MAST30013 Techniques in Operations Research
Group Project Report

Cameron Chandler 993990, Alexander Westcott 994344, Anand Bharadwaj 745574

May 2020

1 Summary

The objectives of this report include: modelling the cubic relationship between travel cost per unit of traffic and traffic demand, and comparing the performance of various descent methods at minimizing the error in the estimated relationship; modelling the flow of traffic through a simple system such that the minimum travel cost through the network can be found. We found that most algorithms have a certain trade-off between speed and correctness of solutions found, and that this trade-off can be made by altering the stopping tolerance. In conclusion, optimal values for both modelling the cubic relationship and reducing travel cost in the network could be found, and we recommend that future work could involve a deeper sensitivity analysis that what we have conducted, perhaps even involving determination of how much we would be willing to pay for additional links.

2 Introduction

In this report, we analyse the traffic assignment problem and outline the theoretical and computational strategies that we used to tackle this problem. Traffic assignment is a widely studied problem in operations research, with applications in areas as diverse as urban planning, defence logistics, electronics and applied microeconomics. For example, a city council might wish to predict the effect of building a new highway or widening a lane on travel times, or the managing authority of a power network might desire to find the equilibrium current/voltage loads across different wires (Krylatov et al. (2019)).

The traffic assignment problem essentially involves allocating a certain quantity of traffic along different source-to-destination paths in a transportation network, subject to flow conservation constraints. However, as more traffic is allocated to a particular road (arc), congestion effects occur, and we model this by assuming that the travel cost for each road is increasing in the quantity of traffic allocated to that road. The physical interpretation of the travel cost is typically either as a travel time per driver, or in economic terms, a per-driver disutility.

As Wardrop (1952) writes, the traffic assignment problem involves a combination of empirical and theoretical analysis. The empirical aspect of the problem lies in meaningfully modelling the travel cost function. For example, we might collect data for transit times versus the level of traffic on a particular road and using regression, fit a model capturing the relationship between these variables. A range of such *congestion functions* have been documented in the literature, including the well-known quartic volume-delay function used by the Bureau of Public Roads. A study of the empirical validity of such cost functions was conducted by Florian & Nguyen (1976), who tested these functions in the context of traffic data from Winnipeg, Canada. In our study, we pursue a similar approach to those outlined above, fitting a cubic regression to model the travel cost/traffic demand relation.

The second aspect of the traffic assignment problem, namely, predicting the equilibrium traffic allocation of the system given a specific traffic network and cost function for each arc, requires a more theoretical approach. Wardrop (1952) proposed two different equilibria: 1) The user-optimal equilibrium (where traffic is allocated to routes such that there is no incentive for any individual driver to unilaterally deviate to another route; this is essentially a Nash equilibrium) and 2) The system-optimal equilibrium (an allocation where the aggregate travel cost, measured as the sum of the products of per-unit travel cost and traffic demand over all arcs, is minimised). One interesting application of these equilibrium concepts can be found in the microeconomic problem of implementing optimal road tolls to modify user behaviour from the first to the second equilibrium (e.g. Knight (1924), Cole et al. (2006) and Fotakis et al. (2010)). However, in this report, we solely consider the system-optimal equilibrium concept and implement an algorithm for finding such an equilibrium in a specific network.

The remainder of this report is devoted to a discussion of the theoretical and computational strategies we used for tackling the travel cost/demand regression and the system-optimal traffic assignment problems. We set up the former as an unconstrained convex problem, applying a range of standard unconstrained optimisation algorithms (e.g. steepest descent, Newton's method and BFGS) to find the optimum and performed a computational study of these algorithms and compared their performance to that of standard Python libraries. The latter problem, on the other hand, was set up as a constrained optimisation problem; however, to solve the model, we implemented an interior-point algorithm using a log-barrier penalty function, thereby converting the problem into an asymptotically equivalent unconstrained optimisation problem. For comparison, we also implemented the well-known Frank-Wolfe optimisation algorithm (Frank & Wolfe (1956)), investigating the relative performance of these two methods.

3 Methods

3.1 Problem 1 - Q2

If x is the travel demand and y is the travel cost/time per unit of traffic, we will fit a cubic of form $y = \beta_1 x^3 + \beta_2 x^2 + \beta_3 x + \beta_4$. This will create predictions for the travel costs of each of the data point that we already have, and these predictions will have errors. The coefficient of the regression curve can be found by minimising the sum of the squares of all the errors.

We begin by defining several variables. We let

$$\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix}$$

As such, β holds the coefficients of the cubic equation. We let

$$X = \begin{bmatrix} x_1^3 & x_1^2 & x_1^1 & x_1^0 \\ \vdots & \vdots & \vdots & \vdots \\ x_n^3 & x_n^2 & x_n^1 & x_n^0 \end{bmatrix}$$

Where x_i is the ith entry in the x column of the provided data, and n is the number of entries. For the data provided, $n = 100$. We also let

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Where y_i is the ith entry in the y column of the provided data, and n is the number of entries. As noted above, for the data provided, $n = 100$. To minimise the squares of the errors, we first must obtain the square of the errors. We note that a vector of errors can be obtained as $y - X\beta$, and to sum the squares of its entries, we can left multiply this vector by its transpose. We define $f(\beta)$ to be this result, which is the sum of the squared errors for the cubic fitted with coefficients β.

$$f(\beta) := (y - X\beta)^T (y - X\beta)$$

We now seek to simplify this expression.

$$\begin{aligned} f(\beta) &= (y - X\beta)^T (y - X\beta) \\ &= (y^T - \beta^T X^T)(y - X\beta) \\ &= y^T y - y^T X\beta - \beta^T X^T y + \beta^T X^T X\beta \end{aligned}$$

Since $y^T X\beta \in \mathbb{R}^1$, and $(\beta^T X^T y)^T = y^T X\beta$, the two must be equal. So we see

$$f(\beta) = y^T y - 2\beta^T X^T y + \beta^T X^T X\beta$$

We are now in a position to define the minimisation problem that will let us find β. The problem is:

$$\min_{\beta} f(\beta) = y^T y - 2\beta^T X^T y + \beta^T X^T X\beta$$

We now seek to show that the problem is unconstrained and convex. That the problem is unconstrained is immediate from the expression of the problem: there are no constraints on β. That the problem is convex will be shown by demonstrating that $f(\beta)$ is C^2 and that the Hessian of the function is positive semidefinite.

