8-20-2021

# Digitally Reporting Trail Obstructions in Forest Park

Colton S. Maybee
*Portland State University*

Citation Details
Maybee, Colton S., "Digitally Reporting Trail Obstructions in Forest Park" (2021). *REU Final Reports*. 28.
https://pdxscholar.library.pdx.edu/reu_reports/28

# Digitally Reporting Trail Obstructions in Forest Park

Colton Maybee

August 20, 2021

**Abstract**

The inclusion of technology on the trail can lead to better experiences for everyone involved in the hobby. Hikers can play a more prominent role in the maintenance of the trails by being able to provide better reports of obstructions while directly on the trail. This paper goes into the project of revamping the obstruction report system applied at Forest Park in Portland, Oregon. Most of my contributions to the project focus on mobile app development with some research into path planning algorithms related to the continuations of this project.

## 1 Introduction

Forest Park in Portland, Oregon, is a collection of hiking trails prevalent amongst locals. These trails are being maintained by a group of employees that spend much of their time regularly checking for hiking hazards and obstructions. The more time employees have to spend searching for obstructions, the less time they have to remove them. The solution we have planned can reduce the amount of time taken to check these trails. Our solution includes the replacement of a pen and paper recording system with a digital approach. A system like this has organizational benefits and includes new opportunities for sharing trail status to visitors without asking an employee. Our solution includes three separate projects. One is a web-based interface designed to be an informant of trail hazards and visitor frequency. The second is a mobile app designed to be used on the trail to report obstructions. Third, being a beginning of season trail sweep event optimized with path planning algorithms for the quickest experience possible.

## 2 Methodology

### 2.1 GPX

The standard tie between all three of these projects is the presence of GPX files. These are files that are designed to store large quantities of coordinates with attached elevations, grouped into routes and tracks. These files are the way all of our projects will be getting trail location information.

## 2.2 Mobile App

The goal of the app is to provide visitors a way of seeing the activity of a trail. This can include the last time it was hiked and what kind of obstructions are currently blocking trails. The way that the trails are currently displayed is using the google maps API. This API is usually expensive compared to other interactive map options but is free if the app is free on the app store and does not have any form of monetization.

Each of the trail gpx files is stored in the assets folder of the app. At launch, each gpx file is read. If the title of the trail isn't included in the local database, a new row in the SQLite database is added. The rows name being the trail name and the current date and time as the last time the trail was hiked.

After each trail is read from the asset file, the map starts, and each of those read gpx files is drawn using the polyline function from the google maps API. We supply it with a list of LatLng values from the gpx file and the line's color. Each trail's date is used to calculate how many days ago the trail was hiked, and the amount of days is linked to the trail's drawn color. Currently, if a trial is less than 14 days, the color is green. If the trail is between 14 and 28, it is yellow. Any trail older than 28 days is colored red.

Each trail also has a marker placed at the first coordinate in the gpx file on the map view. This marker is interactable. One tap shows the name of the trail. Another tap on the name of the trail clears the map are redraws the single trail in blue. This is useful if there is a dense section of trails or overlapping trails. A third tap on the title of the trail returns the map to include all of the trails.

The first button in the top left of the app opens the list view. The list is formated using a recycler view. Each entry is a trail title and the time since last hiked. This view can also be sorted alphabetically, numerically, and the inverse of each.

Back to the map view, the second button in the top left opens the obstruction report UI. Here a user can select the type of obstruction from a drop-down and type a description of the obstruction. After all the information is inserted, the user taps the submit button, and a marker is added to the map. This marker is placed at the device's current location and shaped like a caution sign. These obstruction markers can be interacted with just like the trail markers. When tapped, the obstruction type is displayed, and if tapped again, the information of the obstruction is displayed. The displayed information includes the chosen type, the typed description, and the date and time of the report. Temporarily, if the title is taped and held, the information panel includes a delete button that removes the obstruction report and redraws the map.

If we go back to the list view, there is a settings button in the top right. That opens a drop-down that lets the user chose the contents of the list. Trails or Obstructions can be displayed. The same sorting functions apply to both lists.

Included on the map and list pages are refresh buttons. Currently, these buttons reload the app's current open activity.

## 2.3 Web App

The goal of the web app is to inform users of the status of trails. This will be similar to the mobile app, excluding the ability to report trail obstructions. We want the web app to be a place where visitors can check the status of their trails before they go on their trip. The web app can also be the center of where Forest Park employees get the obstruction information. The advantage of the web app over the mobile one is that the web app is hosted from a web browser, which means that users of the web app do not have to download anything to get their trail statuses.

This web app development was focused on by Adan Robles and his paper describes the web app interface with more detail. [1]

## 2.4 Path Planning Event

The goal of the path planning-based event is that many people can do a total search of the trail network for any obstructions created during the winter months of little activity at the beginning of the season. The initial paths of the hikers will be calculated using multi-agent-based path planning algorithms. Integration with the mobile app would also allow the path planning algorithms to recalculate during the event. The hikers' locations and currently explored trails could be used to redirect hikers to different routes based on their pace.

