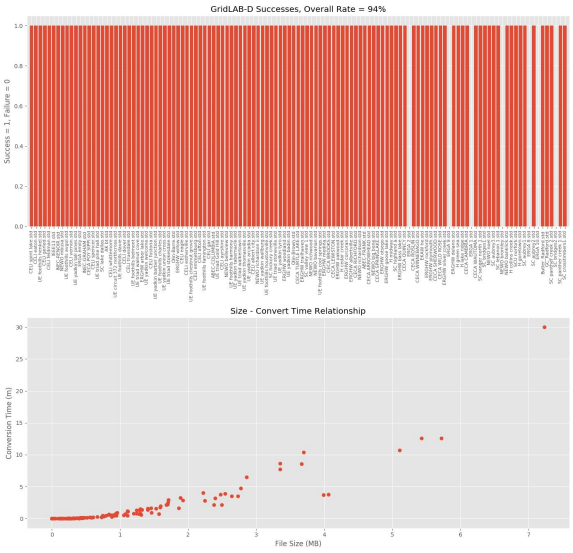
- Key grid visualization issue: fully integrated T&D networks can have millions of elements but existing interactive visualizations struggle around 10k elements, lack customizability, or involve extremely heavy tech stacks.
- Custom flask app built that uses datashader (python library)
  - Displays a visual representation of a circuit with nodes as points and edges as lines
  - User controls for interactivity to explore the graph with zoom, pan, and box-select
  - Overlays graph on a map with leaflet.js
  - Datasets work well up to 100k nodes, with datasets of 1 million points usable (20 seconds on each re-render).
- Code location: https://github.com/dpinney/omf/tree/e9ac4cf95650ce8f5441f508e99c4dbdbd3c582a/omf/scratch/dataShader
- Test procedure: see "zoomGraph.py" in above repository.





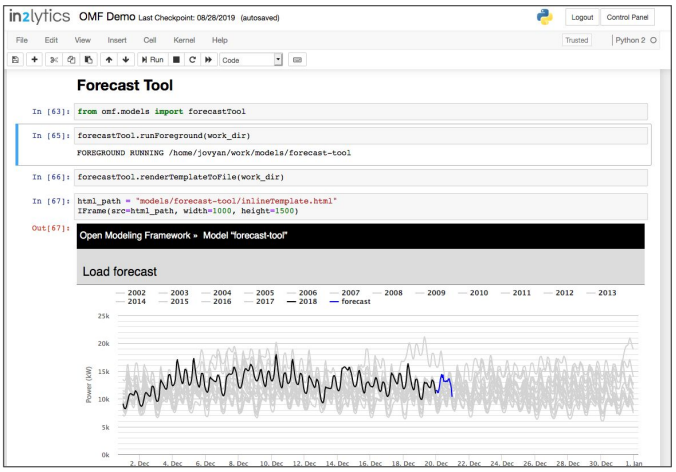
Platform - Circuit Import and Conversion

- Circuit import/conversion a key platform feature.
- Pre-GRIP, OMF.coop conversion success rate for Windmil was 5%, i.e. only 5% of circuits ran powerflow after conversion.
- Test suite enhanced and automated for 100 representative circuits.
- Current success rate 94%.
- Cymedist conversion testing limited by lack of data. Existing code was extensively refactored in anticipation of data partner recruitment.



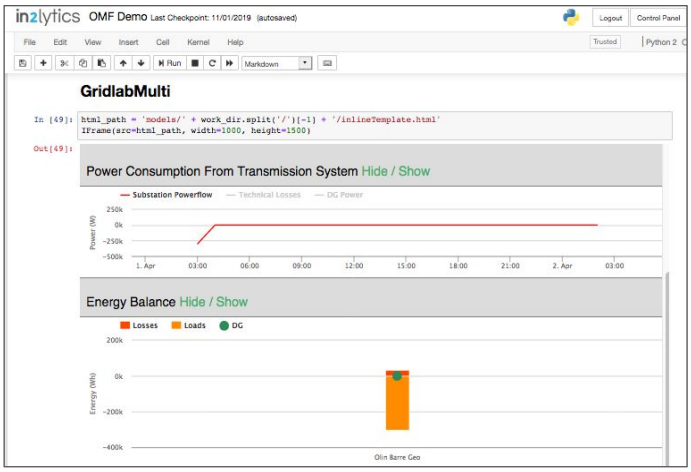
Platform - Realtime Data Ingest Demo (GridState)

- The majority of the GRIP analytics can benefit from feeds of real time sensor data (from e.g. SCADA, AMI, etc.).
- We deployed the GRIP OMF analytics inside a realtime data capture platform called "GridState".
- This integration demonstrates that the modeling functions of the OMF can reach a wider audience by providing an easy and flexible deployment option that combines a Jupyter notebook with an OMF scripting API.
- The flexibility of this deployment option will make it easier for users to load data into the OMF from a wide variety of sources, such as the realtime data supplied by GridState.
- Future work will include studying the benefits of ingesting realtime data with GRIP analytics.



