Smart Fuel Optimisation: An Investigation into the Applications and Usefulness of Smart Fuel Optimisation Algorithms

Diarmuid Ó Muirgheasa

B.A.I. Engineering

Final Year Project  April 2012
Supervisor: Dr. Mike Brady

School of Computer Science and Statistics

O'Reilly Institute, Trinity College, Dublin 2, Ireland
Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

______________________________
Diarmuid Ó Muirgheasa, April 4, 2012
Permission to Lend

I agree that the Library and other agents of the College may lend or copy this thesis upon request.

Diarmuid Ó Muirgheasa, April 4, 2012
# Table of Contents

Acknowledgements ...................................................................................................................................... 7

Abstract .................................................................................................................................................. 8

Chapter I - Introduction .......................................................................................................................... 9
  i. Motivation ........................................................................................................................................ 9
  ii. Objectives ..................................................................................................................................... 9
  iii. Report Outline ............................................................................................................................. 10
    a. In Brief ...................................................................................................................................... 10
    b. In Detail ..................................................................................................................................... 10

Chapter II - Background ......................................................................................................................... 11
  i. Fuel Prices Ireland .......................................................................................................................... 11
  ii. GasBuddy.com .............................................................................................................................. 12
  iii. Original Inspiration ....................................................................................................................... 12

Chapter III - Solution Design .................................................................................................................. 14
  i. User Interface ................................................................................................................................ 14
  ii. Valuing Time ................................................................................................................................. 17
  iii. Algorithm for Filtering List of Candidate Filling Stations .......................................................... 19
  iv. Application of Algorithm in Experimental Procedures ............................................................... 23
  v. Screen-Scraper .............................................................................................................................. 23

Chapter IV - Technology Stack ................................................................................................................ 24
  i. Google Maps ................................................................................................................................. 24
  ii. Ajax (Asynchronous JavaScript and XML) .................................................................................... 24
  iii. C# 25
  iv. External Libraries ......................................................................................................................... 25
    a. Movable Type Latitude/Longitude Script .................................................................................... 25
    b. Underscore ................................................................................................................................. 25
    c. Marker Clusterer ......................................................................................................................... 25
d. Html Agility Pack

Chapter V - Implementation

i. Google Maps API

a. Google Maps API - Limitations
b. Google Maps API – Usage

ii. Screen-Scraper for Harvesting Near Real-Time Prices

Chapter VI - Evaluation of Usefulness

i. Testing Methodology

ii. Presentation of Results

iii. Analysis of Results

iv. Testing GoFuel in Different Deployment Environments

Chapter VII - Further Development

i. Deployment Issues

a. Server-Side Scripting

ii. Algorithm Improvements

a. More Advanced Fuel Efficiency Calculator
b. More Advanced Optimisation
c. Predicting Future Vehicle Usage
d. More Alternative Top-Level Routes
e. Reducing Dependence on Google

iii. Greater Vehicle Integration

iv. Recording User Behaviour

v. Applications in Freight Transport

Chapter VIII - Conclusion

i. Summary of Results

ii. Learning Outcomes

Chapter IX - Bibliography

Chapter X - Appendices

i. Appendix A – Code listing
ii. Appendix B – Experimental Results ................................................................. 50
Acknowledgements

Whoever You Are,

For your time, or patience, or encouragement. Or your inquisitiveness, or your total lack of interest. Or your scathing comments, or kind comments, or encouraging comments, or disparaging comments. Or your looks of awe. Looks of boredom. Or that slight head tilt that says “that really isn’t very good though, is it?”

Or the raised eyebrows that say “hey, maybe you’re not an idiot”. Or the slight nod of “well, it’s better than it was last week”. Or the infinitely reassuring “you’ll get there in the end”. Or the gentle but firm “no, no, don’t do it that way. Let me show you…”.

Each of you

Thanks for, you know – everything.

Oh, and the laughs. Thanks for them especially.
Abstract

Over the past number of years fuel prices have been almost perpetually in the news. Conscious efforts to minimise the cost of motoring have, as a result, been very much on the rise. The phrase “hypermiling” has been coined, to describe the process of extracting the maximum possible range from a tank of fuel, and has extremely quickly entered common usage. Protests have been held against the rise of fuel prices, and car manufacturers have scrambled to produce cars which will provide greater fuel efficiency.

There has also been a certain amount of focus on the price disparity between different fuel stations but, given the cultural and political backdrop, perhaps surprisingly little.

The aim of this project is to investigate whether it is possible to notably reduce the cost of running a car, in both monetary terms and in terms of time taken, by the use of a smart optimisation algorithm which calculates when and where to refuel.

Within this larger question the project also addresses the question of whether this optimisation realises significant enough gains to be worthwhile not just in theory, but also in practice.
Chapter I - Introduction

i. Motivation

Although prices have dropped somewhat since the record highs of 2008, the cost of crude oil (and consequently petrol and diesel) remains extremely high by historical standards [1]. At least in the short term, there is a general consensus that prices are likely to remain at or close to current levels [2]. This has led to widespread concern about the cost of petrol and diesel, and the impact that motoring costs might have on individuals, businesses and the economy as a whole [3] [4] [5].

At the same time, technological literacy continues to grow. Smartphone proliferation in particular continues its upward curve [6], and the average person, certainly in western society, is becoming more and more 'connected' all the time. Unsurprisingly, many consumers are choosing to use the capabilities of this technology to save themselves money in all sorts of situations [7].

In this context, the idea of using technology to attempt to save money on fuel hardly seems an original one, and indeed it isn’t, at least not entirely. Internationally there are a significant number of crowdsourcing fuel price sites, and at least one of those (www.gasbuddy.com, [8]) also provides route optimisation functionality.

The key difference between that system and the system presented with this project, however, is that in this project the fuel station recommended is not simply the cheapest station, but also takes into account the time taken to reach that station, and whether the value of the user’s time negates this advantage or not. The algorithm design is discussed in further detail in later chapters.

ii. Objectives

In terms of deliverables, the primary objective of this project was to build a working system which would allow a user to plan when and where to refuel their car, and which would optimise their route with minimal effort on their part. The optimisation performed was to take into account not only the price of fuel at the pump, but also the amount of fuel used to get to that pump, and the value of the driver’s time used in the process.