That $f(\beta)$ is C^2 follows from the fact that it is a polynomial of β with degree two. We now calculate the gradient of $f(\beta)$:

$$\nabla f(\beta) = -2X^T y + 2X^T X\beta$$

And using that gradient we calculate the hessian of $f(\beta)$:

$$\nabla^2 f(\beta) = 2X^T X$$

To see that the Hessian of $f(\beta)$ is positive semidefinite, we let $a \in \mathbb{R}^4 \setminus 0$, and determine $a^T(\nabla^2 f(\beta))a$.

$$\begin{aligned} a^T(\nabla^2 f(\beta))a &= a^T X^T X a \\ &= (Xa)^T(Xa) \\ &= \|Xa\|^2 \\ &\geq 0 \end{aligned}$$

Thus the Hessian of $f(\beta)$ is positive semidefinite, and therefore we conclude that $f(\beta)$ is convex. Moreover, we conclude that the minimisation problem defined above is convex. As such, we have shown that the minimisation problem is convex and unconstrained, as required.

3.2 Problem 2 - Q1

To model this traffic assignment problem using constrained optimisation, we can start by observing that our objective function is the total travel cost C, which is given by summing the product of the travel cost on that edge and the traffic demand on that edge over all edges of the graph. Moreover, we require that at each node, the incoming traffic demand is equal to the outgoing traffic demand (traffic flow conservation condition). More formally, we model the problem as follows:

$$\begin{tikzcd}
d & v_3 \arrow[l, "x_1"'] & v_1 \arrow[l, "x_4"'] \arrow[d, "x_6"] & o_1 \arrow[l, "30"'] & 30 \\
 & v_4 \arrow[ul, "x_2"] \arrow[u, "x_3"'] & v_2 \arrow[l, "x_5"'] & o_2 \arrow[l, "40"'] & 40
\end{tikzcd}$$

In this case, the variable (or constant) shown adjoining each edge will represent the traffic demand on that particular road. To be explicit

- x_1 represents the traffic demand for travelling on the edge between v_3 and d
- x_2 represents the traffic demand for travelling on the edge between v_4 and d
- x_3 represents the traffic demand for travelling on the edge between v_4 and v_3
- x_4 represents the traffic demand for travelling on the edge between v_1 and v_3
- x_5 represents the traffic demand for travelling on the edge between v_2 and v_4
- x_6 represents the traffic demand for travelling on the edge between v_1 and v_2

For parsimony, we will not assign variables for the roads from o_1 to v_1 and that from o_2 to v_2, since the traffic demand along these roads is trivially determined from the traffic flow conservation rule. The cost for each segment is given by the units of traffic multiplied by the traffic cost per unit of traffic. Therefore, the objective function is

$$\begin{aligned}C &= 30(\beta_1 \times 30^3 + \beta_2 \times 30^2 + \beta_3 \times 30 + 20) \\ &\quad + 40(\beta_1 \times 40^3 + \beta_2 \times 40^2 + \beta_3 \times 40 + 20) + \sum_{n=1}^{6} x_n(\beta_1 x_n^3 + \beta_2 x_n^2 + \beta_3 x_n + \beta_{4,n})\end{aligned}$$

where $\beta_{4,n}$ is the edge-specific parameter in the demand-cost relation, (Note that $\beta_4 = 20$ for the roads from o_1 and o_2). Using $(\beta_1, \beta_2, \beta_3) = (0.002, -0.01, 0.1)$, we obtain that:

$$C = 0.002\sum_{n=1}^{6} x_n^4 - 0.01\sum_{n=1}^{6} x_n^3 + 0.1\sum_{n=1}^{6} x_n^2 + 50x_1 + 400x_2 + 80x_3 + 900x_4 + 50x_5 + 30x_6 + 7480$$

Now, we can first derive the conservation constraints (forming the equality constraints of the problem):

$$\begin{aligned}x_4 + x_6 - 30 &= 0 \quad \text{(conservation at } v_1) \\ x_6 + 40 - x_5 &= 0 \quad \text{(conservation at } v_2) \\ x_3 + x_4 - x_1 &= 0 \quad \text{(conservation at } v_3) \\ x_2 + x_3 - x_5 &= 0 \quad \text{(conservation at } v_4)\end{aligned}$$

Finally, we must also ensure that the traffic assigned to each road is non-negative for physical feasibility. Thus, we add to our constraint set the constraints $x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$. Some of these constraints may indeed turn out to be redundant, but there is no harm in retaining them.

To summarise, our optimisation problem is the following:

$$\min_x C = 0.002\sum_{n=1}^{6} x_n^4 - 0.01\sum_{n=1}^{6} x_n^3 + 0.1\sum_{n=1}^{6} x_n^2 + 50x_1 + 400x_2 + 80x_3 + 900x_4 + 50x_5 + 30x_6 + 7480$$

$$\begin{aligned}\text{subject to } & -x_i \leq 0,\ i \in \{1, \ldots, 6\} \\ & x_4 + x_6 - 30 = 0 \\ & x_6 + 40 - x_5 = 0 \\ & x_3 + x_4 - x_1 = 0 \\ & x_2 + x_3 - x_5 = 0\end{aligned}$$

3.3 Problem 2 - Q2

Now to derive the KKT conditions. We can write down the Lagrangian as

$$\begin{aligned}L(x, \lambda, \eta) &= 0.002\sum_{n=1}^{6} x_n^4 - 0.01\sum_{n=1}^{6} x_n^3 + 0.1\sum_{n=1}^{6} x_n^2 + 50x_1 + 400x_2 + 80x_3 + 900x_4 + 50x_5 + 30x_6 + 7480 \\ &\quad + \eta_1(x_4 + x_6 - 30) + \eta_2(x_6 + 40 - x_5) + \eta_3(x_3 + x_4 - x_1) + \eta_4(x_2 + x_3 - x_5) \\ &\quad - \lambda_1 x_1 - \lambda_2 x_2 - \lambda_3 x_3 - \lambda_4 x_4 - \lambda_5 x_5 - \lambda_6 x_6\end{aligned}$$

