Highway Data and Map Visualizations for Educational Use

James D. Teresco
Siena College
515 Loudon Rd.
Loudonville, NY, USA
jteresco@siena.edu

ABSTRACT

It is often a challenge to find interesting and appropriate data sets to use as examples to demonstrate graph data structures and algorithms. Goals for the data are often conflicting. The data should include examples small enough to work through in a class example by hand, but some large enough to demonstrate important behaviors of a structure or algorithm. Data should be freely available in a convenient format and should have some real-world relevance. Visualization of the data and results computed from it is helpful.

This paper describes a collection of graph data sets generated from the Clinched Highway Mapping Project's highway data and some examples of their use in undergraduate courses on data structures and algorithms. The source data, the process used to convert the data into a more useful format, some examples of its use, and a visualization tool using the Google Maps API, are described.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer science education; E.1 [**Data Structures**]: Graphs and networks; G.2.2 [**Graph Theory**]: Graph algorithms

General Terms

Algorithms

Keywords

Pedagogical tools, Graph data structures, Algorithm visualization, Google Maps API.

1. INTRODUCTION

Both students and instructors enjoy class examples and assignments that have both relevance to a real problem and an interactive and visual component [6]. The challenge is often in finding appropriate data sets. This paper describes a collection of data sets representing highway data that can be used to help study graph and other structures and algorithms. The project's motivation was to improve upon an often-used example when studying Dijkstra's algorithm for computing single-source shortest paths on a graph. A small number of vertices represent places (e.g., cities, airports, intersections, rooms), and edges represent the cost in distance or time to travel from one to the other. Figure 1 (top) shows such an example graph.

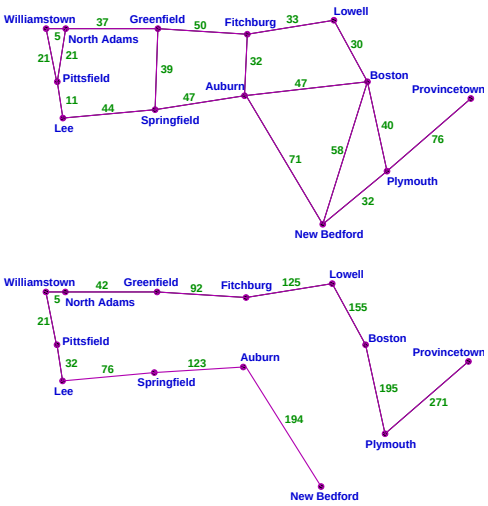

Figure 1: Top: a simple graph of towns in Massachusetts that can be used to demonstrate graph structures and algorithms. Bottom: Computed single-source shortest paths from Williamstown.

The graph has some appealing characteristics. It has some basis in reality. An adjacency matrix or adjacency list representation can be drawn by the instructor or students during class. Algorithms such as Dijkstra's Algorithm can be traced to completion (Figure 1, bottom), step-by-step at the start, and more quickly once students have the basic idea.

A somewhat larger and more interesting data set previously used by the author includes 466 segments of highways in the United States Interstate highway system gathered from an old Rand McNally road atlas. Each entry is defined by the two town names at the endpoints, the name of the highway connecting the two, and the distance in miles between them. For example,

```
albany_ny binghamton_ny I-88 142
```


SIGCSE'12, February 29–March 3, 2012, Raleigh, North Carolina, USA.

indicates that I-88 connects Albany, New York, with Binghamton, New York, over a distance of 142 miles.

This data set is too large to study manually, but small enough to serve as a good data set for testing a laboratory or homework program. While this data could be visualized in an abstract graph form, it lacks any information about where the towns at the graph vertices are located.

This paper describes a collection of data sets that are similar in spirit to the above examples, but with some important advantages. This data is extracted from the Clinched Highway Mapping (CHM) project [9]. The vertices represent places (usually intersections) and the edges represent roads connecting those places. Place data consists of a name and a latitude-longitude pair that allows the places to be plotted on a map. Each data set represents one or more highway systems in North America or Europe, ranging in size from a few dozen vertices and edges up to over one hundred thousand. The smaller data sets can be used in class examples or for debugging programs, while the larger sets can be used to test programs more thoroughly, and to analyze their performance meaningfully.

