ISBN: 978-972-8939-81-6 © 2013 IADIS

LOCATION-AWARE MOBILE LEARNING OF SPATIAL ALGORITHMS

Ville Karavirta
Department of Computer Science and Engineering, Aalto University
PO. Box 15400, FI-00076 Aalto, FINLAND

ABSTRACT

Learning an algorithm – a systematic sequence of operations for solving a problem with given input – is often difficult for students due to the abstract nature of the algorithms and the data they process. To help students understand the behavior of algorithms, a subfield in computing education research has focused on algorithm visualization – learning material showing the steps and data used by an algorithm. As the use of mobile devices has risen together with the capabilities of the devices, mobile learning is more important then ever. It also opens possibilities to contextualize the learning experience. In this paper, we present our work towards location-aware mobile learning of spatial algorithms that adapts the learning material to the location of the student.

KEYWORDS

Spatial Algorithms, Location-Aware Learning, Algorithm Visualization

1. INTRODUCTION

An *algorithm* is a set of rules for a sequence of operations typically used for some input data to solve a calculation problem. For example, sorting emails in inbox based on the subject or finding the fastest bus-route from place A to B. In Computer Science (CS) education, algorithms are an important topic for students to understand. They should be able to choose and compare different algorithms for a given problem in programming. However, in programs, the algorithms work on abstract data, making learning and understanding algorithms difficult for students. To help students, numerous *algorithm visualization* tools have been developed (see, for example, Malmi *et al.* (2004) and Rößling and Freisleben (2002)). The educational tools visualize the data used by the algorithm and the sequence of operations it performs on that data. *Spatial Algorithms* are an area of algorithms that work on data tied to locations. For example, testing if a data point falls inside a certain area on a map could be solved with a spatial algorithm. Algorithm visualization tools for spatial algorithms have also been developed (Nikander *et al.*, 2010).

Mobile devices nowadays include more and more sensors and modern smartphones are able to get a location of the device. Location is one of the most-often used attributes to provide *contextual-learning.* That is, learning experience changes based on the context. Wang (2004) defines this context to be made of six dimensions: identity of the learner, spatio-temporality, facility, activity, learner, and community. In this paper, we focus on the spatio-temporal dimension, more specifically location-aware learning.

As spatial algorithms work on location data, using maps to visualize that data is natural. Existing educational systems like the spatial extension to the TRAKLA2 learning environment by Nikander *et al.* (2010) work on abstract data and do not use maps. In the work presented in this paper, the goal is to provide engaging algorithm visualizations for learning spatial algorithms that change based on the user's location. In this short paper, we present our work in progress towards that goal.

The rest of the paper is structured as follows. Section 2 briefly presents related work on the fields of educational (spatial) algorithm visualizations and location-aware learning to provide context for our work. Section 3, in turn, describes our new location-aware content for spatial algorithms. Finally, Section 4, discusses the contributions and the planned future of this project.

2. BACKGROUND

Algorithm visualizations (AV) are actively researched for computer science. There are more systems developed than can be mentioned within the page restrictions of this paper. The systems range from those targeted for a single topic to those being general and applicable for (almost) any algorithm. The AV research suggests that user engagement of the AVs is the key to educational effectiveness (Naps *et al.,* 2003; Hundhausen *et al.,* 2002). A good example of engaging AVs is the TRAKLA2 system (Malmi *et al.,* 2004). In TRAKLA2 exercises, the student has to simulate the steps of the actual algorithm by manipulating visualizations.

There are several tools for learning a single spatial algorithm, such as the one by Fisher (2004) for Voronoi diagrams. However, they typically only let students view a visualization instead of engaging them more. Furthermore, there are relatively few educational systems capable of visualizing multiple spatial algorithms. Only such system we are aware of is the spatial algorithm extension to the aforementioned TRAKLA2 system (Nikander *et al.,* 2010). In the exercises provided by the extension, a student has to manipulate visualizations to mimic the behavior of the actual algorithm. Equally engaging, though only for one algorithm, is the exercise for Dijkstra’s algorithm introduced by Karavirta and Korhonen (2012). That exercise is also similar to the work presented in this paper in that it uses a map to visualize authentic location data.

There are many definitions of context-aware computing and context-aware learning (see, for example, Wang (2004)). However, location is part of the variables of the context in practically all of them. Indeed, location has been widely used in mobile learning to adapt the learning experience. For example, situated simulations (Orkelbog, 2010) use location as one parameter in an augmented reality learning application. Another area where location has been used as part of the context is language learning (Kasaki *et al.,* 2012). Moreover, it has also been used in location-aware learning of natural sciences (Chu *et al.,* 2010).