Now, we have the following KKT conditions. KKTa:

$$\begin{aligned}\frac{\partial L}{\partial x_1} = 0 &\implies 0.008x_1^3 - 0.03x_1^2 + 0.2x_1 + 50 - \eta_3 - \lambda_1 = 0 \\ \frac{\partial L}{\partial x_2} = 0 &\implies 0.008x_2^3 - 0.03x_1^2 + 0.2x_2 + 400 + \eta_4 - \lambda_2 = 0 \\ \frac{\partial L}{\partial x_3} = 0 &\implies 0.008x_3^3 - 0.03x_3^2 + 0.2x_3 + 80 + \eta_3 + \eta_4 - \lambda_3 = 0 \\ \frac{\partial L}{\partial x_4} = 0 &\implies 0.008x_4^3 - 0.03x_4^2 + 0.2x_4 + 900 + \eta_1 + \eta_3 - \lambda_4 = 0 \\ \frac{\partial L}{\partial x_5} = 0 &\implies 0.008x_5^3 - 0.03x_5^2 + 0.2x_5 + 50 - \eta_2 - \eta_4 - \lambda_5 = 0 \\ \frac{\partial L}{\partial x_6} = 0 &\implies 0.008x_6^3 - 0.03x_6^2 + 0.2x_6 + 30 + \eta_1 + \eta_2 - \lambda_6 = 0\end{aligned}$$

KKTb: We require $x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$, $\lambda \geq 0$ and $\lambda_1 x_1 = \lambda_2 x_2 = \lambda_3 x_3 = \lambda_4 x_4 = \lambda_5 x_5 = \lambda_6 x_6 = 0$
KKTc: $x_4 + x_6 - 30 = 0$, $x_6 + 40 - x_5 = 0$, $x_3 + x_4 - x_1 = 0$, and $x_2 + x_3 - x_5 = 0$.

3.4 Theory/Algorithms

To solve the model in Problem 2, we chose to convert the problem into an unconstrained optimisation problem by minimising the corresponding log-barrier penalty function with penalty parameter $\alpha_k = k$ and taking the penalty parameter to ∞. We can write down this function:

$$\begin{aligned}P_{\alpha_k}(x) =& 0.002\sum_{n=1}^{6} x_n^4 - 0.01\sum_{n=1}^{6} x_n^3 + 0.1\sum_{n=1}^{6} x_n^2 + 50x_1 + 400x_2 + 80x_3 + 900x_4 + 50x_5 + 30x_6 + 7480\\ &- \frac{1}{k}\sum_{n=1}^{6}\log(x_n) + \frac{k}{2}\left[(x_4 + x_6 - 30)^2 + (x_6 + 40 - x_5)^2 + (x_3 + x_4 - x_1)^2 + (x_2 + x_3 - x_5)^2\right]\end{aligned}$$

For this function, the gradient is

$$\nabla P_{\alpha_k}(x) = \begin{bmatrix} 0.008x_1^3 - 0.03x_1^2 + 0.2x_1 + 50 - \frac{1}{kx_1} - k(x_3 + x_4 - x_1) \\ 0.008x_2^3 - 0.03x_2^2 + 0.2x_2 + 400 - \frac{1}{kx_2} + k(x_2 + x_3 - x_5) \\ 0.008x_3^3 - 0.03x_3^2 + 0.2x_3 + 80 - \frac{1}{kx_3} + k(2x_3 + x_2 + x_4 - x_1 - x_5) \\ 0.008x_4^3 - 0.03x_4^2 + 0.2x_4 + 900 - \frac{1}{kx_4} + k(2x_4 - x_1 + x_3 + x_6 - 30) \\ 0.008x_5^3 - 0.03x_5^2 + 0.2x_5 + 50 - \frac{1}{kx_5} - k(x_2 + x_3 - 2x_5 + x_6 + 40) \\ 0.008x_6^3 - 0.03x_6^2 + 0.2x_6 + 30 - \frac{1}{kx_6} + k(2x_6 + x_4 - x_5 + 10) \end{bmatrix}$$

Thus, letting $f(x_i) = 0.024x_i^2 - 0.06x_i + 0.2 + \frac{1}{kx_i^2}$, we obtain that the Hessian is:

$$\nabla^2 P_{\alpha_k}(x) = \begin{bmatrix} f(x_1)+k & 0 & -k & -k & 0 & 0 \\ 0 & f(x_2)+k & k & 0 & -k & 0 \\ -k & k & f(x_3)+2k & k & -k & 0 \\ -k & 0 & k & f(x_4)+2k & 0 & k \\ 0 & -k & -k & 0 & f(x_5)+2k & -k \\ 0 & 0 & 0 & k & -k & f(x_6)+2k \end{bmatrix}$$

The problem has thus been converted into a form amenable to the application of a standard unconstrained optimisation algorithm (e.g. steepest descent, Newton's method or BFGS).

In pseudocode, the algorithm can be described as follows:

```
$\mu \leftarrow 2$ (this can be any constant greater than 1)
$n \leftarrow 1$
$EPS \leftarrow 10^-5$
**while** $|x_n - x_{n-1}| < EPS$ **do**
    $x_n \leftarrow UnconstrainedOptim(P_k(x))$
    $n \leftarrow n + 1$
    $k \leftarrow \mu k$
**end while**
```

As a final theoretical point relating to Problem 2, we can show that we are in fact dealing with a convex program. Indeed, first observe that all the inequality constraints are linear (hence convex); and moreover, the equality constraints are all affine. Next, letting $h(x_i) = 0.024x_i^2 - 0.06x_i + 0.2$, we find that the Hessian of C is:

$$\nabla^2 C(x) = \begin{bmatrix} h(x_1) & 0 & 0 & 0 & 0 & 0 \\ 0 & h(x_2) & 0 & 0 & 0 & 0 \\ 0 & 0 & h(x_3) & 0 & 0 & 0 \\ 0 & 0 & 0 & h(x_4) & 0 & 0 \\ 0 & 0 & 0 & 0 & h(x_5) & 0 \\ 0 & 0 & 0 & 0 & 0 & h(x_6) \end{bmatrix}$$

But writing $h(x_i) = 0.024(x - 1.25)^2 + 0.1625$, we find that $h(x_i)$ is always positive, meaning that the Hessian of C is positive-definite (the eigenvalues are simply given by the main diagonal elements in this case). Putting all this together, we have shown that we have a convex program. Since all the constraints are affine, a constraint qualification is satisfied everywhere, meaning that the KKT conditions are satisfied at a point if and only if it is a global minimiser for the objective function. Thus, if our sequence of minimisers for the penalty function indeed converges to a cluster point, this cluster point will be feasible and stationary for the NLP and moreover a global minimiser.

