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

Evan Wang | https://github.com/tinker10/D3-Labeler

Motivation and Problem

- Labeling data is important but time-consuming
- Automatic label placement (ALP) is used widely by map generators (Agrawala et al., 2001)
- Problem: There exists little published or empirical evidence that individuals use ALP
- For many advanced plotting tools (D3, Matlab, Matplotlib, Mathematica) favored by scientists and engineers, there is no sophisticated built-in ALP

Proposed Solution

- A possible solution is to create a plug-in (extension, add-on) for an existing plotting tool that performs ALP
- ALP plug-ins do exist, with varying sophistication and ease of use
- For D3, there is currently no good ALP plug-in
- My project: write a D3 plug-in for automatic label placement that implements simulated annealing and easily incorporates into existing D3 code, with syntax mirroring other D3 layouts

Approach

Components of a labeling problem

Anatomy of a labeling problem. Each label corresponds to an anchor point. A leader line may be used to help with the label-anchor point correspondence. None of the elements may cross the graph boundary.

Label placement rules

The general rules for label placement have been studied most notably by Imhof (Imhof, 1962) and more quantitatively formulated by Yeoli (Yeoli, 1972).

- Spatial overlap: labels should not overlap with each other or anchors
- Unambiguity: labels should be unambiguously identified with its anchor
- Legibility: labels should be easily readable
- Distance: labels should be close to its anchor
- Position: (right, top) is preferred over (left, bottom)

Implementation

Search space

- The search space consists of the collection of label positions
- Each configuration can be written as \( V = \{ r_1, r_2, r_3, \ldots r_N \} \)
- All configurations that satisfy the boundary conditions are permissible.
- \( 2N \) degrees of freedom

Energy (cost) function

Algorithm: simulated annealing

- Labeling problem: essentially an optimization problem on a complex and high-dimensional energy landscape
- Many classes of algorithms can be applied, including exhaustive search, greedy algorithms, local search algorithms, stochastic search algorithms, genetic algorithms
- They be categorized into local or global optimization methods
- Global search is preferred over a local one because global algorithms can avoid being trapped in a local minima
- Simulated annealing is one of the more favored algorithms because of its simplicity, flexibility, and intuitive physical basis

Results

Benchmark results. Each result is the average of 100 simulations (1000 Monte Carlo sweeps each).

<table>
<thead>
<tr>
<th>Labels</th>
<th>Time (sec.)</th>
<th>L-L overlap</th>
<th>Leader intersect.</th>
<th>L-A overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>0.13</td>
<td>0.01</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0.45</td>
<td>0.14</td>
<td>0.05</td>
<td>1.25</td>
</tr>
<tr>
<td>75</td>
<td>1.10</td>
<td>0.62</td>
<td>0.13</td>
<td>4.35</td>
</tr>
<tr>
<td>100</td>
<td>1.84</td>
<td>1.77</td>
<td>0.37</td>
<td>9.88</td>
</tr>
<tr>
<td>125</td>
<td>2.91</td>
<td>5.52</td>
<td>0.57</td>
<td>23.59</td>
</tr>
<tr>
<td>150</td>
<td>4.40</td>
<td>17.17</td>
<td>1.01</td>
<td>47.60</td>
</tr>
</tbody>
</table>

Snapshots. Number of labels (a) 25 (b) 50 (c) 75 (d) 100 (e) 125 (f) 150.

Visualization of energy penalties.

Visualization of Monte Carlo moves. To ensure an ergodic and efficient sampling of the conformational space, we use a combination of (a) translation and (b) rotation moves.

API (full version on Github)

Installation

Usage

Future Work

- Implementing additional graph-type specific rules for a variety of graph types (bar-graph, pie-chart, time-series).
- Adapt the simulated annealing layout for other optimization problems, such as drawing node-link diagrams.

Acknowledgements

I would like to thank Prof. Maneesh Agrawala and members of the class for insightful comments and suggestions.