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 $\pi/8$
- Want to encourage multiples of $\pi/2$ even more

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

Four 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 $k \in \mathbb{Z}$
4. "Encourage" angles of $\pi/2$

Continuous Optimization Approach

Central idea: relax constraint

Compose *fitness function* from three terms

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

Fitness Function

Take weighted sum of these:

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

Problem: How to choose α, β, γ?

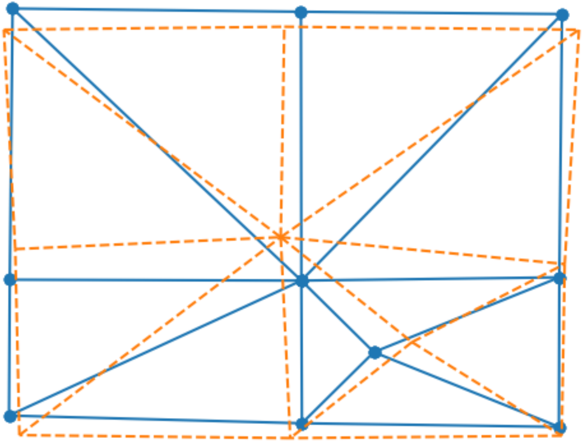
- Eyeball it

Black-Box Optimization Results

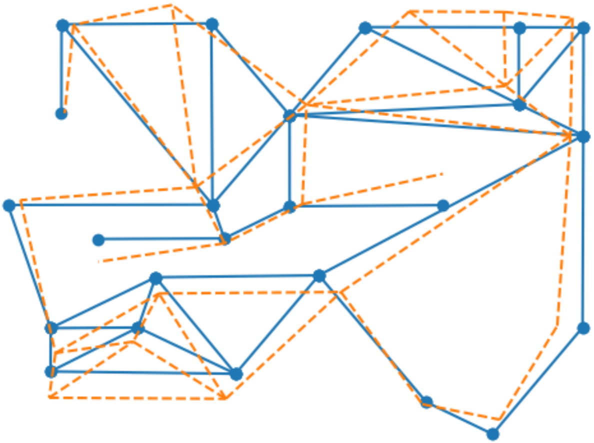
Many black-box optimization tools available

- Derivative-free optimization
  - Powell seemed to provide better results than Nelder-Mead
- Basin-hopping
  - Very expensive
  - Did not improve results enough visually to be used often
- Differential Evolution
  - Slower than Powell
- CMA-ES
  - DEAP (an evolutionary comptation framework for Python)
  - CMA-ES initial results were vastly different

