University of Glasgow | School of Computing Science

Selfish Traffic Routing: A Visual Tool

Euan Halliday
2557097H

ABSTRACT

Modern traffic networks face increasing complexity due to urbanisation, congestion, and decentralised decision-making. Theoretical frameworks such as Wardrop Equilibrium and Social Optimality highlight the relationship between individual route choices and overall efficiency. However, practical applications are hindered by computational challenges, limited visualisation capabilities, and a lack of research tools. This work presents an interactive visual tool that integrates theoretical traffic flow models with modern numerical optimisation techniques to compute and visualise Wardrop Equilibrium, Social Optimality, and the Price of Anarchy in real time. Using a Flask-based Python backend, Sequential Least Squares Programming (SLSQP) is used to find the equilibrium and socially optimal flows. At the same time, a React-based frontend offers an intuitive interface for constructing networks, defining cost functions, and dynamically visualising routing outcomes. Validation on established examples such as the Pigou and Braess networks confirms that the computed results align with theoretical predictions. Additionally, performance evaluations on randomly generated networks demonstrate scalability and robustness. The tool not only combines the application of theoretical models with practical traffic management but also provides a platform for researchers and practitioners to experiment with and analyse the impact of decentralised, selfish routing on overall network efficiency. Users are invited to try the tool themselves at https://2557097h.netlify.app/.

1 INTRODUCTION

Traffic congestion is a critical challenge in modern urban environments, contributing to lost productivity, higher levels of pollution, and significant economic costs. Urban commuters in metropolitan cities spend up to 200 hours per year in traffic, with congestion costing these cities billions of dollars annually [15]. However, traffic congestion refers not only to urban transportation but also to various types of networks, including telecommunications, logistics, and energy distribution [18, 32, 33]. Therefore, 'traffic' can be thought of as a flow of entities that move between interconnected nodes through edges [14]. Optimising this flow, especially in decentralised networks where individual users or agents independently seek to minimise costs, presents a unique challenge in network planning and efficiency. Unlike cooperative network models, selfish routing reflects real-world behaviours where users act independently to reduce costs [34]. In large networks, this behaviour can be modelled as a non-atomic network, which is a system where each user's impact on the overall traffic flow is negligible, allowing traffic to be treated as a continuous flow rather than as discrete units [20].

This selfish behaviour in non-atomic networks can lead to a stable state known as Wardrop Equilibrium, where each user has no incentive to change their route. Wardrop Equilibrium represents a form of Nash Equilibrium tailored to large, decentralised networks [4] and is especially relevant in contexts where users behave independently, such as road traffic, telecommunications, or power distribution [36]. However, while Wardrop Equilibrium is individually stable, it often results in suboptimal outcomes for the entire network, as it can lead to inefficient traffic patterns compared to the socially optimal solution. This phenomenon is known as the Price of Anarchy, highlighting the inefficiency caused by uncoordinated decision-making [31].

Despite the theoretical advancements in understanding Wardrop Equilibrium and selfish routing behaviours, there remains a significant challenge. Computational barriers make it difficult to analyse very large non-atomic networks, while the lack of practical tools limits experimentation and exploring strategies for improving network efficiency [19]. Prior research into algorithms that solve Wardrop Equilibrium highlights ways to decrease this computational overhead [10], although practical tools for simulating and visualising Wardrop Equilibrium remain limited.

1.1 Research Questions & Objectives

This work will therefore guide the core research questions:

RQ1: How does the computed Wardrop Equilibrium compare to the Social Optimum in various network topologies?

RQ2: What are the computational challenges involved in solving equilibrium conditions numerically using convex optimisation techniques?

RQ3: How can visualisation aid in the understanding and analysis of network congestion?

To address these questions, the primary objectives are:

(1) To formalise the mathematical model underlying traffic flow and equilibrium conditions.
(2) To implement a backend Wardrop Equilibrium, Social Optimality, and PoA calculator using Python [16] and Flask [25].
(3) To design a user-friendly React-based [28] frontend that enables interactive construction and visualisation of traffic networks.
(4) To conduct empirical evaluations comparing the calculator's performance and examining the impact of network topology on equilibrium states.

2 BACKGROUND

2.1 Wardrop's Principles and the Foundations of Traffic Flow Theory (1952) [34]

Wardrop's work emerged during rapid urbanisation and increasing vehicular traffic. Cities were experiencing congestion at unprecedented levels, thus needing systematic approaches to managing traffic flow. Wardrop sought to formalise traffic dynamics to address both individual behaviour and system-wide efficiency, laying the foundations for equilibrium-based traffic modelling. In 1952, John Glen Wardrop proposed what is now called the 'Wardrop Equilibrium'. His paper introduced two principles that have since formed the basis of modern traffic flow theory and equilibrium analysis. These principles contrast individual optimisation and collective optimisation in traffic systems. They also highlight how individual user decisions impact overall network performance.

Wardrop's first principle, referred to as **Wardrop Equilibrium** (or **User equilibrium**), asserts:

Principle 1. *In a state of equilibrium, the travel time on all routes used is the same and less than or equal to the travel time on any unused route.*

This principle defines a state where no driver can individually reduce travel time by switching to a different route. At equilibrium, drivers selfishly lower their trip times, disregarding how this may impact the overall network. Wardrop noted that this behaviour is standard since users prioritise personal efficiency over collective benefits.

To formalise this idea, Wardrop investigated the relationship between journey time and traffic flow. He observed that **journey time** t_i is not constant but increases as traffic volume q_i approaches a route's capacity p_i. This relationship can be expressed as:

$$t_i = \frac{b_i}{1 - \frac{q_i}{p_i}}, \tag{1}$$

(b_i is the baseline (free-flow) travel time on route i)

In this relationship, travel time t_i increases non-linearly as q_i approaches p_i, reflecting the congestion effect. This shows how heavily congested roads can experience sharp increases in travel time, which requires traffic management to balance flows.

Wardrop also implied that the **equilibrium conditions** can be represented by:

$$t_i = t_j \quad \forall i, j \text{ where } q_i > 0, \quad \text{and} \quad t_i \geq t_j \quad \forall i, j \text{ where } q_i = 0.$$

These conditions reflect the selfishness of drivers. Therefore, equilibrium is achieved when no driver can improve their travel time by switching routes.

While the First Principle models individual selfish behaviour, the Second Principle seeks to optimise the system as a whole. Wardrop's Second Principle, known as **Social Optimality**, states:

Principle 2. *Traffic is distributed to minimise the total travel time across the network.*

According to this principle, Social Optimality aims to reduce the overall travel time that all vehicles in the system must endure. However, as drivers inherently prioritise their journey times over system efficiency, reaching a socially optimal condition will likely require external interventions like congestion pricing or tolling to align independent choices with systemic advantages. This reflects the trade-off between personal and collective efficiency.

Wardrop defines the **average journey time per vehicle** as:

$$T = \frac{\sum_{i=1}^{D} q_i t_i}{Q},$$

where Q is the total traffic demand and D is the number of routes. Since Q is constant, minimising T reduces to minimising the **total vehicle-time Z**:

$$Z = \sum_{i=1}^{D} q_i t_i. \tag{2}$$

This representation demonstrates that the Second Principle involves balancing flows across all routes to achieve the minimum Z rather than equalising travel times as in the First Principle.

Wardrop derived the conditions for Social Optimality by introducing the concept of **marginal travel time**. The marginal travel time on route i is the derivative of the total vehicle time with respect to the flow on that route, defined as:

$$\begin{aligned} \frac{d(q_i t_i)}{dq_i} &= \epsilon, \qquad \text{for used routes } (q_i > 0) \\ \frac{d(q_i t_i)}{dq_i} &\geq \epsilon, \quad \text{for unused routes } (q_i = 0) \end{aligned} \tag{3}$$

This framework underlines that traffic flow is dispersed so that no route in use has a marginal travel time larger than any other, and unused routes have marginal travel times greater than or equal to ϵ. Thus, this principle reduces total travel time by ensuring network balance and that adding flow to any route increases the total system-wide cost at the same rate.

In practice, marginal travel time is the extra delay caused by adding another vehicle to a route. Managing traffic to ensure that marginal travel times are balanced among routes can decrease bottlenecks and enhance network efficiency. For instance, on a congested highway, marginal travel time increases significantly as capacity is approached.

(Wardrop's relationships for the total flow distribution and constraints under both Wardrop Equilibrium and Social Optimality, can be found in Appendix A.1)

Under Wardrop Equilibrium	Under Social Optimality
Drivers choose routes to minimise their travel times.	Traffic is distributed to minimise the total travel time across the network, even if individual times are unequal.
Travel times on all used routes are equal.	Marginal travel times on all used routes are equal.
Can lead to inefficient outcomes due to overuse of certain routes.	May require interventions to achieve, as it doesn't align with individual self-interest.

Table 1: Comparison between Wardrop Equilibrium and Social Optimality

2.1.1 Comparison Between Wardrop Equilibrium and Social Optimality. Using a **Pigou network**, Wardrop's comparison between Wardrop Equilibrium and Social Optimality can be highlighted clearly. A Pigou network is a simple model that illustrates the inefficiency that can arise under Wardrop Equilibrium. It consists of two parallel routes between an origin and a destination [31].

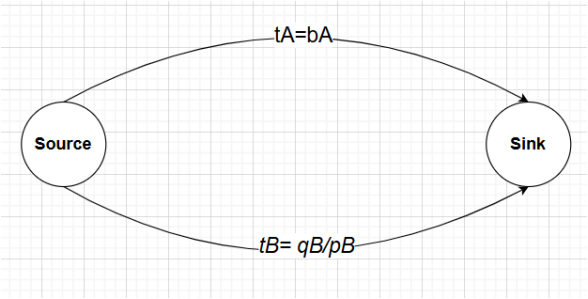

Figure 1: Illustration of the Pigou Network using Wardrop's foundational relationships. Route A has a constant travel time $t_A = b_A$, unaffected by flow (e.g. a longer but uncongested road). Route B has a travel time $t_B = \frac{q_B}{p_B}$, which increases linearly with the flow (e.g. a shorter but congested road).

Under Wardrop Equilibrium, all drivers may choose Route B if it initially offers a shorter travel time, leading to congestion and increased travel times for everyone. Under Social Optimality, some drivers would be redistributed to Route A to minimise travel time across both routes, so that marginal travel times on both routes are equal, resulting in balanced overall network efficiency.

