Equilibria in network congestion games

Bachelor's Project Applied Mathematics

July 2023

Student: R. de Jong

First supervisor: Prof. dr. J. G. Peypouquet

Second supervisor: Prof. dr. S. Trenn

Abstract

Network congestion games are a class of games that have been extensively studied from both a game theoretic and mathematical point of view. We develop a mathematical basis for analysing non-atomic instances of network congestion games, with attention to the proofs behind basic concepts. We furthermore establish an upper bound on the inefficiency of selfish equilibria. Having investigated the mathematical theory for this class of games, we analyse a standard numerical method to compute equilibrium flows. A conjugate based extension is discussed to remedy slow convergence. The methods are compared by analysis on a model of the city of Groningen.

Contents

1 Introduction

In this thesis, we develop a broad basis in the theory of network congestion games. As will be formalized later, network congestion games deal with transport problems. Typical examples of this include urban planning uses, like analysing road congestion. For example, the framework has been used to analyse the traffic network in the city of Winnipeg [Florian and Nguyen, 1976]. However, this can also be extended to other problems like scheduling problems [Zhou and Yang, 2019] or more computer science focused applications like data sharing [Liu and Wu, 2008, Law et al., 2012].

These kinds of problems are somewhat "new" compared to other mathematical problems. One of the first mentions of the problem in a more economical setting was made by Pigou in his book on economic welfare [Pigou, 1920]. This example will be discussed in section 4. A very influential contribution in this field came with the paper by Beckmann, McGuire and Winsten [Beckmann et al., 1955]. In their work, it was proven that the user equilibrium model can indeed be formulated in terms of a mathematical optimization problem. We will revisit their contribution in section 3. After this, Rosenthal proved in 1973 that network congestion-like games always had a pure Nash equilibrium [Rosenthal, 1973]. Another main source of this thesis is the work of Roughgarden [Roughgarden, 2005], which will be extensively used to prove inefficiency bounds for selfish behaviour.

This paper aims to tackle three fundamental ideas in this broad field. Some preliminaries relating to basic convex analysis, optimization and algorithms can be found in appendix A. As a first problem, we investigate how to compute equilibria in network congestion games. In section 2 we discuss both selfish equilibria and user equilibria. A fundamental question in optimization theory is the existence and uniqueness of problems, which is discussed in the context of our topic as well. Further characterizations of optimal flows in terms of marginal cost functions are discussed and their connection with Beckmann's formulation of the optimization problem to find user equilibrium flows. After establishing how to compute optimal and selfish flows, we ask ourselves if there is a bound on how bad these selfish flows can be compared to the optimal case. In section 3 we prove that such a bound does indeed exist when we consider polynomial cost functions. Finally, we investigate how to compute equilibria numerically in section 4. As with any mathematical problem, when scaling to larger problems it becomes unfeasible to perform all computations manually. We discuss the Frank-Wolfe algorithm and a conjugate adaptation to speed up convergence. In section 5 we apply these methods to a model of the road network of the city of Groningen. We compare the speed of convergence of both of these algorithms, and briefly discuss the results obtained.

Acknowledgements

I would like to thank my supervisors, professor Peypouquet and Trenn for their supervision. Their questions and ideas have contributed to this thesis a lot and sparked more questions during and after working on this thesis. I would also like to thank my friends and family for supporting me. Being able to talk about my theories and getting both math related and non-math related advice has helped a lot in the process of making this thesis.

2 Computation and characterizations of equilibria

A natural question to ask is if it is possible to determine optimal flows for an instance $I = (G, r, c)$. A simple idea might be to compute a flow that minimizes the cost equation $(2.1.7)$. While this gives a mathematically valid answer to the question, this situation will practically never occur in real life. People will never perfectly adhere to a prescribed traffic route, if they believe it is not in their own best interest. A more realistic situation might we where users take the path that is best for them personally. In his paper "Some Theoretical Aspects of Road Traffic Research" [Wardrop, 1952] Wardrop formulated both these principles as follows;

1. The journey times on all paths are equal, and less than what any single vehicle would experience if switching to another route,

2. The average (and thus total system cost) journal route cost is at a minimum

We refer to these situations as *Wardrop's first and second principle* respectively. In this section we look at both these notions and develop the mathematical theory behind them. Before we do this however, we go over some basic definitions that we will use throughout the paper.

