A D3 plug-in for automatic label placement using simulated annealing
Motivation

- **Labeling features or data points** (point-feature label placement) is important but **time consuming** [1].

- Given the large cost of time, automatic label placement (ALP) is used widely by **map generators** [2].

- **Problem:** There exist little published or empirical evidence indicating that **individuals** frequently use ALP

- For **advanced plotting tools** favored by scientists and engineers, there is often **no sophisticated built-in ALP**.

Project proposal

- A possible solution to the problem is to create a plug-in (extension, add-on) for an existing program that performs ALP

- ALP plug-ins do exist for some of the aforementioned plotting tools, with varying sophistication and ease of use [1, 2].

- For D3, there is currently no good ALP plug-in

- My project: write a D3 plugin for automatic label placement that implements a good algorithm (simulated annealing) and is easy to use

- Note: not designing a new algorithm, but addressing a problem that can potentially help a lot of people.

(1) http://bl.ocks.org/MoritzStefaner/1377729
(2) http://bl.ocks.org/ZJONSSON/1691430
Related Work

- The **general rules** for ALP have been carefully studied by many, most notably by Imhof [1, 2] and more quantitatively formulated by Yeoli [3]

  1. Spatial overlap: labels should not overlap with each other or point features
  2. Unambiguity: labels should be unambiguously identified with its graphical feature
  3. Legibility: label should be easily readable

- In addition to these rules, Imhof [1] described a set of **more specific stylistic rules** pertaining to specific label positions

  1. Label position on the right is preferred over left
  2. Label position on the top is preferred over bottom
  3. Label position closer to the corresponding point feature is preferred.

Search space and cost function

- The **search space** consists of the collection of all possible label positions.
- All label configurations that **satisfy** the boundary conditions are **permissible**.
- Search space contains **2N degrees of freedom**, where \( N \) = number of labels.
- Not all points in the search space are equal.
- To distinguish between the quality of configurations, construct a function which takes as input amount of overlap, distance constraints, stylistic preferences [1].
- Energy/cost function

\[
C_i = f(A_{ll}, A_{lp}, L_{lp}, \theta)
\]

**General dependence**

**Cost for each individual label** \( i \)

\[
C_i = \sum_{j \neq i}^N A_{ll}W_{ll} + \sum_{j \neq i}^N A_{lp}W_{lp} + L_{lp}W_{lp} + \theta W_\theta
\]

**Total cost**

\[
C_{tot} = \sum_{i}^N C_i
\]

Why simulated annealing?

- Many **possible classes of algorithms** can be applied to automatic label placement. These include random placement, exhaustive search algorithms, greedy algorithms, local search algorithms, stochastic search algorithms, genetic algorithms.

- Can generally be divided into **local or global** search algorithms.

- Landscape of label configurations is **rough and high-dimensional** → **global search**

- **Simulated annealing** [1,2] with Metropolis acceptance criterion:

  ![Simulated annealing illustration]

  (a) High temperature  
  (b) Low temperature

---

Specifics of my implementation

- Focus on **ease of use**

```javascript
<script src="anneal.js"></script>
```

```javascript
// Implement simulated annealing
var sim_anneal = d3.anneal()
    .nodes(label_centers)
    .anchor(anchor)
    .width(width).height(height)
    .sim_anneal(0.25);
```

```javascript
// Draw data points
circ = svg.selectAll(".circ")
    .data(label_centers)
    .enter().append("circle")
    .attr("class", ".circ")
    .attr("r", 20.0)
    .attr("cx", function(d) { return (d.x); })
    .attr("cy", function(d) { return (d.y - offset); })
    .style("fill", 'red');
```

5 lines only
Video demo 1: data in a straight line
Video demo 2: scatter plot
Remaining timeline

Nov. 22: Implement MC rotation moves, stylistic rules (Imhof)

Nov. 27: Implement more accurate calculation of label area, overlap area, optimize weights

Dec. 2: Presentation

Thanks for listening!