2.2 Wardrop Equilibrium

2.2.1 Beckmann, McGuire, and Winsten: Formalising Wardrop Equilibrium (1956) [6]. In 1956, Beckmann, McGuire, and Winsten extended Wardrop's first principle into a more advanced mathematical framework, providing more theoretical clarity for traffic equilibrium analysis. This was the most substantial advancement since Wardrop had released his paper 4 years prior, and the approach incorporated optimisation theory into traffic flow modelling

Beckmann formalised **Wardrop Equilibrium**, which built upon Wardrop's first principle. He expressed the principle using the concept of a potential function $H(x)$, which represents system-wide travel costs:

$$H(x) = \sum_{i,k} \int_0^{x_{i,k}} g_{i,k}(x)\,dx - \frac{1}{2} \sum_{i,j} \int_0^{x_{i,j}} h_{i,j}(x)\,dx, \quad (4)$$

Where:

- $g_{i,k}(x)$ represents the **inverse demand function**. This can be defined as the generalised travel cost for trips from an origin i to a destination k, as a function of the total flow $x_{i,k}$ between those points. This function reflects how costs change as more trips are demanded between i and k.
- $h_{i,j}(x)$ represents the **cost function**, which describes the travel cost (in terms of time or expense) for using a specific road or link i-j, as a function of the flow $x_{i,j}$ on that road. This cost increases with congestion and is linked to the capacity of the road.

Equation 4 utilises the framework of convex optimisation to ensure that minimising it will result in the flow configuration that satisfies the conditions of Wardrop's first principle. Beckmann demonstrated that Wardrop Equilibrium corresponds to the solution of a convex optimisation problem, where $H(x)$ is minimised over all feasible flow configurations. This means that $H(x)$ accounts for all possible flow arrangements in a network. However, these arrangements are only possible if they satisfy that all traffic is routed from sources to sinks while adhering to flow conservation and non-negativity constraints [31]. Formulating the potential function as a convex optimisation problem is hugely significant. Convexity guarantees at least one global minimum, meaning that the solution of equilibrium traffic flow can be computed in a finite amount of time and is unique in terms of system-wide cost. Equation 4 not only enables the development of efficient algorithms for large-scale traffic assignment but also provides a foundation for comparing Wardrop Equilibrium and Social Optimality. This is because it mathematically highlights how Wardrop Equilibrium prioritises individual gains at the cost of increased system-wide inefficiency, which offers insights into how to solve the problem between individual and system-wide efficiency.

In this context, Beckmann's work showed that the traffic flow at Wardrop Equilibrium satisfies the following optimisation problem:

$$\min H(x),$$

Subject to the following constraints:

(1) **Non-Negativity of Flow** The flow on any edge must be non-negative:

$$x_{ij,k} \geq 0, \quad \forall (i,j), k, \quad (5)$$

where $x_{ij,k}$ represents the flow on the edge connecting node i to node j, for trips destined for k.

(2) **Flow Conservation** The flow originating from a node i to a destination k must respect flow conservation across the network:

$$x_{i,k} = \sum_j (x_{ij,k} - x_{ji,k}), \quad (6)$$

where the outgoing flow from node i is equal to the net difference between flow exiting i and flow entering i.

(3) **Edge Flow Aggregation** The total flow on an edge (i, j) is the sum of flows from all incoming and destinations:

$$x_{ij} = \sum_k (x_{ij,k} + x_{ji,k}), \quad (7)$$

which ensures that all source-destination pairs' flow is correctly aggregated across the network.

These constraints collectively ensure that the optimisation problem minimises the potential function $H(x)$, which reflects the total system-wide travel costs under Wardrop Equilibrium.

Beckmann's framework directly influenced the development of non-atomic congestion games. These networks generalise their equilibrium principles to systems with infinitely small agents [20]. The potential function introduced by Beckmann parallels the potential function used in non-atomic congestion games, highlighting how relevant his work is today.

2.2.2 Non-Atomic Congestion Games. [31]

Non-atomic congestion games extend the equilibrium analysis to scenarios that involve a continuum of players, with each player controlling an infinitesimally small fraction of the total traffic flow. Therefore, they can capture the dynamics of large-scale traffic networks more realistically than a model that assumes discrete players. These games also focus on path-based flows, which suit decentralised decision-making in traffic networks. This is a significant improvement as path-based models align with Wardrop's first principle more naturally as they operate at the level of specific routes rather than Beckman's aggregated edge flows. This granularity allows a more accurate representation of user behaviour and traffic dynamics, particularly in scenarios where route-level decisions significantly impact overall congestion and network performance.

A non-atomic congestion game instance is defined as (G, r, c), where:

- $G = (V, E)$: A directed graph representing the network. V is the set of vertices, and E is the set of edges.
- r: A demand vector specifying the total traffic demand for each source-sink pair (s_i, t_i).
- c: A set of cost functions $c_e(f_e)$ for each edge $e \in E$, where f_e is the total flow on edge e. The cost functions are continuous, non-negative, and non-decreasing, reflecting congestion effects.

Wardrop's first principle states, Principle 1. The modern mathematical formalisation for this can be denoted as a flow f which is at equilibrium if, for every source-sink pair (s_i, t_i) and all paths $P, P' \in P_i$ with $f_P > 0$:

$$c_P(f) \leq c_{P'}(f),$$

where

$$c_P(f) = \sum_{e \in P} c_e(f_e),$$

is the total cost of using path P, with $c_e(f_e)$ denoting the cost function on edge e as a function of the flow f_e.

This formalisation highlights that paths with positive flow ($f_p > 0$) must have costs equal to or lower than any unused path, ensuring that no user has an incentive to switch.

The potential function framework introduced by Beckmann is simplified in non-atomic congestion games as:

$$\Phi(f) = \sum_{e \in E} \int_0^{f_e} c_e(x)\, dx \tag{8}$$

This function has several key properties. Firstly, equilibrium flows minimise the potential function $\Phi(f)$ over the set of all feasible flows. The convexity of $\Phi(f)$ is derived from the non-decreasing nature of the cost functions c_e and ensures the existence of at least one equilibrium flow. Furthermore, the potential function provides a framework for characterising optimal flows, extending the optimisation principles established in 1956.

The non-atomic model builds on the flow constraints which Beckmann articulated in Equations 5, 6, 7. However, they are extended to account for paths rather than edges. These constraints are:

(1) **Non-Negativity of Flow:**

$$f_P \geq 0, \quad \forall P \in P_i, \forall i, \tag{9}$$

where f_P represents the flow on path P between source s_i and sink t_i.

(2) **Flow Conservation**

$$\sum_{P \in P_i} f_P = r_i, \quad \forall i, \tag{10}$$

ensuring that the total flow between each source-sink pair equals the specified demand r_i.

(3) **Edge Flow Aggregation:**

$$f_e = \sum_{P \in \mathcal{P}: e \in P} f_P, \quad \forall e \in E, \tag{11}$$

which aggregates the flow across all paths using a particular edge e.

In summary, non-atomic congestion games extend Beckmann's work by providing a more granular framework for analysing traffic flows by emphasising route-level decisions rather than aggregated edge flows. Equation 8 paves the way for innovative strategies to manage congestion and improve efficiency in complex transportation systems. This is pivotal for exploring topics such as Social Optimality and algorithmic approaches to computing Wardrop Equilibrium.

2.3 Social Optimality

2.3.1 Formalising Social Optimality. Wardrop's Second Principle, known as **Social Optimality (SO)**, states Principle 2. This contrasts to **Wardrop Equilibrium**, which states Principle 1. Therefore, Social Optimality represents the ideal flow arrangement where efficiency for the whole system is maximised.

The formalisation of Social Optimality as an optimisation problem was developed by Dafermos and Sparrow in 1969 [11]. They defined the optimal flow distribution of the system as minimising the total travel cost across a network. This problem, referred to as **Problem $\mathbf{P_1}$** in their paper, is mathematically formulated as:

$$\min C(F) = \sum_{a \in \mathcal{L}} c_a(f_a),$$

where $C(F)$ represents the total cost of the network, F is the flow distribution on the network, and $c_a(f_a)$ is the cost associated with an edge a.

In addition, it is also observed that the cost functions $c_a(f_a)$ are continuous, strictly increasing, and convex, whilst the flow pattern F satisfies the flow constraints defined in Equations 9, 10, and 11, ensuring that the demand between all origin-destination pairs is met. This framework was significant in modelling traffic flow as a system-wide optimisation problem, providing theoretical and computational power for analysing transportation networks.

Modern developments in traffic modelling and Social Optimality use the ideas of Dafermos and Sparrow but refine them to focus on explicitly minimising total travel time. This relates to Wardrop's original work, where Social Optimality focuses on travel time instead of just cost (Equation 2). Therefore, the **optimisation-based formulation** of system-wide travel cost Z, subject to the same flow constraints, can be represented as:

$$\min Z = \sum_{e \in E} c_e(f_e) f_e, \tag{12}$$

Where $c_e(f_e)$ represents the cost (or travel time) on edge e as a function of the flow f_e, where f_e is the total flow on edge e[31].

Here, including the min operator explicitly defines Social Optimality as a minimisation problem, targeting the network's lowest possible total travel cost. This modern formalisation specifically helps with traffic engineering problems as its sole focus is reducing total travel time in a system. However, Dafermos and Sparrow [11] represent Social Optimality in a more general context for a broader range of components (tolls, fuel consumption, etc.) and is often used in contexts where planners need to factor in multiple dimensions of cost.

At Social Optimality, the **marginal social cost** for all used routes is equalised to achieve the most efficient traffic distribution across the network.

Therefore, a modern formalisation can be derived from Wardrop's marginal travel time (Equation 3), which has been adapted to align with Equation 12. For a route e with flow $f_e > 0$, the marginal cost condition is expressed as:

$$\frac{d}{df_e}\left(f_e \cdot c_e(f_e)\right) = \lambda, \quad \text{for used edges } (f_e > 0),$$
$$\frac{d}{df_e}\left(f_e \cdot c_e(f_e)\right) \geq \lambda, \quad \text{for unused edges } (f_e = 0).$$

Where λ is a constant representing the equalised marginal social cost across all used edges.

This framework builds on Wardrop's Second Principle by generalising the concept of marginal travel time to marginal social costs. Using costs over time means factors like tolls and environmental impacts can be included [22]. Equalising marginal costs across all used routes and ensuring unused routes have higher costs results in efficient traffic distribution. Therefore, this formalisation highlights the ability to use tools like congestion pricing, which aligns individual behaviour with system-wide efficiency and extends Wardrop's ideas to modern traffic management challenges, such as balancing multiple cost factors to optimise network performance [22].

2.4 Price of Anarchy

The **Price of Anarchy (PoA)** is an essential concept in congestion games, highlighting the inefficiency arising in decentralised systems where agents act selfishly. In traffic networks, PoA measures the ratio between the total system cost under a selfish equilibrium (Wardrop Equilibrium in this case) and the optimal

social cost [31]. This concept has evolved from earlier theoretical explorations to a framework for analysing real-world traffic inefficiencies.