Since the problem is indeed convex, this led us to consider an alternative computational strategy for solving the optimisation problem, namely, the Frank-Wolfe algorithm (Frank and Wolfe, 1956). An iterative, first-order optimisation procedure, the Frank-Wolfe algorithm essentially minimises a first-order Taylor approximation of the objective function around the current candidate point and moves iteratively in the direction of this local minimiser, albeit at a progressively diminishing rate. Formally, the pseudocode of the algorithm is the following:

```
$k \leftarrow 0$
$x_k \leftarrow$ arbitrary feasible point
while $k \leq k_{max}$ do
    $s_k \leftarrow argmin_{s_k} s_k^T \nabla C(x_k)$
    $\gamma \leftarrow \frac{2}{k+2}$
    $x_{k+1} \leftarrow x_k + \gamma(s_k - x_k)$
    $k \leftarrow k + 1$
end while
```

4 Experimental Results

This section contains our solutions to problem 1 and problem 2 (4.1, 4.2), as well as a computational study of the performance of different algorithms and parameters for solving problem 1 (4.3). Finally, we look at some extensions of problem 2 in doing empirical sensitivity analysis, and the conclusions we can draw from that (4.5).

4.1 Solution to Problem 1

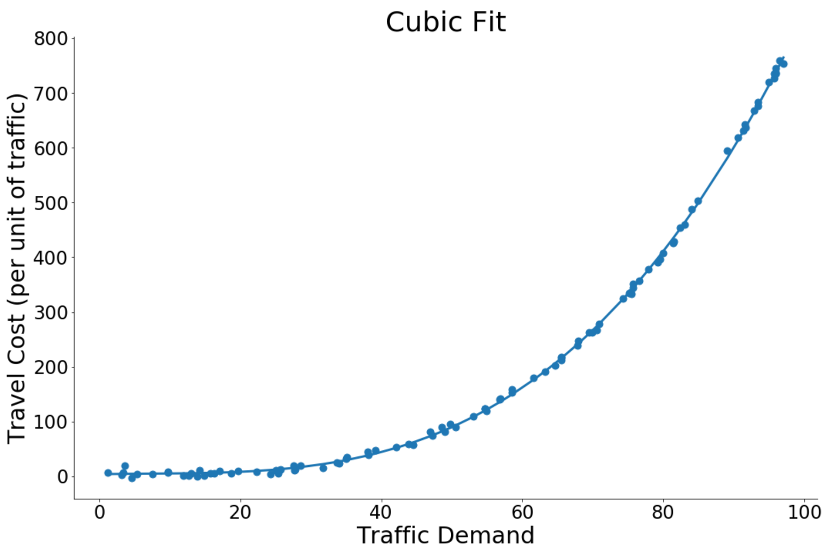
The demand cost relation can be interpreted as a measure of the cost of travel along a particular route by a unit of traffic, potentially with a unit of time. That the function fitted increases with increasing demand indicates that the cost of taking a route for a given unit of traffic increases as the number of units of traffic increases, meaning that total cost can increase quickly with increasing demand. It is therefore likely that traffic will seek to use different routes relatively evenly, with variation based upon varied β_4 figures in the later parts of the problem.

The least squares cubic fitted for the data provided is as follows:

$$y = 0.00104x^3 - 0.0229x^2 + 0.262x + 3.94$$

With that curve plotted against the original data appearing as follows:

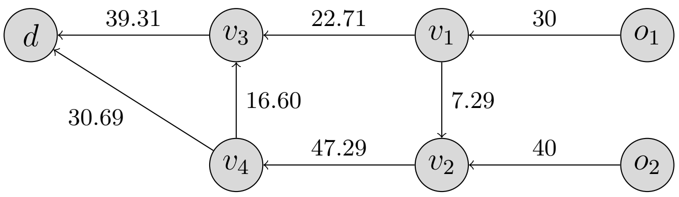
Figure 1: Least Squares Cubic Plotted With Original Data



It is worth noting here that whilst we used a descent algorithm to obtain the solution and explored other optimisation algorithms, in general we are able to find a closed form solution to fitting a least squares polynomial to a data set. However, that is not the focus here, so we will not discuss it further.

4.2 Solution to Problem 2

The solution to problem 2, being the traffic demand along eagch arc of the network that minimises the total travel cost, is presented below. The minimum travel cost is approximately 61750.



An interesting point to note is that this arc-based assignment does not pin down a unique equilibrium in terms of the paths that drivers choose to use. While in this specific network, the volume of traffic assigned to the path o_1, v_1, v_3, d is uniquely determined as 22.71, we have a family of such route assignments over the paths o_1, v_1, v_2, v_4, d and o_2, v_2, v_4, d that all yield the same traffic volume over the relevant arcs. Intuitively, this is because the flow of traffic is 'memoryless', in that, once some traffic reaches a vertex such as v_2, it is inconsequential whether that traffic originated from o_1 or o_2 (at least for the purpose of optimising our objective function). To this end, when the outgoing traffic from v_2 is routed either directly to d or via v_3, the assignment of routes in terms of the origin point of this traffic is arbitrary.

4.3 Algorithm Experiments

A computational study was conducted on not only the implementation of Newton's Method and the BFGS algorithm, but also on three additional descent methods from the Python SciPy library (SciPyCommunity (2019)).

These three methods are the Nelder-Mead method, Newton-CG (conjugate gradient), and Trust Region Newton-CG. Nelder-Mead maintains a simplex − a special type of polytope − with $n + 1$ vertices in $\mathbb{R}^n$, and moves the worst performing vertex at each iteration. It only requires function evaluations. Newton-CG functions as Newton's method does but approximates the inversion of the Hessian, and so is less computationally expensive. Trust region methods find a minimum within a given radius using an approximation to the objective function, such as a second order Taylor series approximation. In this case we use Newton-CG to solve this subproblem, before moving the center of the region to the newly found point. The radius adjusts as the algorithm continues.

The parameters that were changed in the experiment are:

- A step size constant (T) used in Newton's method and BFGS to determine the width of an interval to search
- the tolerance for each algorithm's stopping criterion

Trialled values for T ranged from 10^{-2} to 10^{2} inclusive, and values for the tolerance ranged from 10^{-3} to 10^{6} inclusive. These parameters were used to observe not only how each would effect the time taken for the algorithms to run, but also the value of their minimum. This is to evaluate whether the fastest evaluations are truly the most desirable, or if we need to compromise to find the point we are truly after.

