ISSN: 2473-3857
v3 n4325

Improved Highway Data and Map Visualizations for Teaching Data Structures and Algorithms

James D. Teresco
jteresco@siena.edu
Department of Computer Science
Siena College
Loudonville, NY 12211 USA

Abstract

Teaching and learning about algorithms and data structures is more interesting and enjoyable when working with data that has a real-world connection and with results that can be visualized. The project described here is an ongoing effort to meet these needs. The goals of this project, Map-based Educational Tools for Algorithm Learning, are to provide a set of data files based on highway systems worldwide in a convenient graph format, and support tools for the visualization of this data and the results of programs that implement algorithms that use this data. It also provides sample assignments and solutions for instructors who wish to make use of this work. The data itself is derived from a travel hobbyist project named Travel Mapping, and is provided in the form of plain text data files that describe systems of highways in different parts of the world in the form of a graph. Especially attractive features include the variety of data sizes available and the ability to visualize this data on a map using the Google Maps API.

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

1. INTRODUCTION

Examples and assignments in courses like data structures or algorithms can sometimes seem dry, boring, and contrived. This can make it hard for everyone involved to remain engaged in study of even very interesting algorithms.

The Map-based Educational Tools for Algorithm Learning (METAL) project is an effort to provide interesting data sets and meaningful visualization capabilities for use in class examples and assignments for the benefit of both instructors and students. It is the successor project to the Clinched Highway Mapping as Pedagogical Tool project (Teresco, 2012). The current project builds upon the previous work by expanding the scope and improving the quality of the data, providing that data in multiple formats, and by extending and improving the capabilities of the visualization system.

This paper describes the overall project, with the recent improvements highlighted. The motivation for the project is in Section 2. A summary of the available graph data is given in Section 3. The Highway Data Examiner (HDX) visualization tool is described in Section 4. Example usage of the data sets and visualization in classes is in Section 5. Section 6 presents results of a survey of students who used METAL in class work. Background information about the Travel Mapping hobbyist project (TM), which provides the raw data for this project, is described in Section 7. Section 8 details how TM data is converted into METAL's graph data. Conclusions and some thoughts on the current and future directions of the project are discussed in Section 9.

2. MOTIVATION

This project grew out of the author's dissatisfaction with a data structures assignment about Dijkstra's algorithm for computing single-source shortest paths on a graph. The first enhancement of that assignment that used highway mapping data and provided

ISSN: 2473-3857
v3 n4325

visualization capabilities using the Google Maps API was very successful (Teresco, 2010), and led to the greatly expanded project described here.

Choosing a data set for Dijkstra's algorithm and related topics presents some challenges. To be explored thoroughly in class or as part of a homework assignment, the data must necessarily be small, but should still be interesting. A common approach is to find or construct a graph that has some small collection of locations (e.g., cities, airports, rooms in a building) to use as vertices, and edges that are labeled with a cost (e.g., time, distance, price) to travel between adjacent locations. Many textbooks include such examples, e.g., (Bailey, 2007) Exercise 16.7, p. 436, and Figure 1 (top) shows an example graph often used by the author (Teresco, 2012).

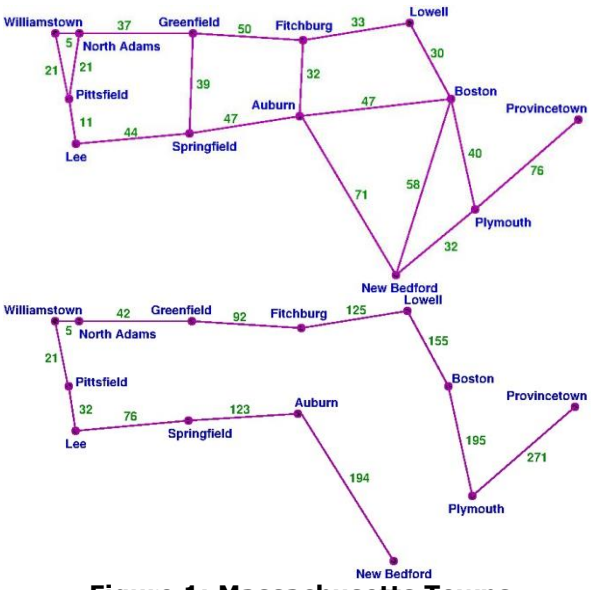

Figure 1: Massachusetts Towns

Figure 1 (Top): a simple graph of towns and the distances between them in Massachusetts that can be used to demonstrate graph structures and algorithms. (Bottom): Single-source shortest paths computed on the graph.