The origins of PoA can be traced to the **Coordination Ratio**, introduced by Koutsoupias and Papadimitriou (1999) [21]. Their important work on "Worst-Case Equilibria" investigated the inefficiency of decentralised decision-making by selfish agents. They defined the Coordination Ratio as:

$$\text{Coordination Ratio} = \frac{\text{Cost of Worst Nash Equilibrium}}{\text{Cost of Optimal Solution}}, \quad (13)$$

Which highlights the possible difference between individual and socially optimal outcomes. Although their initial focus was on resource-sharing scenarios (hence the Nash equilibrium), their findings laid the groundwork for later studies of inefficiency in traffic networks.

Building on this foundation, Papadimitriou (2001) [26] formalised the term Price of Anarchy in the context of systems aligned with game theory, including traffic networks. His work applied PoA to multi-commodity flow networks and demonstrated that selfish routing leads to suboptimal outcomes and that the inefficiency is bounded by the structure of cost functions [26]. A "multi-commodity" flow network can be defined as one with multiple types of traffic (commodities) moving through it simultaneously, with each commodity representing a different origin-destination pair [5].

2.4.1 *Roughgardens Contributions to PoA.* [31]

Roughgarden extended the study of PoA to non-atomic selfish routing games where individual agents control infinitesimally small portions of the total flow. A key insight from Roughgarden's work was that PoA depends solely on the properties of the edge cost functions (agreeing with [26]), regardless of the network's size, topology, or the number of commodities [29]. Therefore, PoA unified the analysis of traffic inefficiencies across a wide range of network models, providing a foundational understanding of selfish behaviour in transportation systems.

The result for PoA in non-atomic routing builds on the characterisation of equilibrium flows using the potential function in Equation 12 where Z is replaced with $C(f)$. Let $G = (V, E)$ represent a network, where $c_e(f_e)$ is the cost function for edge e.

Equilibrium flows f minimise $\Phi(f)$, which ensures their existence and uniqueness in terms of the total cost. Roughgarden used the potential function to establish an upper bound on the PoA:

$$C(f) \leq \gamma \cdot \Phi(f^*),$$

where f^* is the optimal flow and γ depends on the properties of the cost functions.

This bound is significant because it provides a worst-case guarantee on the inefficiency of Wardrop Equilibrium compared to Social Optimality. This means that even in networks with selfish agents, the total cost cannot exceed a predictable multiple Φ of the cost under optimal conditions. Therefore, the insight is crucial for network design, as it helps planners estimate the maximum inefficiency caused by decentralised decision-making and informs strategies like tolling to minimise this inefficiency.

For polynomial cost functions of degree p, $c_e(x) = ax^p$, Roughgarden, derived a precise bound:

$$\text{PoA} = 1 + \frac{p}{p+1}.$$

This result shows that inefficiency grows with the degree of the cost functions. For example, linear functions ($p = 1$) yield a PoA of $\frac{4}{3}$, while quadratic functions ($p = 2$) yield a PoA of $\frac{3}{2}$. These bounds highlight the relationship between nonlinearity and inefficiency.

Roughgarden uses two examples to illustrate the implications of his work:

- **Pigou's Example**: A simple network with two parallel edges demonstrates that selfish routing can lead to significant inefficiency, with a PoA matching the polynomial bound $1 + \frac{p}{p+1}$.
- **Braess's Paradox**: occurs when adding a new route to a network increases total travel cost under selfish routing. Roughgarden showed that even in scenarios involving Braess's Paradox, the PoA is at most $\frac{4}{3}$ for networks with affine cost functions.

(Further proofs on the PoA bounds established by Roughgarden, can be found in Appendix A.2)

2.5 Algorithmic Approaches to Computing Equilibrium[7]

Creating an efficient algorithm for the computation of Wardrop's Equilibrium and Social Optimality is essential for applying theoretical traffic models to large-scale transportation networks. Various algorithms have been developed to compute Wardrop Equilibrium and Social Optimality, where each algorithm leverages different optimisation strategies and computational approaches. These algorithms are summarised in [7] and can be defined into three different types:

2.5.1 ***Link-Based Algorithms.***

Link-based algorithms focus on optimising flows at the level of individual network links. They achieve this by iteratively updating link-level traffic volumes until all used paths are balanced, thus resulting in Wardrop Equilibrium. These computationally efficient methods formed the backbone of many early traffic assignment models.

Method of Successive Averages (MSA). MSA is a heuristic algorithm that uses relative travel times to adjust flow proportions along paths. The flow adjustments at each iteration are averaged with the current solution, and eventually, the algorithm will converge to equilibrium. Its fixed step sizes make it computationally simple, but results in slow convergence, particularly near equilibrium.

- **Advantages:** Simple to implement and low computational resource requirements.
- **Limitations:** Extremely slow convergence for congested networks or high-level precision needs and unsuitable for non-linear cost functions.

Frank-Wolfe Algorithm (FW). The Frank-Wolfe algorithm simplifies solving Wardrop Equilibrium by linearising each step's non-linear objective function (Equation 8). This linear approximation solves the shortest path problem, determining the best direction to adjust traffic flows. Furthermore, it blends the current and new flows using an optimal step size to verify steady improvement.

- **Advantages:** Avoids costly projections onto the feasible set by solving a linear sub-problem at each iteration.

It also offers faster convergence than MSA and is suitable for large networks. However, solving the linear subproblem can become computationally expensive as the network grows.
- **Limitations:** Suffers from slow convergence near equilibrium due to restricted search directions. This can often lead to zig-zagging behaviour where the algorithm changes between feasible points without making significant progress toward the equilibrium.

Conjugate Frank-Wolfe Algorithm (CFW). CFW refines the FW method by introducing conjugate search directions, reducing zig-zagging and improving convergence rates. It is particularly effective near equilibrium, where FW struggles. However, it does come with the caveats of higher computational complexity and implementation challenges compared to FW.

2.5.2 ***Path-Based Algorithms.*** Path-based algorithms manage traffic flows explicitly at the route level, which provides greater precision than link-based approaches. These methods are computationally intensive but ideal for applications requiring high accuracy.

Gradient Projection Method. This algorithm optimises path flows using the objective function (Equation 8) while staying in the feasible region. It iteratively adjusts path flows by computing gradients of the objective function and moving in that direction. The method then modifies the gradient direction to ensure the updated flows remain in the feasible region.

- **Advantages:** Handles constraints effectively and has steady convergence with minimal oscillations.
- **Limitations:** Computationally intensive and requires significant memory for path-level flow tracking.

Projected Gradient Method. This variation of the Gradient Projection Method optimises path flows by performing an unconstrained gradient descent step and then explicitly projecting the updated flows back into the feasible region.

- **Advantages:** Efficiently balances gradient descent with strict constraint enforcement. It is also robust against infeasibility errors by correcting violations in a separate projection step.
- **Limitations:** Complex to implement due to the need for an explicit and potentially computationally expensive projection process.

2.5.3 ***Bush-Based Algorithms.*** Bush-based algorithms are a more modern range of algorithms that leverage the acyclic nature of feasible path sets. They focus on sub-graphs (bushes) for each origin-destination (OD) pair, which balances the efficiency of link-based approaches and the precision of path-based methods. Bush-based algorithms operate within acyclic sub-graphs and iteratively update flows within bushes to meet equilibrium conditions.

- **Advantages:** Reduced computational complexity, highly effective for large-scale networks, memory-efficient, scalable and achieves rapid convergence.
- **Limitations:** Complicated to implement and requires careful implementation to maximise computational advantages.

Algorithmic approaches to computing Wardrop equilibria have evolved to handle the various difficulties of extensive transportation networks. Path-based algorithms offer increased precision and scalability, while link-based approaches offer improved computing efficiency. In addition, bush-based algorithms provide the best of both worlds but are very hard to implement correctly. The advancements of algorithms ensure that theoretical traffic models are more efficient at informing congestion management.

2.5.4 Applicability to Social Optimality. The algorithms above are primarily designed to compute Wardrop Equilibrium. However, they can also be adapted to calculate Social Optimality. This is achieved by modifying the cost functions to reflect marginal social costs rather than individual travel times using Equation 12.

2.6 Visualisation Techniques

Visualising Wardrop Equilibrium is essential for understanding traffic flow dynamics within complex networks. Visualisations can help identify congestion hotspots, compare Wardrop Equilibrium with Social Optimality, and communicate insights to researchers. In addition, they connect the theory to the practical application, which enables researchers and decision-makers to interpret the network behaviour and design informed interventions. Various visualisation techniques have been developed to represent traffic networks and their equilibria. These methods cater to different aspects of traffic flow analysis, with each offering its advantages:

Network Diagrams *[17].* Network Diagrams are static graph-based representations that use nodes to highlight intersections and edges to represent roads or routes. They effectively provide a broad, spatially organised view of traffic networks and identify critical routes or congestion points. They incorporate flow arrows and weighted edges to illustrate traffic dynamics like flow direction and congestion levels. However, their static nature makes them less effective in representing temporary changes or dynamic adjustments in traffic flows, limiting their use for real-time equilibrium analysis.

Heat Maps *[24].* Heat Maps are used to depict traffic intensity or travel times using gradient colours over a geographic area. Therefore, overlaying traffic data on geographic maps can provide contextual insights into real-world conditions. They are ideal for identifying high-demand areas and tracking temporal changes in congestion patterns. However, heat maps cannot show direct flows, making them less effective for analysing traffic distribution.

Flow Maps *[27].* Flow Maps are used to visualise traffic movement using lines or arrows, where thickness corresponds to flow volume. These dynamic maps can adapt to real-time or simulated conditions, illustrating how traffic redistributes during equilibrium adjustments. They help understand how flows evolve in response to network changes or interventions. Despite this, flow maps can become cluttered in dense networks with overlapping paths, making it harder to identify traffic intensity.

Interactive Simulations *[35].* Interactive Simulations are used to integrate user-driven inputs with real-time feedback, enabling dynamic exploration of traffic patterns. Users can manipulate network parameters such as flow, cost functions, and road capacities to observe how flows evolve toward equilibrium. Interactive simulations facilitate comparative analysis and help researchers understand the impacts of different configurations. The limitations of interactive simulations are that they require more significant computational resources and a robust UI to ensure accessibility and effective visualisation.

2.6.1 Related Visualisation Tools. Although few tools allow researchers and decision-makers to examine Wardrop Equilibrium or Social Optimality directly from network visualisations, many traffic simulation tools use the theoretical concepts. However, these tools are directly designed for road planning and traffic management, which obscures their applicability for equilibrium-focused research due to their broader scope and complexity. Despite this, they still provide valuable insights and features that can inform and enhance the development of a specialised research tool for studying and visualising traffic equilibrium.