For each algorithm and combination of parameters, 100 points (x) were chosen in $\mathbb{R}^4$ such that $x_i \in [-5, 5]$. The algorithms would then minimise the non linear program described in problem 1. In each run, the time taken, function value and number of iterations are recorded to measure their performance.

BFGS and Newton's method were implemented in Python 3.8, making use of NumPy datatypes for precision. The three additional algorithms came from the python SciPy library, and so all experiments were run using Python. The machine used for testing ran Linux and had an 8th Gen Intel I7 processor.

Various visualisations of results and some comments follow, preceding a more comprehensive conclusion. Note that the shaded regions around each line represent 95% confidence intervals that arise from the random sampling of initial points.

Figure 2: Performance of Newton's Method against Tolerance. The algorithm found the global minimum regardless of the tolerance

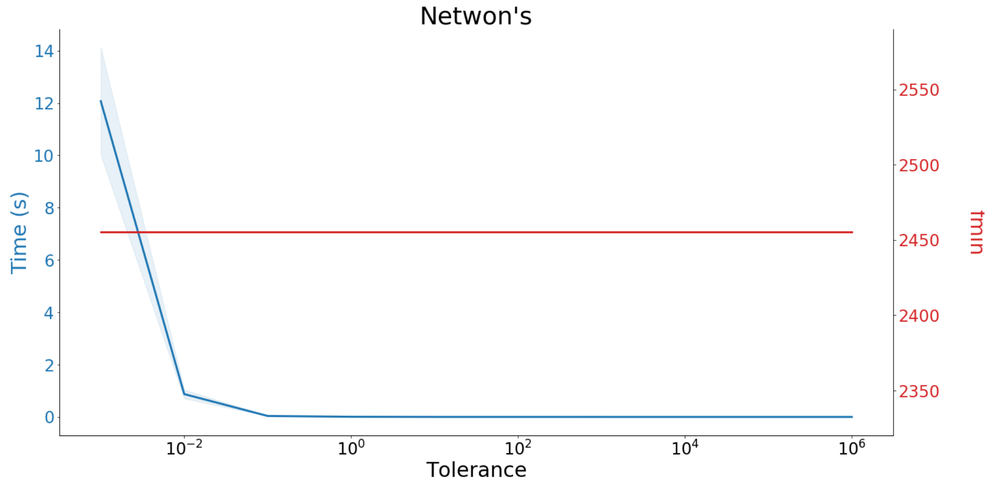


Figure 3: Performance of BFGS against Tolerance

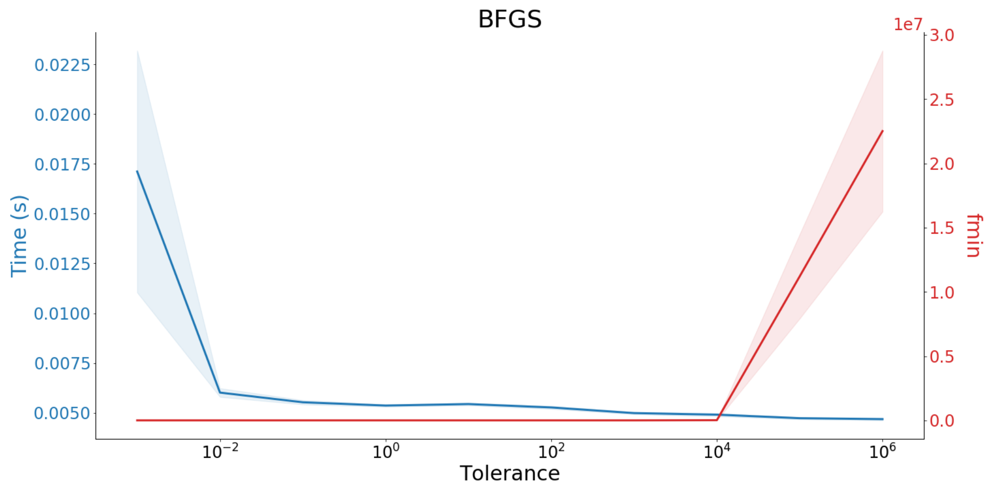


Figure 4: Performance of Nelder-Mead against Tolerance. It is clear that the algorithm's performance is invariant to the stopping tolerance