The CHM project is described in more detail in Section 2. Section 3 details the CHM data and how it is converted into formats more convenient for academic use. In Section 4, the mapping system, based on the Google Maps API [2], is described. Examples of data sets and their uses are presented in Section 5. Finally, some concluding remarks and ideas for future extensions are discussed in Section 6.

2. THE CLINCHED HIGHWAY MAPPING PROJECT

The CHM project, created and managed by Timothy Reichard, allows users to track segments of highways on which they have traveled (or, in "roadgeek" terminology, "clinched"). They can compare their travels to fellow users and generate maps of their travels. Users submit a list of highway segments they have clinched, specifying the highway by name and region specifier, and the endpoints of the segment they have traveled. The project began with the Interstate highway system in the United States, but has expanded to include many other national and state/provincial highway systems in North America and Europe. The highway systems in the project are constantly being expanded and existing systems are being updated, corrected, and enhanced by a small team of volunteer collaborators (including the author).

2.1 CHM Project Data Granularity

A key feature of the CHM project is the granularity of the data used to represent the highways. Which intersections should be included in the description of a highway? In the early stages of the project, this decision was quite easy. The only highways represented were the U.S. Interstates, and the only way to enter or exit these highways is at a freeway interchange or a highway terminus. A CHM user could specify their clinched segments by the interchange numbers (or, state or international boundary or other terminus) for each highway. Interchanges are usually located every few miles, so the straight lines connecting interchanges form a reasonable approximation of the shape and length of the highway. Some exceptions exist, such as interchanges 2 and 3 of Interstate 90 in Massachusetts (Figure 2). They are 30.3 miles of driving distance apart along a road that winds

Figure 2: The straight line connecting interchanges 2 and 3 of I-90 in Massachusetts. The direct distance is 28.29 miles, but the actual driving distance is two miles longer as the road winds through mountains.

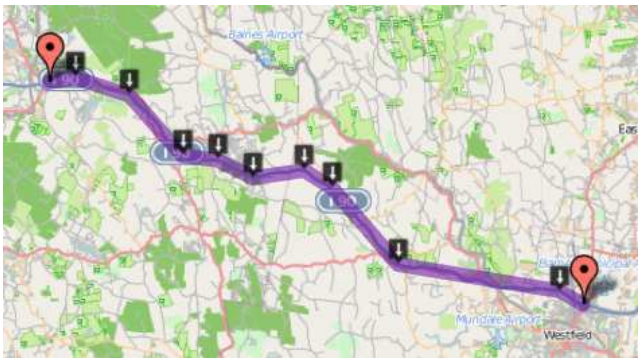

Figure 3: The route between interchanges 2 and 3 of I-90 in Massachusetts after adding "shaping" points to improve the approximation of the road. The distance via shaping points is 30.1 miles.

its way over and through the Berkshire Mountains, but the straight line distance is only 28.29 miles.

When the CHM project was expanded beyond limited access roads to include those with numerous "at-grade" intersections, new decisions were needed. Guidelines were developed to determine which intersections to use to define a route (called *waypoints* in the project's terminology), and how to name those points. It would be impractical to attempt to include every intersection, especially for longer routes and routes through urban areas, nor would it add much value to the project. Most travelers will enter and exit the highways at major intersections. On average, waypoints are spaced less than two miles apart.

Support for "hidden" waypoints was also added, allowing more accurate highway segment lengths and maps for curved road segments. These are points that are generally not useful to CHM users as potential endpoints of a traveled segment but which significantly improve the accuracy of the approximated route. For example, the segment between interchanges 2 and 3 of I-90 in Massachusetts has 9 hidden waypoints, as shown in Figure 3, which increase its approximated length to 30.1 miles.

For the purposes of CHM users, the waypoints are found using the project's Highway Browser, which plots all of the

waypoints for a route within a given region using the Google Maps API. But for CHM collaborators and for the purposes of the project described here, a more detailed look at how the data is gathered and stored is necessary.