***PTV VISUM** [2].* Mainly used for macroscopic traffic modelling and strategic planning. (Macroscopic modelling analyses overall traffic flow patterns across entire networks rather than individual vehicles) Its strength lies in long-term planning for urban traffic, which uses Wardrop Equilibrium and Social Optimality.

SUMO (Simulation of Urban Mobility)[3]. It focuses on microscopic simulation, where SUMO models individual vehicle behaviours and dynamic traffic conditions. (Microscopic simulation focuses on individual vehicles' detailed interactions and behaviours) It is often used for operational studies like optimising signal timings or testing autonomous vehicle algorithms.

AIMSUN NEXT[1]. Offers multi-scale simulations, from macroscopic to microscopic. (Multi-scale simulation integrates both broad network-level analysis and detailed vehicle-level interactions) It bridges strategic planning and operational studies, providing tools for dynamic traffic assignment and congestion analysis.

Identified Gaps in Current Tools. As can be seen, these existing tools contribute valuable functionalities. However, a clear gap remains in integrating theoretical analysis with an intuitive, interactive user interface. Specifically, the following gaps have been identified:

(1) **Lack of Integrated Equilibrium Analysis:** The existing tools focus on detailed vehicle-level simulations and large-scale strategic planning without directly computing theoretical equilibrium conditions, or, if they include equilibrium analysis, they don't offer dynamic visual feedback. This limitation restricts users' ability to observe and interpret how equilibrium conditions evolve in response to changes in network configurations or cost functions.
(2) **Usability and Accessibility Challenges:** The complexity and steep learning curves of many current platforms limit their accessibility, particularly for users who need a tool for both research and practical applications. There is a strong need for a tool that simplifies these advanced concepts while maintaining theoretical rigour.
(3) **Limited Real-Time Interaction:** Although some tools support real-time simulation, few allow for interactive experimentation with network parameters and immediate visualisation of equilibrium states. This interactivity is essential for both educational purposes and practical traffic management applications.

3 METHODOLOGY

This chapter presents the methodology underpinning the tool's design and implementation. The approach was devised by addressing the gaps identified in Section 2.6.1 and integrates mathematical models of traffic flow with modern numerical optimisation techniques and an interactive visualisation frontend. The system is built to translate Wardrop Equilibrium, Social Optimality, and the Price of Anarchy into practical computational methods that enable real-time analysis of traffic networks. Additionally, it offers a minimalistic, intuitive platform for researchers and learners to experiment with different network configurations dynamically, analyse traffic congestion, and explore mitigation strategies. It also serves as a quick solution for practitioners, such as transportation engineers, who must address immediate traffic problems or compare various network scenarios in real time. An overview of the architecture is provided, which details the integration of the Flask [25] backend with the React [28] frontend.

To overcome the limitations of existing tools, three design principles were adopted:

(1) **Theory Meets Practice:** Numerical optimisation techniques (implemented using Python [16], Numpy [13], Sympy[9] and Scipy [8] via a Flask [25] backend) are integrated with a React-based [28] frontend. The tool computes and displays Wardrop Equilibrium, Social Optimality, and PoA in real time, and calculates the corresponding edge flow values for both Equilibrium and Social Optimality. In addition, it traces all possible routes from each source using a depth-first search. Therefore, this combined approach directly links the underlying theoretical analysis with interactive user feedback.
(2) **User-Friendly Interface:** The interface aims at simplicity and intuitiveness, making advanced traffic modelling concepts accessible to a wider range of users. Dynamic visualisations allow users to adjust network parameters (e.g., adding nodes/edges, defining cost functions) and instantly observe the resulting impact on equilibrium states.
(3) **Real-Time Feedback and Analysis:** The platform's interactive nature enables users to conduct comparative analyses, observe convergence properties, and evaluate the efficiency of various network configurations on the fly. This immediate feedback represents a significant improvement over traditional, static simulation tools.

The proposed integrated approach completes an existing gap, benefiting academic traffic modelling research and equilibrium computation while also being practical in urban traffic management and planning.

3.1 Mathematical Formulation

Following the gap analysis, the next step was developing a mathematical framework for calculating Wardrop Equilibrium, Social Optimality, and the PoA. This framework translates the theoretical principles outlined in the Background (see Section 2) into a practical optimisation problem. Wardrop Equilibrium in a network can be formulated as a constrained optimisation problem. In this formulation, a flow distribution that minimises a global objective function while satisfying the physical constraints of the traffic flow is aimed for. When defining the optimisation problem, both the decision variables and the objective functions for

achieving Wardrop Equilibrium and Social Optimality must be made clear:

3.1.1 Decision Variable. The decision variable in this optimisation problem is the edge flows. Each edge e_i in the network is associated with a flow variable x_i representing the traffic volume on that edge. The flow vector gives the complete set of decision variables:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n],$$

where n is the total number of edges. As a non-atomic model, individual contributions to the flow are assumed to be infinitesimally small, which means x_i can be treated as continuous variables. This continuous formulation enables the application of standard numerical optimisation techniques, such as integration, to compute Wardrop Equilibrium.

3.1.2 Objective Functions.

Potential Function $PF(\mathbf{x})$: Wardrop Equilibrium states that, in equilibrium, no driver can reduce their travel time by unilaterally switching routes. This is equivalent to minimising a potential function that encapsulates the cumulative travel cost along all edges. For an edge e_i with a cost function $c_i(x_i)$, the potential function is defined as:

$$PF(\mathbf{x}) = \sum_{i=1}^{n} \int_0^{x_i} c_i(x)\, dx. \tag{14}$$

The integral captures the total cost of increasing the flow on edge e_i from zero to x_i. When $PF(\mathbf{x})$ is minimised, it yields the equilibrium flow configuration, which means that any infinitesimal shift in flow will not lead to a lower overall cost.

Total Cost Function $TC(\mathbf{x})$: In contrast to the individualistic behaviour modelled by Wardrop's Equilibrium, the Social Optimal solution is achieved by minimising the total travel cost incurred over the entire network:

$$TC(\mathbf{x}) = \sum_{i=1}^{n} c_i(x_i) \cdot x_i. \tag{15}$$

In effect, this function aggregates the individual costs experienced on every edge to measure the network's overall efficiency. Minimising $TC(\mathbf{x})$ produces a flow distribution that minimises the sum of all travel costs, leading to an optimal state throughout the system. This represents a perfect hypothetical scenario in which traffic is allocated so that the total cost across the network is minimised, even if some individual routes might have higher travel times than in a purely selfish routing scenario.

3.1.3 Constraint Formulation. The optimisation problem must also respect several constraints that represent the physical realities of traffic flow:

1. Flow Conservation Constraints: The total incoming flow must equal the total outgoing flow for every intermediate node (nodes that are neither sources nor sinks). Mathematically, for a given intermediate node v:

$$\sum_{e \in I(v)} x_e - \sum_{e \in O(v)} x_e = 0, \tag{16}$$

where $I(v)$ and $O(v)$ denote the sets of edges entering and leaving the node v. This constraint guarantees no accumulation or loss of flow at any node, reflecting the conservation of entities (vehicles, data packets, energy, etc.) in the network.

2. Source Flow Constraints: For each source node s in the network, there is a corresponding, fixed total flow associated with it, denoted as TotalFlow_s. The sum of the flows on all edges leaving a source node must equal this value:

$$\sum_{e \in O(s)} x_e = \text{TotalFlow}_s. \tag{17}$$

This constraint guarantees that the network is supplied with the correct amount of flow from each source.

3. Non-Negativity and Upper Bound Constraints: Each edge flow x_e must be non-negative:

$$x_e \geq 0 \quad \forall e \in E. \tag{18}$$

Moreover, for edges leaving a source node s, an upper bound is enforced such that:

$$x_e \leq \text{TotalFlow}_s. \tag{19}$$

These bounds are directly enforced in the optimisation routine as variable limits. This means that the computed flows remain within feasible and realistic ranges.

3.2 Backend Implementation: Flask Server

This section describes the implementation of the Flask-based [25] backend server, which hosts the Python-written [16] algorithm to compute Wardrop Equilibrium, Social Optimality, and PoA. The algorithm transforms the network flow problem using the mathematical formulation from Subsection 3.1 into a constrained optimisation task. The server processes user input, constructs a network model, builds objective functions and constraints (as specified in Equations 14, 15, 16, and 17), performs numerical optimisation, and returns the results in JSON format to the React frontend.

3.2.1 Input Processing and Network Construction. The server accepts a JSON payload that defines the network as a directed graph and contains two main components:

- **Nodes:** Each node is described by a unique identifier and a type (e.g., source, sink, or intermediate). Furthermore, the source nodes have an attribute called `totalFlow` that describes how much flow goes into the network.
- **Edges:** Each edge consists of a unique identifier, the identifiers of its source and target nodes, and a `costFunction` (a mathematical string such as `x` or `x**2+5`) defining the travel cost as a function of flow.

The function `parse_network` extracts nodes, edges, node types, and source flows from the JSON input. This parsed information is then used to build the network topology and form optimisation constraints.

3.2.2 Objective Function Construction. Using Sympy [9], the function `build_potential_and_total_cost_functions` constructs two key objective functions:

- The **Potential Function** $PF(\mathbf{x})$ (Equation 14), whose minimisation yields the Wardrop Equilibrium.
- The **Total Cost Function** $TC(\mathbf{x})$ (Equation 15), whose minimisation corresponds to the Social Optimal flow.

These symbolic expressions are converted into numerical functions using `lambdify` for integration into the optimisation routine.

3.2.3 Constraint Formulation. The optimisation problem is subject to a set of constraints (Equation 16, 17, 18, and 19), which are implemented within the backend algorithm. The function `build_constraints` dynamically constructs the flow conservation constraints for intermediate nodes and the source flow constraints for source nodes based on the parsed node and edge data. However, non-negativity and upper bound constraints are not constructed as functions but are enforced through the variable bounds in the optimisation routine.

3.2.4 Numerical Optimisation. The numerical optimisation is performed using SciPy's [8] `minimise` function with the Sequential Least Squares Programming (SLSQP) method, which is well-suited for the convex optimisation problem. This is because, SLSQP is capable of handling non-linear objective functions, equality constraints (such as the flow conservation and source flow constraints detailed in Equations 16 and 17), as well as variable bounds (Equation 18 and 19). Although various algorithmic approaches (e.g. MSA, Frank-Wolfe, gradient projection, and bush-based algorithms) were proposed to compute Wardrop equilibrium and Social Optimality in Section 2.5, the implementation of the SLSQP method was opted for several reasons. The first reason is that SLSQP offers a unified optimisation framework that handles non-linear objectives, equality constraints, and bounds directly. This eliminates the need for a custom iterative calculator tailored to link or path-based flow updates. The next reason is that SLSQP is extensively tested and robust, guaranteeing convergence to the global optimum in convex problems. This is critical for accurately computing both equilibrium and socially optimal flows. The final reason is its ease of implementation. Leveraging SLSQP via SciPy allows more focus on integrating the theoretical model with the interactive visualisation frontend, rather than implementing complex algorithms from scratch.

