The Title

Author 1
Affiliation
Affiliation
author@a.com

Author 2
Affiliation
Affiliation
author2@b.com

ABSTRACT
In this paper we describe the formatting requirements for SIGCHI Conference Proceedings, and offer recommendations on writing for the worldwide SIGCHI readership. Please review this document even if you have submitted to SIGCHI conferences before, for some format details have changed relative to previous years. These include the formatting of table captions, the formatting of references, and a requirement to include ACM DL indexing information.

Author Keywords
put author keywords here

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Miscellaneous—*Optional sub-category*

INTRODUCTION AND MOTIVATION
Our project creates an interface for manually creating and sharing metro maps of news stories in a clean and clear way. The primary work on which it is based is Dafna Shahaf's Connecting the Dots project []. In her formulation, the selection of stories, the metro lines that connect them, and label display are all determined by computer algorithm, while layout is an entirely manual process.

We propose a system that makes layout an efficient and easy task. This is important because a primary advantage of metro maps over simple ordered lists of news stories is their ability to inform users about key events more effectively. On information summarization tasks, a metro map with Shahaf's algorithmically-chosen lines allows users to more quickly grasp important points (as determined by experts) of complex events like the Greek Debt Crisis [].

If we give non-designer users a simple and powerful tool for metro map visualization of news, we allow people to connect different news stories, and visually summarize complex series of events, and also frame existing content with appropriate context that's likely missing from hyperlinks within the text. A key advance in our system is a force-directed layout algorithm that assists the editor in creating a visualization that conforms to the constraints demanded by a metro map of news stories. Allowing the user to meet all these constraints while still allowing effective user control of the layout formed a serious barrier for our development efforts to overcome, and in this paper we share practical advice.

RELATED WORK
Metro map layout as a specific problem is a somewhat studied area of research, first initiated by [1]. These papers consider a graph representing a real world metro system, and attempt to lay out an aesthetically pleasing map of it. [2, 3] There are many similarities; for example, much of our aesthetic criteria (including the octilinearity of edges) are shared with the related work. However, our work is distinct from these previous pieces of work in the following ways. First, these papers consider metro systems that correspond to a physical geography; thus, they are subject to extra constraints such as preservation of topology and relative positioning of stations. However, since metro maps of the news are abstract entities, we have more flexibility in choosing the topology of our generated maps. Second, these papers consider the generation of a metro map to be a rare occurrence, and thus invest considerable time into the solution to find as globally optimal a layout as possible (sometimes requiring running time on order of hours.) Our work seeks to perform layout in realtime, and relies on user interaction to discover a global optimum. Finally, the temporal nature of news articles mean we have extra constraints to work with when laying out our metro maps of the news.

METHOD

Force Layout
The heart of our project was implementing a layout algorithm for metro maps. Because fully automatic layout of metro maps is a difficult task, our layout algorithm adopted a hybrid approach, using force layout in combination with custom aesthetic constraints and user input.

We started with D3.js's implementation of force layout. The default implementation offers repulsive charge forces, a pseudo-gravity force and fixed-distance geometric constraints for links. All of the features are implemented on top of position Verlet integration, which is a method for simulating Newtonian systems.

One important property of Verlet integration is that custom

Submitted for review to CHI 2009.

constraints can easily be implemented on top of it. In D3.js, these custom constraints are implement in a `tick` handler, in which a constraint can inspect the physical location of all nodes in the diagram and make adjustments to their location based on constraints. This is technically distinct from adding "custom forces" to the diagram, which would need to be incorporated by the Verlet algorithm; instead, we simply "nudge" nodes to where we would like them to be.

Custom constraints were primarily used to fulfill aesthetic criteria. The extra constraints we implemented where "octilinearity", which enforces 45 degrees angles on edges, "monotonicity", which enforces a relative ordering of nodes based on their date, and "time-alignment", which enforces a fixed X position based on the date.