Making use of this tool, the secondary objective of the project was to analyse how useful this kind of optimisation is. Is it worth the time and effort involved to save a few cents per litre of fuel? How much time is it worth, how much effort, and what is the trade-off? How easy would it be, under certain circumstances, for a user to cost themselves money by going too far out of
their way to refuel at a slightly cheaper station? Answering these questions was also a key objective of the project.

iii. Report Outline

a. In Brief
This report is divided into eight chapters, with this introduction the first of those. Chapters two, three and four concern the background of the project and the planning and design phases of the project, while the final four chapters address the specifics of the implementation phase, and go on to analyse the outcome in some detail.

b. In Detail
Background discusses the wider context in which the solution presented in this project is being offered. Solution Design discusses the identification of the building blocks required to implement a solution, and also discusses issues around the user interface design in some detail. Furthermore this chapter contains a detailed description of the optimisation algorithm designed for this project. Technology Stack discusses in detail the different elements of the technology stack used.

Implementation contains details on the issues and challenges involved in the implementation of the project. Evaluation of Usefulness is a detailed look at the potential benefits of the system, including a detailed discussion of the simulations performed to evaluate the usefulness of the system.

Further Development highlights some of the elements which featured in the original plan for the project but which proved excessively ambitious given the time constraints. It also describes some other features which suggested themselves during development but which didn't make it as far as implementation, as well as other tweaks and fixes which could be performed in the future. Finally, this section contains a brief ethical discussion regarding data protection, the rights of users, and the responsibilities of those who would operate a system like this.

Conclusion is a recap of the key points in the rest of the report, and a final summary of the key findings and outcomes of the project.

A Bibliography and brief Appendices make up the remainder of the project.
Chapter II - Background

As mentioned in Introduction, cost of fuel has become a major issue in recent times. Although the impact of global warming may be having more of an effect on a national, legislative scale [9], for individual motorists the price of fuel has been a major factor in the increased demand for vehicles which deliver greater fuel efficiency.

In a UK survey, carried out by the RAC and published in March 2012 [3], 27% of 4,000 motorists questioned chose excellent fuel economy as their top priority when searching for a new car. 22% of that same group claimed that if petrol prices were to hit £1.40 per litre they would start searching for a new car, more fuel efficient vehicle.

Against this background sites such as pumps.ie, a crowdsourcing fuel price website, have received significant praise for empowering consumers to make informed purchasing decisions [10].

i. Fuel Prices Ireland

In the Irish context, filling stations are required by law to display their prices prominently, in such a way as to be visible to passing motorists [11]. No such law exists to compel them to display their prices online, however, and few if any filling stations choose do so. It is this gap that consumers have filled by taking part in efforts to crowdsource this data and make it available to the general public.

As mentioned above, Pumps.ie is an Irish fuel price website, which is based on the principle of crowdsourcing. It lists filling stations all around the country, along with the crowd sourced petrol and diesel prices.

While the nature of crowdsourcing means that these prices are often at least a day or two old, in some cases much more, there is still a huge amount of up to the minute information on the site at any given time, and the “freshness” of each individual station's price listing is clearly displayed. As one might expect, more densely populated areas tend to have more consistently up-to-date information. The greater Dublin area in particular seems to enjoy a comprehensive and up-to-date database. By contrast, some more remote locations seem to have little if any activity from contributors.

---

1 As of 03/04/12, prices in the UK have reached peak highs of almost £1.50 per litre in certain areas, with a nationwide average comfortably above that £1.40 threshold [5].
2 The term “crowdsourcing” refers to a process of problem solving which involves outsourcing tasks to a network of third parties. The word is a portmanteau of “crowd” and “outsourcing”, and was coined in a Wired magazine article in 2006 [21].
ii. **GasBuddy.com**

Similar to Pumps.ie, GasBuddy.com provides free access to a database of crowdsourced fuel prices, in its case from stations all over the US and Canada. Above and beyond what Pumps.ie offers, however, GB also offers a trip cost calculator, which plots routes via the cheapest filling stations available near to those routes. What GB doesn't offer, however, is a solution to the dilemma posed by the original inspiration for this project, which is shown in Figure 1 and discussed below.

iii. **Original Inspiration**

The original inspiration for this project was the following webcomic:

![Figure 1 - Mouseover text: And if you drive a typical car more than a mile out of your way for each penny you save on the per-gallon price, it doesn't matter how worthless your time is to you—the gas to get you there and back costs more than you save. Source: xkcd.com/951. Reprinted Creative Commons 2.5](image)

In addition to the idea of optimising refuelling, this comic begged the further question: could the current interest\(^3\) in fuel prices, and in getting the cheapest per litre price for fuel, be counter-productive?

---

\(^3\)And indeed obsession, in some circles
Although Pumps.ie (and other similar resources) certainly empower individuals to make more informed decisions, could it be that they are actually providing information which ultimately hurts consumers, at least in certain circumstances?

In the case of GasBuddy, its route calculator takes only fuel prices into account, not the value of the consumer's time. Could this give rise to the situation mocked in the above comic, where individuals are going dramatically out of their way and spending significant amounts of time to save a paltry few cents?
Chapter III - Solution Design

The core of this project is a series of routing algorithms, built on top of the functionality provided by the Google Maps API with the purpose of choosing an appropriate filling station for a given start and end point. As well as optimising based on a balance between time taken and fuel cost\(^4\), the codebase also implements routing based on much simpler criteria, such as shortest total distance, which provides a baseline for comparing the “optimal” optimisation against in the experimental phase of the project. These experiments, and their results, are discussed in further detail in Implementation and Evaluation of Usefulness.

With the experimental phase addressed in detail in later chapters, this chapter focuses primarily on the GoFuel application, which is the consumer-targeted application of the optimisation algorithm. Included in this analysis is a detailed discussion of the optimisation algorithm used to filter the list of filling stations, as well as a discussion of the problems associated with equating a user’s time with a monetary value.

GoFuel is presented here as a web application, although alternative deployment options are equally feasible. Some of these alternatives are briefly discussed later in the report.

---

\(^4\) Which is the heart of the project, and the basis of the GoFuel application submitted as part of the project.
In the case of GoFuel there were requirements for specifying fuel type and trip urgency, as well as optional advanced settings such as starting fuel level, average fuel efficiency and refuel zone⁵.

In the interests of simplicity the application is presented to users as two separate screens. First the input screen is shown, where the user specifies their route and any other criteria necessary, and then the mapping screen, where their optimised route is displayed.

There are several features of note on the input screen. The “From” and “To” fields implement a feature of the Google Maps API to provide the user with prompts.

![Image of GoFuel application](image)

**Figure 3 - Input prompts, which greatly simplify interacting with the programme**

Once the user begins typing they are provided with suggestions, with the suggestions provided being heavily dependent on the location boundaries specified in the code. In the code submitted with this report the autocomplete is bound to Dublin, but in a release targeted at a wider audience there is no reason that the autocomplete couldn’t be programmed to bind to the user’s current location.