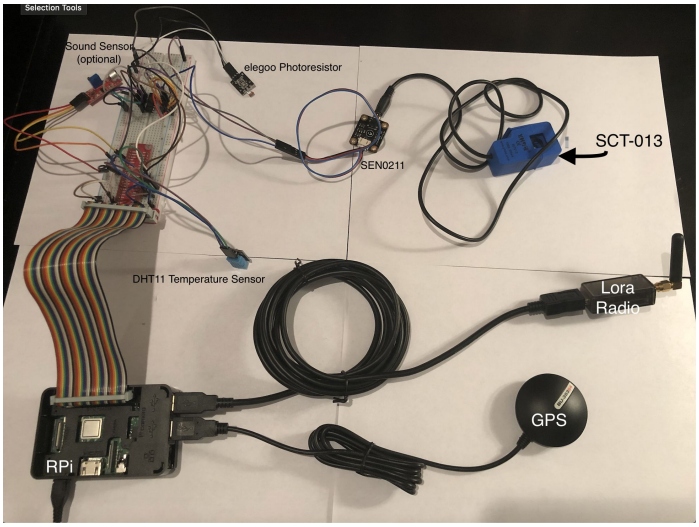
Platform - Realtime Data Ingest Demo (GridState) Pt. 2

- One benefit of ingesting realtime data with GRIP analytics is that power flow analysis can be performed in near realtime.
- Understanding the benefits of realtime power flow analysis to utilities is a large area of interest.
- Future work will include studying additional benefits of ingesting realtime data with GRIP analytics.



Platform - Realtime Data Ingest Demo (LCH)

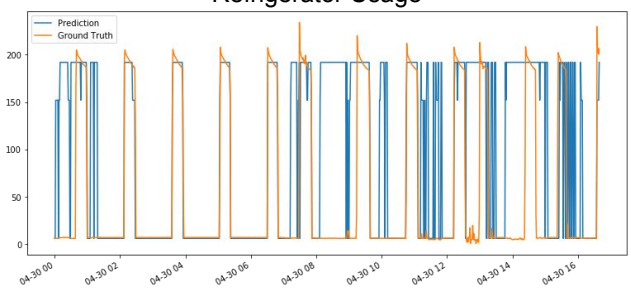
- Alternate realtime ingest mechanism: direct data capture from low-cost hardware platform
- BOM Cost: $53.91 (Raspberry Pi, SCT013, MCP3008 ADC)
- Sample Rate and Data Resolution: 15K-17K samples per second
- Data transmission:
  - LoRa usable for ~1 sample/second with 3-5 mile range
  - Wifi/GSM required for full sample rate
- Full investigation: https://docs.google.com/presentation/d/1PKk2rHSGdkpF1wqHeMnnWnu3GVldkLB-y3YhvCuiGF0/edit#slide=id.g6b5bcf6f15_0_6



Anticipate (ML) – Load Disaggregation

- Goal: Perform disaggregation on collected data using publically available appliance labeled data for model training so that we may inform consumers of their consumption habits, and develop better load models for distribution planning activities
- Methods: Used the open-source tool NILMTK and the REDD data set from MIT to train models for disaggregation prediction and validate performance given known labels
- Conclusions: disaggregation prediction seems feasible
- Work is ongoing to fit disaggregation models to other publically available datasets and perform disaggregation analysis on collected data

Refrigerator Usage

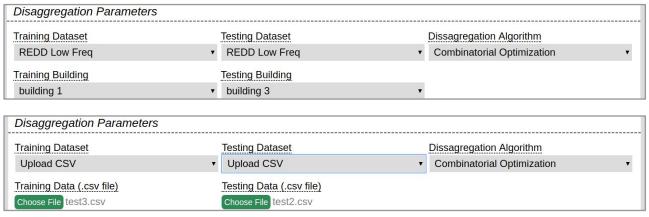


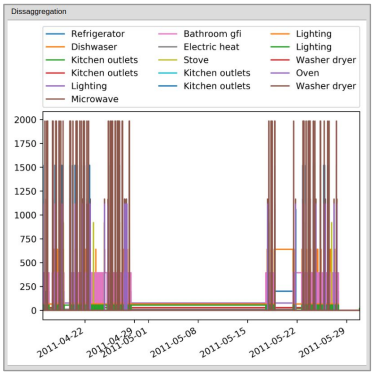
Dish washer	192.064718
Fridge	111.039010
Light	102.623030
Microwave	170.115434
Sockets	37.054990

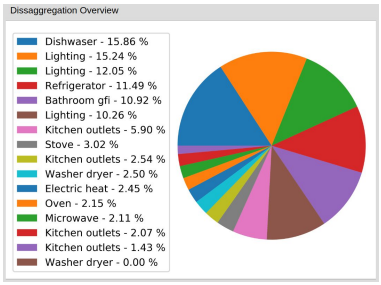
Root Mean Squared Error for the top 5 appliances in the dataset

Anticipate (ML) - Load Disaggregation Pt. 2

- Breaks down provided power consumption data from a given meter into the constituent appliances.
- Relies on training data to make accurate predictions and returns both an overall appliance level disaggregation and a disaggregation over time.
- Users can use provided sample training and test data or upload their own as CSV files
- Utilizes the open source *NILM toolkit* for a bulk of the data analysis processing
- Can help inform consumers who are inadvertently using a ton of energy about their consumption patterns.
- Can lead to better load models for distribution planning