This graph works well for several reasons. It is derived from real-world information and it is small enough to trace through an example in its entirety. Graphs with similar structure can easily be created and used for exam questions. The data sets presented here would not necessarily replace these simple examples, but more likely supplement them.

The beneficial attributes of METAL data and tools described in the following sections include:

- The data is based on actual places: the *waypoints* along major highways from the TM data.
- The granularity of the data is very convenient. It is fine enough to be able to represent many major roads and their intersections, but coarse enough to ignore minor places and intersections.
- Provided data files, all in a common format, range in size from just a handful of vertices and edges, useful for hand-traced examples or for debugging code, to hundreds of thousands of vertices and edges, useful for meaningful performance analysis, and everything in between. Each file represents a subset of highways in North America, Europe, Asia, or Africa.
- Each graph is provided in two formats (Section 8). For "simple" graphs, edges are straight lines, while "collapsed" graphs have edges with intermediate points included that improve the accuracy of the edge's shape and length relative to the road segment it represents.
- The data and results produced by programs that use the data can be visualized in map form.

The use of visualizations in learning data structures and algorithms has a long history. See, e.g., (Naps, Rößling, Almstrum, Dann, Fleischer, Hundhausen, Korhonen, Malmi, McNally, Rodger & Velázquez-Iturbide, 2003), (Shaffer, Cooper, Alon, Akbar, Stewart, Ponce & Edwards, 2010). The focus here is on the highway-based graph data and on the pre- and post-processing visualization capabilities of METAL, and their use in courses.

3. AVAILABLE DATA SETS

METAL graph data is regenerated (as detailed in Section 8) each time there is an update to the underlying TM highway data (as detailed in Section 7). Graphs include both simple and collapsed formats for each of the following: (1) one set for all TM data, (2) one set for each region (e.g., state, province) and country, (3) one set for each large multi-region highway system (e.g., U.S. Interstates, European "E" roads), (4) one for each continent, (5) a set of area graphs for highways within a specific radius of a place like a college or a major city, and (6) a set of graphs spanning a specific group of regions (e.g., the New England states in the U.S.). As of this writing, there are 596 graphs in the ever-expanding set, ranging in size from 3 vertices and 2 edges up to 439,746 vertices and 486,665 edges.

Small graph data sets are useful for debugging of programs or even manual tracing. Somewhat

ISSN: 2473-3857
v3 n4325

larger sets allow for more realistic situations to be tested. Scalability and performance analysis studies are possible with the largest sets. Graphs are derived from data in various parts of the world leading to differences in spatial scale, density, and connectivity, as well as size.

4. HIGHWAY DATA EXAMINER

A key feature of METAL is its open-source *Highway Data Examiner (HDX)* visualization tool (http://courses.teresco.org/metal/hdx/), leveraging the Google Maps API. HDX can visualize raw TM data, METAL's graph data, or the results of computations that use the data. Users can access METAL's HDX installation or could host their own instance. HDX now supports more file, its user interface has been redesigned to focus more on the map, and many usability improvements have been made compared to earlier versions (Teresco, 2012).

HDX has several modes of operation, depending on the format of the file loaded. Some are intended for TM or METAL developers, while others are intended for students and instructors. Figures in the next section are HDX screen captures. The HDX map inherits a full set of Google Maps features, including zoom and pan functionality, and a variety of map tile backgrounds can be selected.

5. USAGE IN COURSES

METAL and its predecessor projects have been used and improved in the author's courses nearly every semester for several years, and other instructors have used it in courses elsewhere. This section highlights some of the tasks students have been assigned. A few of these uses have been described previously (Teresco, 2010, 2012), others are new, but all have been improved and expanded.

