# Final Report on the Black Arcs Problem

July 6, 2018

# Problem Summary

Natural graphs of city layouts don't "look nice"

# Problem Summary

What does it mean to “look nice”?

- Heuristic idea: sharp angles
- Ideal to have small set of allowable angles
- We take these to be multiples of $\pi/8$

Problem: how to translate graph nodes/edges to make angles more like multiples of $\pi/8$?

# The Data

Get map data in list format:

- List of $n_v$ vertices as $(x_i, y_i)$ coordinates
- List of $n_e$ edges as vertex pairs

Look to translate nodes $(x_i, y_i) \to (\tilde{x}_i, \tilde{y}_i)$ so to "improve" angles formed by edges

# The Data

Get map data in list format:

- List of $n_v$ vertices as $(x_i, y_i)$ coordinates
- List of $n_e$ edges as vertex pairs

Look to translate nodes $(x_i, y_i) \to (\tilde{x}_i, \tilde{y}_i)$ so to "improve" angles formed by edges

Three principles:

1. Don't let nodes move a lot
2. Don't let angles change a lot
3. Make angles equal to $k\pi/8$ for some $k \in \mathbb{Z}$

# Continuous Optimization Approach

Central idea: relax constraint

Compose *fitness function* from three terms

1. Penalize node movement: $\displaystyle\sum_{i=1}^{n_v} \left((\tilde{x}_i - x_i)^2 + (\tilde{y}_i - y_i)^2\right)$
2. Penalize angle change: $\displaystyle\sum_{j=1}^{n_e} \left(\tilde{\theta}_j - \theta_j\right)^2$
   - Given edge $e_j = \{(x_k, y_k), (x_\ell, y_\ell)\}$, define $\theta_j = \mathrm{arctan2}(x_\ell - x_k, y_\ell - y_k)$
   - "Four-quadrant inverse tangent"
3. Penalize difference from $k\pi/8$: $\displaystyle\sum_{j=1}^{n_e} \sin^2\left(8\tilde{\theta}_j\right)$

# Fitness Function

Take weighted sum of these:

$$f(\tilde{x}, \tilde{y}) = \alpha \sum_{i=1}^{n_v} \left( (\tilde{x}_i - x_i)^2 + (\tilde{y}_i - y_i)^2 \right)$$
$$+ \beta \sum_{j=1}^{n_e} \left( \tilde{\theta}_j - \theta_j \right)^2 + \gamma \sum_{j=1}^{n_e} \sin^2 \left( 8\tilde{\theta}_j \right)$$

Problem: How to choose $\alpha$, $\beta$, $\gamma$?

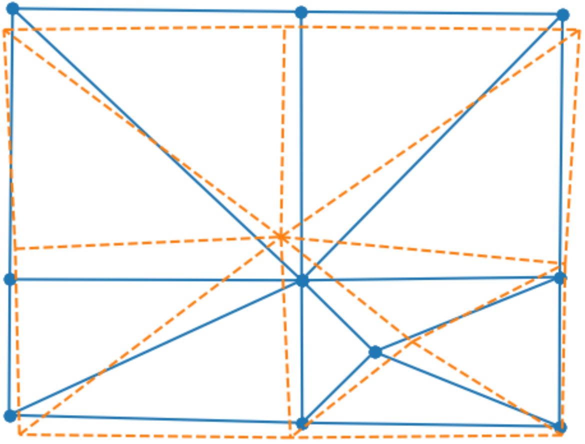
- First pass: experiment!
  - Choosing equal weights leads to difficult optimization
  - Moving nodes a lot (in coordinate space) is okay
  - Roughly $\alpha = 10^{-5}$, $\beta = \gamma = 1$

# Black-Box Optimization

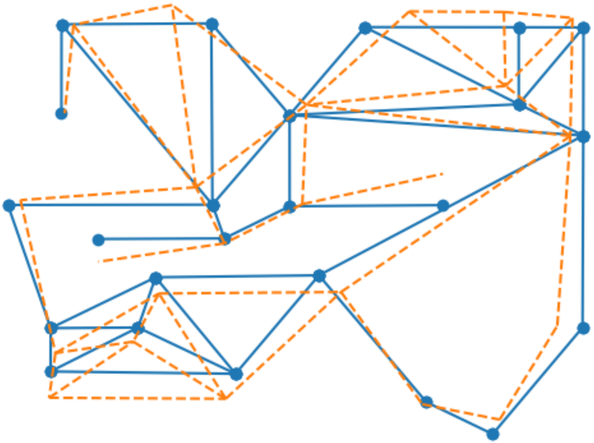
Many black-box optimization tools available

- Derivative-free optimization
  - Nelder-Mead, Powell, . . .
  - Try to identify search direction, go downhill
- Basin-hopping
  - Combine these with local searches above
- Differential Evolution
  - Metaheuristic method
  - Create population of candidate solutions, recombine to improve optimality
- Many others
  - Chose to use just those available in scipy

# First results



# Second results



# Measuring Quality

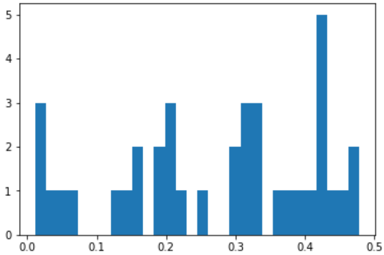
Visual quality seems good, but what about some numbers?