"Octilinearity" was a property that we found most prior work on metro maps posited as an aesthetic criterion for a "good metro map" (though it seems not to be universal, as can be seen from the California Bart map.) This was implemented by considering all edges in our layout, and computing the closest 45-degree aligned angle to the current layout of the edge. The constraint is applied by rotating the endpoints around the centroid of the edge so that the angle is fulfilled. The effect of octilinearity is primarily local. Our implementation generalizes so any set of angles can be preferred; for example, our final console also allows hexilinear layout, which uses 60-degree aligned angles.

"Monotonicity" was a property unique to abstract metro maps, and can be thought as a replacement of the topological constraints physical metro maps labor under. This force ensures that progress from left-to-right and from top-to-bottom corresponds to progress in time. This corresponds to a lot of important property users might expect out of maps, including the fact that two parallel lines should also be parallel in time. This is a global constraint on layout, but since it is relatively flexible it can be thought to "rotate" the force layout into a good direction.

"Time-alignment" was another property unique to abstract metro maps, and is more strict that monotonicity. It expresses the idea that metro maps are timelines, and should be "to scale" with the underlying interval data that they express. There are a few possibilities for a constraint like this: one could imagine a constraint where the length of the line connecting two stations accurately reflects the amount of time that passed between two events; however, our implementation simply fixes the X-coordinate of nodes, so that they only have degrees of freedom in the Y-axis.

Coordinating these custom forces along with D3.js's default forces was a nontrivial task, which we discuss in more detail in Discussion.

Debug Console

A reusable component we built as a part of this project was a debug console for force layout. In many force layout tasks, the resulting visualization is highly sensitive to the parameters of the force layout; our goal was to make it easy for a designer in this field to make adjustments to these parameters. Each parameter is represented as a slider. We also added sliders which controlled the degree to which our custom forces were applied to the layout. We also added the ability to pause and unpause the force layout algorithm, so that multiple changes to parameters and node positions could be made. Finally, we implemented the ability to completely serialize layout to JSON, so that it can be reloaded at some future point. We believe it would be relatively simple to build a social sharing capability to our application, although we judged it out of scope for this project.

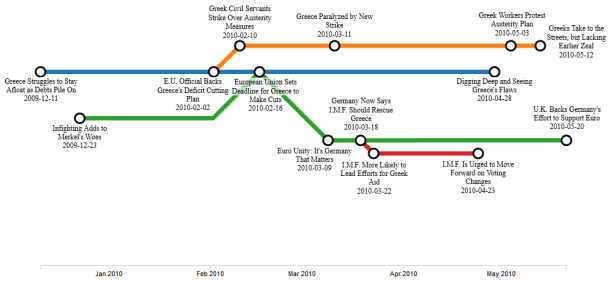
You can view the debug console by appending `#debug` to our demo.