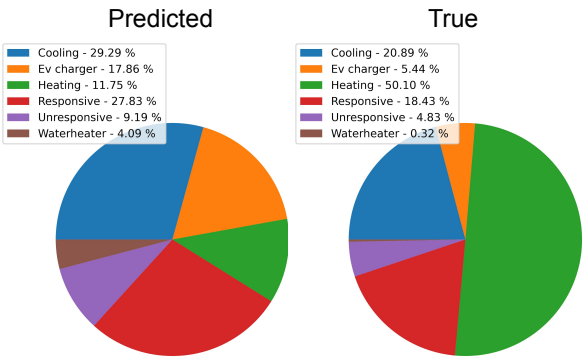




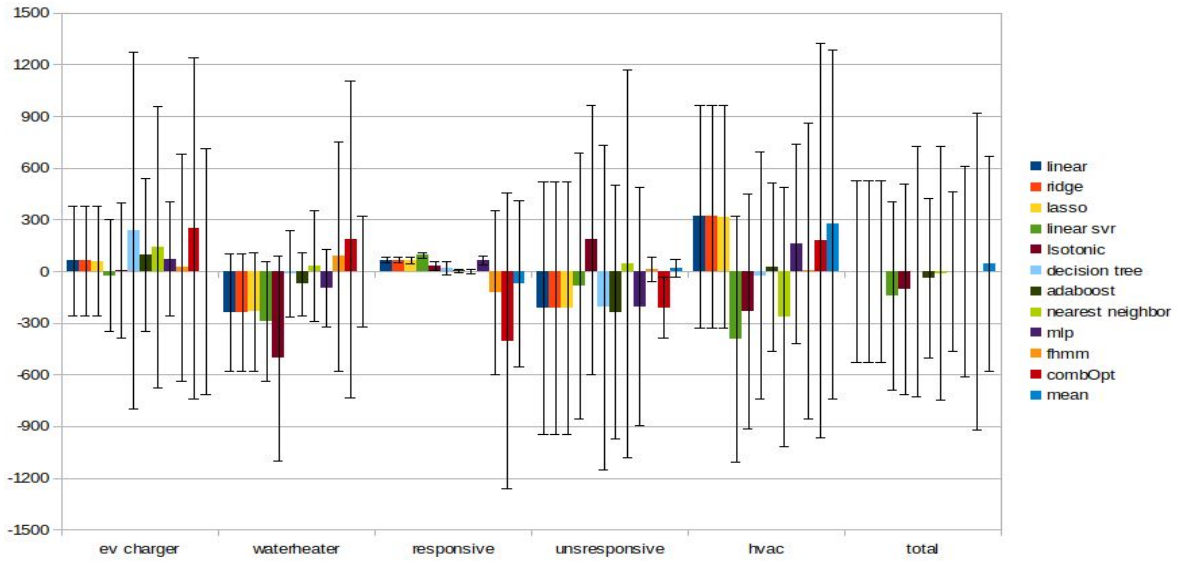


Anticipate (ML) - Load Disaggregation Pt. 3

- We use Gridlab-D to simulate loads and meter data for a single household
- 90% of the data is used as training data for NILMTK disaggregation algorithms
- NILMTK makes a prediction on the remaining 10%
- Predictions are compared to the simulated ground truth to quantify prediction errors
- As shown at right, NILMTK predictions are very inaccurate



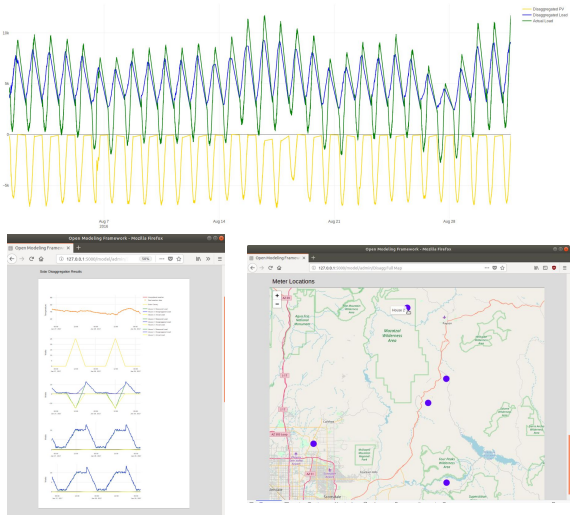
Anticipate (ML) - Load Disaggregation Pt. 4



Advanced work: NILMTK alternatives for disaggregation, working on choosing model, feature engineering.

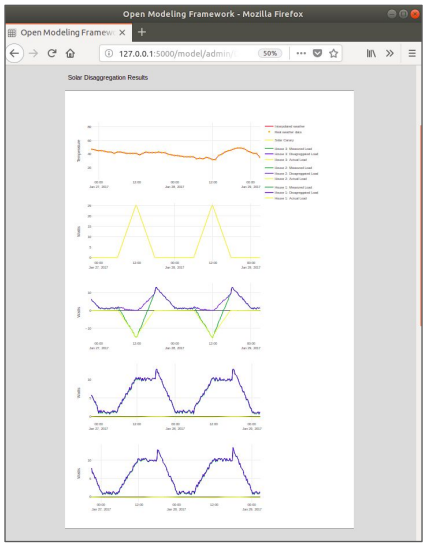
Anticipate (ML) - Solar Disaggregation Model

- Solar Disaggregation model CSSS integrated in to OMF code base. (CSSS = Contextually Supervised Source Separation).
- CSSS gives utilities the ability to detect solar installations on their systems that were not disclosed to the utility
- Model Inputs:
  - User uploads 3 .csv files, enters a start date, and selects an ASOS station
  - Net load for each house measured at 15 minute intervals
  - Solar proxy load measured at 15 minute intervals
  - Latitude/Longitude coordinates for each house and solar proxy
  - ASOS station, based on the nearest city
- Model outputs:
  - Temperature data points from ASOS station and temperature interpolated to 15 minute intervals for CSSS
  - Line graph of solar proxy load input
  - Line graph of each net load, with hidden solar and estimated actual load from CSSS disaggregation
  - Leaflet.js map with marker clusters for each net load source, solar proxy, and ASOS station



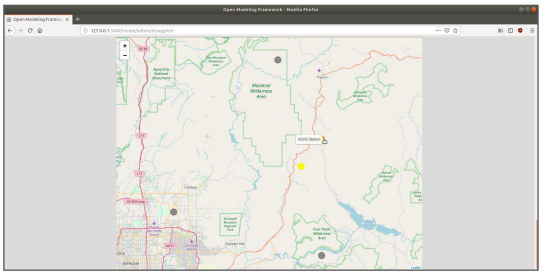
Anticipate (ML) - Solar Disag. Utility Application