Using Waypoint/Vertex Data Only

The first time students are asked to work with METAL data, the task is typically centered around reading and parsing the contents of TMG files (Section 9), but limited to the vertex data. Students write a program that searches for the vertices that are at the northernmost, southernmost, easternmost, and westernmost locations, and searches the vertex labels for the longest, shortest, first alphabetically, and last alphabetically. For a more challenging task, students could be required to keep track of all vertices that tie for the "win" for a given search criterion. In a data structures course, for example, this can be an engaging way to refresh prerequisite knowledge.

A subsequent task involves creating a data structure to encapsulate the information of a graph vertex (label and coordinates), and storing an instance for each vertex in an array or ArrayList. This collection can then be used in a simple interactive search such as printing all labels that have a given prefix, substring, or length.

Building a collection of objects representing the vertices also presents an excellent opportunity when studying sorting. This data has many natural sorting criteria, which motivates and demonstrates the power of a comparator-based sort, where different comparators allow that same implementation to sort by various criteria.

An especially successful task is associated with a lab where students implement a generic data structure that keeps track of the *k* "best" (by some criteria, ideally determined by a comparator) of *n* values added to the structure. Bailey (2007) refers to this as a "Best Of" ordered structure in a lab assignment (p. 275). Requiring students to find the *k* best for different values of *k*, for different comparison criteria, and for the larger of the METAL data sets forces them to make their implementations sufficiently generic and efficient.

Even in these vertex-only examples, HDX can be used to visualize the input data and the results in both map and tabular form. It can be difficult to look at some numbers in console output and decide that they appear to be correct. It is much easier to see that, for example, the directional extreme points are correct, when viewing in HDX. Figure 2 shows a small example graph, the primary highway system in Andorra (the "*Carreteras Generales"*). Figure 3 shows a visualization of the extreme points as written to a "waypoint list" (.wpl) file.

Figure 1: METAL's GRAPHS

Figure 2: METAL's graph of highways in Andorra as shown in HDX. The scrollable table on the right lists all vertices and edges.

ISSN: 2473-3857
v3 n4325

Among the more advanced assignments restricted to vertices only that have used METAL is the computation of the convex hull of the vertex points. The resulting polygon can also be visualized by HDX.

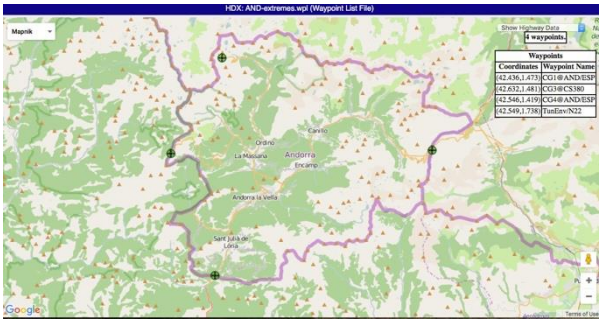

Figure 3

Figure 3: A visualization of the results of a search for the extreme points in the Andorra graph from Figure 2, shown in HDX. It is much more quickly evident from the map than from text output that the answers are correct.

Introducing Edge Data

Students in sufficiently advanced classes construct and work with a graph structure. Students can be provided with or be required to build their own graph data structure and appropriate support structures to use to store graph vertex and graph edge labels. For collapsed format graphs, the edge structures are non-trivial.

With both vertices and edges available, search algorithms can be extended to find features such as the vertex with the highest degree, the longest and shortest edges, the edges with the most intermediate shaping points, and the edges with the most concurrent routes (multiple routes that share the connection represented by an edge are separated by commas in the edge labels in TMG files). The "best of" ordered structure can also be reused to find the *k* longest or shortest or other features of edge data, reinforcing the importance and power of building a generic data structure. These can also be visualized with HDX, using its "near miss point" (.nmp) format, illustrated in Figure 4, which shows the 5 longest edges in the Andorra graph. HDX takes pairs of waypoints from an NMP file and plots them on the map with an edge connecting them.