The next line of the interface is exactly what it looks like – the user simply chooses one of Petrol, Diesel or Electricity as their fuel of choice. Petrol and Diesel work in the same way, with the

⁵ “Refuel zone” here refers to the upper and lower bound on fuel levels which specify the “zone” in which the vehicle should be refuelled. The default settings are four and two litres, respectively, and the algorithm will attempt to find filling stations near the point on the route where the user is predicted to have that amount of fuel remaining.
same set of filling stations considered (apart, that is, from the rare situation of a station selling only petrol or only diesel). Electricity works off a different, much more limited set of electrical charging stations⁶.

The next part of the interface after that is the “trip urgency” slider, which is slightly less straightforward. Essentially the slider exists to put a money-value on the user’s time, but without specifically asking the user to put a value on their time. Simply asking the user for a number, although present in earlier version of the design, proved an unnecessarily difficult question for the user to answer (it is addressed in more detail in the next subsection, Valuing Time).

![Map Screen](image)

The buttons marked “Route (desktop version)” and “Route (mobile version)” are intended to simplify the separation between desktop and mobile versions of the website. The idea is that every user would see the same welcome screen, and input their details on the same screen, but that mobile users could then specifically request a simplified map screen⁷.

The “Advanced Settings” button exists to allow users specify settings such as the fuel efficiency of their vehicle, the size and position of their desired refuelling zone, and so on.

---

⁶ The sample datasets used covers most of the filling stations in Dublin, and contain 72 “traditional” (petrol & diesel) stations and 8 electrical charging points.

⁷ This feature has not yet been implemented.
The map screen is a simpler design again. On the right hand side there is a directions pane, while the rest of the screen is occupied by the map. The GoFuel logo is on the top left, and can be used as a "Home" button, bringing users back to the input screen.

The question mark logo ("?") on the top right of the map displays helpful instructions, as well as a legend explaining the meaning of the various symbols on the screen.

Most of the standard Google Maps interface has been stripped away. There may be situations in which users need to be able to click on the screen to pan or zoom, but for most users, most of the time, these represent a waste of screen real-estate, and simply distracted from actually using the programme.

Similarly the street view control has been removed, although this decision was less straightforward, and a final decision on whether or not to include it would have to involve feedback from users.

Finally the layers control has also been removed, with only the “Map/Satellite” control present in their place. Access to satellite images might help a user navigate – access to webcams, traffic information and photographs are unlikely to be so useful, and represent unnecessary clutter.

With regards the other icons, green filling station logos represent filling stations in the database, while the green filling station with a golden background represents the recommended filling station. The blue circles represent groups of filling stations – this functionality is based on an external, non-Google library, and is discussed in more detail in Implementation.

As one might expect, the blue line between “A” and “C” represents the route.

   ii.  Valuing Time

Placing a precise value on an individual’s time is an almost impossible task, particularly when talking about their free time and trying to quantify opportunity costs. Doing so to any degree of accuracy is far beyond the scope of this project, and indeed is research belonging in another field altogether.

For the purposes of this project, the "Urgency" slider in the input screen is set to represent figures between €0/hr and €150/hr, in €5 increments. This very approximately represents a

---

8 Point A and C in the directions pan here represent the start and end points specified by the user, while B represents the recommended filling station.

9 Again, as with any user interface decision, this would have to be finalised in consultation with a representative group of users.

10 Presumably both sociologists and economists would have some claim over this question, as might philosophers, and doubtless a number of other groups as well.
spread of hourly wages, from the unwaged (or those who just don't value their free time) to those on very large incomes.

Although this measure is crude, it has been positively received in informal testing, and provides a perfectly good starting point for any future development on this application. The idea of placing some value on a driver's time is perhaps a more important one than the specific number used. As will be further discussed in Evaluation of Usefulness, the routes recommended for optimised routes turn out to be much the same whatever the driver's time is valued at.
iii. **Algorithm for Filtering List of Candidate Filling Stations**

There are 12 steps in the filtering of filling stations implemented in this project. The images used to illustrate these steps are very much not indicative of the number of points on the polyline of a typical route, which even for trips of just a few kilometres can often run towards or into the hundreds. Although there is of course huge variation, a full 24km refuelling zone is often in the order of 100 points long. *[The steps in italics are performed mostly or entirely using the Google Maps API]*

1. Read in XML files of filling stations\textsuperscript{11} and charging points\textsuperscript{12}. In this Beta version there are two files of filling stations, one sorted by latitude and one by longitude\textsuperscript{13}.
2. Parse these lists using JavaScript code in the user’s browser and convert into arrays of objects.

3. **Geocode\textsuperscript{14}** the start and end addresses.
4. Using the co-ordinates from the geocoding in step 3, use Google Maps API to plot an initial route.

---

\textsuperscript{11} These files are generated by the C\# screen-scraping application, described in more detail on pages 18 and 22.
\textsuperscript{12} The charging points file is an 8-entry XML file, which represents a small sample of the approximately 80 modern charge points currently in operation around the country.
\textsuperscript{13} The justification for this approach is explained in more detail in Implementation.
\textsuperscript{14} Geocoding is the process of converting an address (for example “Trinity College, Dublin”) into co-ordinates.
5. Based on starting parameters for fuel efficiency, fuel level and refuel zone, find points on the map which correspond to the refuel zone$^{15}$.

6. Find the extreme points (north, east, south and west) in the refuelling zone.

$^{15}$ As described earlier in the report, the “refuelling zone” is defined as the portion of the route between where fuel levels first become low and where they become critically low, and represents the area within which we would like to refill.
7. Using the north and south extrema (with a small amount of padding\textsuperscript{16}) as boundaries on the latitude-sorted array of filling stations, and east and west extrema (with that same padding) as boundary points on the longitude-sorted list of filling stations, create two shortlists of filling stations.

8. Find the intersection of these lists. This intersection represents the first draft of the final shortlist.

\textsuperscript{16}Initially the extrema are padded by 0.005°, which in Ireland is approximately 600 meters. If subsequent expansions of the zone are required, this padding is incremented by that same 0.005° each time (up to a maximum of 40 expansions, or 24 kilometres).
9. Find the straight line distance from each station on this shortlist to the route, and
determine if the station is within $2^n$ km of the route, iterating for $n=0, 1, 2...$ Stop

10. If less than 10 stations are found, return to step 7 and increase the padding by 0.005°, up
to a maximum of 0.2°.

11. For the filtered shortlist from step 10, make use of the Google Maps API to calculate a
separate route from start to finish via each of the shortlisted stations.