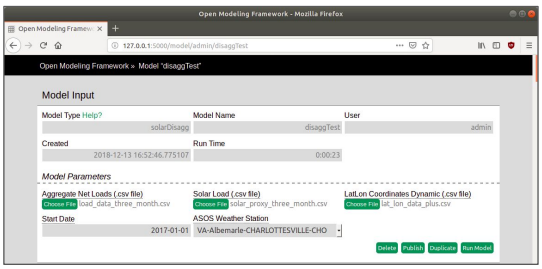
- CSSS is the disaggregation of a time series of source signals from observations of their sum
- Model outputs:
  - Temperature data points from ASOS station and temperature interpolated to 15 minute intervals for CSSS
  - Line graph of solar proxy load input
  - Line graph of each net load, with hidden solar and estimated actual load from CSSS disaggregation
  - Leaflet.js map with marker clusters for each net load source, solar proxy, and ASOS station
- Future work: expose interface on platform production interface, expand test datasets.
- Code location:
  - Frontend: https://github.com/dpinney/omf/blob/2f6207eeb9ed2d3710d9a98dde9a5906fff903e9/omf/models/solarDisagg.html
  - Backend: https://github.com/dpinney/omf/blob/2f6207eeb9ed2d3710d9a98dde9a5906fff903e9/omf/models/solarDisagg.py
- Test it in production:
  - https://omf.coop/newModel/solarDisagg/gripQ4Report



Anticipate - Solar Disag. Utility Application Pt. 2

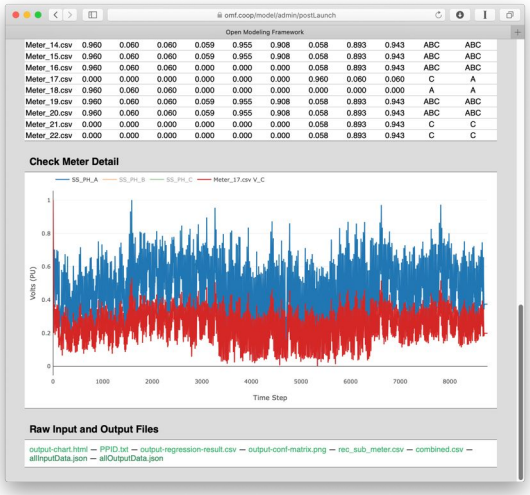
- Solar Disaggregation model CSSS integrated in to OMF code base.
- CSSS gives utilities the ability to detect solar installations on their systems that were not disclosed to the utility
- Developed web-based frontend to allow utility staff to easily set up and run model
- Model Inputs:
  - User uploads 3 .csv files, enters a start date, and selects an ASOS station
  - Net load for each house measured at 15 minute intervals
  - Solar proxy load measured at 15 minute intervals
  - Latitude/Longitude coordinates for each house and solar proxy
  - ASOS station, based on the nearest city





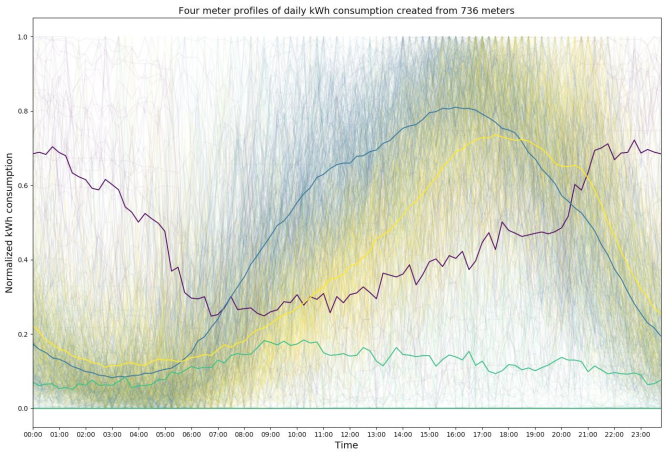
Anticipate (ML) - Power Phase Identification

- Problem: phase of customer meters sometimes incorrect, leads to energy loss, decreased motor efficiency
- Solution: model that uses meter readings to predict their phase
- Implementation: correlation method comparing meter voltages to substation voltages over time
- Results 100% correct for real and simulated test data
- Working with Horry EC to deploy the model.



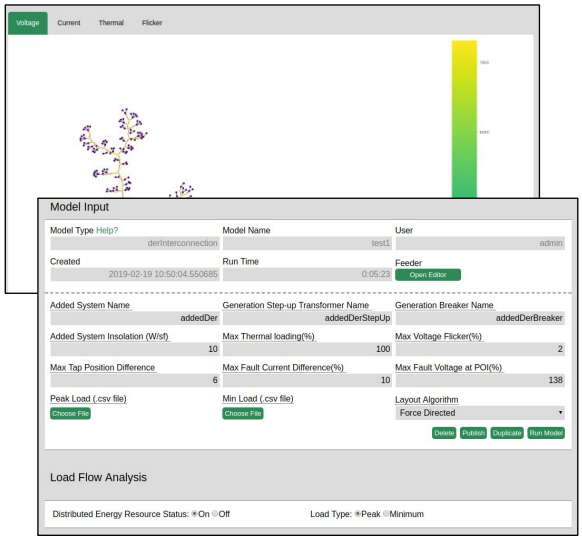
Anticipate (ML) – AMI Meter Data Clustering

- The objective of meter data clustering is to quantitatively identify relative kWh consumption patterns that are shared by subsets of utility customers.
- The K-means clustering algorithm is used to group meters together based on how their relative consumption varies during the day.
- By identifying kWh consumption patterns, we can help utilities perform market segmentation to optimize how they serve customers.
- For example, distinct segments can receive targeted rates.
- The initial investigation shows that we can qualitatively identify meaningful, distinct consumption patterns.
- The result of fitting four clusters to the data is shown. This suggests that there are at least four daily consumption patterns that are distinct enough to be considered market segments.