3. LOCATION-AWARE CONTENT FOR SPATIAL ALGORITHMS

To provide location-aware algorithm visualizations for spatial algorithms, we have implemented a simple JavaScript library that helps creating such AVs. We have used the library to create content for two algorithms: kd-Tree and Point-Region Quad-Tree. For the PR Quad-Tree algorithm, there is a visualization for two modes: *learning* and *assessment* (for kd-Tree, only learning mode is currently supported). In the learning mode, the student can view the algorithm process a set of data. In assessment mode, the student needs to simulate the algorithm.

3.1 Learning Mode and the User Interface

In the learning mode, the student gets visualizations of the selected algorithms. The visualization shows each step of the algorithm processing the random input data. Student can control the visualization by moving steps backward and forward or to the beginning or the end. Each step in the visualization includes an explanation on what the algorithm does at that point. Furthermore, the visualization of the tree data structure used by the algorithms as well as the tree visualized as a map are shown (see Figure 1 for an example).

The interface for the existing visualizations has the same components. Figure 1 shows an example of the kd-Tree visualization. In the top, there is the name of the algorithm as well as text that changes based on the state of the visualization. In the learning mode, it has explanations of the current step of the algorithm. The bottom of the screen contains buttons to control the visualization and a progress bar. The majority of the screen in the middle shows the data structures and the data on a map used by the algorithm.

In both learning and assessment modes, the visualization content checks the location of the student’s device and initializes itself with data points for real locations around the student’s location. The data points are randomized every time the visualization or exercise is loaded, so student can view different examples and try solving the exercise multiple times.

All the visualizations support screens of varying sizes. The content resizes itself based on the available space. Thus, they can be accessed and used on device with any size of a screen. Naturally, there is a lower limit for the screen size, below which the visualizations are not usable.

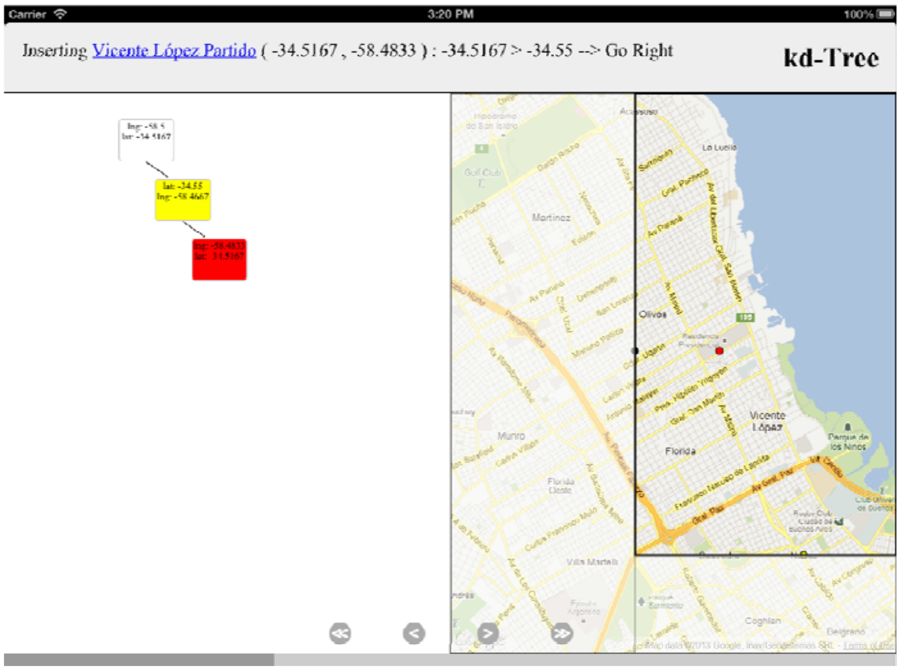
Inserting Vicente López Partido (-34.5167 , -58.4833) : -34.5167 > -34.55 --> Go Right

Figure 1. Example of the kd-Tree visualization in learning mode on the iPad. The red circle on the map is a point that is currently being inserted.

3.2 Assessment Mode

In the assessment mode, student has to interact with the data structure to simulate the algorithm. To explain this interaction, an example of the PR Quad Tree visualization in assessment mode is shown in Figure 2. Student's task is to insert the locations into the data structure following the algorithm in question. In the figure, the student has selected a node in the tree by clicking or tapping it. Selecting an empty node will insert the data into the node. Selecting a node with an existing value (like in the figure) will show the *"Split"* button. Pressing this button will split the node (and the matching area on the map) into four sub-areas. To complete the exercise, a student needs to find the correct positions for all the input data locations and split the nodes as necessary. Although students interact only with the data structure, the map is constantly updated to match the state of the tree structure.

