Blake Heard
Meterrific
9/30/19

Routing Algorithms for a Smart Parking Meter Application

Introduction

Routing algorithms are crucial for an application in which consumers expect to find the closest parking spot to a particular destination. In a study from ParkMobile, it was found that the average person spends a total of 17 hours per year and an estimated $345 in costs looking for a parking spot [1]. One solution to this issue is having an application that could inform users of the shortest path to an available parking spot close to a given destination. Numerous routing algorithms can be used to find the shortest path from a destination, and many commercial applications, such as Google Maps, take full advantage of these algorithms. This technical review examines both existing shortest-path routing algorithms and industry standards for path planning in navigation-based applications.

Shortest-Path Algorithms

Shortest path algorithms like Dijkstra's, Bellman-Ford, and D* already exist as standards for path-planning. Dijkstra's algorithm was the first of these implemented in 1959, and it finds the shortest path from a source to all other vertices in a graph with positive edge weights [2]. Since this algorithm will find the shortest path to all vertices in a graph, it can be used to find the shortest path from a source to a *particular* vertex. Dijkstra's algorithm has a runtime efficiency of $O(|E| + |V|^2)$, where E represents the number of edges and V represents the number of vertices in a path network. Bellman-Ford is basically the same algorithm as Dijkstra's except that it also utilizes negative edge weights. These negative weights could be assigned based on conditions like real-time traffic, and the Bellman-Ford algorithm has a slightly better runtime than Dijkstra's [3].

The A* algorithm is another extension of Dijkstra's algorithm and is also a standard when it comes to path-planning. A* also finds the shortest path to a destination based on distance, but it also takes into account a heuristic to further optimize path-planning. For example, Manhattan distance can be used as a heuristic that the A* algorithm will use to find the shortest-path even faster. By using heuristics to optimize path-planning, A* achieves slightly better performance with respect to time over Dijkstra's algorithm [3].

Commercial Applications that Utilize Path-Planning

There are currently several navigation-based apps that utilize path-planning algorithms to find the shortest path from a location to a destination. For example, Google Maps utilizes an "arc-routing" algorithm (in which arcs are streets) to find the shortest path that will traverse given streets until the path stops at a set destination [4]. This algorithm, filed under U.S. Patent 14/157,913, takes in a source and destination and assigns graph weights based on real-time traffic data from an outside source. Once weights are assigned, the algorithm computes the two or three shortest paths and relays this information to the user [5].

Waze, another navigation app, uses a form of path-planning similar to the algorithm used for Google Maps. Waze still utilizes a shortest path algorithm and real-time traffic data to assign weights to graph edges, but Waze also adds routing penalties, like whether or not a road is a "toll road," to calculate an optimal route for the user [6]. The addition of a heuristic like routing penalties makes Waze's algorithm appear to be similar to the A* algorithm and also provides flexibility in determining the shortest path to a destination.

Parking apps have also started to implement a form of path-planning for parking spots. One such application, ParkMobile, utilizes an algorithm that "calculates a general outlook of likely parking availability based on variables such as location and time of day" [1]. The app does not navigate a user to a particular parking spot, but instead, it informs the user of the likelihood of a parking spot being available on a particular street given some sort of heuristic. While this is not the real-time parking solution that Meterrific looks to remedy, general parking availability based on an algorithm similar to ParkMobile's algorithm could be beneficial in informing users of standard parking availability at different times of day.

References

[1] E. Mandel, “Parkmobile adds spot-finding feature for on-street parking in Atlanta,” *bizjournals.com,* March 28, 2018. [Online]. Available: https://www.bizjournals.com/atlanta/news/2018/03/28/parkmobile-adds-feature-for-on-street-parking-in.html. [Accessed Sept. 28 2019]

[2] M. Yan, Class Lecture, Topic: "Dijkstra's algorithm", *Massachusetts Institute of Technology. Regexstr*, 2014.

[3] H. Reddy. “PATH FINDING - Dijkstra’s and A* Algorithms,” *Indiana State University*, [online document], Dec. 13, 2013. Available: http://cs.indstate.edu/hgopireddy/algor.pdf. [Accessed Sept. 28 2019]

[4] Google. “Routing | OR Tools | Google Developer.” *Google LLC.* [Online]. Available: https://developers.google.com/optimization/routing. [Accessed: Sept 27 2019]

[5] R. Geisberger, “Route Planning,” U.S. Patent 14, 157,913, 24 Feb., 2014.

[6] Waze. “Routing Server.” *Google LLC.* [Online]. Available: https://wiki.waze.com/wiki/Routing_server#Routing_algorithm_refinements. [Accessed: Sept 27 2019]