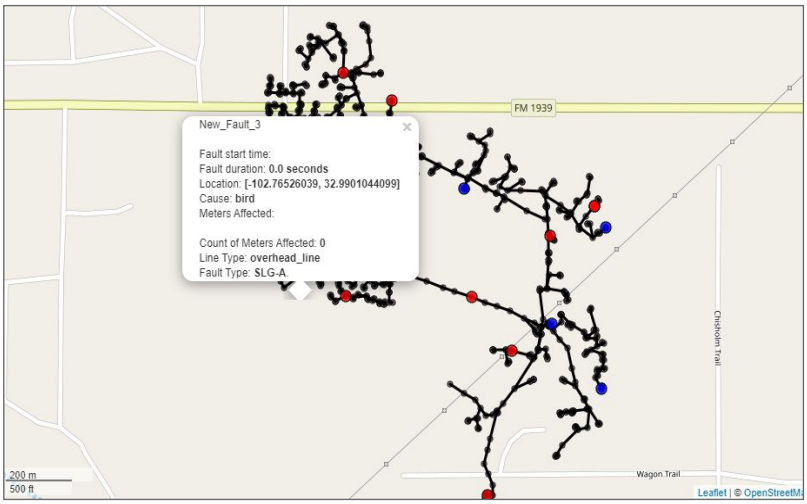
Anticipate - DER Resilience Analytics

- Automated screening process to assess impacts of DERs on resilience/reliability of arbitrary distribution circuits.
- Visualizations for load flow analysis
- Supplementary sortable tables for flicker, reverse power flow, thermal, short circuit, and fault current violations
- Allows distribution engineers to quickly discover problems with the DER without expending resources in manual interconnection modeling (multiple powerflows and circuit changes calculated automatically).



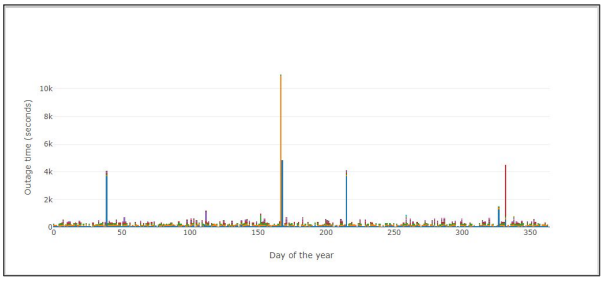
Anticipate - Outage Cost Analysis

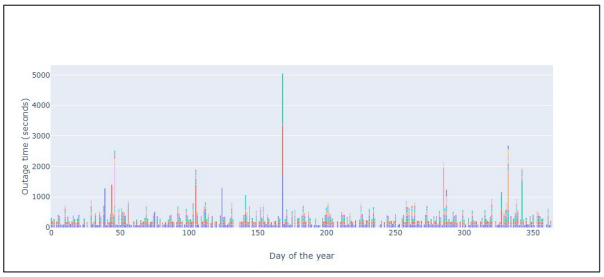
- Goals:
  - Visualize outage data over the period of a year (with filtering based on duration, date the outage occurred, meters affected, and cause of the outage),
  - Calculate the reliability metrics associated with the outage data and a feeder system specified by an .omd file.
  - Generate a new set of outages and calculate the reliability metrics and graph this new data (with filtering as well).
- Methods implemented:
  - Leaflet (geoJSON format).
  - Lattice heat map (for outage location generation)
  - Scipy distributions (for outage duration generation)



Anticipate - Outage Cost Analysis pt. 2

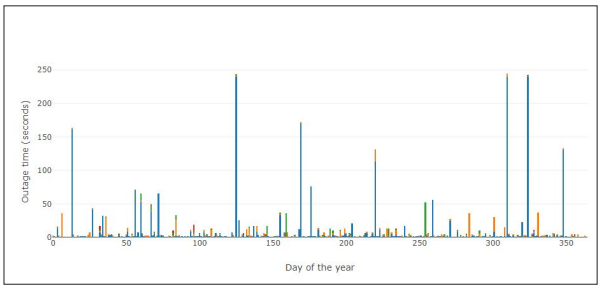
- Test data
  - Olin Barre Fault
  - ieee37nodeFaultTester
- Conclusions
  - Data generation and visualization methods can be used to help utilities predict the distribution of future faults and associated reliability metrics.
  - The variety of different fault generation methods that can be employed give utilities the freedom to fine-tune parameters based on expected dependencies between variables (location, cause, fault type, start time, and duration) which become inputs to other resilience models.
  - Core XAIXI calculations useful for validation of other models that impact these metrics.

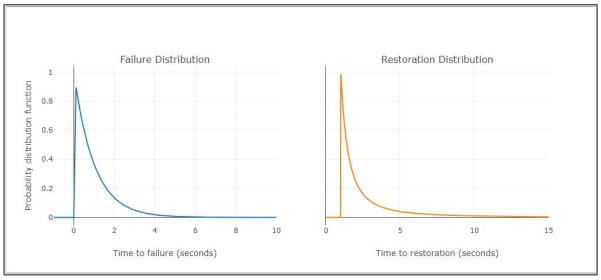




Anticipate - Smart Switching

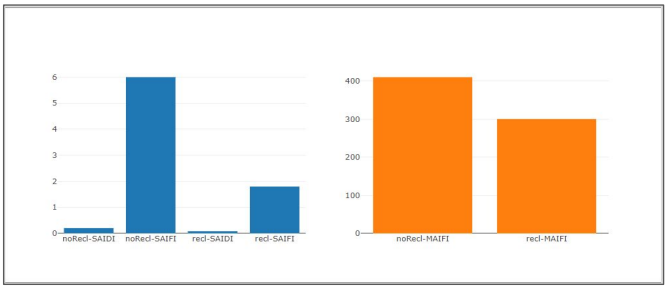
- Goal: Provide utilities with a way to measure the benefit of adding a recloser to a distribution circuit on a given line.
- Model generate random faults throughout the period of one year provided some fault distribution and a maximum outage length, then compare SAIDI/SAIFI/MAIFI values and outage costs generated from the original circuit and from the circuit with an added recloser, taking into account a threshold for what is classified as a sustained outage.
- Tools used:
  - GridLab-D.
  - Random fault generator.
  - Reliability metrics object.
  - Plot.ly for graphing.