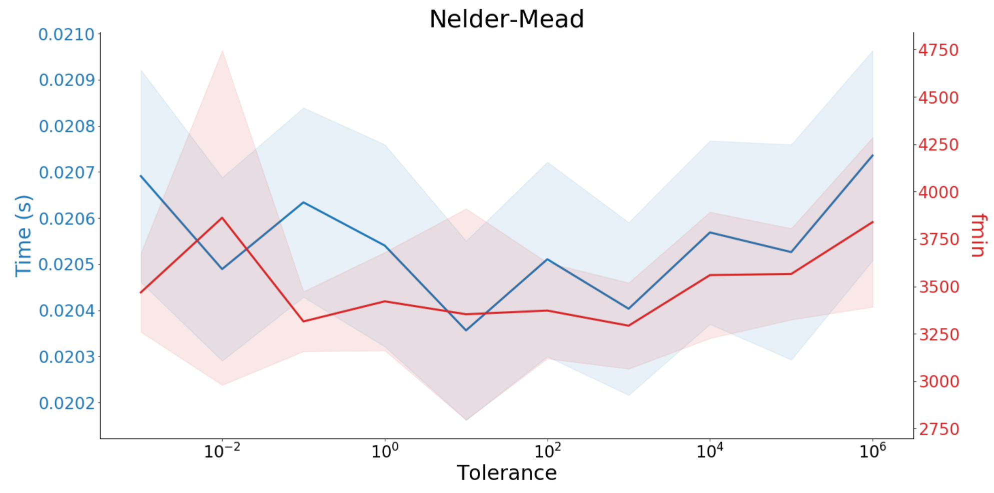


Figure 5: Performance of Newton's CG against Tolerance

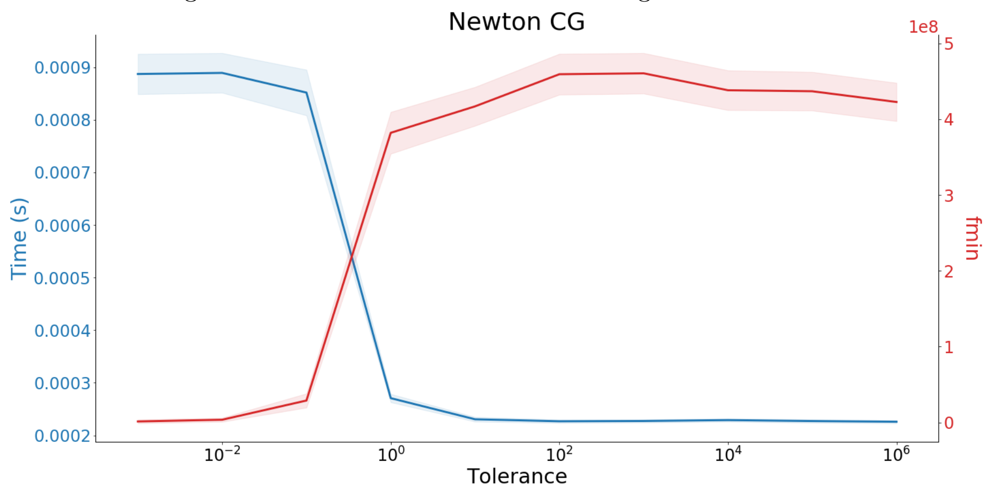


Figure 6: Performance of Trust NCG against Tolerance

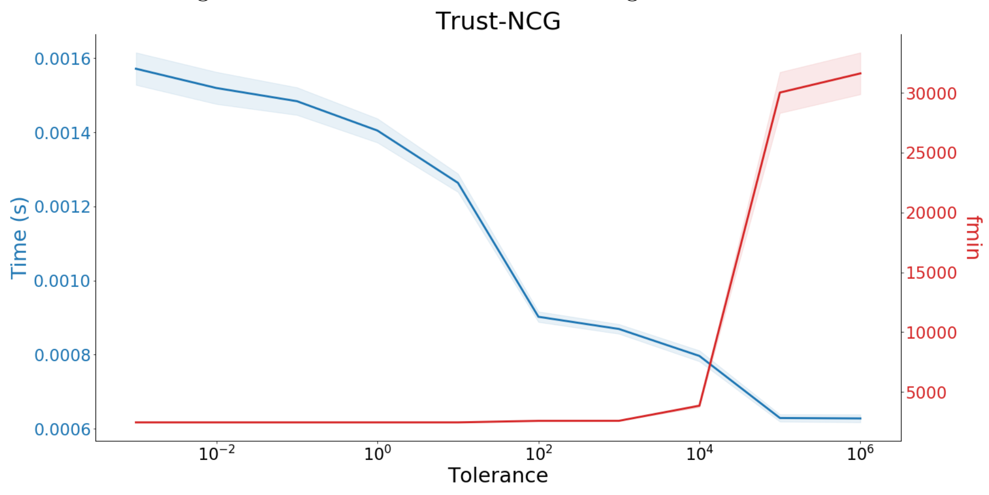


Figure 7: Performance of BFGS against Tolerance and T

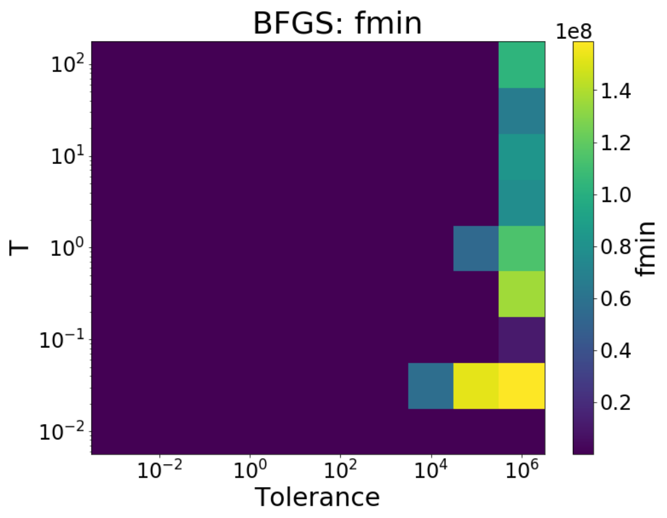


Figure 8: Performance of BFGS against Tolerance and T

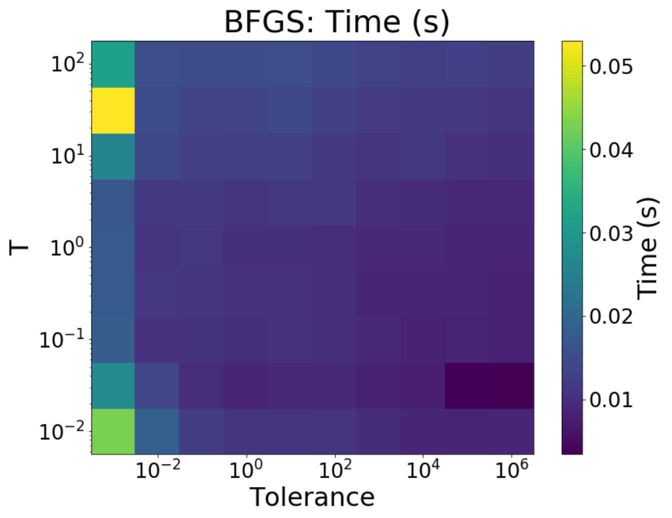
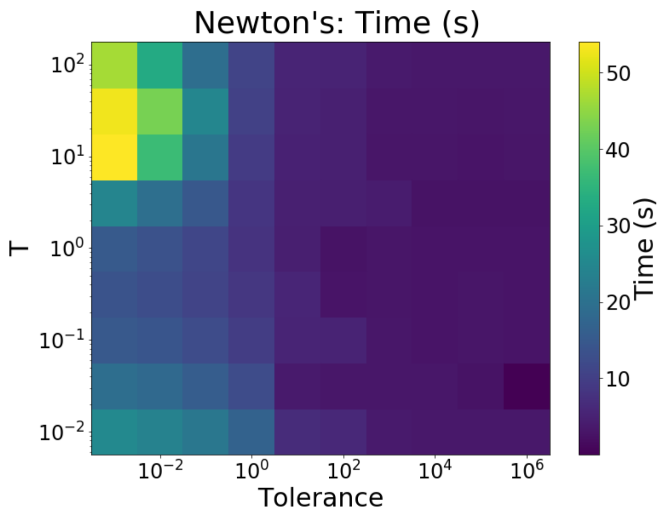


Figure 9: Performance of Newton's Method against Tolerance and T



4.3.1 Computational Study Conclusions

The main conclusions from this study are summarised below.