2.1 Basic definitions

In this section, we build up the basic definitions used in the analysis of congestion games. The definitions used here stem from [Roughgarden, 2005] and [Diestel, 2017]. We encode our network as a directed graph $G = (V, E)$. Here V will be the set of vertices, and $E \subseteq V \times V$ is the set of directed edges between the vertices. We assume that the graph is connected in the sense that any vertex can be reached from another vertex. On each edge, we assign a nonnegative, continuous and nondecreasing cost function $c_e(\cdot) : \mathbb{R} \to \mathbb{R}_{\geq 0}$. Moreover, we assume that the cost on an edge e is only determined by the traffic on the edge e:

$$\frac{\partial c_e}{\partial x_{\tilde{e}}} = 0 \text{ and } \frac{\partial c_e}{\partial x_e} \geq 0, \tag{2.1.1}$$

for $e \in E$ and $e \neq \tilde{e} \in E$. These conditions give a physical intuition to these functions; The more traffic on a road, the higher the cost on the corresponding edge (one could see travel time as a cost) and the flow increases continuously with traffic increase. Of importance in routing games are origin-destination pairs.

Definition 2.1 (Origin-Destination Pairs). Let $G = (V, E)$ be a connected, directed graph encoding the traffic network. The set of *origin-destination pairs (OD-pairs)* $\mathcal{D}$ is an indexed set consisting of ordered pairs $(a, b) \in V \times V$. That is:

$$\mathcal{D} = \{d_1, \dots, d_i, \dots, d_N\} = \{(s_1, t_1), \dots, (s_i, t_i), \dots (s_N, t_N)\}, \tag{2.1.2}$$

where $N \in \left[|V|^2\right]$ is the number of origin destination pairs. The sequence $(s_i)_{i \leq N}$ is the sequence of origin vertices and $(t_i)_{i \leq N}$ is the sequence of destination vertices.

We also refer to these pairs as *commodities*. If we have a single origin-destination pair we call the network a *single commodity network*, while we call the network a *multicommodity network* if there is multiple origin-destination pairs. As a physical interpretation of these origin-destination pairs, we can imagine that the graph G encodes the road network of a city, where each node in V represents either a source of traffic, an intersection of roads or a goal for traffic to reach. One possible origin destination pair could then be a neighbourhood and a shopping centre.

This definition also gives merit to our assumption that the graph G is connected. Suppose that G was disconnected. Then, we can have two cases. First, it could be that all the OD-pairs are in the same connected component. In this case, the other part of the graph does not contribute to the problem. Thus it suffices to consider the graph $\tilde{G} = G \setminus D$ where D is the disconnected component that did not contain any origin or destination nodes. In the other case where some of the pairs are not in the same connected component, the problem would be impossible to solve. More precisely, suppose that $\mathcal{D} = \{d_1\} = \{(s_1, t_1)\}$ where s_1 and t_1 are not in the same connected component. Then, whatever nonzero traffic rate we assign it will be impossible to solve the problem since there is no path between s_1 and t_1.

Definition 2.2 (Traffic rate). Let $G = (V, E)$ be a connected directed graph. On this graph, define a set of origin-destination pairs $\mathcal{D}$. The *traffic rate* r_i corresponding to the OD-pair $d_i \in \mathcal{D}$ is a positive real number which represents the amount of traffic to be diverted from s_i to t_i.