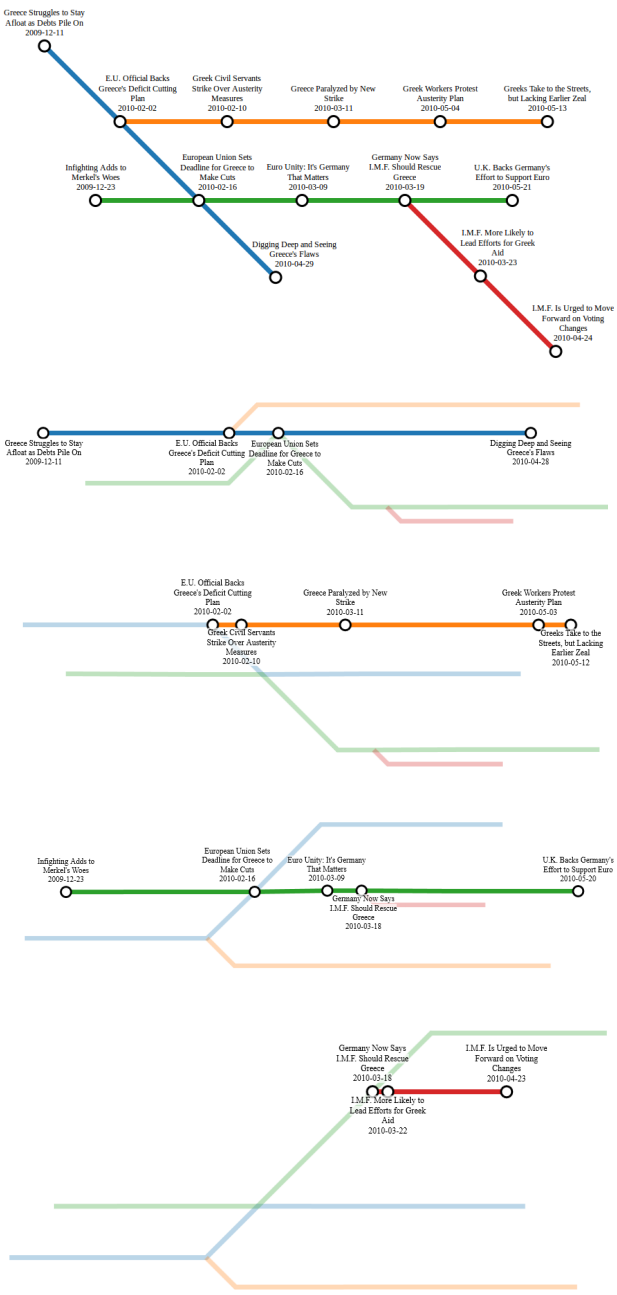
Presentation

Our project seeks to provide an accessible viewer for metro maps, a new way of looking at complex stories which cannot be summarized on a simple timeline. The hope of the demo was to introduce the audience to the concept of a metro map, and how to interpret it. In order to facilitate faster comprehension of the metromap (and on the suggestion of Jeffrey Heer), we added the feature to focus on specific news thread and dim the rest, as well as node highlighting when viewing the full text of a news article and a short tutorial. The slideshow we introduce to walk through functionality served successfully as a tutorial. Users during informal testing and at the poster session demos found the layout aesthetically pleasing, liked the transitions between line focuses and time constraints, and found node and edge highlighting both to be clear.

Regarding the metro maps concept in general, most demo users saw the aesthetic appeal but not all found it convincingly superior to, say, a simple color-coded timeline in conveying information. A more formal user study with a wider variety of document sets could replicate the superior performance on information summarization that Shahafs original research demonstrated [].

RESULTS





DISCUSSION

We learned a number of non-obvious lessons about the interaction between our custom forces and the default layout. We describe some of these principles in this section.

Last constraint wins

When adding custom constraints to a force layout, it is important to understand that your constraints are not being fed to a magic constraint-solving oracle; rather, you are simply moving nodes around on the canvas. As a result, if one constraint moves a node to one location, and another constraint moves it back, the end result is that the first constraint is ignored. We knew about this property from the very beginning of the project, but it was interesting to see how it manifested during real layout.

The most salient effect is that custom forces hugely impede the ability of normal charge repulsion to avoid edge-crossing. For example, if octilinearity is turned on while initial layout is occuring, the layout algorithm usually failed to find a layout with few edge crossings. Additionally, the strength of the forces contributes hugely to this effect: for example, when edge lengths were long, the octilinearity force is magnified (nodes have to be pushed "further" in order to achieve the desired angle), and the incidence of edge-crossing correspondingly increased.

Staging

One step we took to mitigate the impact of local custom forces was to stage the application of forces. For example, if one was creating only an octilinear layout, we got good results if we first allowed the layout to operate without any octilinear forces, and then only gradually ramp up the force near the end of layout as a "quantization" step. We have a default scaling of custom forces implemented in the debug console—the sliders have two knobs, one indicating how strong the force should be at the beginning of layout, and another knob indicating how strong the force should be at the end of layout. The strength of the force is inversely scaled with the graph "cooling" parameter alpha, which is normally used to turn of simulation in order to conserve CPU.

On the other hand, global custom forces such as time-alignment and monotonicity did not benefit much from staging. Since these forces can be thought of as adding global constraints to diagram layout, it is better for them to be applied immediately. This was especially evident for time-alignment, where relaxing the constraints upon edits resulted in nodes "contracting" together (since they were no longer fixed in X-position) before bouncing back to their original locations.