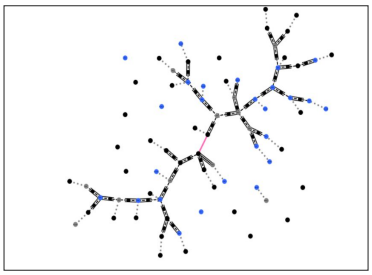




Anticipate - Smart Switching Pt. 2

- Test data:
  - test_ieee37nodeFaultTester.glm
- Conclusion: There is evidence that adding a recloser to a distribution circuit can help cut costs in both residential and commercial areas, depending on the placement of the recloser. It is useful to run the model multiple times with the same input data, to take into account the potential variance of outages in a given year.

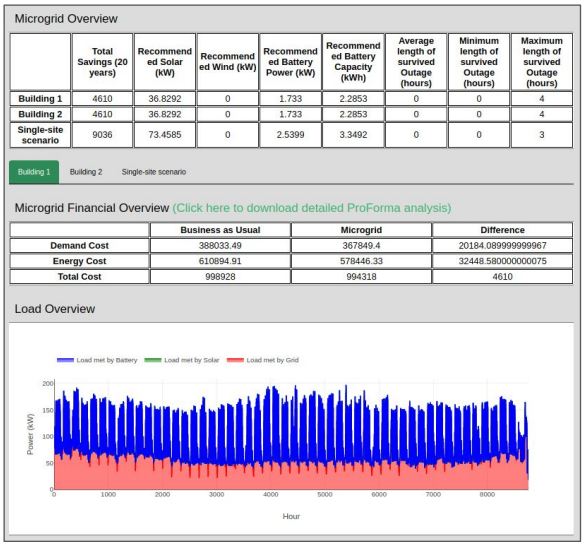




	No-Recloser	Recloser
Lost kWh Sales	14408	5950
Restoration Labor Cost	2350	2227
Restoration Hardware Cost	416	404
Outage Cost	17174	8581

Anticipate - Microgrid Design Tool

- Goal: Given load shapes over a 1yr time period, determine what combination of solar + wind + batteries increases resiliency and minimizes cost.
- Methods: NREL provides an API that can be used to optimize for economics and resiliency.
- Our design allows for single-site or multi-site microgrid design. Multi-site analysis provides results both if:
  - each site is treated as a separate microgrid
  - if a single microgrid were to support all sites
- Results include recommended technology size, detailed financial analysis (proforma), time-series plots for solar and wind generation, as well as solar and wind and battery consumption. Basic power outage survivability information is also provided.
- Allows load operator (a utility or a building owner) to determine the requirements for and benefits of, establishing a microgrid for their specific loads


Microgrid Overview

	Total Savings (20 years)	Recommended Solar (kW)	Recommended Wind (kW)	Recommended Battery Power (kW)	Recommended Battery Capacity (kWh)	Average length of survived Outage (hours)	Minimum length of survived Outage (hours)	Maximum length of survived Outage (hours)
Building 1	4610	36.8292	0	1.733	2.2853	0	0	4
Building 2	4610	36.8292	0	1.733	2.2853	0	0	4
Single-site scenario	9036	73.4585	0	2.5399	3.3492	0	0	3

Building 1 | Building 2 | Single-site scenario

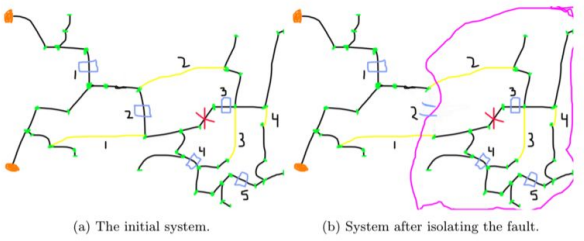
Microgrid Financial Overview (Click here to download detailed ProForma analysis)

	Business as Usual	Microgrid	Difference
Demand Cost	388033.49	367849.4	20184.089999999967
Energy Cost	610894.91	578446.33	32448.580000000075
Total Cost	998928	994318	4610

Load Overview

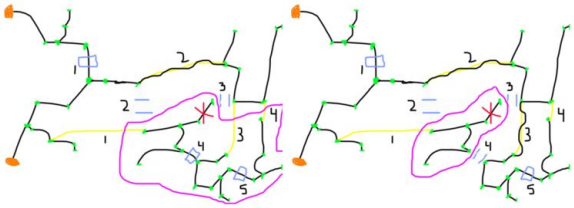
Absorb - FLISR

- Goal: Quickly isolate a fault and restore power in a distribution system by opening and closing switches already present in the system.
- Methods implemented:
  - Adjacency list representation of connectivity
  - Maximal connected subgraphs
  - Depth-first search
- Test Data:
  - Modified Oline Barre Fault (with 16 initially open tie lines and 15 initially closed reclosers)
- Conclusion: The FLISR algorithm provides sub-second restoration for a system with ~3000 nodes, and a proof is provided that the solution is optimal for both radial and non-radial feeder systems. The algorithm has the potential to drastically reduce the amount of time it takes to find an optimal solution and implement it in a faulted system.



(a) The initial system. (b) System after isolating the fault.