For ease of notation, we usually refer to the traffic rates as a vector $r \in \mathbb{R}^N$ which has positive entries r_i corresponding to the traffic rate of the i-th origin destination pair. To connect this back to the previous example, the rate demand could for example be the number of people that live in the previously mentioned neighbourhood that shop at this shopping center.

Of course, when talking about routing over a network a natural way of formalizing the possible routes belonging to a commodity is that of a path on a graph.

Definition 2.3 (Path set). A *directed path* on a directed graph $G = (V, E)$ is a graph $H \neq \emptyset$ with distinct vertices $x_0, \dots, x_k$ and edges $e_0, \dots e_{k-1}$ such that e_i is an edge between x_i and x_{i+1}. For a network G, with commodities $\mathcal{D}$, denote $\mathcal{P}_i$ (which we assume to be non-empty) as the set of directed paths from s_i to t_i. The *path set* $\mathcal{P}$ is then the union of these sets,

$$\mathcal{P} = \bigcup_{i=1}^{N} \mathcal{P}_i. \tag{2.1.3}$$

Next, we need to define a way to measure how much traffic flows over a given path.

Definition 2.4 ((Feasible) flow). A *flow* is a nonnegative vector f with entries indexed by paths $P \in \mathcal{P}$, indicating how much traffic gets routed over any possible path on the network G. A flow is called *feasible* if it routes the prescribed amount of traffic for each commodity. That is, a flow f is feasible if for all i

$$\sum_{P \in \mathcal{P}_i} f_P = r_i. \tag{2.1.4}$$

This definition can be naturally extended to a *flow on edges*. Namely, the flow on an edge $e \in E$ is given by

$$f_e = \sum_{P \in \mathcal{P} \text{ s.t. } e \in P} f_P. \tag{2.1.5}$$

Using these definitions, we can finally formalize the notion of a routing game.

Definition 2.5 (Instance of a routing game). An *instance of a routing game* is a triple $I = (G, r, c)$ where $G = (V, E)$ is a connected directed graph encoding the network and r and c are a vector of traffic rates and a vector of cost functions respectively.

Example 2.6 (Example of instance). Consider the following network G as in figure 1, with origin destination pair $d_1 = (s_1, t_1)$ and $r_1 = 1$. The triple $I = (G, r_1, c)$, with $c = (2x + 1, 5x + 3, 0.5x + 1, 4x + 1, x + 2)$ forms an instance of a routing game.

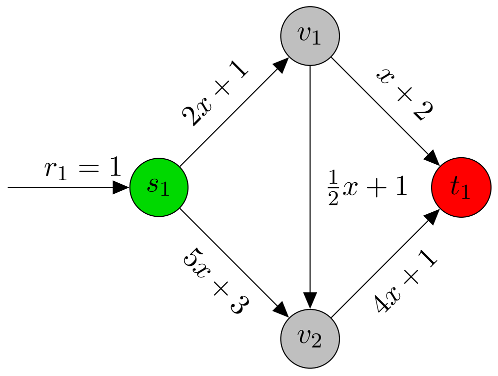


Figure 1: Example of an instance of a routing game

The set of possible paths $\mathcal{P}$ will be the set of paths between s_1 and t_1. Therefore, we have that

$$\mathcal{P} = \{(s_1, v_1, t_1), (s_1, v_2, t_1), (s_1, v_1, v_2, t_1)\}.$$

A feasible flow could then be $f = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ or $f = (1, 0, 0)$. △

Next, we provide a few notions that have to do with the cost of flows.

Definition 2.7 (Cost of path and cost of flow). The *cost of a path* corresponding to a flow f is given by

$$c_P(f) = \sum_{e \in P} c_e(f_e), \tag{2.1.6}$$

whereas the *cost of a flow* is given by

$$C(f) = \sum_{P \in \mathcal{P}} c_P(f) f_P. \tag{2.1.7}$$

The cost of a flow is also referred to as social cost since it can be seen as the total cost that everyone on the network experiences [Díaz et al., 2017].

We note that we can also write the cost of a flow in terms of flow on edges.