In the bottom of the screen, student sees buttons related to the exercise. Pressing the *model answer* button shows a model solution as a visualization with explanations similar to the learning mode. Assessment mode supports two different modes for feedback: *continuous* and *"on request"*. With continuous feedback, the exercise informs student immediately after each action if it was incorrect. As this can lead to a trial-and-error solving strategy, the "on request" mode only gives a student feedback when requested by pressing the *grade* button. In the on request feedback mode, the exercise cannot be continued after viewing the model solution. Furthermore, the *reset* button initializes the exercise with new random input. Finally, the *undo* button can be used to undo one operation.

The feedback in the "on request" mode is the number of steps the student solution has correct. The grading functionality in both modes is based on comparing the states in the student solution sequence and the model answer. This functionality is implemented in the JSAV algorithm visualization library (http://github.com/vkaravir/JSAV) used by our visualizations.

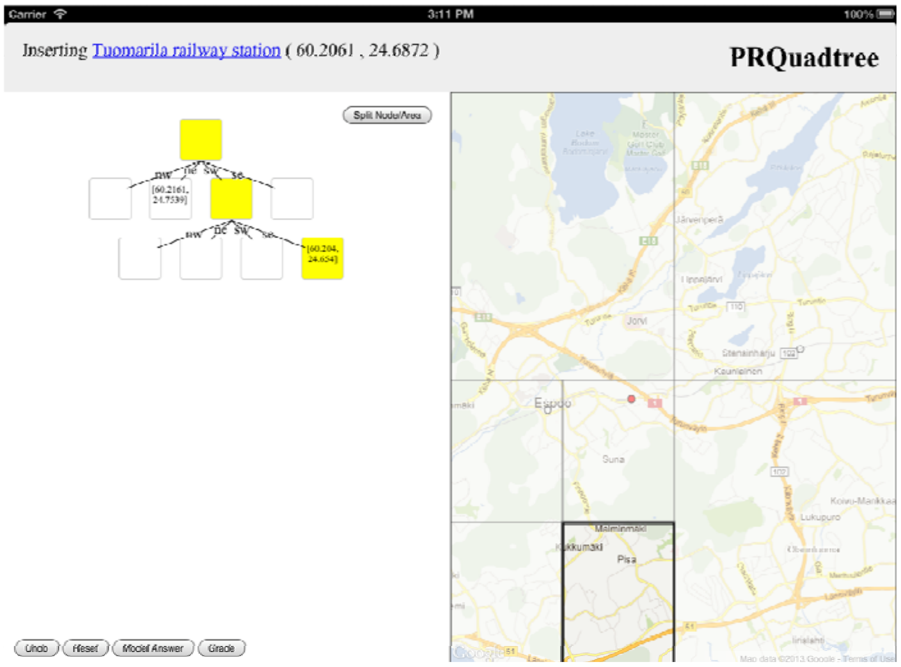

Figure 2. Example of the PR Quad-Tree in assessment mode on the iPad.

3.3 Technology

From the technical point of view, the visualizations are implemented with HTML and JavaScript. They use the W3C Geolocation API to get the users current location. Based on this location, semantic data is fetched from DBPedia, the semantic version of Wikipedia (Auer *et al.*, 2008). After processing the data, visualizations use the JSAV algorithm visualization framework to show the visualization or the exercises. JSAV also grades the student answer in the assessment mode by comparing the student-generated visualization to the model visualization. Showing the data on a map uses Google Maps. Furthermore, we have tested the visualizations in browsers of both iOS and Android 4.0, several desktop browsers, and as a native iOS application using PhoneGap (a framework intended for wrapping HTML and JavaScript as native applications).

4. DISCUSSION AND FUTURE WORK

We believe the new location-aware content presented in this paper is motivational and interesting for students. Alas, we have not used it in teaching with students yet. One reason for this is that we do not feel comfortable in using these as a required part of a course, since some students might not be willing to share their location. However, as previous research has shown, engaging algorithm visualizations can well be educationally effective.

We see the trend of mobile web technology use only increasing in the future with more focus given to interactive learning material that works across devices. While we want to be part of this trend, at the same time, we want to explore ways the new technologies can be used to provide better learning experiences in addition to converting existing material to new devices and platforms. Thus, in this ongoing work, we have taken first steps toward adapting algorithmic learning material to the location of the students as well as using authentic data in the exercises.