Figure 4: HDX visualization of the 5 longest edges (without shaping points shown) in the Andorra graph from Figure 2. The visual feedback of the map makes it clear that these are long edges, and likely the correct result.

Figure 4

The original motivation for this project and still a very successful assignment is "Dijkstra's Road Trip," where students are required to implement Dijkstra's algorithm for single-source shortest paths. Once a graph is in memory, Dijkstra's algorithm is used to find the shortest path (in distance, not time, since METAL does not have speed limit information) from a starting graph vertex to a destination graph vertex. The algorithm itself brings together several topics from a data structures or algorithms class including graph operations, priority queues, and tables. Using METAL data for this adds a real-world aspect to the assignment, and HDX allows students to visualize the shortest paths they have computed. For example, the shortest path computed from TunEnv/N22 to CG6@CS600 in the Andorra graph of Figure 2 is shown in Figure 5. Here, HDX is displaying the contents of a "waypoint path" (.pth) file produced by a Java program that implemented Dijkstra's algorithm.

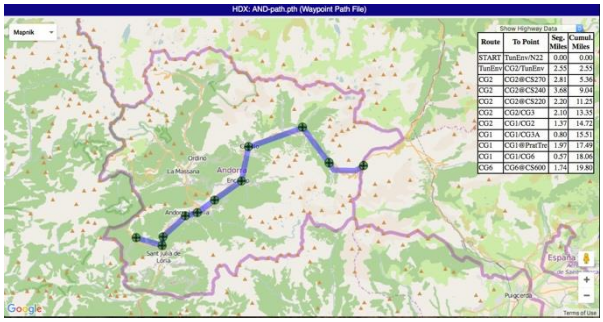

Figure 5

Figure 5: HDX visualization of the shortest path computed by Dijkstra's algorithm from TunEnv/N22 to CG6@CS600 in the Andorra graph from Figure 2. The fact that this appears on the map to be a reasonable route from the source to the destination gives useful feedback about the correctness of the result.

Breadth-first and depth-first graph traversals have also been part of assignments using METAL data and HDX.

In summary, METAL has been used in courses for vertex and edge search algorithms, including straightforward searches for a maximum or minimum value and finding the *k* "Best Of" values, finding convex hulls, performing depth-first and breadth-first traversals, and using Dijkstra's algorithm for single-source shortest paths. In each of these cases, the ability to see the input data and results on the map can aid in the understanding of what the algorithm is trying to compute and to see that the results are correct.

Details of assignments using METAL, including handouts, starter code, and sample solutions, are available to instructors upon request.

6. STUDENT EVALUATION

METAL's data and visualization capabilities have enabled class examples and assignments that students in several of the author's courses have found engaging and challenging.

Students in the most recent course to work with METAL, the author's Spring 2017 Analysis of Algorithms course at Siena College, were given the option to take a short survey about their experience (Siena IRB #04-17-049). 34 students responded and the results were overwhelmingly positive. The first three questions focused on items related to the parts of METAL presented in this paper. Table 1 summarizes the responses.

	Q1	Q2	Q3
Strongly Agree	8.8%	35.3%	35.3%
Agree	47.1%	47.1%	50.0%
Somewhat Agree	29.4%	11.8%	8.8%
Neither Agree/Dis	8.8%	2.9%	8.8%
Somewhat Disagree	2.9%	0.0%	0.0%
Disagree	0.0%	0.0%	0.0%
Strongly Disagree	2.9%	2.9%	0.0%

Table 1: Student survey responses.

Q1: "Working with the highway graph data is more interesting and engaging than working with other data sets." Q2: "The ability to view the highway graph data in map form makes it more interesting and engaging to me." Q3: "The fact that the highway graph data has a direct real-world connection makes it more interesting and engaging to me."

These responses, while a small sample, are very encouraging. Almost all students agreed to some extent with the project's assumptions that the real-world highway data and ability to visualize in map form are interesting and engaging, and much of their agreement was quite strong. For question 1, the intent was that they compare using map data against synthetic examples they had seen in class and assignments.

7. THE TRAVEL MAPPING PROJECT

This section provides background information about the Travel Mapping project, which provides METAL's raw data.