- Newton's Method found the global minimum regardless of the tolerance, however, it was slower for smaller tolerances – very significantly in some cases.
- The performance of Nelder-Mead is invariant to the tolerance parameter.
- All other algorithms had performances dependent on tolerance. High tolerances stopped running very quickly, but they were far from the global minimum, whilst low tolerances took a long time to run, but terminated very near to the global minimum.
- Changing T had no significant effect on how far each algorithm terminated from the global minimum.

Table 1: Summary of Computational Study for a fixed T=1, tolerance=10^{-3}

Algorithm	**Av. fmin**	**Av. Iterations**	**Av. Time (s)**
Newton-CG	10729.77	4.13	0.00084
Trust-NCG	2455.19	10.39	0.00151
BFGS	2455.19	13.03	0.00749
Newton's Method	2455.19	28.06	0.01640
Nelder-Mead	3235.94	441.18	0.02077

4.4 Discussion of computational strategies for Problem 2

To solve the system-optimal traffic assignment problem, we pursued two strategies: iterative minimisation of the log-barrier penalty function using the BFGS algorithm for increasingly larger values of the penalty parameter k, as well as the Frank-Wolfe algorithm. The following figures show the trace plots of x_{min} for the different algorithms.

Figure 10: Convergence of x_{min} across BFGS iterations

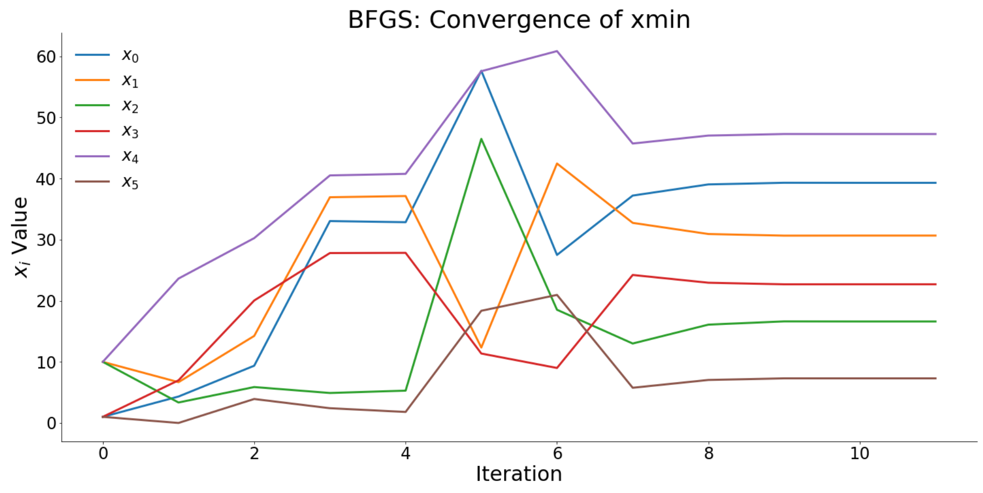


Figure 11: Limiting values of x_{min} versus the penalty parameter K (small values of K)

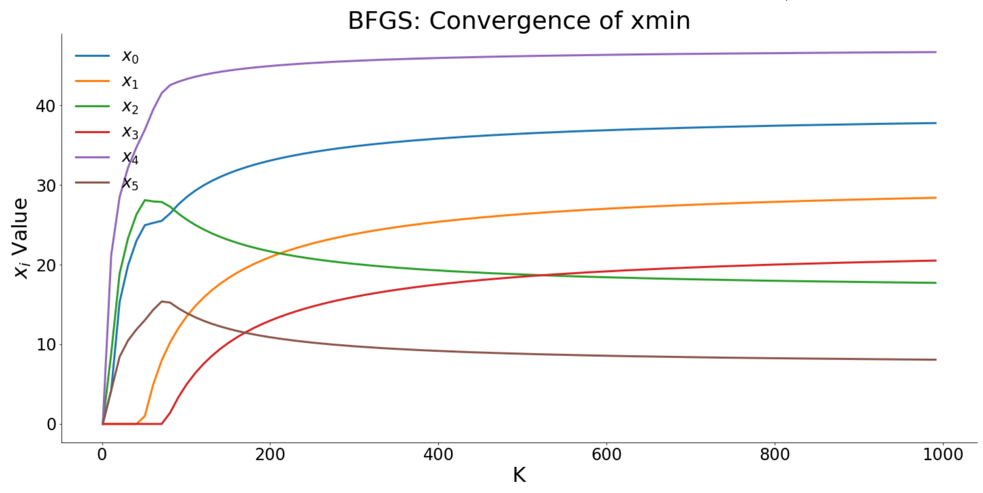


Figure 12: Limiting values of x_{min} versus the penalty parameter K (log scale)