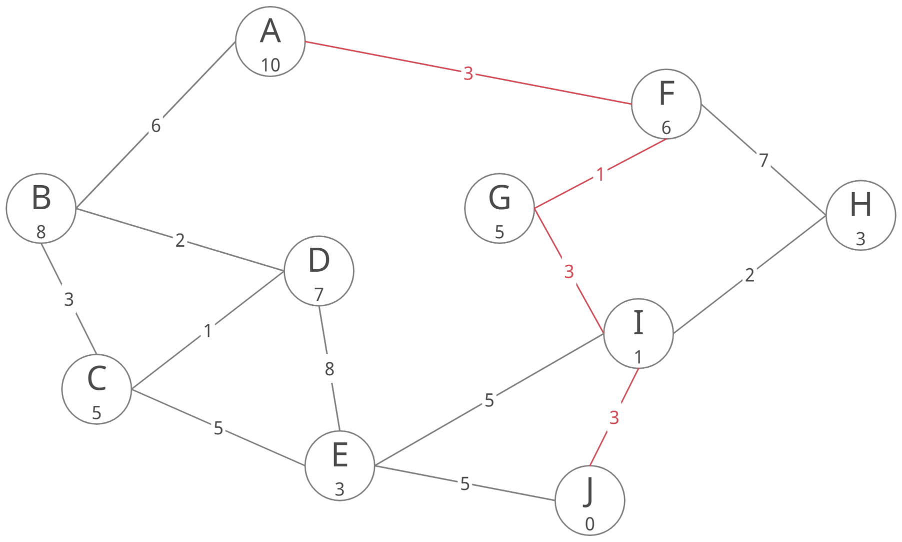
*****functionality of the path planning based event can be described by Erin*****

### 2.4.1 Path Planning Algorithms

When beginning research into path planning algorithms, I started with understanding some of the more basic algorithms first before finding ones that tailor to our desires. There are two main types of algorithms that I studied, sample-based and search-based. A* [2] is the standard search-based algorithm. A* finds the shortest path through a graph by assigning every point in the graph an estimated distance to the goal node. This estimate is usually a straight-line path, not taking into account obstacles or connectivity. Using this extra value, A* can quickly find the shortest distance between two points without evaluating all the possible paths.

Sample-based algorithms work slightly differently. Rapidly-Exploring Random Trees [3] is the standard for these sample-based algorithms. In RRT, a random point is selected on a graph. At the beginning of the algorithm, a set maximum distance variable is assigned. When the random point is selected, another node is created along the line between the random point and the starting node. A node that is either the maximum distance away from the starting node or in the exact location as the randomly chosen point, whichever has the

Figure 1: An example of the A* algorithm.



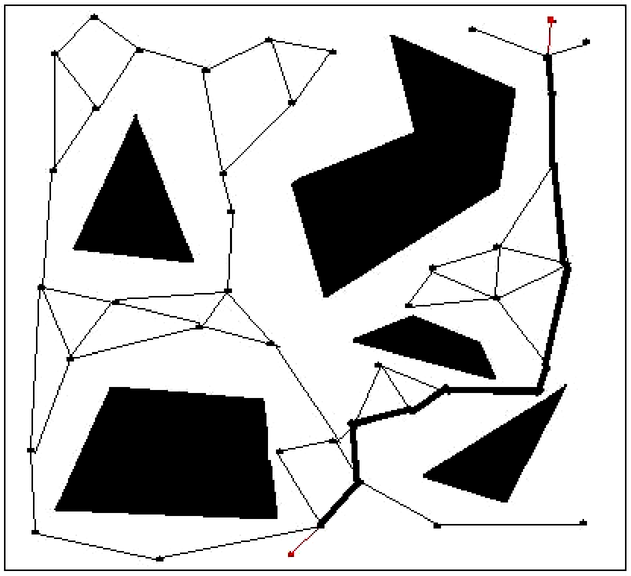
shortest distance. If this line between these nodes is ever intersecting with an impassable obstruction, then the node is not added, and a new random point is chosen. This process continues, with new random points being chosen, each connecting to the nearest existing node. After some time, this algorithm can find an existing path between the start and goal nodes.

With these algorithms come optimizations that improve their efficiency. Some making the algorithm computational faster, and others improving their chance of success. With A* already being so optimized based on its search-based methods, most of the optimizations I found relate to RRTs. RRT* [4] is an optimization to RRT that makes the algorithm much better at finding a shorter path. What makes RRT* different is what node is selected to be connected to the randomly chosen point. If the randomly chosen point is between two existing nodes, then the new node can be connected to both existing nodes, creating a bridge between these two points that is shorter than the previous path. After the path between the start and goal node is complete, randomly chosen nodes continue to be created, pruning off unnecessary bends, and ultimately, getting very close to the shortest path possible.

Another optimization to RRTs that exists is Bidirectional RRTs [5]. This is very similar to normal RRTs but with a slight twist. Instead of the tree ”growing” from the starting node, both the starting and ending node trees are selected to be connected to by the random points. This results in two trees that eventually connect to each other instead of one tree trying to find the single goal node.