12. For each of the routes from step 11 calculate total route cost based on the following
formula:

$$
(fuelRequired \times ppl) + (totalTime \times timeValue) + \left(\frac{routeDistance \times ppl}{fuelEfficiency}\right)
$$
Where: ‘fuelRequired’ is the quantity of fuel (in litres) to be purchased\textsuperscript{17}; ‘ppl’ is the price per litre of fuel at the chosen station; ‘totalTime’ is the length of the journey in hours; ‘timeValue’ is the cash-value placed on the user’s time (in €/hr); ‘routeDistance’ is the total length of the route; and ‘fuelEfficiency’ is the efficiency of the vehicle in km/L.

13. Choose the lowest-cost route from the costs calculated in 12.
14. Ensure that the distance from the start to the fuel station is less than the starting range (this is a basic sanity check, and protects against a situation extremely unlikely to arise in practice). If this is not the case, find the next lowest cost option from the list assembled in step 12.

iv. Application of Algorithm in Experimental Procedures
For the experimental part of the project this algorithm is used much as it is here, except that step 12 is extended such that the route is optimised based on a number of different criteria, and the results of each optimisation recorded for comparison.

v. Screen-Scraper
The screen-scraper’s function is to harvest fuel price information from pumps.ie, and to output it to a format which can be used by the web application. It is relatively simple in design: it reads a list of URLs from a text file, where each URL is the address of a page on pumps.ie corresponding to a particular filling station; it then loads that URL and searches through the code for that page until it finds the latitude, longitude, petrol price and diesel price; it stores this information until every page has been parsed in this way; and then finally it sorts the stations into two lists, one by latitude and one by longitude, and writes the details of the stations to two separate XML files. This is discussed in slightly more detail in Implementation.

\textsuperscript{17} This figure is taken as a static 50 litres in this programme. Although this figure could easily be replaced with a dynamic figure representing a maximum fill (and indeed, the programme did, at one point in development, work this way) it was decided that if this figure was to be variable then the programme would need to take into account how much fuel was left at the end of the journey. How this would be valued and compared threw up many more questions, without adding significantly to the value of the analysis, hence the static figure used here.
Chapter IV - Technology Stack

The project made use of a number of technologies, listed below. Some more of the specifics of their use are explored in the next chapter, Implementation.

i. Google Maps

Google provide both JavaScript and Flash APIs for their Google Maps system (mainly for reasons of compatibility, the JavaScript version was used in this project). These APIs expose essentially all the functionality of Google Maps, allowing the developer to choose exactly which parts of the Google Maps system to implement, and providing a solid platform upon which to build further functionality.

Aside from displaying the map itself, some of the key features used in this project included route planning, route display and marker display. Route planning in particular is central to the implementation of the algorithm described in Solution Design. The GDirections object takes as parameters the start point, end point and any waypoints on the route, and then plots that route. GDirections returns the route as a polyline\(^\text{18}\), which allows for fine-grained analysis of the route, step by step.

From the perspective of this project, the biggest limitation of the API is the usage limit of 5 directions queries per second. This, to a large degree, is what necessitates the careful pruning of candidate filling stations described Solution Design. This limitation is enforced by Google, and any queries over this level do not receive a response.

ii. Ajax (Asynchronous JavaScript and XML)

Ajax is a group of interrelated client-side web technologies, generally defined as primarily consisting of XHTML, CSS, DOM, XML, XSLT, XMLHttpRequest and JavaScript. XHTML and CSS are used to define how content is to be presented; DOM (Document Object Model) is used for dynamic display and interaction; XML and XSLT are used for data interchange and manipulation; XMLHttpRequest is used for asynchronous data retrieval; and JavaScript is used to bind all those technologies together.

Ajax is particularly characterised by asynchronous data retrieval, which Google Maps makes heavy use of and which allows for data to be retrieved from the server without requiring a page

\(^{18}\) A polyline (also known as a polygonal chain) is a series of connected line segments used to approximate a curve. In this case the GDirections object receives a list of co-ordinates, which can either be attached directly to the Map object, or else parsed manually (both approaches are used at different points in this project).
The application presented is written primarily in JavaScript (including some JQuery, for the user interface). HTML and CSS are used for the website.

Ajax has been steadily gaining popularity for the past number of years, and is currently used by a number of significant websites, including Flickr, Gmail, Facebook and the Apple online store.

iii. **C#**

C# is a Microsoft-designed object-oriented programming language. It is strongly influenced by C++ and Java, and is (superficially, at least) somewhat similar to those languages.

It is used here as, with the addition of the "Html Agility Pack" library, it provides a simple way of downloading, parsing and processing files from the web.

iv. **External Libraries**

Aside from the core Google Maps services, there are three external libraries used in the JavaScript part of this project, and one in the C# part.

a. **Movable Type Latitude/Longitude Script**

This library [12] provides an implementation of the haversine formula, which is used to calculate "straight line" distances in km, based on input start- and end-points in co-ordinate form. The haversine formula is a special case of the law of haversines, which relates the side and angle of spherical triangles. Strictly speaking, the haversine formula gives great-circle distances.

b. **Underscore**

The underscore library [13] is a "utility-belt library for JavaScript that provides a lot of the functional programming support that you would expect in Prototype.js (or Ruby), but without extending any of the built-in JavaScript objects" [13]. It provides the basis for the method used to find the intersection of the two lists of filling stations.

c. **Marker Clusterer**

Marker Clusterer [14] is a JavaScript library developed within Google, which automatically combines markers into single composite markers as the user zooms. This prevents excessive clustering of markers as the user zooms out, but preserves the fidelity of the data for when the user zooms in.

---

19 It is worth noting that, strictly speaking, the code written specifically for GoFuel would not actually be considered Ajax, as it has no need for and thus makes no use of XMLHttpRequest. It does however make use of most of the other elements of Ajax, and the Google Maps API which is used is considered to fall under the heading Ajax. Of course, this is a semantic question, and of little real importance either way.
d. **Html Agility Pack**

The Html Agility Pack (HAP) [15] is a lightweight HTML parser, which provides a simple and efficient way of parsing HTML files from C#, and extracting information of interest. It is particularly tolerant of malformed HTML, such as is often found “in the wild”.
Chapter V - Implementation

This chapter discusses the more technical details of the design and implementation of this project, focusing in particular on difficulties encountered in the course of the project and on the technical reasons for certain design decisions.

i. Google Maps API

As mentioned previously, Google provide JavaScript and Flash APIs for their Maps service. The JavaScript API was central to this project. Apart from the map display functionality, the API’s routing functions and marker display functions were also used, while the API also allowed the map’s user interface to be dramatically changed from Google’s default layout.