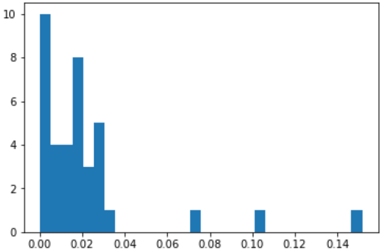
- For first example
  - Reduce fitness function from 6.35 to 0.67
  - Reduce scaled deviation from angles of $k\pi/8$ from 0.45 to 0.10

# Measuring Quality

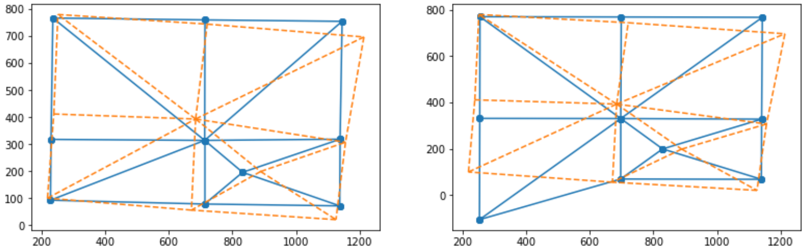
Visual quality seems good, but what about some numbers?

- For second example
  - Reduce fitness function from 21.29 to 1.61
  - Reduce scaled deviation from angles of $k\pi/8$ from 0.48 to 0.15



# Some choices




Two results from slightly different optimizer options

- Graph at right has much lower deviation in angles from $k\pi/8$
- Achieving lower deviation comes at trade-off with node displacement

# Next steps

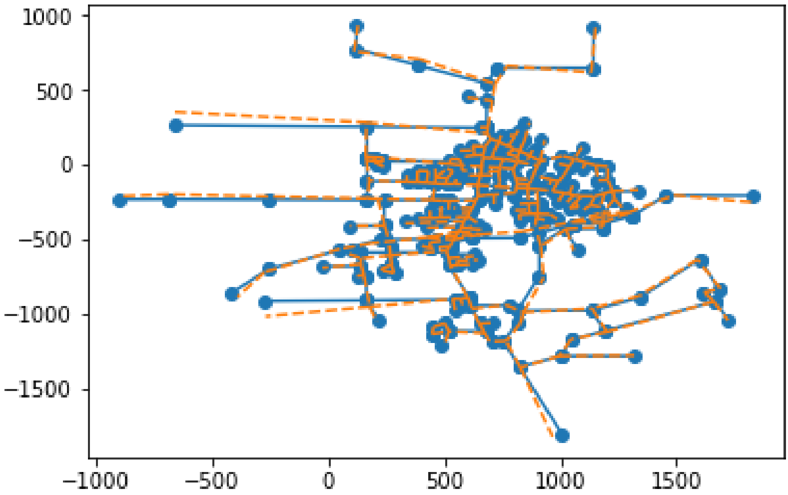
Working with Sackville map adds some complexity

- Direct application of optimization approach only reduces maximum deviation from 0.50 to 0.42

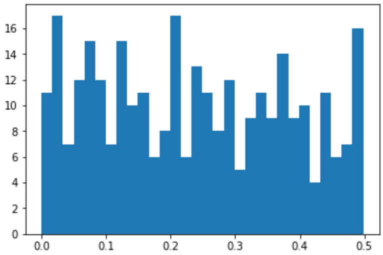
Idea: rotate first, then optimize relative to rotated map

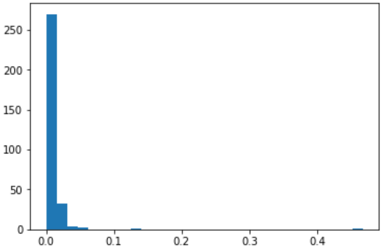
- Eyeball $23^\circ$ rotation, from histogram of initial angles
  - Can we automate this reliably?
- Working on optimization after rotation

# Hot off the press!

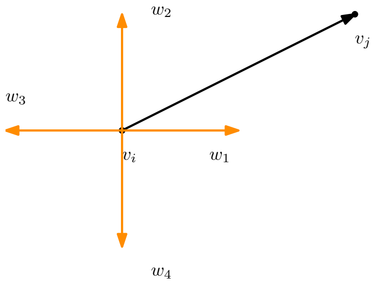


# Hot off the press!



# (Integer) Linear Models




$$\min \sum_{(i,j)\in E} |\epsilon_{i,j,1}| + |\epsilon_{i,j,2}|$$

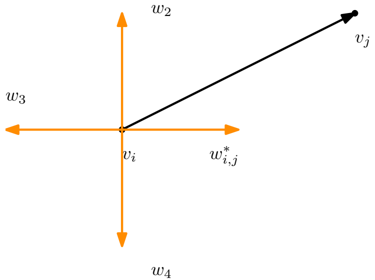
$$\tilde{v}_i + \ell_{i,j} \sum_{d \in 1\ldots 4} c_{i,j,d} w_d = v_j + \epsilon_{i,j}$$

$$c_{i,j,d} \in \{0, 1\}$$

$$\sum_d c_{i,j,d} = 1$$

$$\ell_{i,j} = \|v_i - v_j\|$$

# Linear Models




$$\max \sum_{(i,j)\in E} \langle z_{i,j}, w^*_{i,j} \rangle$$

$$z_{i,j} = (\tilde{v}_i - \tilde{v}_j)/l_{i,j}$$
$$w^*_{i,j} = \text{closest } w_d$$
$$\ell_{i,j} = \|v_i - v_j\|$$

In both cases we need penalties/bounds to prevent vertices from moving too far.