Figure 1: Beginning to solve a system with our algorithm.

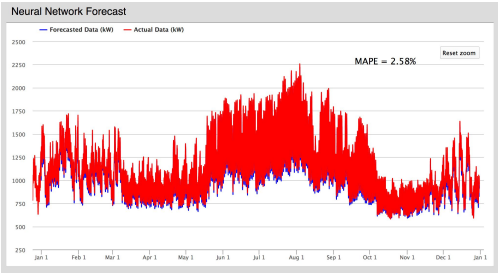


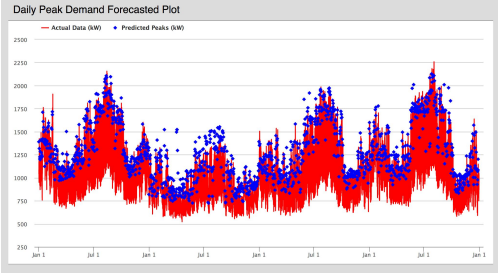
(a) After one tie line is closed. (b) The solved system.

Figure 2: Continuing to find a solution.

Absorb (ML) – Load Forecasting

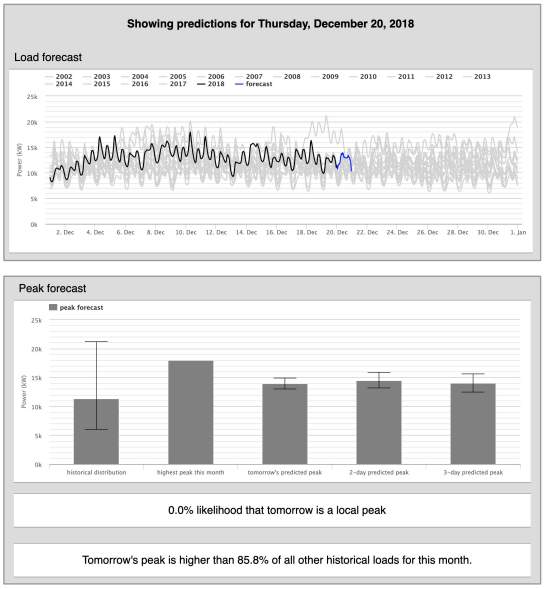
- Goal: provide open, free load forecasting methods to utilities. Forecasts basis of many load control and energy storage dispatch tasks, and these tasks are critical in resilient design and restoration activities.
- Methods implemented:
  - Artificial Neural Network
  - Rolling Time-series Regression (ARIMA)
  - Double Exponentially Smoothed ARIMA
  - Daily Peak Support Vector Regression
- Test data: ERCOT North Region 2002–06
- Conclusion: strong results for all methods, neural network forecast had clearest day-ahead hourly MAPE success (2.58%).





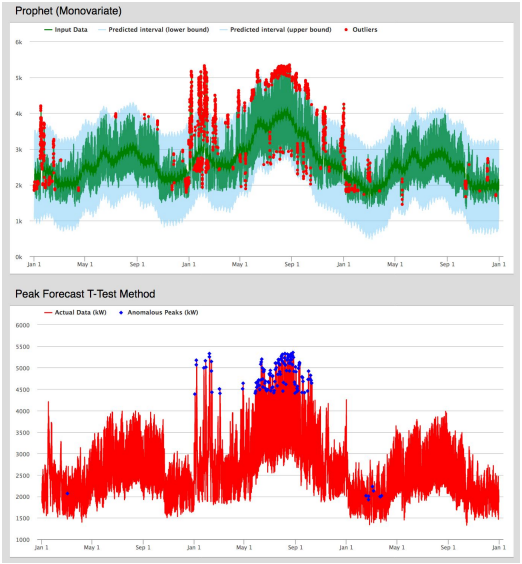
Absorb (ML) – Load Forecasting Pt. 2

- A utility may decide to invest in batteries or a direct load control program to improve system resilience; however, these efforts are only as effective as the utility's ability to predict the day's energy consumption.
- Forecasting methods we developed were integrated in to a model giving statistical likelihood that tomorrow will be the monthly peak.
- Detailed description of the method: https://towardsdatascience.com/an-electric-utilitys-3-part-guide-to-peak-shaving-with-neural-networks-de5c7752d946
- Conclusion: testing on ERCOT data showed the model could find a once-weekly dispatch strategy that would only miss the monthly peak every few years.



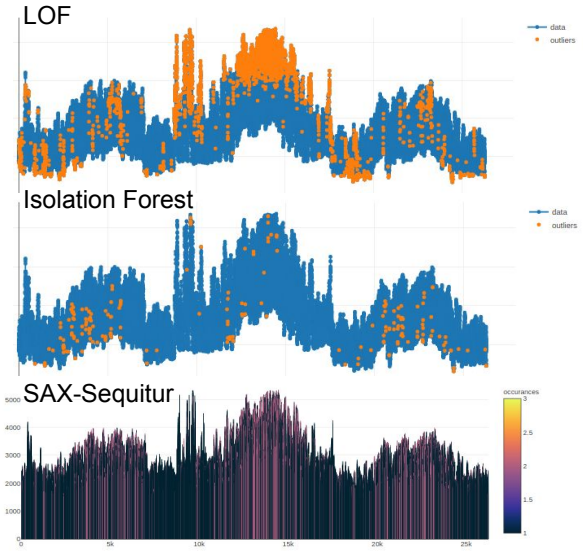
Recover (ML) – Anomaly Detection

- Goal: detect equipment failure and/or non-technical losses from meter data streams.
- Methods implemented:
  - Facebook Prophet
  - Daily Peak SVM T-Test
  - Local Outlier Factor (LOF)
  - Isolation Forests
  - Symbolic Approximation (SAX) - Sequitur
  - Etc.
- Test data:
  - ERCOT South Region 2002–05
  - Synthetic data from GridLAB-D
- Conclusion:
  - Near perfect identification of theft and hardware failures when test/train split on same meter
  - Cross-training results pending
  - Maybe our first **supercomputing** application
- Full methodology and results: https://docs.google.com/presentation/d/1rksH_8D2fFzRF0uUN2ovPkpgCerzEp20ZoHIaUZi5KY/edit#slide=id.p