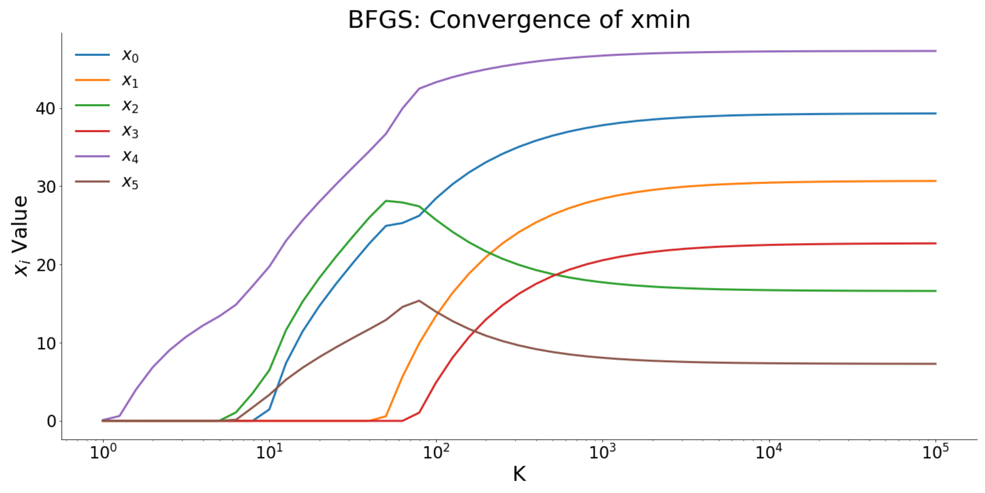
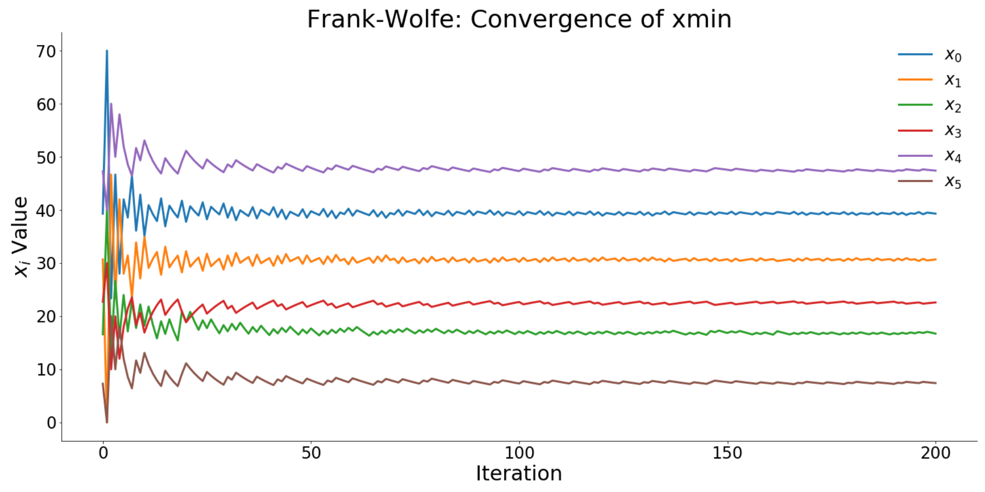


Figure 13: Convergence of x_{min} across iterations in the Frank-Wolfe algorithm



Note that the first graph shows the convergence of x_min for a fixed value of K, while the second and third graphs show the convergence for different values of K. Our findings are the following. We find that both BFGS and the Frank-Wolfe algorithm exhibit convergence to the aforementioned minimiser in Section 4.2. Graphically, we can see that the convergence of BFGS for fixed k is faster than that for Frank-Wolfe (which is somewhat noisy); this is as expected, since BFGS is known to exhibit superlinear convergence, while the convergence of Frank-Wolfe is sublinear. However, the tradeoff is that the penalty parameter needs to be quite large before convergence has occurred, meaning that a sizeable number of 'outer iterations' are required for the penalty function-based algorithm to perform effectively. We leave a detailed comparative analysis of the runtimes of the two algorithms for future work.

4.5 Problem 2 Empirical Sensitivity Analysis

With a model for the traffic network implemented we are able to tweak some parameters, such as the initial demand, to see the effect on the total cost of the network. This could have practical applications in making decisions on where to zone more housing, or where to provide additional public transport links to relieve traffic.

We made changes to demand levels of the two source nodes and recorded the change in total cost, as tabulated below.

Source node	Change in demand	Change to total cost
o_1	+1	+1717
o_2	+1	+1970
o_1	+10	+19526
o_2	+10	+22869
o_1	-1	-1674
o_2	-1	-1910
o_1	-10	-15140
o_2	-10	-16786

We can see that the network is more sensitive to changes in o_2 than o_1, so o_1 is more suited to take more traffic if there is a choice, and if traffic can be removed it should be removed from o_2. We also note that the rate of reduction of cost appears to decrease with increasing the magnitude of removing demand, and in the opposite direction the rate of increase of cost increases with increasing magnitude of adding demand, regardless of the source node where the demand changes.

We also made changes to the β_4 values assigned to each arc in the network, to see which have the greatest impact upon the network as a whole. The change in total cost for the network is tabulated below.

Arc	Cost change, increase β_4 by 10	Cost change, decrease β_4 by 10
o_1v_1	+300	-300
o_2v_2	+400	-400
v_1v_2	+72	-74
v_1v_3	+216	-228
v_2v_4	+472	-474
v_4v_3	+164	-167
v_1d	+392	-394
v_1d	+306	-308

The magnitude of the values corresponds closely to the traffic demand for the respective arcs in the original solution. The fact that the positive and negative changes are very similar indicates that there appears to be very little change in traffic behaviour across the network: the ratios of demand remain very similar. Changing these constants in the demand cost relation thus appears to have a relatively constant effect upon the network. If there were to be a practical variable relating to this, these results would indicate that prioritising the reduction of this value on the roads with the greatest demand would have the greatest positive effect upon the network.

5 Conclusion

After a discussion of related problems and literature in the introduction, the cubic relationship between travel cost per unit of traffic and traffic demand was successfully modelled in conjunction with a computational study of various optimisation methods where the trade-off between speed and correctness of solutions was investigated. An optimal solution to minimise the travel cost through a simple network was also found, with two algorithms for this problem, namely, an interior-point approach based on the log-barrier penalty function and the Frank-Wolfe algorithm, implemented and benchmarked. Furthermore, sensitivity analysis was performed to ascertain the impact of varying demand at incident nodes in the network.

References

Cole, R., Dodis, Y. & Roughgarden, T. (2006), 'How much can taxes help selfish routing?', *Journal of Computer and System Sciences* **72**(3), 444–467.

Florian, M. & Nguyen, S. (1976), 'An application and validation of equilibrium trip assignment methods', *Transportation Science* **10**(4), 374–390.

Fotakis, D., Karakostas, G. & Kolliopoulos, S. G. (2010), On the existence of optimal taxes for network congestion games with heterogeneous users, *in* 'International Symposium on Algorithmic Game Theory', Springer, pp. 162–173.

Frank, M. & Wolfe, P. (1956), 'An algorithm for quadratic programming', *Naval research logistics quarterly* **3**(1-2), 95–110.

Knight, F. H. (1924), 'Some fallacies in the interpretation of social cost', *The Quarterly Journal of Economics* **38**(4), 582–606.

Krylatov, A., Zakharov, V. & Tuovinen, T. (2019), 'Optimization models and methods for equilibrium traffic assignment'.

SciPyCommunity (2019), 'Optimization (scipy.optimize)'.
URL: *https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html*

Wardrop, J. G. (1952), 'Road paper. some theoretical aspects of road traffic research.', *Proceedings of the institution of civil engineers* **1**(3), 325–362.