The TM project (Travel Mapping, 2017) is being developed and maintained by a group of travel enthusiasts as a way to track and view their cumulative travels, inspired by the dormant Clinched Highway Mapping (CHM) project (Reichard, 2010). TM's users submit lists of highway segments they have traveled, and those are matched with highway data to produce statistics and maps of the user's travels.

Travel Mapping Data

There are many ways to obtain data to construct graphs of roads and places. For example, the OpenStreetMap project (OpenStreetMap, 2016) provides a massive amount of data that could be adapted. The TM data described below has proven to be a convenient and appropriate source of data for METAL's graphs.

TM highway data consists of thousands of plain-text files, organized both by region (often an entire country, but states/provinces/equivalents for larger countries), and highway system, such as the U.S. Interstate Highways, the New York State Touring Routes, or German Autobahns. For example, the segment of Interstate 90 within Ohio would be represented as a TM route, and would be placed in the OH region (Ohio) and the usai highway system (U.S. Interstates). Individual routes, or segments of routes within a region, are each represented by a waypoint file. These files approximate the route of the highway using a list of waypoints in the order they occur from one end of the route to the other. A waypoint consists of a label, normally the name of an intersecting road or exit number, and an OpenStreetMap URL that encodes its coordinates. The file representing Washington's Route 523 in Seattle is shown in Appendix A.

TM highway data is always being expanded, corrected, and otherwise improved by a team of volunteers. As of this writing, 33,011 routes across North America, Europe, Asia, and Africa have been plotted for a total of 984,592 miles (1,584,547 km). 450,038 unique locations have been plotted as starting, ending, intersecting, or

ISSN: 2473-3857
v3 n4325

other interesting points along these routes. Data is derived only from permissible, open sources (http://tm.teresco.org/credits.php). Routes are normally plotted from map imagery from the OpenStreetMap project or open government satellite or topographic imagery. In particular, since it is unclear whether data created by selecting points from Google Maps would be permitted by Google's terms of use, TM volunteers take care not to derive project data from Google Maps or other proprietary imagery. However, TM and METAL do use the Google Maps API extensively, in accordance with Google's terms of use.

Shaping Points

The granularity of TM data makes it an ideal source from which to generate METAL graphs. Intersections to be included as waypoints are chosen to obtain a reasonably accurate approximation of the route. Less significant intersections that would add memory and computational costs to the data processing, database, and mapping capabilities without much improvement in accuracy of the route are omitted. Choosing only major intersections usually leads to routes with a sufficiently accurate approximation. Consecutive waypoints are rarely more than a few miles apart and are often closer.

In situations such as limited-access routes with long sections between interchanges or curvy routes, the usual points might not provide sufficient accuracy. *Shaping points*, invisible to TM users other than as improvements in map accuracy, can be added to the route (as regular waypoint entries whose labels begin with '+') to improve its approximation. For example, Figure 6 shows the section of Interstate 90 in Massachusetts between interchanges 2 and 3, a limited-access road through a mountainous area. The direct distance between the two endpoints (shown by the straight purple line in the figure) is 28.3 miles, but the actual road distance is 30.3 miles. By adding 9 shaping points, the segment is represented more accurately by 10 shorter, but still straight line, segments totaling 30.1 miles.

TM data and more details about its organization are available in its public repository at https://github.com/TravelMapping/HighwayData.

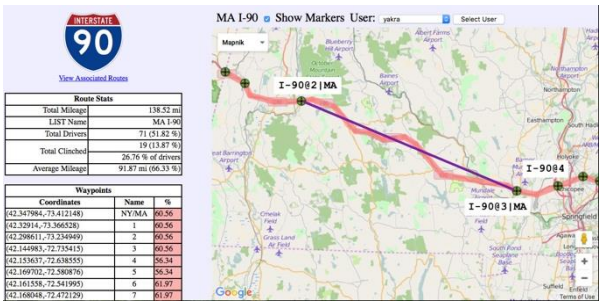

Figure 6

Figure 6: The route between interchanges 2 and 3 of I-90 in Massachusetts demonstrating the use of shaping points to improve the approximation of the road. The straight purple (darker, if viewing in greyscale) line indicates the direct path between the interchanges. The pink (lighter, in greyscale) path includes 9 intermediate shaping points.

