Students Tackle the Routing Problem for In-Traffic Emissions Tests

by Walter Tichy

Editor's Introduction

Vehicle emissions tests used to be done entirely in the laboratory. However, certain car manufacturers cheated on those tests. In response, the European Union introduced emissions tests in real traffic. To make such tests meaningful, they must be performed on routes that meet certain criteria, such as the difference in elevation between start and end points and the proportion of urban and country roads. Finding suitable routes is a complex search problem. Undergraduate students from Karlsruhe Institute of Technology, Germany, developed the first fully automatic solution for finding such routes. In this interview, they share how they did it.

Students Tackle the Routing Problem for In-Traffic Emissions Tests

by Walter Tichy

In 2015, the Volkswagen emissions scandal broke. Independent testers discovered that on the road certain diesel cars emitted more than 40 times the amount of nitrogen oxide (NO_x) permitted by law. However, the cars had all passed the prescribed emissions tests. How was that possible? It turns out the cars were equipped with a special software that sensed whether a vehicle was being tested. Under test conditions, emission controls operated normally. However, when on the street, the emission controls were reduced or turned off entirely, subjecting people around the world to unnecessarily high levels of NO_x . The cars remained under the limit only in the laboratory. In response, the European Union (EU) mandated emissions testing in real-world conditions called Real Driving Emissions (RDE), in addition to laboratory tests. RDE tests work with portable systems for emissions measurement. But there is a problem. RDE tests must be comparable, which means they must be carried out under similar driving conditions, for example with similar acceleration and braking profiles. What good would it be if some cars were tested in cities, others on highways, and others still on downhill courses, all with very different speeds and emissions? The problem is how to find routes satisfying the comparability requirements of RDE tests. A team of five undergraduate students at Karlsruhe Institute of Technology (KIT) in Germany tackled that problem.

Team members Daniel Bösch, Tobias Danner, Patrick Fetzer, Julian Roßkothen, and Alina Valta are all informatics students at KIT. They developed a program that finds comparable routes by adapting a classic search algorithm. The students were in their third semester at the time. Their software shows how powerful algorithms in the hands of relatively inexperienced programmers can produce impressive results.

Why did you choose the RDE route finding problem?

For our bachelor's degree in informatics, we were required to do a practical training course in software engineering (called PSE). This is a three-month team project that is supposed to deliver about 10,000 lines of code plus the attending documentation. Former students working for the auto parts division of P3 Group in Stuttgart, Germany, who had taken part in a PSE project years

ago, suggested the problem of finding RDE routes to Prof. Tichy, who passed it on to us. We thought it was an interesting problem.

Has this problem been solved before?

Currently, there are RDE routes available, but they seem to be created manually. As far as we know, no algorithmic solution has been implemented before.

First things first. How are emissions measured when a car is moving?

Real-driving emissions are measured with the portable emissions measurement system (PEMS), which is a box stowed in the trunk or hooked to the back of a car and connected to the exhaust pipe (see Figure 1). It can record numerous measurements during driving.

Figure 1. Portable Emissions Measurement System, photo by LSDSL; <http://bit.ly/2Z1yHNn>

What are the requirements for testing real driving emissions?

Since emissions vary due to speed, grades, elevation, outside temperature, and stops at traffic lights and such, the EU stipulated requirements for emissions tests to ensure comparable results. The elevation difference between start and end point must be less than 100 meters to prevent tests where the vehicle drives mostly downhill. The elevation must be less than 700 meters (or less than 1,300 meters in extended mode) throughout the drive. The total time of a test must be between 90 and 120 minutes. The most difficult criterion involves the types of road to be used: the cars must be tested on urban roads (0–60 km/h), rural roads (60–90 km/h), and motorways (90–145 km/h) in a certain distribution. Currently, the RDE test requires 29–44 percent urban roads, 23–43 percent rural roads, and 23–43 percent motorways. Our algorithm takes into account all those requirements.

Why couldn't you simply use the classic route-finding algorithm by Dijkstra?

Dijkstra's algorithm calculates the shortest path to a target point. In our case, there is no target point; only a starting point. An RDE route may end anywhere. Also, Dijkstra's algorithm decides which route segment to take next by local decisions at each waypoint. In our case the criterion for urban percentage, etc., can't be evaluated locally, since in the beginning the exact total time is unknown, and hence the percentage can't be calculated exactly. It is also unknown whether a chosen road segment eventually leads to a motorway to complete the motorway percentage.

So you needed a different algorithm. How does it work?

The algorithm uses a priority-driven variant of breadth-first search. All visited paths, even if they are from previous iterations, are kept in a priority queue. In the first iteration we extend our (empty) search tree from the starting point to every directly reachable intersection (from a parking lot, that's typically two). From then on, we always extend the highest priority path, where the priority reflects the quality of that path. So, unlike classical breadth-first search, the tree is expanded unevenly. The algorithm stops as soon as a path (=route) satisfies all predefined requirements.

How is the quality of a route determined?

We define the route quality with factors derived from RDE requirements. These include, amongst others, the overall duration, elevation, and percentages of city, rural and motorway driving. We also include some factors to increase drivability.

For example, to calculate the urban, rural and motorway percentages, we estimate the driving duration of each edge using an estimated average speed. From the estimated duration, we can approximate the time share of each street type.