During numerical optimisation, an initial guess is required to begin the iterative optimisation process. It was decided that a vector of ones would be used as the initial guess for the algorithm. This uniform starting point is chosen because it is simple and generally satisfies the variable bounds for each edge (Equation 18 and 19).

The objective functions $PF(\mathbf{x})$ (Equation 14) and $TC(\mathbf{x})$ (Equation 15) were originally constructed to accept multiple arguments. However, SciPy's optimiser requires a single array as input. To bridge this gap, wrapper functions are defined using lambda expressions:

$$\texttt{potential\_wrapper} = \lambda f : \texttt{potential\_func}(*f),$$

$$\texttt{total\_cost\_wrapper} = \lambda f : \texttt{total\_cost\_func}(*f).$$

These wrappers unpack the array of edge flows and pass them as separate arguments to the corresponding functions.

The optimisation execution was carried out using two calls:

(1) To compute the Wardrop Equilibrium, the potential function is minimised:

```
eq_result = minimize(
    potential_wrapper,
    initial_guess,
    method='SLSQP',
    bounds=bounds,
    constraints=constraints,
)
```

(2) To compute the Social Optimal flow, the total cost function is minimised:

```
social_result = minimize(
    total_cost_wrapper,
    initial_guess,
    method='SLSQP',
    bounds=bounds,
    constraints=constraints,
)
```

If either optimisation fails to converge, an error message is returned.

3.2.5 ***Results and Error Handling.*** If the optimisation converges successfully, then the total travel cost for both the Wardrop Equilibrium and the Social Optimal solutions is computed by evaluating their respective cost functions and extracting the edge flow distributions for each scenario. The Price of Anarchy is then determined by

$$\text{PoA} = \frac{\text{Equilibrium Cost}}{\text{Social Optimal Cost}},$$

which quantifies the inefficiency of selfish routing. The computed flows, travel costs, and PoA are rounded to 3 decimals. This level of precision is chosen to balance accuracy while maintaining that the results are easily interpretable. The results are then aggregated into a JSON response for return to the frontend. Error handling is integrated throughout the process, and if any exceptions are encountered during input parsing, constraint construction, or optimisation, then user-friendly error messages are presented to the user via a dedicated error handler.

3.2.6 ***Visual Summary of the Backend Process.*** :

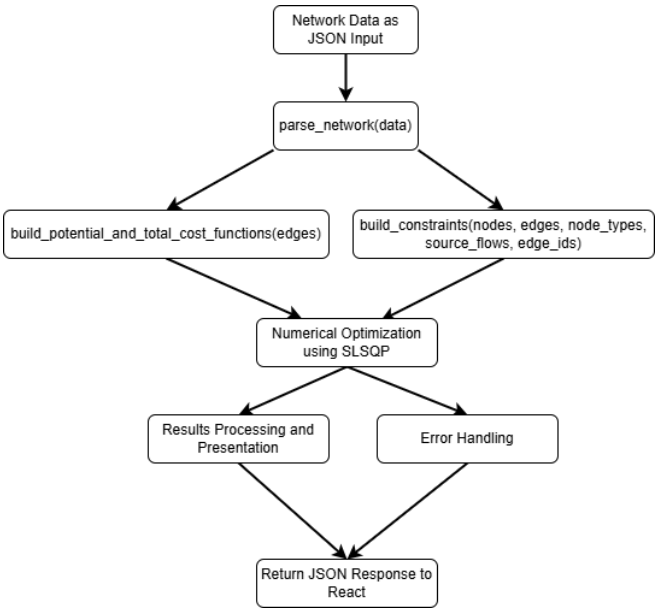

Figure 2: Flowchart of the Flask server backend process

3.3 Frontend Implementation: React Visualisation Tool

This section describes the implementation of the interactive frontend, where users can construct network diagrams for traffic

analysis, define cost functions and source flows, and then visualise computed results in real time. Therefore, network diagrams and interactive simulations were adopted from the discussion in Section 2.6 for a combined approach. Network diagrams provide an intuitive overview of the network topology. At the same time, interactive simulations let users dynamically explore how adjusting network parameters will affect Wardrop Equilibrium and Social Optimality in real time. The frontend was developed using React [28], a popular JavaScript library for building user interfaces, and the react-flow-renderer library [12], which provides a framework for graph-based visualisations.

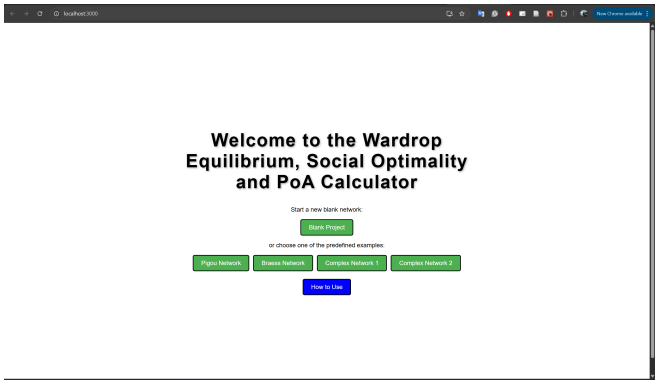

Figure 3: Landing page of the React application

Figure 3 shows the application's landing page, which serves as the entry point for users to access the tool. Users can build their own network from scratch or choose a pre-defined example (e.g., Braess Network).

3.3.1 User Interactions and Network Construction. The application supports a range of user interactions that let them construct networks. Users can drag and drop nodes (sources, sinks, or intermediate nodes) onto a canvas. The `DnDFlow` component manages the network's state using `ReactFlow`'s hooks (`useNodesState` and `useEdgesState`). When nodes are dropped, unique identifiers are generated (via functions like `getNodeId` and `getEdgeId`), and properties such as their position on the canvas and their label are assigned. Each edge is associated with a cost function (defined in terms of x) that the user enters via input fields in the sidebar. These cost functions are specified as strings representing mathematical expressions (e.g. `x` or `x**2+5`). Sympy [9] supports a wide range of mathematical symbols and operations, which allows complex expressions to be defined. However, the cost functions must be convex to ensure that the subsequent numerical optimisation is valid and converges. Users are advised to verify that their expressions represent convex functions, as using more complex or non-convex symbols may lead to computational issues. Additionally, the application supports dynamic modification of the network. Users can delete nodes and edges through keyboard shortcuts and adjust the node handles interactively. This lets them increase/decrase node handles to maintain visually apparent networks with spaced out edges.

3.3.2 Integration with the Flask API. The frontend communicates with the backend Flask [25] server to compute Wardrop Equilibrium, Social Optimality, and PoA in real time. The function `getNetworkData` retrieves the current state of the network from the frontend. It collects information from each node, such as the unique identifier, type (e.g. source, sink, or intermediate), and,

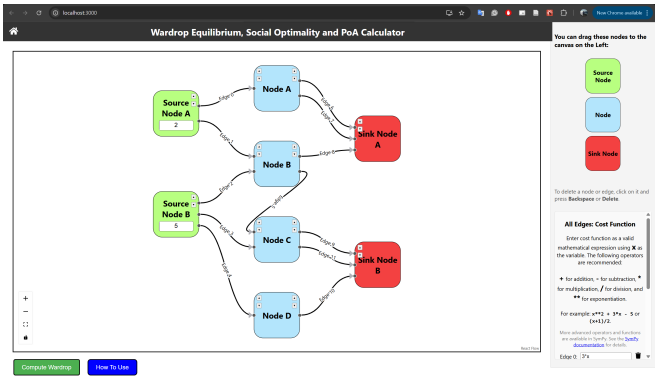

Figure 4: An example of a network diagram (ComplexNetwork1 from the pre-defined examples) which shows the main interface where users can interact with and modify network elements, such as nodes and edges, to observe the resulting traffic flow computations

for source nodes, the specified total flow. Then, from each edge, it retrieves the unique identifier, the source and target node identifiers, and the cost function. This information is then structured into a JSON object that accurately represents the network configuration and is sent to the backend for processing. Upon clicking the "Compute Wardrop" button, the `handleCompute` function initiates an asynchronous POST request to the Flask server endpoint (`/calculate-equilibrium`). This request, implemented via the Fetch API, sends the current network data (nodes, edges, and associated parameters) in JSON format. Once the backend computes the equilibrium results, it sends a JSON response with edge flow distributions, total travel costs for Wardrop Equilibrium and Social Optimality, and the PoA to the frontend. Additionally, the frontend performs a depth-first search to determine all viable source-to-sink routes incorporated into the final results. Any errors returned by the server are captured and displayed to the user via an error message component, ensuring that issues (such as invalid network configurations) are clearly communicated.

3.3.3 Graph Rendering and Result Visualisation. The network is rendered on an interactive canvas using `ReactFlow`. Custom node components (e.g. `SourceNode`, `DefaultNode`, and `SinkNode`) and a custom edge component (`CostEdge`) are used to visually represent the different elements of the network. These components support additional interactions such as updating cost functions or adjusting node properties. Once computation is complete, the results (including edge flows, overall network metrics and source-sink routes) are displayed in a modal window (`ResultsModal`) in real time (as shown in Figure 13 at Appendix B.1). In Addition, the application includes a set of predefined network configurations (e.g., Pigou, Braess, and complex networks) that users can select to quickly visualise and test different traffic scenarios. This feature aids in benchmarking the system and evaluating its performance under various network topologies.

3.4 Overview of System Architecture

The system consists of two main components: a React-based [28] frontend for interactive network design and visualisation and a Flask-based [25] backend server for computing Wardrop Equilbrium, Social Optimality, PoA, and edge distributions. This

section provides a high-level overview of the data flow between them.

3.4.1 Data Flow. The data flow in the system proceeds through the following steps:

(1) **Network Construction:** The user interacts with the React frontend to build the network by adding nodes and edges and defining parameters (e.g., cost functions and total flows).
(2) **Data Aggregation:** The `getNetworkData` function aggregates the current state of the network into a JSON object.
(3) **Data Transmission:** An asynchronous POST request is sent from the frontend to the Flask server endpoint (`/calculate-equilibrium`) using the Fetch API.
(4) **Backend Processing:** The Flask backend parses the input (using `parse_network`), constructs the network model, builds objective functions (`build_potential_and_total_cost_functions`) and constraints (`build_constraints`), and performs numerical optimisation to compute the Wardrop Equilbrium and Social Optimum.
(5) **Result Return:** The computed results (edge flows, total costs, and PoA) are processed and aggregated into a JSON response and sent back to the React frontend.
(6) **Visualisation Update:** Upon receiving the results, the frontend updates the network visualisation in real time. A depth-first search (DFS) algorithm is also executed to determine all possible source-to-sink routes, which are then displayed.