2.2 Gathering CHM Data

CHM collaborators create route descriptions by specifying a list of waypoints, gathered using a range of sources and tools[1]. In many cases, it is as simple (and at times, tedious) as centering an online map on the desired waypoint and copying a URL that represents the waypoint to the highway's waypoint list. Care is taken to use only freely-available map data in this process. Since most Google Maps data is copyrighted, routes gathered could be considered an inappropriate derivative work. Therefore, routes are generally plotted from maps in the OpenStreetMap project [7] or government satellite or topographic imagery.

2.3 CHM Data Formats

The underlying CHM data is contained in a number of plain-text files. First, a CSV-format *system file* lists the routes that make up a highway system (e.g., the U.S. Interstates, the New York state numbered highways, the German autobahn system). Entries have the format

```
usany;NY;NY5;;;;ny.ny005
usany;NY;NY52;Bus;Bea;Beacon, NY;ny.ny052busbea
usany;NY;NY114;;Gre;Greenport, NY;ny.ny114gre
```

where the 7 semicolon-separated entries represent (1) the internal CHM system name – in this case, `usany`, which is the New York state numbered highways, (2) the region name – usually a standard abbreviation for the state or province, sometimes a country, (3) the route name – `NY5` means New York route 5, (4) a banner – usually empty, could be `Bus` to represent a business route as in the second entry above, `Alt` for an alternate route, `Byp` for a bypass route, etc., (5) an abbreviation – usually empty but used to distinguish between multiple routes with the same number and/or banner within a region, (6) a city – when an abbreviation is used, a human-readable place name corresponding to that abbreviation, and (7) a waypoint file name – the name of the file that contains the waypoint data for the route.

A *waypoint file* lists the points that comprise a single route. For example, the file `ny.ny005.wpt` which describes New York state route 5, includes the lines shown in Figure 4.

Each line represents one waypoint by specifying a waypoint name, which must be unique within a route, and an OpenStreetMap URL that centers on the waypoint, from which only the the latitude and longitude entries in the query string are important. Waypoint names that begin with a '`+`' are treated as hidden waypoints.

3. CONVERTING CHM DATA

This collection of files works well for the CHM project, but is not convenient for the project described here. The waypoints in different files need to be matched up by coordinates and edges need to be added for adjacent waypoints within a route to create a graph.

[1]See `http://cmap.m-plex.com/docs/sources.php`

3.1 Creating an Initial Graph

An author-developed Java program is used to create a graph in memory that represents the highways in one or more CHM systems, optionally restricted by region. The basic idea is simple – for each waypoint file, a graph vertex is created at each waypoint, and a graph edge is created connecting each pair of consecutive waypoints, labeled with the name of the route being processed.

Graph vertex labels usually take the form

```
NY5@ChaDr 42.929044 -74.175761
```

where the name is the route name followed by `@` and the waypoint name from the route's list. Edges are defined by their vertex endpoints and are labeled with the name of the highway connecting them. Edge lengths are not stored, as they can be computed as needed from endpoint coordinates.

However, there are complications to consider even as the initial graph is being created:

- Intersecting highways will each have a waypoint for the other. These will have the same (or at least very nearly the same) coordinates and must be represented by a single graph vertex.
- Highways often overlap each other for part of their routings (sometimes called a "concurrency" or a "multiplex"). For example, Interstates 80 and 90 share the same roadway between western Indiana and central Ohio. Vertices and edges along a concurrency should represent the points and segments of all concurrent highways, but each should exist only once in the graph.

Vertex and edge labels representing multiple routes are handled by adding the information from the new highway to the appropriate vertex or edge as each highway is processed. Vertex labels, however, can end up being quite complex using this approach. When a waypoint corresponding to an already-existing vertex is encountered, the vertex label is extended to include information about the new highway that visits that location. For example, a simple intersection between two highways, `NY5` and `NY103`, would be given the label `NY5@NY103&NY103@NY5`. A number of techniques are applied to simplify such labels. This one would be simplified to `NY5/NY103`. Even after applying several simplifications, each of which looks for a pattern, some waypoints remain unsimplified and have fairly long labels. Each highway encountered that has a segment connecting the same two vertices results in an augmentation of the corresponding edge's label with the new highway's name.