Proposition 2.8. *The cost of a flow can be written as:*

$$C(f) = \sum_{e \in E} c_e(f_e) f_e. \tag{2.1.8}$$

Proof. The proposition follows quite easily using the fact that we can swap finite sums, keeping in mind the indices:

$$\begin{aligned} C(f) &= \sum_{P \in \mathcal{P}} c_P(f) f_P = \sum_{P \in \mathcal{P}} \left(\sum_{e \in P} f_P \cdot c_e(f_e) \right) \\ &= \sum_{e \in E} c_e(f_e) \left(\sum_{\substack{P \in \mathcal{P} \\ \text{s.t. } e \in P}} f_P \right) = \sum_{e \in E} c_e(f_e) f_e, \end{aligned}$$

where the second equality follows from definition 2.4 and the final equality follows from equation 2.1.5. □

Example 2.9. Consider again the instance (G, r_1, c) from example 2.6. Consider the feasible flow given by the flow $f = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. In figure 2 we give the network with on edge specified the flow on that edge under this flow vector. We can then compute the cost of this flow as

$$\begin{aligned} C(f) = \sum_{e \in E} c_e(f_e) f_e &= \frac{2}{3}\left(2 \cdot \frac{2}{3} + 1\right) + \frac{1}{3}\left(\frac{1}{3} + 2\right) + \frac{1}{3}\left(\frac{1}{2} \cdot \frac{1}{3} + 1\right) \\ &+ \frac{1}{3}\left(5 \cdot \frac{1}{3} + 3\right) + \frac{2}{3}\left(4 \cdot \frac{2}{3} + 1\right) = \frac{121}{18}. \end{aligned}$$

Instead, taking $f = (1, 0, 0)$ we simply get

$$C(f) = \sum_{e \in E} c_e(f_e) f_e = 1 \cdot (2 \cdot 1 + 1) + 1 \cdot (1 + 2) = 6.$$

△

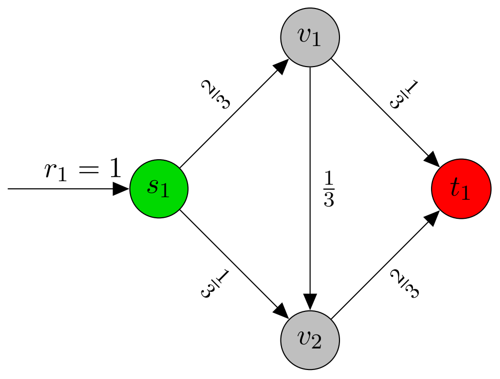


Figure 2: Flows on edges with $f = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$

Another notion will we use is that of a marginal cost function.

Definition 2.10 (Marginal cost function). Let $c_e(f_e)$ be a cost function for an edge $e \in E$. Then, the *marginal cost function* is defined as:

$$c_e^*(x) = \frac{\mathrm{d}}{\mathrm{d}x}\left(c_e(x) \cdot x\right) = c_e'(x) \cdot x + c_e(x). \tag{2.1.9}$$

The notion of a marginal cost has a nice interpretation. Namely, consider a case where we slightly increase flow of traffic on an edge. In this case, the marginal cost will give us the cost experienced by this traffic and also the increase in congestion caused by this traffic.

We finally note that here we consider *nonatomic* congestion games. That is, we consider a game with an infinite number of “players” that all control an infinitesimal amount of traffic. In the atomic case we have a finite number of players that all control larger amount of traffic. Practically speaking, in a nonatomic game we can freely split traffic in whatever way we desire, since each player controls a negligible amount of traffic.

2.2 Wardrop’s second principle: System optimal equilibria

We first examine the second case, which is also referred to as a *system optimal* or SO flow assignment. In this case, we strive for a distribution of flow that is more desirable from a societal aspect [Angelelli et al., 2021].