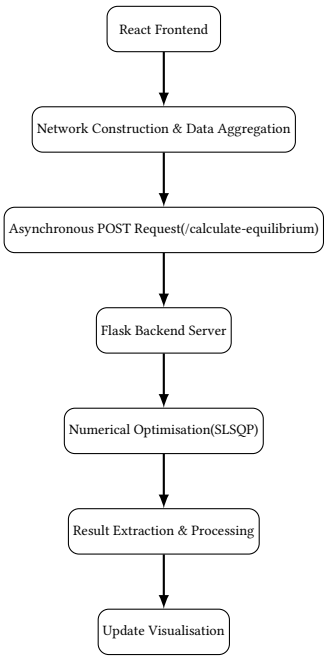

Figure 5: High level system architecture diagram: interaction between React frontend and Flask backend

4 EVALUATION AND RESULTS

In this section, the performance and accuracy of the tool are evaluated. The evaluation is organised around two primary objectives. Firstly, the tool's results are validated by comparing them against theoretical benchmarks of well-established networks such as the Pigou and Braess networks. Secondly, the calculator's computational efficiency, convergence properties, and PoA values are analysed through experiments on randomly generated networks of various sizes and complexity profiles. These evaluations confirm that the tool replicates established theoretical results while suggesting areas for further algorithmic optimisation.

4.1 Correctness Validation

The correctness of the calculator was verified by running it on two standard networks: a Pigou network and a Braess network. These networks are extensively discussed in the literature and have well-established properties that facilitate a direct comparison between the computed results and expected outcomes. In particular, the analyses presented in [30] and [31] serve as theoretical benchmarks for validating the calculator's results. Therefore, the networks in the React application are constructed using the exact parameters specified in these texts, meaning a direct comparison can be conducted.

4.1.1 Pigou Network. The Pigou network has two parallel edges connecting Source Node A to Sink Node A with a total flow demand of 1. Cost functions are defined as $c(x) = x$ for Edge 0, and $c(x) = 1$ for Edge 1.

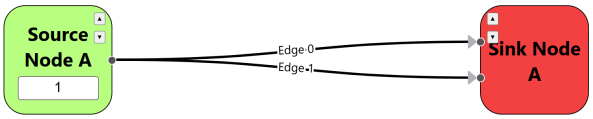

Figure 6: Visualisation of the predefined Pigou network as rendered by the React application

Roughgarden [30] emphasises that in a unique Wardrop equilibrium, all flow is routed on the constant-cost edge (Edge 1), resulting in an equilibrium total cost of 1. However, the socially optimal solution splits the flow equally between the two edges, leading to a total cost of:

$$C_{\text{opt}} = 0.5^2 + 0.5 = 0.25 + 0.5 = 0.75.$$

Thus, the Price of Anarchy (PoA) is given by:

$$\text{PoA} = \frac{1}{0.75} \approx \frac{4}{3}.$$

Table 2 contains the computed results for the Pigou network using the React application. The results match the classical theoretical analysis, thus validating the correctness of the calculator.

Table 2: Pigou network results

	Wardrop Equilibrium	Social Optimal
Edge 0 Flow	0	0.5
Edge 1 Flow	1	0.5
Total Cost	1	0.75
Price of Anarchy	1.333	

4.1.2 Braess Network. The Braess network is analysed in two configurations: before and after the addition of a zero-cost middle link.

Before Adding the Middle Link. The network contains four nodes with two disjoint paths from Source Node A to Sink Node A. In the first path, the edge departing from the source is assigned a variable cost of $c(x) = x$, while the edge entering the sink has a constant cost of 1. In the second path, the cost functions are swapped where the edge leaving the source has a constant cost of 1 and the edge entering the sink carries a cost of $c(x) = x$.

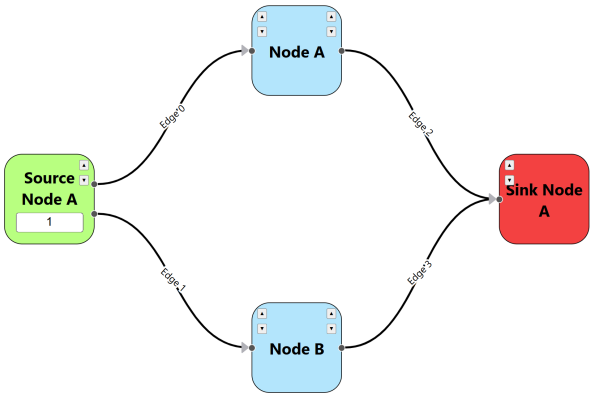

Figure 7: Visualisation of the predefined Braess network before adding the middle link as rendered by the React application

Roughgarden [31] describes that at equilibrium, flow splits evenly between the two paths (0.5 on each), resulting in a path cost of 1.5. Since the equilibrium flow is socially optimal in this configuration, the total social cost is 1.5, and the PoA is 1.

Table 3 contains the results from the React application, which confirm the theoretical predictions.

Table 3: Braess network results (before adding middle link)

	Wardrop Equilibrium	Social Optimal
Edge 0 Flow	0.5	0.5
Edge 1 Flow	0.5	0.5
Edge 2 Flow	0.5	0.5
Edge 3 Flow	0.5	0.5
Total Cost	1.5	1.5
Price of Anarchy	1.0	

After Adding the Middle Link. The network is modified by adding a zero-cost middle link that connects the intermediate nodes. The cost functions for the original edges remain unchanged.

Roughgarden [31] shows that with the inclusion of a zero-cost edge, the equilibrium shifts all flow along the edge, incurring a total cost of:

$$1 + 0 + 1 = 2.$$

Since the social optimum remains at 1.5, the PoA is:

$$\text{PoA} = \frac{2}{1.5} \approx \frac{4}{3}.$$

Table 4 summarises the computed results from the React application after adding the middle link, confirming the theoretical analysis.

The results for both the Pigou and Braess networks match the theoretical predictions presented in [30] and [31]. This agreement validates the correctness of the calculator as implemented

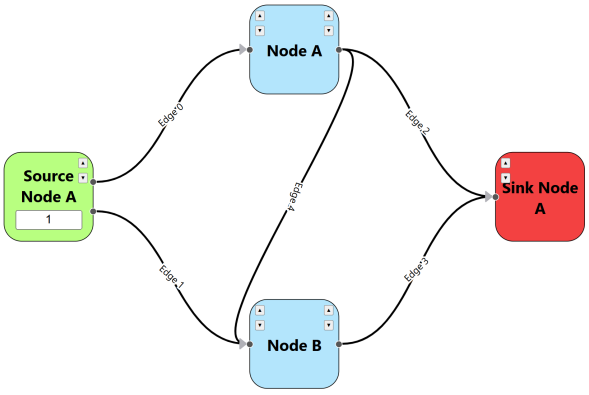

Figure 8: Visualisation of the predefined Braess network after adding the middle link as rendered by the React application

Table 4: Braess network results (after adding middle link)

	Wardrop Equilibrium	Social Optimal
Edge 0 Flow	1	0.5
Edge 1 Flow	0	0.5
Edge 2 Flow	0	0.5
Edge 3 Flow	1	0.5
Edge 4 Flow (Middle Link)	1	0
Total Cost	2	1.5
Price of Anarchy	1.333	

in the React application and that it can reliably replicate well-established outcomes.

4.2 Performance Analysis

4.2.1 Experimental Setup. A test framework was developed to generate random directed networks of different sizes and cost function profiles. Each network has a set of nodes where the first node is designated as the source (with an initial flow) and the last node as the sink. Thus, this sets up a network structure with clear entry and exit points. Next, all possible directed edges between distinct nodes are determined, and a random subset is selected such that the number of edges is four times the number of nodes. Therefore, this creates a sufficient number of connections for realistic routing scenarios. Each edge is assigned a cost function based on one of three modes:

- *linear_random* mode: In this mode, the cost function for each edge is defined as $a \cdot x + b$.
- *quadratic_random* mode: Here, each edge's cost function is defined as $a \cdot x^2 + b$.
- *mixed* mode: Each edge is randomly assigned a linear or quadratic cost function with a probability of 50% for each.

For each cost function mode, the calculator is evaluated on networks with node counts of 10,20, ..., 100. Furthermore, 15 independent instances are generated for each configuration to average out any inconsistencies and the effect of randomness on network generation and cost assignment. The coefficients a and b are randomly drawn from the ranges [0.5, 2.0] and [0, 1.0] to simulate

different levels of sensitivity to traffic flow. In each run, key metrics are recorded to provide a basis for evaluating the calculator's computational efficiency and scalability:

- **Execution Time**, which was measured using wall-clock time, is the primary metric indicating the calculator's overall computational efficiency and scalability.
- **Iteration Count** is the number of iterations required for the SLSQP optimiser to converge for both the Wardrop Equilibrium and Social Optimality computations.
- **Price of Anarchy**, which calculates the ratio of total costs for Wardrop Equilibrium vs Social Optimality.

4.2.2 Results. :

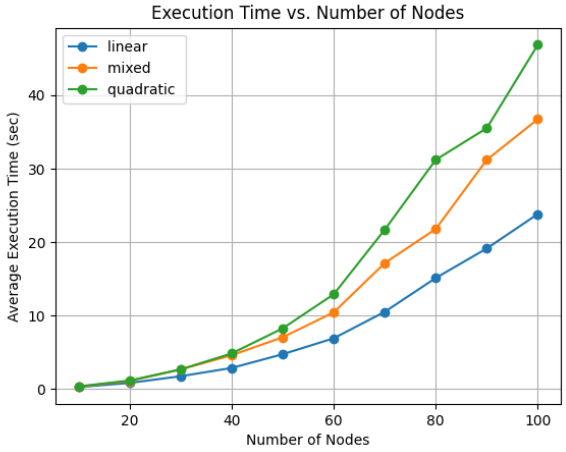

Figure 9: Average execution time vs. network size

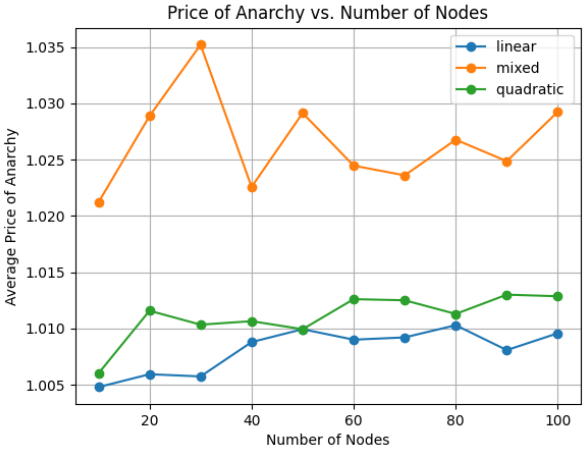

Figure 10: Average Price of Anarchy vs. network size

4.2.3 Discussion. Key observations can be made from the Figures 9, 10, 11, 12, and the performance of each mode.

1. Execution Time: Figure 9 shows that the overall execution time increases roughly polynomially with the network size in all modes. However, it should be noted that the linear mode consistently achieves lower execution times and less sharp increases compared to the quadratic and mixed modes. Since linear cost