3.2 Graph Data File Formats

After simplifications have been applied, the in-memory representation of a graph is written to a file. The file format is straightforward and reasonably easy to read back in to a graph structure. The file consists of three sections: (*i*) one line containing the number of vertices $|V|$ and edges $|E|$ in the file, (*ii*) $|V|$ lines specifying vertices with a waypoint label and the point's coordinates, and (*iii*) $|E|$ lines specifying the edges, with two vertex numbers (determined by their order in the vertex list section of the file) and an edge label specifying the routes that traverse this edge. For example, the file that includes all CHM-plotted highways within the state of New York looks like this:

```
NY67_W http://www.openstreetmap.org/?lat=42.956689&lon=-74.239748
GuyParkAve http://www.openstreetmap.org/?lat=42.955041&lon=-74.220500
NY30/67 http://www.openstreetmap.org/?lat=42.937480&lon=-74.192777
ChaDr http://www.openstreetmap.org/?lat=42.929044&lon=-74.175761
```

Figure 4: Waypoint data file entries for New York state highway 5 near Amsterdam. The entire route consists of 234 waypoints of this form.

```
8260 9189
I-78@NJ/NY 40.727649 -74.021244
I-78@HudSt 40.723714 -74.007908
... 8257 more vertex entries ...
NY878@AtlBeaBr 40.596304 -73.736973
0 1 I-78
2 3 I-81
... 9186 more edge entries ...
8258 8259 NY878
```

4. HIGHWAY DATA EXAMINER

Perhaps the most desirable feature of the data sets created by this project is the ability to visualize them in map form using a tool called the *Highway Data Examiner (HDX)*. HDX is built upon the Google Maps API. It can visualize both the graph data itself and the results of computations that use the data.

Google provides a number of application programmer interfaces to access their mapping capabilities, including one that allows embedding of a map into a web page using JavaScript. Developers must obtain an API key from Google to use the Google Maps API[2], but the service is free for non-commercial use. The HDX web-based viewer is available for use by instructors, so instructors (and students) need not obtain an API key or even do any JavaScript programming unless they wish to host and possibly extend HDX at their own site, or wish to develop other Google Maps applications.

Since its release in 2005, the Google Maps API has been used in countless web sites, including some whose primary purpose is educational. Some are focused on web development (e.g.,[5]), while others use the maps as a tool in a course (e.g., [4, 8]).

4.1 Using the Google Maps API

HDX uses several Google Maps API JavaScript objects and methods[3]. The map itself is displayed in an HTML `div` with `id="map"` by constructing a `GMap2` object

```
map = new GMap2(document.getElementById("map"),
        { mapTypes: [mapMapnik, mapOsmarender] });
```

The `mapTypes` parameter directs the `GMap2` to include OpenStreetMap map images in addition to the default map images from Google. A desired waypoint (graph vertex) at latitude `lat` and longitude `lon` with is added to `map` as follows:

```
marker = new GMarker(new GLatLng(lat, lon));
map.addOverlay(marker);
marker.bindInfoWindowHtml(markerinfo);
```

where `markerinfo` is the HTML to be displayed when a user clicks the marker on the map. A line connecting waypoints, representing a graph edge or a path among a collection of waypoints is added as a `GPolyline` object:

```
conn = new GPolyline(edgePts, "#0000FF", 10, 0.4);
map.addOverlay(conn);
```

where `edgePts` is an array of `GLatLng` representing the coordinates of the points along the path, and the remaining parameters indicate that the line should be blue, have a width of 10, and a transparency of 0.4. These objects are created and destroyed over the life of a map as different graphs or routes or other items are displayed.

4.2 Using HDX

HDX is launched by visiting the appropriate URL, `http://courses.teresco.org/chm/viewer/`, specifying either a query string parameter or using a file selector on the page to choose the data file to be visualized.