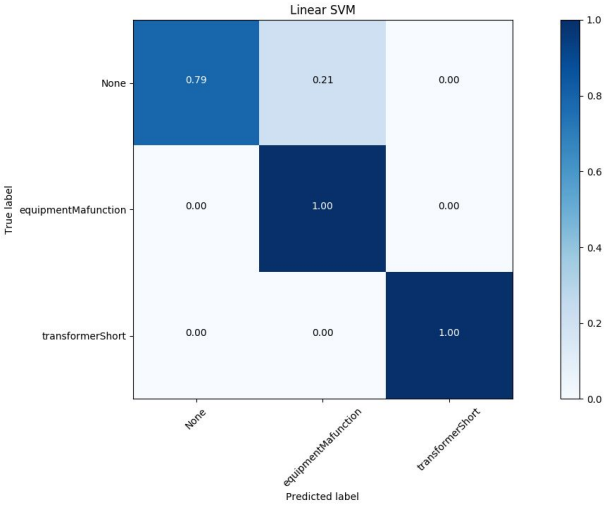
Recover (ML) - Fast Anomaly Detection

- Goal:
  - Detect equipment failure (residential, transformer) and non-tech losses from meter data
  - Implement methods with sub-hour train/test performance to enable realtime detection
- Methods:
  - Local Outlier Factor (LOF)
  - Isolation Forests
  - Symbolic Approximation (SAX) - Sequitur
  - SVM
  - etc.
- Test Data:
  - Tagged datasets from utilities uncommon
  - Generated test data using 2 utility circuits and GridLAB-D
- Details on model specification and selection: https://docs.google.com/presentation/d/1S3PXIdQUw_d-frCVtvmGZbtilRssprUiKGCXxHqjh8k/edit#slide=id.g75f2d9c050_0_8



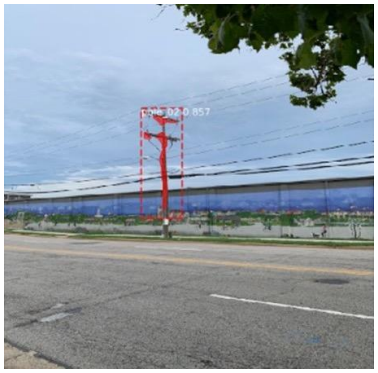
Recover (ML) - Fast Anomaly Detection Validation

- Conclusion:
  - Detection methods reliable with 6 months of training data.
  - Scores lower for cross-trained models (train on one meter, test on another)
  - Future work will focus on cross-trained test case because this is the likely deployment method (single meter methods would only detect further issues on that 1 meter).
  - Non-technical losses frequently confused with other anomalies, so we're redefining how this anomaly is defined
  - Full validation results: https://docs.google.com/presentation/d/1rksH_8D2fFzRF0uUN2ovPkpgCerzEp20ZoHIaUZi5KY/edit



6 month Olin Barre GH test case

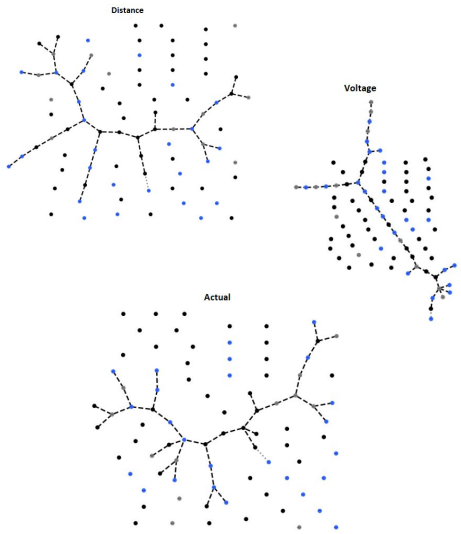
Recover (ML) - Image Recognition for poles



- Generated images database (from real and synthetic sources)
- Built, trained and tested neural network engine (Mask R-CNN) for image recognition
- Created tool to perform image recognition from video data
- We are working to gather more data, required for these kinds of techniques.

Recover (ML) – Network Structure Learning

- Goal: Find the likely connectivity of a feeder system, given the location information for each node and some voltage data
- Methods implemented:
  - Prim's Algorithm (find minimal spanning trees)
  - scikit-learn SVM model
- Test data:
  - ieee37nodeFaultTester
  - test_ieee123nodeBetter
  - Taxonomy Feeders
  - Distance data is obtained from the .omd file and voltage data from running GridLAB-D on the .glm
- Conclusion: Connectivity can be approximated with very good accuracy for smaller feeder systems. Further work must be done in making sure the model is viable for any arbitrary system.



Recover - Black Start

- Goal: Find a sequence of energizing generators and opening/closing switches that restores power in a transmission network and an effective means of visualizing the sequence using powerflow.
- Methods Implemented:
  - pandapower
  - NetworkX
- Test Data:
  - Randomly generated transmission networks
- Conclusion:
  - We can find the best possible path to black start a transmission network and verify our results by running powerflow with pandapower (restoration sequence bottom right).
  - Using NetworkX, we can visualize the results of powerflow at each step of blackstart where, for each node, size is the power generation and color is the power demand. The arrows in the graph show the direction of current flowing and the width of the edges show how much power is transferred from one node to another (powerflow results top right).