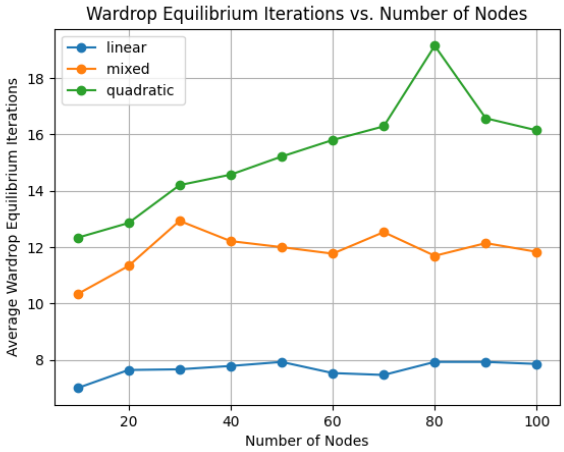

Figure 11: Average Wardrop Equilibrium iterations vs. network size

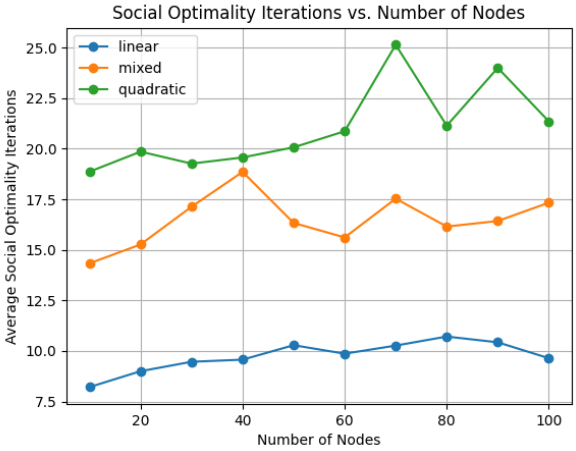

Figure 12: Average Social Optimality iterations vs. network size

functions are computationally simpler to integrate and evaluate than quadratic functions, their execution times are expected to be lower. The quadratic mode suffers additional overhead due to the increased complexity of the cost functions, which is reflected in their longer execution times. The mixed mode, which randomly selects between linear and quadratic cost functions with equal probability, shows the most significant variability in execution time. This is because, although the theoretical probability of choosing either function is 50 percent, the actual distribution between runs could vary from an exact split.

2. Price of Anarchy: Figure 10 presents the average PoA as a function of the network size for the three cost function modes. In general, the PoA values remain very close to one for all network sizes, indicating that the calculator's equilibrium flows are nearly socially optimal. The PoA is lowest for networks with linear cost functions, which agrees with the theory where inefficiency introduced by selfish routing is minimal for affine cost models. Despite this, when in quadratic mode, the PoA values are slightly higher than those in the linear case. This minor increase suggests that

the non-linearity of the cost functions introduces relatively modest inefficiencies and does not significantly disrupt the system's overall efficiency. However, the mixed mode, where each edge is randomly assigned a linear or quadratic cost function, produces the highest PoA values with much greater variability. This shows that the diverse cost function environment in the mixed mode makes some routes more attractive to individual drivers, thereby encouraging selfish routing behaviour. Regardless of the differences among cost modes, one notable observation is that the PoA does not show a monotonic trend concerning network size. Instead, the values oscillate slightly around a narrow band close to 1, where fluctuations are likely a consequence of the randomness in the network generation process, but could also be due to the occasional numerical challenges encountered by the SLSQP optimiser.

3. Convergence Behaviour (Iteration Count): The average number of iterations required for the SLSQP optimiser to converge is shown in Figures 11 and 12. From an overall perspective, the two plots reveal that computing equilibrium flows generally demands fewer iterations than computing social optimal flows. This is because the potential function (Equation 14) for equilibrium is defined with the integration of cost functions, which smooths out the gradient landscape and allows the optimiser to converge in fewer iterations [23]. On the other hand, the total cost function (Equation 15) does not use integration and is more sensitive to small changes in flow, thus requiring more iterations for convergence. In both figures, the linear mode has the lowest iteration count with an average of at most 9 iterations for equilibrium and 11 for Social Optimality across the entire range of node counts. In addition, the linear mode remains relatively flat in both figures, suggesting a stable and well-conditioned optimisation process. The quadratic mode is less flat and escalates slightly with network size, especially for Social Optimality, which peaks above 20 iterations around the higher node counts. This behaviour suggests that, as the network grows, the SLSQP optimisation has to deal with more edges that contain quadratic cost functions. Thus, the number of iterations to converge increases. The mixed mode occupies a position between linear and quadratic when averaged. This highlights that cost function types have the most significant effect on convergence.

4. Invalid Results and Reliability: Of a total of 450 experimental runs, 23 results were invalid (they failed to converge or returned a PoA below 1), which represents approximately 5.1% of all runs. These invalid outcomes were most commonly observed in certain instances of the mixed cost mode and in degenerate network configurations where the optimisation routine struggled to converge. This could be because the networks were randomly created with random cost functions that didn't agree with the constraints of the optimisation problem. These infrequent cases highlight potential numerical instabilities or edge cases that warrant further investigation. When calculating the average performance metrics, these invalid results were excluded to guarantee that the reported trends accurately reflect the typical behaviour of the calculator.

These findings suggest a trade-off: while linear cost functions yield a faster and more predictable performance, introducing nonlinearity may capture more realistic traffic dynamics at the expense of increased computational effort and occasional numerical instability.

4.3 Future Work

The development of this tool opens several promising avenues for further research and enhancement:

- **Algorithmic Improvements:** Future work will focus on further improving the current algorithm to handle more extensive and more complex networks with greater efficiency. This could be done by investigating hybrid optimisation techniques such as combining aspects of Frank-Wolfe, gradient projection, and bush-based methods to improve convergence speed.
- **User Studies:** Structured user studies should be planned to verify that the tool meets end users' practical needs. A sample of academic researchers and urban planners should be recruited, and feedback should be obtained through interviews, surveys, and usability testing. This would point out areas for improvement, simplify complex interactions, and refine the visual design to make the tool more intuitive and accessible.
- **Advanced Visualisation Techniques:** Enhancing the tool's visual component will remain a key priority. Therefore, future work should explore the integration of advanced visualisation techniques, such as animated flow maps and interactive heat maps, to better illustrate the evolution of network equilibria and congestion patterns. This will provide deeper insights into traffic dynamics and help users clearly interpret the effects of various network configurations.
- **Integration of Real-World Data:** Incorporating real-world data will be vital for validating and enriching the tool. An approach is to contact local councils and transportation agencies to gain access to the actual road network and traffic data sets. However, to highlight that Wardrop Equilibrium isn't just applicable to road traffic, other flow domains such as power line networks should be investigated. This integration of real-world data will bridge the gap between theoretical models and practical applications, ensuring that the tool remains relevant and impactful in real-life scenarios.

5 CONCLUSIONS

In this work, an interactive tool was implemented that integrates theoretical models of traffic flow with modern numerical optimisation and dynamic visualisation. The tool comprises a backend calculator with convex optimisation techniques and an intuitive frontend, which lets users build custom network configurations. As a result, the tool successfully computes and visualises the Wardrop Equilibrium, Social Optimality, and the Price of Anarchy in real time. An evaluation on networks, such as the Pigou and Braess networks, confirmed that the tool's calculated values align with established theoretical benchmarks. The promising performance and accuracy of the tool demonstrate its potential both as an academic research instrument and as a practical aid for traffic management and planning. Future work will focus on optimising the algorithm, refining the user interface through structured user studies, and integrating real-world data to enhance its applicability in different flow systems.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Dr. Yiannis Giannakopoulos, for his valuable comments and advice throughout the year.

REFERENCES

[1] 2024. *Aimsun Next User Manual.* https://docs.aimsun.com/next/24.0.1/UsersManual/StochasticRouteChoice.html Accessed: 2024-12-11.

[2] 2024. *PTV Visum User Manual.* https://cgi.ptvgroup.com/vision-help/VISUM_2025_ENG/Content/1_Benutzermodell%20IV/1_5_Gleichgewichtsumlegung.htm Accessed: 2024-12-11.

[3] 2024. *SUMO Documentation.* https://sumo.dlr.de/docs/index.html Accessed: 2024-12-11.

[4] Daron Acemoglu and Asu Ozdaglar. 2009. Applications of Game Theory to Networks. https://dspace.mit.edu/bitstream/handle/1721.1/119628/14-15j-fall-2009/contents/lecture-notes/index.htm Lecture notes for MIT course 6.207/14.15J, accessed December 11, 2024.

[5] Cynthia Barnhart, Niranjan Krishnan, and Pamela H. Vance. 2009. *Multicommodity Flow Problems.* Springer US, 2354–2362. https://doi.org/10.1007/978-0-387-74759-0_407

[6] Martin J. Beckmann, C. B. McGuire, and C. B. Winsten. 1956. *Studies in the Economics of Transportation.* Yale University Press. https://www.rand.org/pubs/research_memoranda/RM1488.html Published for the Cowles Commission for Research in Economics.

[7] Stephen D. Boyles, Nicholas E. Lownes, and Avinash Unnikrishnan. 2023. *Transportation Network Analysis.* Vol. 1. edition 0.91.

[8] The SciPy Community. 2024. SciPy: Open-source software for mathematics, science, and engineering. https://scipy.org/ Accessed: 2024-12-07.

[9] The SymPy Community. 2024. SymPy: A Python library for symbolic mathematics. https://www.sympy.org/ Accessed: 2024-12-07.

[10] José R. Correa and Nicolás E. Stier-Moses. 2011. Wardrop Equilibria. In *Encyclopedia of Operations Research and Management Science*, James J. Cochran (Ed.). Wiley. https://doi.org/10.1002/9780470400531.eorms0962

[11] Stella C. Dafermos and Frederick T. Sparrow. 1969. Traffic Assignment Problem for a General Network. *Journal of Research of the National Bureau of Standards, Section B: Mathematical Sciences* 73B, 2 (1969), 91–118. https://archive.org/details/jresv73Bn2p91

[12] React Flow Developers. 2024. React Flow: The library for interactive node-based visualizations. https://reactflow.dev/ Accessed: 2024-12-07.