Unsatisfiability and instability

The hallmark of an unsatisfiable set of constraints was unstable layout. For example, in an early iteration of octilinearity, we tried to enforce "straight lines" by having nodes communicate edge orientations to one another, and attempt to straighten themselves out. Test runs on this layout quickly resulted in the layout "blowing up", with node x and y coordinates running of the diagram.

While we attempted to design custom forces which would not be unstable, we were not wholly successful. For example, while performing layout for Greek maps, we had two news articles really close together in time, so the timeforce tried to push them together (but only along the x-axis). On the other hand, the octoforce determined that the node was close to vertical, so it attempted to push the nodes to vertical, and combined the node would fly off the page (until it was sufficiently counterbalanced by the gravity parameter).

The most effective mechanism we found for combating this problem was selective mechanisms for *removing* constraints. For example, one constraint was that edges between nodes had to be straight; very early, we added the ability to add

"dummy nodes" which gave edges extra degrees of freedom. Allowing kinks in lines was crucial to achieving good layout in some cases. Additionally (and some what paradoxically), *fixing* nodes to a position also proved effective at combating instability, since fixed nodes were no longer affected by forces.

Editing

Editing was the most important mechanism for performing global optimization of layout; without user input, layouts were prone to getting stuck in local optima that were not aesthetically pleasing. However, we were also pleased to find that all of the local forces minimized the amount of effort necessary for users performing layout; the four precomputed layouts in our demo took on order of five minutes to compose each. Unfortunately, the editing mechanism was not polished enough for us to feel comfortable giving to a general audience.

We believe that there may be a more general story for doing layout of any sort of visual media, which is that a hybrid approach where computer constraints perform "micro-layout" and human input perform "macro-layout" will allow for quick and efficient layout.

FUTURE WORK

Additional features implemented in future work should be focused on making editing more natural and flexible for those without technical backgrounds. Currently, a great deal of the content must be hard-coded, including xml files supplied by the user with most content. While this is still short of the difficulties that making (even static versions of) these timelines in a presentation or image editing program, it's still not as friendly as letting the user drag-and-drop from query results on a fully pre-processed repository of new stories, with the option to add their own. This would fit with a user-friendly node insertion and path drawing mechanism for our long-term goal, being able to make a metro map from scratch with your own document set (and an easy way to import it); this would let people make metro maps in a wide array of domains (for example, legal decisions, scientific literature, or in-depth analysis of single books).

When multiple constraints are being enforced, our interface should do even more to guide the user through the best practices weve learned and described in this work, for instance taking care of the staged enforcement schedules of different constraints that we discuss above.

- Good-looking dynamic focus still requires a fair amount of manual user intervention. A one-click method to focus the map on a single straightened metro line would be useful. Each focused will still have to be manually laid out, but this should be as simple and fast as possible given an initial layout for the full view.

- More complex metro maps, with cycles or with multiple lines crossing a single edge, still present visualization problems. Force-layout can have a difficult time coping with the former, especially when enforcing left-to-right date ordering, while the latter was aesthetically unappealing under schemes we considered.

- More speculative features include a friendlier interface for sharing metro maps, maybe somewhat like manyeyes. Also, a strong alternative to force-based layouts, though it would require significant re-implementation, are mixed-integer programming methods.

ACKNOWLEDGEMENTS

We'd like to acknowledge Dafna Shahaf, for inspiring the project and for many useful discussions, Jeff Heer for teaching the class and important suggestions for features, and Stanford University for making this project possible.

REFERENCES

1. S.-h. Hong, D. Merrick, and H. A. D. Do nascimento. the metro map layout problem. *graph drawing*, 2005.

2. M. Nöllenburg and A. Wolff. a mixed-integer program for drawing high-quality metro maps. *graph drawing*, pages 321–333, 2006.

3. J. Stott and P. Rodgers. automatic metro map layout using multicriteria optimization. . . . *ieee transactions on*, 17(1):101–14, Jan. 2011.