HDX has two plotting modes intended for student and instructor use. Graph mode (Figure 5, top), which is invoked when a file with a `.gra` extension is specified, plots all waypoints and connections between them. Clicking on a waypoint's marker brings up its name, waypoint number, and coordinates. Graph mode is also appropriate for waypoint-only displays by specifying 0 edges. Path mode (Figure 5, center), which is invoked when a file with a `.pth` extension is specified, plots the waypoints in the file with connections drawn between successive points. This mode is suitable for viewing route paths computed using programs such as implementations of Dijkstra's Algorithm.

5. PEDAGOGICAL USES

As with many related projects (e.g., [3]), the motivation for this project is pedagogical. Here, the project began as part of a Data Structures assignment to compute single-source shortest paths using Dijkstra's Algorithm [10]. Students were provided with data sets that were predecessors of those described here and a framework of a Java program that they were required to complete. Their tasks included adding methods to read and parse a graph data file to create an appropriate in-memory representation, to traverse the graph to compute various statistics about the data, to output the waypoints in the graph, and to compute the shortest path between any two waypoints. Students were encouraged, but not required, to use an existing graph implementation in the "structure package" software that comes with Bailey's free textbook [1]. The granularity of the data was ideal for such a project – not so small that it would be considered a "toy" data set, but not so large as to be overwhelming. The fact that students were not only using and programming a very relevant algorithm, but they were generating meaningful results in a context similar to those produced by tools they use regularly made it a very compelling assignment.

[2]The API key is optional in Version 3 of the Google Maps JavaScript API, so instructors and students will be able to install HDX on their own servers without obtaining a key once HDX has been updated to use the new API.

[3]The description here refers to the Google Maps API V2.