The routing features are used not only for displaying the final route, but also for calculating candidate routes via each of the candidate filling stations. Not only do the routing features allow for distance calculations, however, they also allow reasonably accurate time calculations based on average speeds on the various roads that make up the route.

The markers are used to display information about fuel stations. Their position, naturally enough, indicates the position of the fuel stations, while mouse-over text displays the station’s name, and prices for petrol and diesel. In the case of electrical charging stations, only the name is displayed.

a. Google Maps API - Limitations

One of the main limitations of the API is an artificial one, enforced by Google, namely a restriction to just 5 directions requests per second. In most other applications this limitation would have little impact, but unfortunately that isn’t the case here. Because it is necessary to compare distances to a number of different filling stations, it would have been ideal if it were possible to simply directly compute distances to as many fuel stations as seemed worth considering – because of the asynchronous nature of JavaScript, these calculations could have been performed server-side in parallel. In practice, this wasn’t possible due to this limit. Overcoming it meant placing delays between each call to the Google service, resulting in processing delays of several seconds in some cases.

Another issue with relying on Google technology, which only surfaced very late in the project, was that Google do not guarantee to maintain support for undocumented features. This programme iterates through the points in the polyline returned by Google’s servers, a process which falls under the heading “undocumented”. In the final stages of the project Google slightly

---

20 “Candidate filling stations” here refers to the stations which were shortlisted by the shortlisting algorithm, which is described in some detail in Solution Design.
changed the format of the polyline, resulting in a significant amount of time being spent trying to track down a problem in code which had previously been working perfectly.

b. **Google Maps API – Usage**

There are several steps involved in making use of the Google Maps API. Each application making use of the API needs a Google key, generated by registering with Google. This key allows Google to track and limit requests from a given application, but also allows the developer to track the usage of their application. Google also provides tools to allow the developer to limit the use of their key to certain domains.

Once this key is obtained the JavaScript library may be included in the code, with the key as part of the URI\(^1\). For many of the more common features there is plentiful documentation and example code available. For some less commonly used features there is a complete lack of documentation. More significantly, as mentioned above, these undocumented features are subject to change without notice.

Usage is little different from other JavaScript libraries. As JavaScript is an asynchronous language the calls made to Google’s servers occur in the background. Because of the requests per second limitation mentioned above the code executes based on callbacks from the asynchronous function – when the request to Google’s servers is completed (which typically takes in the order of a few milliseconds) this triggers the callback function, which delays for 200 milliseconds\(^2\).

ii. **Screen-Scraper for Harvesting Near Real-Time Prices**

The screen-scrapers is the core of the real-time pricing functionality. It makes use of an external library, the “Html Agility Pack”, to parse pages from Pumps.ie for pricing information.

The download of these pages is based on a manually assembled text file containing a static list of URLs, with each URL corresponding to a specific station’s page. This has the disadvantage of needing to be manually updated for every filling station that opens or closes, and would ideally be replaced with a dynamically generated list, or else made redundant by closer integration with pumps.ie.

Much like the implementation issues encountered with the Maps API, relying on pumps.ie to maintain a consistent format for their data didn’t seem like, and proved not to be, a reliable

---

\(^1\) This code is available at http://domuirgheasa.com/GoFuel/GoFuel.html, and also on the CD included with this report.

\(^2\) This was modified to 300 milliseconds for the test suite, as Google seemed to enforce the limit more strictly when the programme was making hundreds of requests one after another, as against the dozen or so made with the GoFuel application.
solution. During the course of the project the layout of the data on their page did change, which necessitated a small rewrite of the screen-scraping code.

The screen-scaper generates two lists, one sorted by latitude, one by longitude. It could be argued that this sorting would be more sensibly done by the JavaScript code, which would reduce the amount of data transferred during processing. The sorting was done in C# purely on the basis of it being the path of least resistance, rather than because of any over-riding architectural consideration.
Chapter VI - Evaluation of Usefulness

Due to the extremely fluid nature of fuel prices, it is essentially impossible to produce experimental results which could be described as accurately representative of the application's likely performance in the long run. As prices move up and down, so surely do the variations between filling stations, and thus the ability to consistently find the cheapest stations becomes worth more or less over time. With that caveat in place, however, a snapshot of prices at a given time can give a general indication of how useful the application is likely to be. Most significantly, it also gives a good indication of how price-sensitive a consumer would need to be to derive any benefit from it.

i. Testing Methodology

Simulations were performed for each combination of start and end points for seven points distributed around the greater Dublin area. Those points were as follows:

<table>
<thead>
<tr>
<th>Location</th>
<th>Latitude</th>
<th>Longitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Dublin</td>
<td>53.346862</td>
<td>-6.267014</td>
</tr>
<tr>
<td>Cabra</td>
<td>53.365714</td>
<td>-6.292419</td>
</tr>
<tr>
<td>Ringsend</td>
<td>53.343378</td>
<td>-6.227531</td>
</tr>
<tr>
<td>Ballymount</td>
<td>53.313647</td>
<td>-6.348724</td>
</tr>
<tr>
<td>Ballyboden</td>
<td>53.298671</td>
<td>-6.284523</td>
</tr>
<tr>
<td>Blackrock</td>
<td>53.310365</td>
<td>-6.199379</td>
</tr>
<tr>
<td>Dalkey</td>
<td>53.273837</td>
<td>-6.095352</td>
</tr>
<tr>
<td>Donaghmede</td>
<td>53.398479</td>
<td>-6.15097</td>
</tr>
</tbody>
</table>

Table 1 - Co-ordinates of the start and end points of test routes

This yielded a total of 49 routes for testing. This was not by any means intended to be an exhaustive test, but merely to demonstrate a typical range of urban routes, and the potential savings that could be made in that environment.

For each route, a fuel station was chosen based on each of the following criteria:

- Shortest total distance.
- Lowest price per litre.
- Shortest total time.

23Seven of these routes are "return-to-base" type routes (that is, they start and finish at the same location). The other 42 routes could be seen as 21 routes, with each route being considered from A-B, and from B-A.
- Time-optimised, with time value = €0/hour.
- Time-optimised, with time value = €10/hour.
- Time-optimised, with time value = €25/hour.
- Time-optimised, with time value = €70/hour.
- Time-optimised, with time value = €150/hour.