In the field of computer science education, new educational systems are often built with modern web technologies like HTML(5) and JavaScript. For example, Guo (2013) presents a visualization tool for the Python programming language that works on desktop and tablet devices. Karavirta *et al.* (2012) introduced MobileParsons, which is a mobile learning application for Python using web technologies. Galles has a large collection of algorithm visualizations online (http://www.cs.usfca.edu/~galles/visualization/), all implemented with JavaScript. However, when it comes to selecting a library for creating new visualization and exercises, there is relatively little choice. The JSAV library we used is practically the only JavaScript library that supports algorithm visualization and automatic assessment of simulations of algorithms.

As the exercises are implemented with HTML and JavaScript, wrapping them as a native application with PhoneGap works well. We see distribution through official application stores like Apple App Store and Google Play as an important step in distributing educational applications. However, we do not have enough content yet to make it an independent application.

In the future, we will focus on creating content for more algorithms. Hopefully, this can be done in collaboration with the OpenDSA project (Shaffer *et al.*, 2011) and other teachers and researchers in the field. Furthermore, a fruitful addition could be to add a measurement of learner knowledge to our content. This would enable us to better contextualize the material, for example, by first offering the learning mode, then the assessment mode with continuous feedback, and finally ask the student to solve the exercise with on request feedback. Finally, only using the content in teaching will show whether the material actually is motivational and effective for students.

REFERENCES

Auer, S., C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, 2008. DBPedia: A Nucleus for a Web of Open Data. *Proceedings of 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference (ISWC+ASWC 2007)*, pp. 722-735.

Chu, H.-C., G.-J. Hwang, C.-C. Tsai, and J. C. Tseng, 2010. A two-tier test approach to developing location-aware mobile learning systems for natural science courses. *Computers & Education,* 55(4):1618 – 1627.

Fisher, J., 2004. Visualizing the connection among convex hull, voronoi diagram and delaunay triangulation. In *37th Midwest Instruction and Computing Symposium.*

Guo, P. J., 2013. Online python tutor: Embeddable web-based program visualization for cs education. In *Proceedings of the 44th SIGCSE Technical Symposium on Computer Science Education.* To Appear.

Hundhausen, C. D., S. A. Douglas, and J. T. Stasko, 2002. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing,* 13(3):259–290.

Karavirta, V., J. Helminen, and P. Ihantola, 2012. A Mobile Learning Application for Parsons Problems with Automatic Feedback. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research,* pages 11–18. ACM.

Karavirta, V. and A. Korhonen, 2012. Visual algorithm simulation exercises with authentic data sets. In *Towards Learning and Instruction in Web 3.0: Advances in Cognitive and Educational Psychology,* pages 123–137. Springer Verlag, New York, NY, USA.

Kasaki, N., S. Kurabayashi, and Y. Kiyoki, 2012. A geo-location context-aware mobile learning system with adaptive correlation computing methods. *Procedia Computer Science,* 10:593–600. The 9th International Conference on Mobile Web Information Systems (MobiWIS).

Malmi, L., V. Karavirta, A. Korhonen, J. Nikander, O. Seppälä, and P. Silvasti, 2004. Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. *Informatics in Education,* 3(2):267–288.

Naps, T. L., G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. Mc- Nally, S. Rodger, and Ángel J. Velázquez-Iturbide, 2003. Exploring the role of visualization and engagement in computer science education. *SIGCSE Bulletin,* 35(2):131–152.

Nikander, J., J. Helminen, and A. Korhonen, 2010. Algorithm visualization system for teaching spatial data algorithms. *Journal of Information Technology Education,* 9:201–225.

Orkelbog, V. F., 2010. Situated simulation as a learning tool – experiencing forum iulium with the iphone. In *Proceedings of the IADIS International Conference Cognition and Exploratory Learning in Digital Age,* pages 268–272, Timisoara, Romania.

Rößling, G. and B. Freisleben, 2002. ANIMAL: A system for supporting multiple roles in algorithm animation. *Journal of Visual Languages and Computing,* 13(3):341–354.

Shaffer, C. A., V. Karavirta, A. Korhonen, and T. L. Naps, 2011. OpenDSA: Beginning a community active-ebook project. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research,* pages 112–117.

Wang, Y.-K., 2004. Context awareness and adaptation in mobile learning. In *Wireless and Mobile Technologies in Education, 2004. Proceedings. The 2nd IEEE International Workshop on,* pages 154 – 158.