First results



Second results

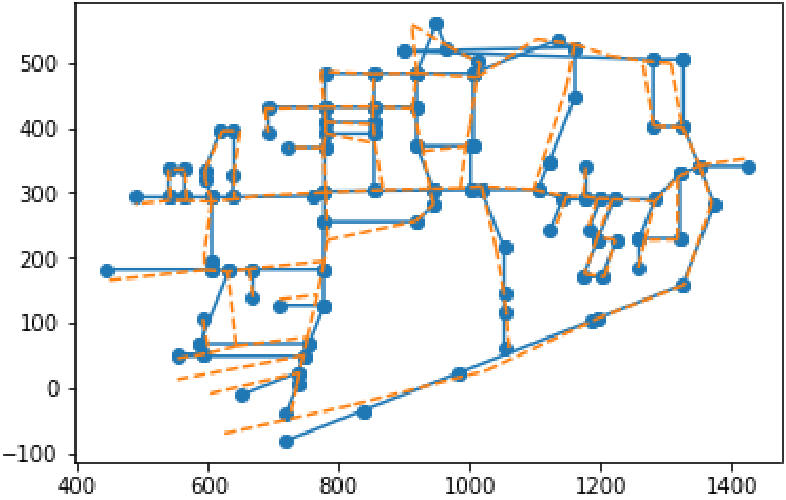


Rotation

Idea: rotate first, then optimize relative to rotated map

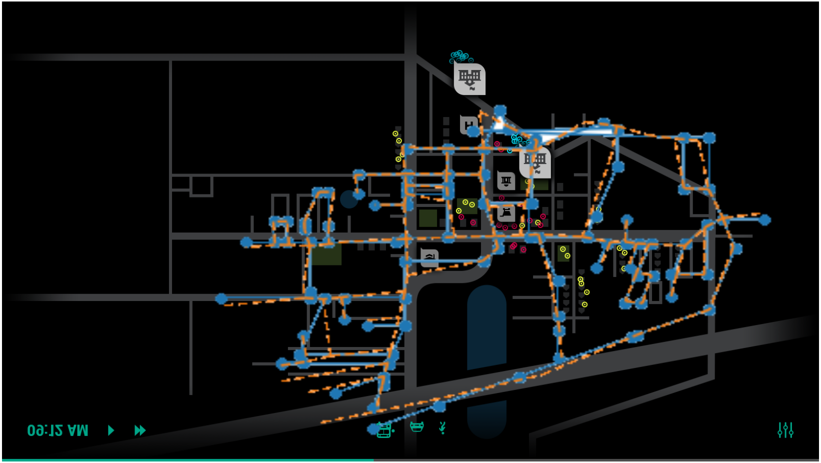
- Eyeball 23° rotation, from histogram of initial angles
  - Did we automate this reliably? NO
- Optimize after rotation

Results

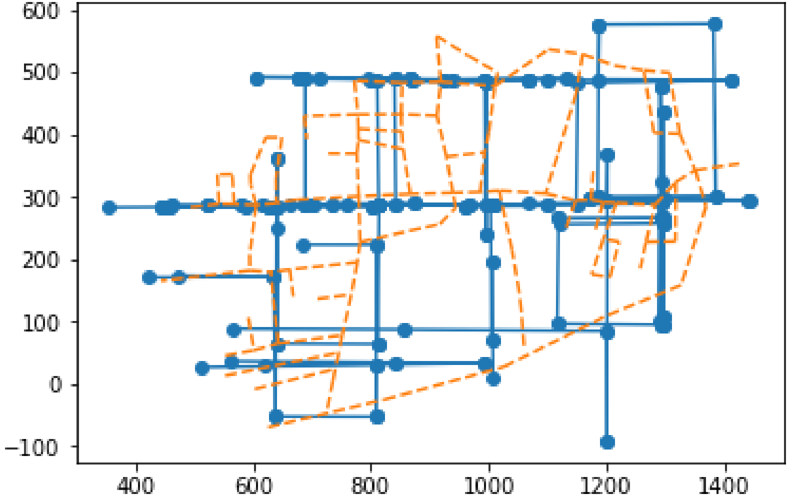


Results

Results



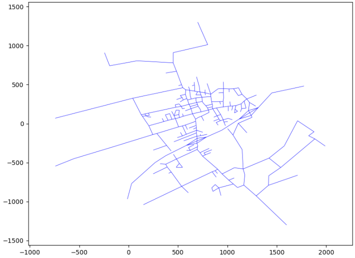
Results



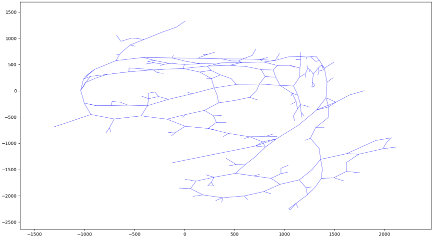
CMA-ES - What Happened?

- Fitness of original map was 7.955456
- $\alpha = 1e - 07$, $\beta = 1.0e - 3$, $\gamma = 1.0e - 1$
- Fitness value of Powell was 1.991282
- Fitness value of CMA-ES was 0.524248
  - γ term dominates (why we see so many 90 degree angles)

Gravity

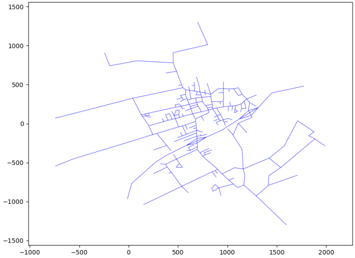


1500
1000
500
0
−500
−1000
−1500
−1000
−500
0
500
1000
1500
2000

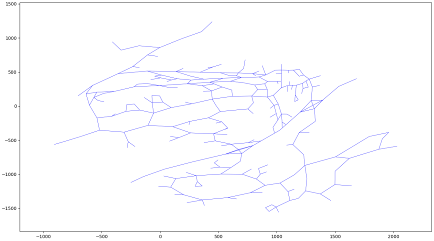


1500
1000
500
0
−500
−1000
−1500
−2000
−2500
−1500
−1000
−500
0
500
1000
1500
2000

Gravity



1500
1000
500
0
−500
−1000
−1500
−1000
−500
0
500
1000
1500
2000



1500
1000
500
0
−500
−1000
−1500
−1000
−500
0
500
1000
1500
2000