For each one of these the total time, total distance, price per litre and total fuel price were recorded. These results are included in their raw form on the attached CD.

For simplicity, all analysis was performed using petrol prices only. A vehicle with fuel efficiency of 12km/litre\(^{24}\) is assumed. The statement "Optimised @ €x" should be taken to mean that the results have been optimised with the user’s time valued at €x/hr.

It is worth noting that while every effort was taken to produce representative comparisons here, the actual method most drivers use when determining what route to take can best be described as a sort of pseudo-arbitrary optimisation based on an incomplete set of imperfect data. This is not something which can practically be reproduced in this kind of setting, and so the comparisons made are between simple optimisations (best price per litre, shortest route by distance and shortest route by time) and the more advanced “smart” optimisation being proposed in this report.

Finally, it is important to note that the routes here are exclusively urban routes. This is largely because reliable data for rural areas was simply not available for testing. As noted earlier in the report, pumps.ie provides extremely accurate and timely fuel prices for the greater Dublin area, but its database for rural areas is not nearly so reliable. Tests could have been run against this partly stale dataset, but rising prices [16] would have made the prices at stations without up-to-date information appear artificially low, thus rendering the results meaningless.

\(^{24}\)12km/litre is approximately 30mpg, which represents a relatively typical car.
ii. Presentation of Results

Some of the key findings from these simulations are shown over the next few pages, and discussed in further detail in the next section, Discussion of Key Results.

Figure 8 - The total monetary cost includes the purchase of a full tank of fuel (50L), as well as the cost of fuel used on the chosen route. The cost of routes on the purple line (“Optimised @ €25”), where the user’s time is valued monetarily, is never more expensive than the shortest distance, but always at least as expensive as “Optimised @ €0” (black line). The black line, in turn, is always at least as good as finding the Shortest Distance route or the Lowest Price per Litre route, all of which goes some small part of the way towards proving the effectiveness of the algorithm.

Figure 9 – Ignoring the cost of fuel used on the route results in figures significantly closer together than the previous graph. The station with the lowest price per litre represents the baseline, with both optimisations giving figures consistently better than the simple Shortest Distance optimisation. Much like the previous graph, it is as important to note how close together the various options are as it is to note the difference.
As would be expected, the time taken to reach the station with the lowest price per litre is sometimes impractically high (for the final route the difference between it and the other three figures (which are identical) is almost 30 minutes). The "Optimised @ €25" figure is, as expected, often but not always the same as the Shortest Distance optimisation.

This graph shows the savings on Fuel Purchased. A saving of €2.50 is the best case, while the worst case is a €0 saving. The average is €1.55, the median €1.50, the standard deviation €0.89.
Figure 12 – This graph shows the savings on total trip cost for the “Optimised @ €0” route against the “Shortest Distance” route.

Figure 13 – There is almost an inverse correlation between time spent reaching cheap fuel and the level at which you value the time you spend going there. The average hourly wage here is €12.91, but as can be seen above the standard deviation is extremely high (€18.79), and in practice routes would need to be analysed on a case by case basis.
Figure 14 – This graph is a cost-benefit analysis of “Optimised @ €25” against “Lowest Price per Litre” optimisation. There is definitely some correlation here between the amounts of time saved versus the amount of money cost, but as would be expected the level of this trade-off is far from constant.
Figure 15 – This scatter chart shows the total monetary cost of trips versus the length in minutes of those trips. It demonstrates the optimisation algorithm’s effectiveness, as the optimised points (orange triangles) are generally clustered to the bottom left of the graph, but it also demonstrates the limitations of the system. The optimisations (orange triangles) cannot get any lower than the black squares (lowest price per litre) or any further to the left than the shortest distance (green diamonds).
iii. Analysis of Results

One of the key questions asked at the outset of this project was whether or not a consumer could ultimately cost themselves money by substituting a closer filling station for a lower priced one which was dramatically out of their way. In the case of the 49 routes analysed here, a consumer would save themselves money in 40 cases by going to the lowest priced filling station as against the closest. Of the remaining 9 cases only two involved differences of over €1, and these involved diversions of 27 and 30 minutes and can thus be discounted as unlikely to occur in practice.

Removing outliers, it was calculated that the average value a user was placing on their time by going to the filling station with the lowest price per litre was €12.91 – although, with a standard deviation of €18.79, this figure is a poor guide as to what one might expect in any given situation. Given this, the answer to that original question is that even going quite far out of their way a consumer is unlikely to actually cost themselves money, but by the same token the amount they're going to save is quite minimal, and those savings tend to be quickly eaten up by the value of their time.

Moving on from this the second key result, which can be seen in almost all of the graphs presented on the previous pages, is that there tends to be very little difference in overall cost between different methods of optimisation. Good optimisation will, on average, save users in or around €1 per fill, with a best case scenario being a saving of €2.50 per fill.

A further key result, and one not displayed on the previous pages, is that once a value of at least €10 per hour is placed on a user's time, adding additional value to that time (test were performed for value of €10-, €25-, €70- and €150-per-hour) results in absolutely no difference in the routes suggested. This might well not be the case for longer journeys, but for journeys of the lengths examined here (up to 20km) the results for these optimisations do not differ in so much as a single case.

While it can be debated back and forth what the exact level of benefit from these optimisations is, it is certainly beyond debate that there are real benefits.

---

25 This figure doesn’t take into account the value of the user’s own time.
26 There were a number of outliers where the time difference was so low as to create an artificially high €/hr equivalent of the driver’s time, so these figures were not used in calculations of averages.
27 This very high level of standard deviation is what one would expect from this kind of experiment.
28 The presumption that a consumer is unlikely to cost themselves money does, of course, assume perfect knowledge of the marketplace on their part – a mistaken belief about fuel prices could obviously result in them costing themselves money.
While optimising refuelling for cars may not yield extraordinary benefits, the obvious next step is the investigate optimisation in the road haulage industry. This is discussed in more detail in the next chapter, **Further Development**. In brief, because trucks tend to cover more distance, and because they consume in or around six times as much fuel as a typical car, the potential savings for a truck could be massively more than the potential savings here. On the other hand, the routes examined here are hardly typical of the routes taken by hauliers, so extrapolating results is unlikely to be accurate.
iv. Testing GoFuel in Different Deployment Environments

The results on the previous pages concern the core algorithms of the project. The GoFuel application was also tested, albeit less formally, in a number of different deployment environments.