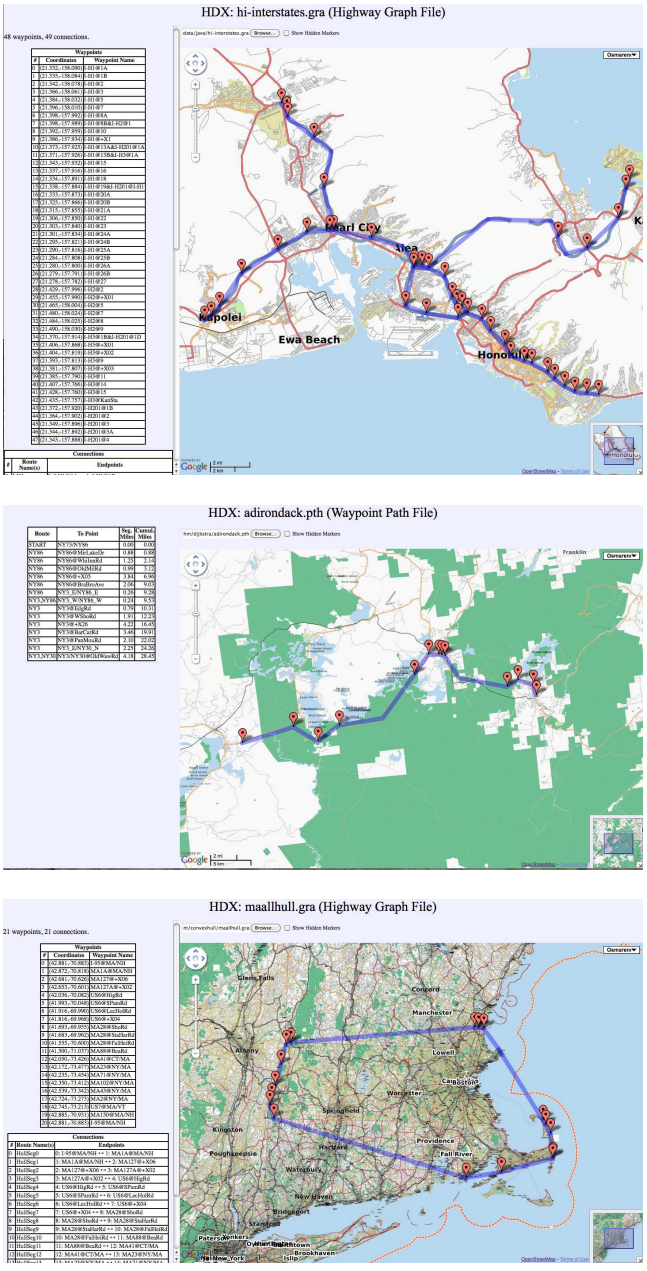

Figure 5: HDX examples. Top: graph mode view of the Hawaii Interstate highways. The raw file contents are shown on the left. Center: path mode view of a computed routing between Tupper Lake and Lake Placid, New York. Route name, waypoint, and distance information about the route shown is on the left. Hidden waypoints are not shown as markers in either case, but may be displayed by selecting the "Show Hidden Markers" toggle. Bottom: a graph mode visualization of a computed convex hull of the vertices at all waypoints in Massachusetts.

The project was expanded for use in an Analysis of Algorithms course [11, 12] to provide some class examples (e.g., Figure 5, bottom, shows a computed convex hull) and an expanded assignment to compute shortest paths.

In each case, students had the ability to visualize the results of their programs using an earlier version of the Google Maps-based viewer. Students did not take advantage of this as frequently as hoped, mainly because of the awkward procedure required at the time. Students needed to create their output files, upload them to a repository on the viewer's web server, then visit an appropriate URL in their browser to see their results. The current version of HDX eliminates the need to upload result files by providing a file selector that allows students to plot data from a file on their local computer in just one step.

5.1 Available Data Sets

The CHM project is constantly expanding to include new highway systems in different parts of the world, and improving the quality of the data in existing systems. This data can then be converted to graph form for use in this project. As an added bonus, the conversion process often points to potential errors and omissions in the CHM data, improving the data for both projects' purposes.

Table 1 shows many of the graphs that have been created and which are available for use in examples and assignments at the time of this writing. While all of these sets represent the same type of data, their sizes and other other characteristics vary widely. The range of sizes is especially important in an academic setting. The smallest data sets can be used to trace through an algorithm manually or to aid in debugging of programs. The moderately sized sets can be good for realistic examples and to test programs that perform correctly for small data sets. The largest sets can be used to test scalability of implementations and to help with performance analysis studies. The data sets also vary in spatial scale, density, and connectivity. The latter can be especially important for testing special cases of some algorithms. For example, systems that include highways on islands not connected by bridges to the mainland will include multiple disconnected components (ferry routes are not plotted).

6. CONCLUSIONS AND FUTURE WORK

The data sets and mapping capabilities described have been used for class examples and assignments in two courses. The system allowed students to work with real-world data and provided a capability for them to visualize their results. The assignments led to interesting discussions about what would be needed to extend to more fine-grained data, to add time estimates in addition to distances, and how routes could be converted to reliable turn-by-turn directions.

The number and quality of data sets has been increased significantly, and HDX is now much easier to use. The system will continue to be used in the author's courses and is available for use elsewhere. HDX works quite well for smaller data (up to a several hundred `Gmarkers` and `GPolylines`), but initial display and manipulation of maps with too many items added bogs down the Google Maps implementation. More efficient operation on larger sets is being investigated.

A second graph data organization is being developed that will make graph data more compact and visualizations less dense. Hidden waypoints are very helpful for improving the accuracy of routes. However, for the purposes of a graph in

Table 1: Available CHM-based graph data sets.