8. CONSTRUCTING GRAPH DATA

TM raw data can be interesting for educational use, but its real value arises once this data has been used to produce METAL's graphs. The basic idea remains similar to that used in the predecessor project (Teresco, 2012): the waypoints from each route become graph vertices, and the connections between consecutive waypoints in a route become graph edges. This section provides details about that process for completeness, but the typical METAL user would only be concerned about the graphs that result from this process.

Graph Construction Process

Graph generation is integrated into TM's site update process, which populates the database used by TM's web front end with information gathered from highway and user data files.

Each graph vertex is labeled based on the route name and the waypoint label. The graph vertex corresponding to the first line for Washington 523's file in Appendix A would be named WA523@WA99, and its coordinates would be taken from the URL. The edge representing the connection between the first two lines in the same file would be labeled with the route name, WA523. The length of the edge is not computed or stored at graph creation time, as it can be computed when needed using the coordinates of its vertex endpoints.

Waypoints on intersecting routes should share a single graph vertex for their junction, so a search is performed as each waypoint is processed to find any existing vertex with the same coordinates. If a vertex already exists, this

ISSN: 2473-3857
v3 n4325

waypoint is added to that vertex. The efficiency of this search is greatly improved by storing the waypoints in an adaptive quadtree structure (Samet, 1985).

Some highways overlap (i.e., have a "concurrency") for part of their routings. For example, part of Washington's Capital Beltway carries designations of both I-95 and I-495. Corresponding waypoints at each location along a concurrency, like intersecting points, should share a single graph vertex. Graph edges along a concurrency should also exist just once, but be designated as carrying all the routes along that concurrency.

Intersection and concurrency detection rely on the coordinates being precisely the same in the source data. Work is ongoing by TM volunteers, using both automatic and manual techniques, to align points where this is not the case.

Graph vertex naming is complicated by locations that have multiple corresponding waypoints due to intersections and concurrencies. Each graph vertex has a "failsafe" label, built by concatenating route/waypoint label information from each concurrent waypoint, separated with '&' characters. Simplification rules then produce a "canonical" label to reduce the number of excessively long labels. Some examples are shown in Appendix B. There is no harm in long, complicated labels other than that the simpler labels are easier to type and display.

Waypoint labels can be repeated in different regions that contain parts of the same route. I-95 along the U.S. east coast has 9 points labeled "1" for an exit numbered 1 (the route's exit numbers reset at state boundaries). To differentiate vertex labels and maintain vertex label uniqueness when the canonical labels conflict, the region can be appended. For example, I-95's Exit 1 in Georgia has vertex label I-95@1|GA.

In the end, all vertices have unique labels, and the majority of complicated failsafe labels are automatically simplified.

Collapsing Shaping Points

An important new feature of METAL is its improved handling of the shaping points from TM data during graph generation. Previously, shaping points were treated the same as any other, except that the visualization tools (Section 4) would hide them on the map. This original "simple" format remains supported, but each graph is also provided in a "collapsed" format.

Since shaping points are not in themselves useful or meaningful points, just accuracy improvements, the collapsed format in METAL treats them as points that help define the edge between two vertices with more accuracy. Using the example from Figure 6, the edge connecting the endpoints of the long, curvy segment, with vertex labels I-90@2|MA and I-90@3|MA, includes the list of latitude-longitude pairs for the shaping points along that path.

The in-memory graph is first built with shaping points represented as graph vertices. The transformation of this simple graph into a collapsed graph requires a single traversal of the graph vertices. Since shaping points are not intersections of routes represented in the TM data (they would be regular waypoints in that case), graph vertices that represent shaping points must have exactly two incident edges. A data error is flagged if this is not the case. To collapse the vertex representing the shaping point, that vertex and its incident edges are removed from the graph, and a new edge is created connecting the vertices at the opposite ends of the removed edges. The coordinates of the shaping point being removed are added to the list of intermediate points along that edge.