The development was performed in Chrome, and the application also tested well in recent versions of Firefox and Safari. Internet Explorer would not run the code with its debugging features in place, but would do so without problem with those lines removed. Opera did not successfully execute the code, nor did some older versions of the other browsers. These issues were not investigated any further.

In the mobile environment the biggest problem was with the layout of the pages, which had not been optimised for mobile use. Processing was slow in poor signal areas, but all browsers tested proved capable of displaying the site and performing the required processing.

Significant further optimisation and testing would be required before a general release.

---

29 Internet Explorer does not support the otherwise-standard "Console.log()" function.
Chapter VII - Further Development

Although the code presented is fully functional, it still carries many of the hallmarks of a “version 1.0” product. There are a large number of features which were part of the plan when it was initially drawn up which didn’t make it to the code presented here, and there were also a significant number of improvements which suggested themselves during the course of the project which weren’t implemented, but which are noted here instead.

   i. Deployment Issues

In the code submitted with this report all processing takes place in JavaScript running on the client computer, essentially following the Ajax model of web development. Although this is certainly an adequate solution for desktop PC users, a wider rollout would highlight that Ajax is not necessarily the optimal way of targeting mobile users. In particular, asking a smartphone to perform a heavy JavaScript workload is perhaps not ideal.

In addressing this, deploying as a smartphone app would seem a logical step. On the other hand, in order to deploy on the four largest smartphone operating systems (iOS, Windows Phone, Android and Blackberry) the entire code base would have to be re-written in Objective-C, C++, and Java using both the Android and Blackberry JDE.

   a. Server-Side Scripting

A more palatable alternative would be to re-work much of the JavaScript, client-side processing to run in PHP (or similar) server-side. In the code included with this submission almost all of “GoFuel.js” could be re-written as server-side code. The client would simply have to pass the start and end points to the server, along with any advanced settings, and the server would perform the processing required and pass back the location of the optimal fuel station.

This would of course increase the load placed on the server, so it is a trade-off which would have to be carefully considered, in particular with regard to how well the Ajax implementation could be optimised.

   ii. Algorithm Improvements

While the algorithm presented here was tweaked repeatedly over the course of the project, there are still a number of improvements which could be made, which together have the potential to further improve the accuracy and reliability of recommendations made by GoFuel, as well as to improve the accuracy of any future simulations.
a. **More Advanced Fuel Efficiency Calculator**

The programme as submitted with this report makes use of a static fuel efficiency, set at 12km/L but notionally user-editable in a future release. While this may be acceptably accurate, some users might find it results in less than ideal optimisation, particularly on long routes.

More accurate optimisation could be performed by using two or more figures for fuel efficiency. Manufacturers typically quote efficiency figures for urban driving and motorway driving, and taking this into account could result in significantly more accurate results. Alternatively, real-time measures of fuel efficiency would of course be even more accurate. Methods by which such measures could be made are discussed later in this chapter.

b. **More Advanced Optimisation**

The optimisation algorithm used in this project takes the most important criteria into account, but there are a number of other, less significant things which could be considered in order to truly optimise the process. These include the impact of fuel weight on fuel efficiency (better to fill up later in a trip, all else being equal, to avoid unnecessarily carrying fuel); the impact of weather on fuel efficiency (better cover as much distance as possible before the wind picks up/the snow starts); even the probability of fuel prices going up/down during the course of the journey (certainly hard to predict, but there is a lag between changes in crude oil prices and changes at the pump, so it isn’t impossible by any means).

c. **Predicting Future Vehicle Usage**

For typical users, the idea of tracking their vehicle usage and building usage patterns is an eminently plausible one. An example of this would be the system knowing that a certain user always drives from their home to their office at 8am, Monday to Friday, and figuring these future trips into any optimisation performed.

This kind of prediction would also allow greater application of the fuel price prediction mentioned above: if crude oil prices have dropped dramatically very recently, it might be useful for a user be told that they should put off refuelling for 24 hours, until the recent changes are reflected in forecourt pricing.

Google offer a machine learning service for prediction called the Google Prediction API, which could form the basis of this feature.

---

30 Motorway fuel consumption is typically in the order of 30% lower than urban fuel consumption. Assuming a combined figure (roughly midway between the motorway and urban figures) is used for a motorway trip of 300km, the estimates for where the refuel zone will actually be could be up to 50km off.
d. **More Alternative Top-Level Routes**

The algorithm as it stands takes a single route from the Google API at the very start of the process, and the entire sequence then runs based on the route (that is to say, the algorithm searches for stations within a given distance of that initial route). For longer journeys in particular there may be multiple routes with very similar journey times, but which are geographically quite distinct\(^{31}\). The restrictions placed on the software by Google’s 5-queries-per-second limit meant that exploring several alternative “top-level” routes turned out to be impractical from a system response time point of view, but if that dependence on Google’s APIs could be broken then this is an avenue which could stand to be explored in greater detail.

e. **Reducing Dependence on Google**

As mentioned earlier in the report, relying on Google for APIs does limit the project in certain ways. If a routing algorithm was built and added to the project, it could help dramatically cut the processing time required by GoFuel, by eliminating the 5 request/second limit. The application might also be made more robust by cutting external dependencies.

iii. **Greater Vehicle Integration**

The processing which is performed by this programme requires certain information which, without vehicle integration, can only be approximated, namely fuel level and fuel efficiency. Integrating with a vehicle would allow this information to be obtained in real-time, and thus would provide the application with a significantly better chance of accurately estimating distances/fuel consumption.

Although direct integration with a vehicle’s computer system would be the ideal way of achieving this, it is also possible to do so by reading vehicle data through the industry-standard OBD-II port\(^{32}\).

Reading data through this port is perfectly straight-forward, and was successfully tested as part of this project by use of a Bluetooth OBD-II module. The project did not advance as far as integrating the data read from this port with the optimisation algorithm.

In a future version, designed specifically for in-car use, this integration could be achieved relatively easily in a variety of different ways. The most straightforward method would be to

---

\(^{31}\) Think of navigating from the NW corner of a square to the SE, using only the square edges. Obviously there are two routes which should take the very same length of time, but perhaps there is a greater availability of fuel stations on one or other. The algorithm currently will only look for candidate stations near the route first suggested.