Data set	$\|V\|$	$\|E\|$
de-interstates: Delaware Interstate highways	26	24
hi-interstates: Hawaii Interstate highways	48	49
canyt: Yukon territorial highway system	351	354
hiall: All plotted highways in Hawaii	549	576
usact: Connecticut state highway system	1524	1751
usama: Massachusetts state highway system	1828	2099
maall: All plotted highways in Massachusetts	2284	2759
usapa: Pennsylvania state highway system	6691	6499
nyall: All plotted highways in New York	8260	9189
manyall: All plotted highways in New York and Massachusetts	10,468	11,906
canada-all: All plotted highways in Canada	16,177	16,436
usai: United States Interstate highway system	21,554	21,505
usaus: United States U.S. highway system	57,281	58,625
usanational: All plotted national highway system routes in the United States	82,362	86,041
usa-all: All plotted (state and national) highways in the United States	160,928	174,157
uscanada-all: All plotted highways in the United States and Canada	177,089	190,593

this project, they are almost always waypoints represented by a vertex of degree 2 and would be unlikely to be meaningful. Such hidden waypoints and the extra edges they induce between visible points can be collapsed into a single edge with the hidden waypoint positions stored in a list associated with that edge. This requires a new graph data structure and file format, and corresponding changes both to programs that use the data, and to HDX.

A longer-term goal for the project is to use the data and an enhanced viewer to implement interactive visualizations of algorithms that operate on these data sets. Adding, removing, and changing the colors and sizes of the `GMarker` and `GPolyline` objects on the map as an algorithm proceeds could enhance student understanding of the algorithm. The range of sizes of the data sets will again be a strength, as small sets can be used to trace through step-by-step interactive visualizations while larger sets would be more appropriate for an automated overview showing progress on a larger scale. This would be similar to the visualization described in [3], but making use of the wide range of data sizes and the additional information (i.e., route names for edge labels) available with the processed CHM data.

7. ACKNOWLEDGMENTS

Thank you to CHM Project leader Tim Reichard and the project collaborators for use of their highway data. Some of the JavaScript code in HDX is based heavily on CHM programs written by Reichard. Thank you to the anonymous reviewers whose feedback helped to improve this paper. And finally, thank you to the students in the Fall 2009 COMSC 211 Data Structures course at Mount Holyoke College and the Spring 2011 CSIS 385 Analysis of Algorithms students at Siena College for their patience and feedback.

8. REFERENCES

[1] D. A. Bailey. *Java Structures, Data Structures in Java for the Principled Programmer*. $\sqrt{7}$ edition, 2007. `http://www.cs.williams.edu/~bailey/JavaStructures/Welcome.html`.

[2] Google Maps API reference. `http://code.google.com/apis/maps/`, 2010.

[3] V. Karavirta. Real-world, student selectable data for education – learning graph algorithms. In *Proceedings of the IADIS International Conference Cognition and Exploratory Learning in Digital Age*, pages 129–136, Timisoara, Romania, October 2010.

[4] D. J. Malan. Reinventing CS50. In *Proceedings of the 41st ACM technical symposium on Computer science education*, SIGCSE '10, pages 152–156, New York, NY, USA, 2010. ACM.

[5] J. McGookey. "Mashup in a day," a mashup lesson plan. Grand Valley State University Technical Library, Paper 56, 2009. `http://scholarworks.gvsu.edu/cistechlib/56`.

[6] T. L. Naps, G. Rössling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and J. A. Velázquez-Iturbide. Exploring the role of visualization and engagement in computer science education. *SIGCSE Bull.*, 35:131–152, June 2003.

[7] OpenStreetMap: the free wiki world map. `http://www.openstreetmap.org/`, 2010.

[8] Y. Ouyang, M. Lehmann, K. Hayden, and D. Kilb. Opportunities presented when developing learning resources for middle schoolers. *J. Comput. Small Coll.*, 24(4):144–150, 2009.

[9] T. Reichard. The clinched highway mapping project. `http://cmap.m-plex.com/`, 2010.

[10] J. D. Teresco. A Dijkstra's algorithm shortest path assignment using the Google Maps API: poster session. *J. Comput. Small Coll.*, 25(6):253–255, 2010.

[11] J. D. Teresco. Graph algorithms using highway mapping data and the Google Maps API. Poster, 2011. SIGCSE 2011, The 42nd ACM Technical Symposium on Computer Science Education.

[12] J. D. Teresco. Using the Google Maps API with highway mapping data as a pedagogical tool: demonstration session. *J. Comput. Small Coll.*, 26(6):58–60, 2011.