The simple graph generated in the area shown in Figure 6 has a vertex labeled I-90@+X12|MA that represents the shaping point between I-90@3|MA and I-90@4. When the vertex I-90@+X12|MA is visited during the collapse procedure, that vertex, and the edges that connect it to I-90@3|MA and I-90@4 are removed, and a new edge is introduced with I-90@3|MA and I-90@4 as endpoints, with the coordinates of I-90@+X12|MA attached to that edge's list of intermediate points.

Care is taken to ensure the ordering of intermediate points remains correct as vertices are collapsed during the conversion process. Even though the graph is undirected, the order of the intermediate points must remain consistent. The edges being removed could each have a list of intermediate points from previous collapsing steps that need to be combined with the new point from the vertex being eliminated. Fortunately, all of this is handled during graph creation, and is not the concern of students or instructors using this data.

Graph Data File Formats

Once the in-memory graph construction is complete, the overall master graphs and

hundreds of subgraphs are created and written to files. The "Travel Mapping Graph" (TMG) file format is straightforward, consisting of four sections:

1. A single header line: the string TMG, a version number, and the string simple or collapsed to specify which graph format is expected.

2. A line containing the number of vertices |V| and edges |E| in the file.

3. |V| lines specifying vertices, each consisting of a waypoint label and the point's coordinates.

4. |E| lines specifying the edges, each consisting of 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 a collapsed graph, the line can then include an even number of floating-point values, that taken in pairs, are the latitudes and longitudes of any shaping points that have been collapsed into this edge.

For example, an excerpt of the collapsed format graph file that includes all plotted highways within the state of Nebraska is shown in Appendix C. One edge shown includes coordinates for one shaping point and another includes coordinates for two shaping points. Note that the simple format graph can be represented in a collapsed file format with no edge shaping points. The distinction is made to allow those loading the file to be able to choose an appropriate data structure (one that can store those shaping points, or not).

9. CONCLUSIONS AND FUTURE WORK

New features of METAL described here include a greatly expanded and higher quality set of graph data files, graph data files provided in both the simple and collapsed formats, and a new version of HDX with significant user interface, functionality, and efficiency improvements. These enhancements, along with an expanded set of classroom-tested student tasks, make METAL an easy-to-adopt supplement to computing courses, especially data structures and algorithms.

The survey results, while not a formal study, provide encouraging feedback that students find the project interesting and engaging. More formal studies are planned to quantify METAL's effectiveness as a teaching tool.

Ideas for new class examples and assignments that could use METAL in the future include quadtree data structures and algorithms, sorting, finding connected components, graph partitioning, spanning trees, and any number of more advanced graph algorithms. Usage to date has only scratched the surface of METAL's potential to supplement computing curricula from the K-12 and undergraduate computing/information technology literacy audiences, to introductory programming, right up to advanced undergraduate and graduate level courses.

The scope and quality of graph data improves every time the underlying TM data improves. Each time METAL is used in a course, that use is expanded and improved based on the previous course's results. New features are being added and other improvements are being made to HDX. An especially exciting development, to be described separately, is the implementation currently underway of interactive algorithm visualization capabilities using METAL data and HDX.

10. ACKNOWLEDGEMENTS

Recent work on METAL was made possible in part due to the support of Summer Scholars 2017 Program, Center for Undergraduate Research and Creative Activity (CURCA), at Siena College.

Thank you to the TM and CHM volunteers for the use of the highway data they have constructed. Some of the code in HDX is based on CHM programs written by Timothy Reichard and TM programs by various contributors. Many colleagues and former students made contributions and suggestions, especially Razieh Fathi. Thank you mostly to the students in courses at Mount Holyoke, Saint Rose, and Siena, for their patience and feedback.

11. REFERENCES

Bailey, D. A. (2007). *Java Structures, Data Structures in Java for the Principled Programmer*. √7 edition., http://www.cs.williams.edu/~bailey/JavaStructures/.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S. & Velázquez-Iturbide, J. A. (2003). Exploring the role of visualization and engagement in computer science

ISSN: 2473-3857
v3 n4325

education. Publishing in ACM SIGCSE Bulletin. *SIGCSE Bulletin Inroads*, 35(2), 131–152.