To prevent switching between different street types too often, we incentivize the algorithm to stay on the same street type. The closer we get to satisfying the percentage of a given street type, the less important it is to stay on it. For example, if we just reached a motorway, we will stay on that motorway until we complete the required percentage and take an exit as soon as we satisfy it.

Where did you get the map data, and how did you handle that?

We used OpenStreetMap for our project, since it is free. Our program has to handle large amounts of data. For example, the map for Germany is more than 50 GB. However, our algorithm seeks routes in a relatively small area, so it is sufficient to load only portions of the map. In addition, OpenStreetMap provides more information than needed by our algorithms such as shops, buildings, or railways. Therefore our program preprocesses the map data into a more compact format by sieving the relevant information and splits the map into dynamically loadable geographic areas.

What special difficulties did you face?

The first version of the algorithm produced routes, which included illegal U-turns like the one in Figure 2. Those occurred because of inaccurately modeled intersections in the map data. We fixed that by banning sharp turns (which would be dangerous anyway).

The algorithm can also fail to deliver a result if it runs out of memory. In that case, one can restart the algorithm, and randomization makes sure different paths are chosen.

Figure 2. U-Turn, Google Maps.

What's the user interface like?

The interface consists of several tabs. In the settings tab (Figure 3) the user can set the route parameters. The program calculates a valid route starting from an address supplied in the routing tab (Figure 4), which is then completed with summary data of the computed route. The route itself is represented as a list of GPS-coordinates and can be saved or exported as GPX, visualized in Google Maps (Figure 5), or downloaded into a navigation system.

RDE Streckenfinder

Settings Routing Vehicle profile

Route parameters:

City (in %): Min: Max:

Rural (in %): Min: Max:

Motorway (in %): Min: Max:

Duration (in min): Min: Max:

Height difference (in m): Max:

Height (in m): Min: Max:

Reset to RDE standard

Change RDE standard

Import map data ...

Open ...

Apply

Apply and save ...

Figure 3. Settings tab.

RDE Streckenfinder

Settings Routing Vehicle profile

Starting point:

No profile selected:

Actual route parameters:

City percentage	38%
Rural percentage	29%
Motorway percentage	33%
Minimal elevation	103m
Maximal elevation	269m
City duration	35min
Rural duration	26min
Motorway duration	30min
Start elevation	125m
Difference start/end	26m

Coordinates:

Latitude: 49.0147239 Longitude: 8.4183819
 Latitude: 49.0141121 Longitude: 8.4183403
 Latitude: 49.0136414 Longitude: 8.4182931
 Latitude: 49.0135536 Longitude: 8.4182829
 Latitude: 49.0134824 Longitude: 8.4182744
 Latitude: 49.0134319 Longitude: 8.4182683
 Latitude: 49.0134126 Longitude: 8.418266
 Latitude: 49.0132505 Longitude: 8.4182466
 Latitude: 49.0123422 Longitude: 8.4181498
 Latitude: 49.0120228 Longitude: 8.4181218

Karlsruhe_10-05-2019_15-03

Save route

Export route

Figure 4. Routing tab with calculated route.

Figure 5. Route displayed in Google Maps.

Looking at the map, I'm struck by the fact that none of the Autobahns (marked with in Figure 5) were used. Is your algorithm buggy?

No, that's not a bug. For RDE tests, motorways are roads with speed limits between 90 and 145 km/h (56 to 90 mph). The German cross-country roads have a speed limit of 100 km/h, so they qualify as motorways.

I understand that the route found by the algorithm may end anywhere, possibly far from the starting point. Could you change the algorithm such that the route returns to the starting point?

The way our algorithm works, this would be quite easy to implement. All we need to do is calculate 50 percent of an actual RDE route and then instruct the user to turn around and drive that same route back to the starting point.

What is still missing, and what are you going to do next?

Currently our algorithm terminates as soon as the first valid route is found. However, it would be better if the algorithm would keep searching for routes that not only fulfill all requirements but are optimal in some way. For instance, one could try to minimize stops in the city or choose routes that are robust despite traffic and inaccurate elevation data. Also, we could optimize our data structure, since breadth-first search takes up a lot of memory.

When will the RDE testing become mandatory in the EU?

RDE will be required for all new cars beginning September 2019, with a conformity factor of 2.1, meaning that the emission requirements may be exceeded by 110 percent. The conformity factor will be reduced over the years.

This was an intense lab course for you. What did you learn about programming in a team?

Because this was our first big software-project, we learnt quite a lot. We learnt how to handle large amounts of data and designing and improving an algorithm. Furthermore, we learnt to organize our team and how important communication and determining the next task for each team member is. Looking back, we put a lot of time and effort into this project. Even so, it was worth it. We received a lot of great feedback from our supervising tutor, Alexander Wachtel, and bonded very well, both as a team and as friends.

How much code did you end up writing?

We wrote about 9,000 lines of code.

What will happen to the code now?

We explored several business models (with your input), but in the end decided to sell the software to our sponsor (P3 Group) for a modest amount. One of our team will continue working on it as a summer job.

You passed that course with flying colors. Congratulations for a terrific project!

About the Author

Walter Tichy has been professor of Computer Science at Karlsruhe Institute of Technology (formerly University Karlsruhe), Germany, since 1986. His major interests are software engineering and parallel computing. You can read more about him at www.ipd.uka.de/Tichy.

DOI: 10.1145/3357506