[13] The NumPy Developers. 2024. NumPy: The fundamental package for numerical computation in Python. https://numpy.org/ Accessed: 2024-12-07.

[14] David Easley and Jon Kleinberg. 2010. *Modeling Network Traffic Using Game Theory.* Cambridge University Press, 207–224. https://doi.org/10.1017/CBO9780511761942.009

[15] Sean Fleming. 2019. Traffic congestion cost the US economy nearly $87 billion in 2018. https://www.weforum.org/stories/2019/03/traffic-congestion-cost-the-us-economy-nearly-87-billion-in-2018/ Accessed 14-November-2024.

[16] Python Software Foundation. 2024. Python: The programming language that lets you work quickly and integrate systems effectively. https://www.python.org/ Accessed: 2024-12-07.

[17] Alexander Galkin and Anton Sysoev. 2021. *Controlling Traffic Flows in Intelligent Transportation System.* Springer International Publishing, 91–101. https://doi.org/10.1007/978-3-030-63563-3_8

[18] Sarah Hatch. 2023. What Is Grid Congestion, and How Does It Impact Prices? Yes Energy Blog. https://blog.yesenergy.com/yeblog/what-is-electricity-grid-congestion Accessed 14-November-2024.

[19] Kai Jungel, Dario Paccagnan, Axel Parmentier, and Maximilian Schiffer. 2024. WardropNet: Traffic Flow Predictions via Equilibrium-Augmented Learning. *arXiv preprint arXiv:2410.06656* (2024). https://doi.org/10.48550/arXiv.2410.06656

[20] Lasse Kliemann and Anand Srivastav. 2009. Models of Non-atomic Congestion Games – From Unicast to Multicast Routing. In *Algorithmics of Large and Complex Networks.* Lecture Notes in Computer Science, Vol. 5515. Springer, 292–318. https://doi.org/10.1007/978-3-642-02094-0_14

[21] Elias Koutsoupias and Christos Papadimitriou. 1999. Worst-case equilibria. In *Proceedings of the 16th Annual Conference on Theoretical Aspects of Computer Science (STACS'99).* Springer-Verlag, 404–413. https://doi.org/10.1007/3-540-49116-3_38

[22] Xin Lin, Chris M. J. Tampère, and Stef Proost. 2020. Optimizing Traffic System Performance with Environmental Constraints: Tolls and/or Additional Delays. *Networks and Spatial Economics* 20, 1 (2020), 137–177. https://doi.org/10.1007/s11067-019-09471-8

[23] Yu. Nesterov. 2005. Smooth minimization of non-smooth functions. *Mathematical Programming* 103, 1 (May 2005), 127–152. https://doi.org/10.1007/s10107-004-0552-5

[24] Rostislav Netek, Tomas Pour, and Renata Slezakova. 2018. Implementation of Heat Maps in Geographical Information System—Exploratory Study on Traffic Accident Data. *Open Geosciences* 10 (2018), 367–384. https://doi.org/10.1515/geo-2018-0029

[25] Pallets. 2023. Flask: A lightweight WSGI web application framework. https://flask.palletsprojects.com/ Accessed: 2024-12-07.

[26] Christos Papadimitriou. 2001. Algorithms, games, and the internet. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing (STOC '01).* Association for Computing Machinery, 749–753. https://doi.org/10.1145/380752.380883

[27] Doantam Phan, Ling Xiao, Ron Yeh, Pat Hanrahan, and Terry Winograd. 2005. Flow Map Layout. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.* IEEE, 219–224. https://doi.org/10.1109/INFVIS.2005.1532150

[28] Meta Platforms. 2024. React: A JavaScript library for building user interfaces. https://reactjs.org/ Accessed: 2024-12-07.

[29] Tim Roughgarden. 2002. The price of anarchy is independent of the network topology. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing (STOC '02').* Association for Computing Machinery, 428–437. https://doi.org/10.1145/509907.509971

[30] Tim Roughgarden. 2007. Inefficiency of Equilibria as a Design Metric. In *Algorithmic Game Theory*, Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani (Eds.). Cambridge University Press, Chapter 17, 443–460. https://doi.org/10.1017/CBO9780511800481.019

[31] Tim Roughgarden. 2007. Routing Games. In *Algorithmic Game Theory*, Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani (Eds.). Cambridge University Press, Chapter 18, 461–486. https://doi.org/10.1017/CBO9780511800481.020

[32] Siim Sild. 2024. Supply Chain Issues: Port Congestion. https://fractory.com/port-congestion-explained/ Accessed 14-November-2024.

[33] Yan tao Tian and Ping ping Xiao. 2010. A review and prospect on the network congestion control. *The Journal of China Universities of Posts and Telecommunications* 17 (2010), 84–90. https://doi.org/10.1016/S1005-8885(09)60601-4

[34] J. G. Wardrop. 1952. Some Theoretical Aspects of Road Traffic Research. *Proceedings of the Institution of Civil Engineers* 1, 3 (1952), 325–362. https://doi.org/10.1680/ipeds.1952.11259

[35] Xin Yang, Shuai Li, Yong Zhang, Wanchao Su, Mingyue Zhang, Guozhen Tan, Qiang Zhang, Dongsheng Zhou, and Xiaopeng Wei. 2017. Interactive traffic simulation model with learned local parameters. *Multimedia Tools and Applications* 76, 7 (2017), 9503–9516. https://doi.org/10.1007/s11042-016-3560-6

[36] Vadim Zverovich. 2021. Traffic Networks: Wardrop Equilibrium and Braess' Paradox. In *Modern Applications of Graph Theory.* Oxford University Press. https://doi.org/10.1093/oso/9780198856740.003.0002

A EQUATIONS

A.1 Wardrop's equations for the total flow constraint and distribution[34]

A.1.1 Wardrop Equilibrium. Wardrop's first principle also provided insights into how traffic flow q_i is distributed across a subset of j routes when journey times are equal. This can be described as:

$$q_i = p_i \left(1 - \frac{b_i}{t}\right), \quad \forall i = 1, 2, \ldots, j,$$

If $b_i > t$, then the flow $q_i = 0$.

This equation calculates the amount of traffic on each route as a function of its baseline travel time (b_i), capacity (p_i), and the shared travel time (t) across all routes in use. This equation shows that routes with higher capacities attract more flow, while routes with higher baseline travel times attract less flow.

Additionally, Wardrop emphasised the **Total Flow Constraint**, which ensures that traffic flow matches total demand:

$$Q = \sum_{i=1}^{j} q_i = \sum_{i=1}^{j} p_i \left(1 - \frac{b_i}{t}\right),$$

Where Q is the total traffic demand between the origin and destination. This constraint allows the equilibrium travel time t to be computed, subsequently determining the flow distribution on each route.

A.1.2 Social Optimality. Wardrop highlighted how the **flow distribution for Social Optimality** differs from Wardrop Equilibrium. It can be derived as:

$$q_i = p_i \left(1 - \sqrt{\frac{b_i}{\epsilon}}\right), \quad \forall i = 1, 2, \ldots, j.$$

The baseline travel time (b_i) determines that routes with higher baseline travel times attract less flow. The capacity (p_i) indicates that routes with higher capacities attract more flow. The marginal travel time (ϵ) ensures that flow is distributed to equalise marginal costs across all used routes.

The **Total Flow Constraint** for Principle 2 is therefore different from Principle 1:

$$Q = \sum_{i=1}^{j} q_i = \sum_{i=1}^{j} p_i - \sum_{i=1}^{j} p_i \sqrt{\frac{b_i}{\epsilon}}.$$

This constraint guarantees that traffic flow is distributed to meet overall demand while adhering to the conditions of Social Optimality.

A.2 Additional PoA bounds

Pigou Bound and Variational Inequalities. To analyse the worst-case inefficiency of selfish routing, Roughgarden, [31] introduced the Pigou Bound:

$$\alpha(C) = \sup_{c \in C} \sup_{x, r>0} \frac{r \cdot c(r)}{x \cdot c(x) + (r - x)c(r)}. \tag{20}$$

This formulation provides a natural lower bound on PoA, particularly for networks with affine or polynomial cost functions. Roughgarden, [31] also uses variational inequalities to demonstrate that equilibrium flows satisfy:

$$\sum_{e \in E} c_e(f_e) f_e \leq \sum_{e \in E} c_e(f_e) f_e^*, \tag{21}$$

This inequality states that the total cost under Wardrop Equilibrium is bounded relative to the cost at Social Optimality.

B FRONTEND

B.1 Results Modal

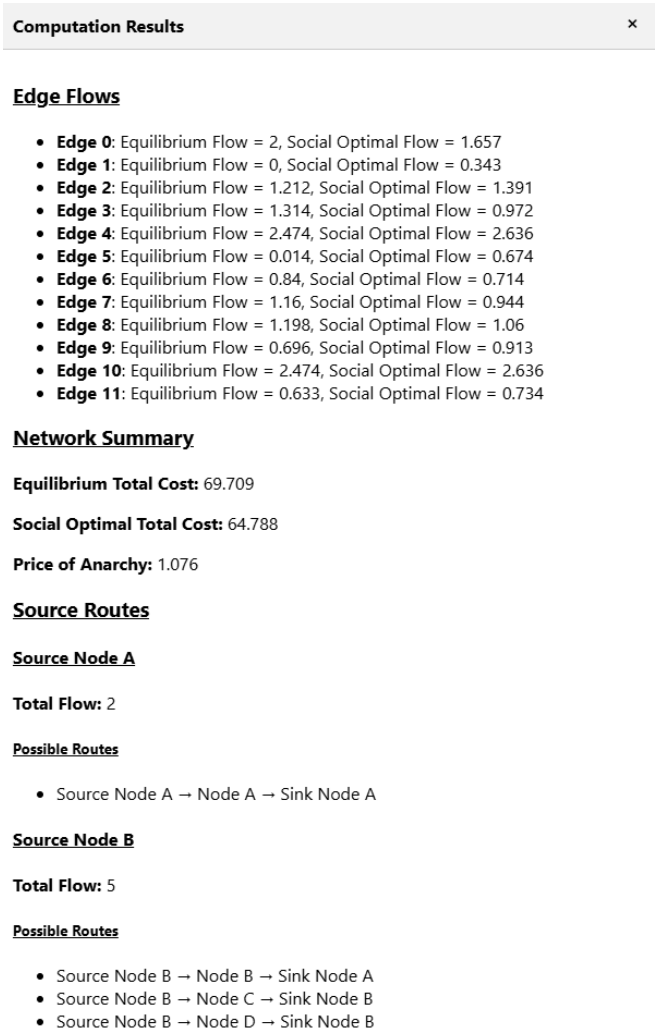

Figure 13: Frontend Display of Computation Results. This screenshot shows the computed edge flows, total travel costs, and the Price of Anarchy (PoA) after the network analysis is performed