\(^{32}\) The OBD-II port has been in wide use since 1994; mandatory in all cars in the USA since 1996; mandatory for all petrol vehicles in the EU since 2001; and mandatory for all diesel vehicles in the UE since 2004.
redeploy the GoFuel application as a mobile phone app, and make use of the Bluetooth radio on
the phone to connect to the Bluetooth OBD-II module. The route could then be recalculated
every few minutes, based on up-to-date fuel readings, rather than relying on static fuel
efficiencies as is currently the case.

iv. Recording User Behaviour
A significant number of the biggest technology companies in the world operate, either partly or
entirely, in order to collect user data. Google’s profits come almost entirely from their adwords
technology, which targets advertising to individuals based on their searching, while Facebook
has monetised its services by targeting individuals with advertisements tailored to their
particular tastes and interests.

With this in mind, there is obviously a case to be made that a mapping application like this could
be used to target individuals with advertising based on businesses they may be passing. In
particular, filling stations might wish to display information about special offers to individuals
whose routes have been plotted via those filling stations, and this would seem like the obvious
way of monetising an application like this.

Along similar lines, keeping track of routes that people take would allow for analysis of where a
new filling station might best be placed.

All of these potential uses have certain ethical risks, however, which would of course need to be
taken into account. Tracking users’ routes without their permission would obviously be a gross
invasion of privacy, not to mention illegal, while targeting users with advertising could be seen
as an unwanted intrusion. Quite rightly, users are nowadays quite protective of their
information, and even asking users to allow you to track them is a risky proposition in terms of
public image.

On top of the potential user backlash, the legal responsibilities which would fall on a company
tracking information like this are not insignificant, and represent an enormous burden of
responsibility. There would need to be a comprehensive cost-benefit analysis conducted before
tracking like this was seriously considered.

v. Applications in Freight Transport
Although the potential benefits to car drivers of this system are relatively small, the benefits to a
haulage company are potentially much more significant. In that use case drivers do have a direct
cash-value to their time, which makes that part of the equation much more straightforward. In
addition, the fuel consumption of an articulated truck is dramatically higher than that of a car
(usually in the order of 2km/L, as against 12km/L), while the added weight of fuel (typically up to 1,000kg, with both tanks full) will have much more of an impact on a truck than it will on a car. Finally a truck can cover well over 150,000km in the course of a year, which again will multiply the potential savings. [17]

Ignoring the weight issues, the potential car savings (€25-50/annum) can be multiplied by 6 (12km/L as against 2km/L), then again by 8 (24,000km for a car, 192,000km for the truck) to give potential savings in the range of €1,200 - €2,400 per annum.

It should be stressed, however, that these are approximate figures. The driving patterns of articulated trucks have not in any way been examined, and a simple multiplication as just performed cannot possibly hope to accurately represent real-life figures. Moreover, not all filling stations are designed to accommodate articulated trucks, so the selection of places to refuel is likely not even close to the selection available to cars.

Nonetheless, this is the most promising avenue of development for this application, and if even a small fraction of that figure could be realised, this application would have significant value to haulage companies.
Chapter VIII - Conclusion

i. Summary of Results

The first significant conclusion to draw from this project is that a consumer would have to be very price sensitive to draw any benefit from an application like GoFuel. For the vast majority of the population, simply finding the nearest filling station is generally as near to optimum as makes no difference.

On the other hand, in the current social climate there are a great number of people who are, or who claim to be, extremely price sensitive. Last January's 2% VAT increase\(^3\) was heavily criticised in many circles \([18] [19]\) even though its effect on prices appeared at first glance to be negligible. The household charge, a burden of some 27c per day, has been opposed even more vigorously \([20]\). With these figures in mind, it is worth looking again at potential savings from this application.

Although the potential savings on short journeys are modest, for the mid-length journeys (10-20km) studied here savings on optimised versus non-optimised routes were found to average around €1.25 per fill. For a high but plausible annual mileage of 24,000km this would result in savings of around €50 per annum. For some, depending on the trips they are taking, savings could be as high as €2.50 per fill. Assuming the same mileage, this would result in annual savings of around €100\(^3\).

Furthermore, the savings calculated here assume that the baseline is a relatively price-conscious consumer who was already using some sort of simple optimisation, whether that was simply the closest fuel station or the lowest price per litre fuel station or some other similarly simple scheme. It is quite possible that many consumers are operating at a baseline below this, travelling further than necessary due to imperfect information, or indeed imperfect application of whatever information they have. It is perfectly plausible that real-life data might show higher potential savings than this.

For those who would or do consider themselves this price sensitive, GoFuel is a perfectly practical and straightforward solution.

Returning to the start of the report, the question posed in chapter one was how to quantify how much time and money is worth expending for each cent saved off a litre of fuel. The answer is: it depends.

\(^3\) Which in actuality was only a 1.6% price increase.
\(^3\) €100 being, of course, the exact same amount as the highly criticised household charge.
The GoFuel application is capable of optimising based on whatever value an individual might place on their time, but the most important element is not what value they place on their time but rather the idea of placing *some* value on their time. This in many ways is the central point being made in this project: that pursuing cheaper fuel prices regardless of the time or effort it takes is simply nonsensical.

Worth re-iterating here is that the testing presented in this report focussed exclusively on the Dublin region. Data from pumps.ie for rural areas was found to be less reliable, with many filling stations’ prices many days or weeks out of date. This lack of reliable data made testing pointless, as rising prices [16] would make the prices at stations without up-to-date information appear artificially low.

If more reliable data could be obtained for rural areas it is eminently possible that the benefits could be more pronounced than they proved to be in the Dublin region. This is predicted as likely primarily due to the typically longer journeys that would be undertaken in rural areas, as well as the fact that relative isolation means the standard deviation of fuel prices in rural filling stations is noticeably higher than in urban or suburban stations.

**ii. Learning Outcomes**

The bulk of the coding in this project was done in JavaScript, a language with which I had no previous experience at the outset of this project. By the same token I had very little knowledge of any of the other elements of Ajax (HTML, CSS, etc.), and very little experience of any sort of web development or deployment, all of which are things I can now approach with a degree of confidence.

Alongside learning these various new technologies, this project was also my first experience of using Google’s APIs, and my first experience of attempting to choose and apply appropriate optimisation algorithms without an assignment sheet to guide me. Finally, a project like this always provides opportunities for learning about and improving one’s planning and organisational skills, and there is no doubt but that the execution of the project became significantly more polished over the course of the college year.
Chapter IX - Bibliography


struck-3067737.html. [Accessed 1 April 2012].


Chapter X - Appendices

i. Appendix A – Code listing
Full source code for all of the elements of the project is available on the attached CD.

The web application can be accessed at http://domuirgheasa.com/GoFuel/GoFuel.html.

ii. Appendix B – Experimental Results
The raw results for the experimental phase of the project are on the attached CD in .csv file format.