Openstreetmap Contributors. (2016). Openstreetmap. http://www.openstreetmap.org/.

Reichard, T. (2010). The Clinched Highway Mapping project., http://cmap.m-plex.com/.

Samet, H. (1985). A top-down quadtree traversal algorithm. Publishing in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence*., PAMI-7(1), 95-98.

Shaffer, C. A., Cooper, M. L., Alon, A. J. D., Akbar, M., Stewart, M., Ponce S. & Edwards, S. H. (2010). Publishing in TOCE. *ACM Transactions on Computing Education*, 10(3), 9:1-9:22.

Teresco, J. D. (2010). A Dijkstra's algorithm shortest path assignment using the Google Maps API (poster abstract). Publishing in JCSC. *Journal of Computing Sciences in Colleges (CCSCNE 2010),* 25(6), 253-255.

Teresco, J. D. (2012). Highway data and map visualizations for educational use. Publishing in SIGCSE. *Proceedings of the 43rd ACM technical symposium on Computer Science Education (SIGCSE '12)*, 553-558.

Travel Mapping Contributors. (2017). The Travel Mapping project. http://tm.teresco.org/

.

Appendix A: SAMPLE TM WAYPOINT FILE

```
WA99 http://www.openstreetmap.org/?lat=47.734121&lon=-122.345123
I–5 http://www.openstreetmap.org/?lat=47.733984&lon=-122.324481
WA522 http://www.openstreetmap.org/?lat=47.733753&lon=-122.292359
```

This file is named wa.wa523.wpt, represents Washington State Route 523, is part of the WA region (the U.S. state of Washington), and is part of the usawa system (the Washington State Highways).

Appendix B: EXAMPLES OF VERTEX LABEL SIMPLIFICATIONS

METAL graph vertices are initially assigned failsafe labels as described in Section 8. A series of simplification patterns are applied to convert lengthy failsafe labels into shorted canonical labels. Three examples are given below.

• In Canada, Yukon Territory routes 1 and 3 intersect in Haines Junction. YT1 has a waypoint labeled YT3, and YT3 has a waypoint labeled YT1. These are combined to form the failsafe label YT1@YT3&YT3@YT1. A simplification rule then renames this as YT1/YT3.

• In Mittenwald, Germany, European route E533 and German route B2 are concurrent, and intersect with a street named Alpenkorpsstraße, each with a point labeled AlpStr. These are combined to form the failsafe label E533@AlpStr&B2Wei@AlpStr, which is simplified to E533/B2Wei@AlpStr.

• In Waipahu, Hawaii, Interstate highway I-H1 has an interchange labeled with its exit number, 5, where it intersects with Hawaii route 76, which uses the label I-H1/750 and Hawaii route 750, which uses the label I-H1/76. These are combined to form the failsafe label I-H1@5&HI76@I-H1/750&HI750@I-H1/76. One of the newest simplification rules renames this as I-H1/HI76/HI750.

Appendix C: SAMPLE METAL GRAPH FILE

This following is an excerpt from the METAL graph TMG file NE-region.tmg. This graph includes data from all highways in the U.S. state of Nebraska that are plotted in TM.

```
TMG 1.0 collapsed
3069 3414
US385/SD79@NE/SD 43.000818 -103.224192
NE2@SD/NE&NE71@NE/SD&SD71@NE/SD 43.001462 -103.653731
NE2/NE71@ToaRd_N 43.00094 -103.641822
US83@NE/SD 42.998496 -100.573397
... 3063 more vertex entries omitted ...
KS99/NE99@KS/NE 40.000923 -96.350698
NE65@KS/NE 40.000778 -96.16274
2941 2927 NE89
2984 2920 NE105
2927 2919 NE89
1468 1387 NE71 41.261937 -103.63811
2429 2296 US83 40.565463 -100.629981 40.640897 -100.632395
... 3408 more edge entries omitted ...
369 364 US20
```

The complete file has 3069 lines representing vertices, followed by 3414 lines representing edges. The NE71 edge includes coordinates for one shaping point, and the US83 edge includes coordinates for two shaping points.