There is one more variation of path planning algorithms that is very important to our project. That variation is multi-agent algorithms. MA-RRT* [6]

Figure 2: An example of a RRT algorithm.



is a multi-agent variant of RRT* that has some of its own problems. With the addition of multiple starting and ending nodes come the addition of higher memory costs. This was solved by the inclusion of a pruning system that removes irrelivant branches from the tree to make space for new ones.

Multi-agent variants also exist for search-based algorithms. M* [7] is an adaptation of the A* algorithm that has been adapted for multi-agent pathfinding situations. The initial pathfinding uses A* to find optimal paths for each agent to their respective goal nodes. After optimal paths have been found, all paths are simulated simultaneously. If any node would contain more than one agent at any given time, the agents redirect their path to a different adjacent node to avoid the collision.

After studying the different types of path planning algorithms, we have to make a decision between sample-based and search-based algorithms. Sample-based algorithms were designed to work well in spatial expressed locations with walls and obstacles. Our project is analyzing trails that can be represented very well as a network graph. Each node is a coordinate on the trail system. This is the reason why I believe that sample-based algorithms are not suited well for our situation. Search-based algorithms like A* will be a better tool for the job.

Now that we know that sample-based algorithms are the better option, we can look deeper to find an algorithm that better solves our situation. MS* [8] is a multi-agent, multi-goal adaptation of A*. What makes M* and MS* different is the number of goal nodes and how they work. In M*, the goal nodes are alto the end nodes for each agent. In MS*, the goal nodes are not ending nodes. The goal nodes are more like checkpoints that need to be reached before the nodes can go to their end nodes. No goal nodes are specifically assigned to any particular agent, letting the agents decide what goals are the most optimal for them to collect.

MS* can be used in our path planning-based event to direct multiple people around the Forest Park trail network. One aspect of this project also involved actively updating the path of an agent based on the other agents' locations. I think that this could also be accomplished using MS* by deleting the goal nodes that have been visited and rerunning the algorithm, updating the agents' starting positions to the current locations of the hikers.

## 3 Results / What Is Next

### 3.1 Mobile App

Before we can start actually getting some results from our mobile app, we need to store the obstruction data in a server-hosted database system. At the current moment, the app stores all of the obstruction data locally on the device. This is because of my lack of experience with databases in general.

Even after getting the database situation all figured out, there will still be some additions and changes that need to take place. One of which is a way to make sure that the reports are relevant and a way to make sure that a person actually visited the trail before we update the hiking activity status. Without these restrictions in place, people at their homes can report obstructions and claim they hiked a trail while on the couch.

There are a couple of different solutions we can put into action but currently seem irrelevant without the inclusion of a server. One solution involves a button someone can push that claims that they hiked the trail. Before the change to the database would take effect, the device's location would be compared to the location of the trail. If these two locations are too far from one another, then a popup message can tell the user that they are not close enough to the trail to report it as being hiked. The second approach involves having the user start a hiking event in the app, then every minute or so; the app would store the device's location. After the user ends their hiking event in the app, these stored locations would be compared to the list of trails. The trails with the most similar coordinates would be updated as being hiked. Another function we would like to integrate in the future is the ability to attach photos to the obstruction reports. Currently, there is a placeholder button in the obstruction reporting interface. This would give forest park employees a lot better idea of what equipment to bring when responding to an obstruction report.

More desired features like using the device's accelerometer to automatically detect a user slowing down and climbing over a fallen tree exist. Before any of these features begin to be developed, integration with a server-hosted database needs to happen first.

### 3.2 Web App

*****Adan would be more qualified to write this section*****

### 3.3 Path Planning Event

*****Erin would be more qualified to write this section*****

## 4 Conclusion

The overall focus of this project is the better the hiking experience in Forest Park for visitors and the employees. One of the most significant pitfalls to fall into is self-doubt. Coming into this project with no experience in mobile development was scary, but these projects are a place to learn, not just perform. Skills in programming are beneficial but not always necessary. Sometimes having an inexperienced person take a look at the problem makes the most significant impact. Another set of eyes is a powerful tool, no matter the experience they carry.

## References

[1] A. Robles, "Forest park trail monitoring," 2021.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE transactions on systems science and cybernetics*, vol. 4, no. 2, pp. 100–107, 1968, publisher: IEEE.

[3] S. M. Lavalle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Tech. Rep., 1998.

[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International journal of robotics research.*, vol. 30, no. 7, pp. 846–894, 2011, place: [Cambridge, MA] : Publisher: MIT Press,.

[5] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001, publisher: SAGE Publishing.

[6] J. Jiang and K. Wu, "Cooperative Pathfinding Based on Memory-Efficient Multi-Agent RRT*," *IEEE access : practical innovations, open solutions.*, vol. 8, pp. 168 743–168 750, 2020, place: Piscataway, NJ : Publisher: Institute of Electrical and Electronics Engineers,.

[7] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, Feb. 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0004370214001271

[8] Z. Ren, S. Rathinam, and H. Choset, *MS*: A New Exact Algorithm for Multi-agent Simultaneous Multi-goal Sequencing and Path Finding*, Mar. 2021.