Given an instance (G, r, c) with N origin-destination pairs, we wish to minimize the social cost. That is, we want to find f^* so that:

$$C(f^*) = \min_f C(f) = \sum_{P \in \mathcal{P}} c_P(f) f_P. \tag{2.2.1}$$

This equation does not describe the full problem however. Recalling the previous section, we require that each flow is non-negative and we require that correct amount of traffic is routed between each origin-destination pair. Therefore, the full problem will be given by:

$$\begin{gathered}\min_f C(f) = \sum_{P\in\mathcal{P}} c_P(f) f_P \\ \text{subject to: } f_P \geq 0 \quad \forall P \in \mathcal{P}, \\ \sum_{P\in\mathcal{P}_i} f_P = r_i \quad \forall\, 0 \leq i \leq N.\end{gathered} \tag{SO}$$

We first investigate the existence of a solution to this problem. For this, we can use Weierstrass' Theorem as given in theorem A.12. Note that we can equivalently write the problem (SO) as a minimization problem over a constraint set:

$$\min_{f\in\mathcal{F}} C(f),$$

where $\mathcal{F}$ is the set of *feasible flows*:

$$\mathcal{F} = \left\{ f \in \mathbb{R}^{|\mathcal{P}|}, \text{ so that } f_P \geq 0 \quad \forall P \in \mathcal{P}, \sum_{P\in\mathcal{P}_i} f_P = r_i \quad \forall\, 0 \leq i \leq N \right\}. \tag{2.2.2}$$

It suffices to argue that the set $\mathcal{F}$ is compact, since the function C is continuous as long as each c_e is. By Weierstrass's theorem as in theorem A.12 the problem will then admit a solution.
Note first the set $\mathcal{F}$ is bounded. For any vector $f = (f_{P_1}, \ldots, f_{P_K})$ with $K = |\mathcal{P}|$, we have that $f_{P_j} \leq r_i$ with $P_j \in \mathcal{P}_i$. Furthermore, each entry can be bounded from below by zero. Therefore,

$$0 \leq ||f||_2 \leq \sqrt{\sum_{i\leq N} r_i^2}.$$

The last inequality gives the worst case scenario where we route all traffic over a single path for each OD-pair. If we were to split traffic, say equally over two paths, the norm would become smaller, since $\sqrt{\left(\frac{1}{2}r_i\right)^2 + \left(\frac{1}{2}r_i\right)^2} = \sqrt{\frac{1}{2}r_i^2} \leq \sqrt{r_i^2}$. Moreover, the set is closed since it is its own closure. The boundary of the set is formed by $f_p = 0$ for all $P \in \mathcal{P}$ and $f_P = r_i$ for all $P \in \mathcal{P}_i$ and for all $i \leq N$. It is possible for f_P to be zero or to be equal to r_i, and thus these boundaries are contained in the set $\mathcal{F}$. Therefore, the set is closed. By combining the fact that $F \subset \mathbb{R}^{|\mathcal{P}|}$ and the fact that $\mathcal{F}$ is closed and bounded, we know the set $\mathcal{F}$ is compact. Since $\mathcal{F}$ is compact, the problem (SO) has at least one solution.

Example 2.11. Consider the network as given in figure 3. On this network, we are given a single OD-pair (s_1, t_1), with an assigned traffic rate $r_1 = 1$. The problem (SO) can then be formulated as:

$$\begin{gathered}\min_f C(f) = f_1 + f_2 \\ \text{subject to: } f_1, f_2 \geq 0 \\ f_1 + f_2 = 1\end{gathered}$$

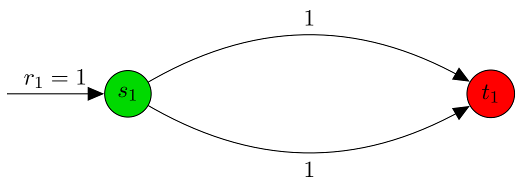


Figure 3: Instance where uniqueness cannot be guaranteed

where we denote f_1 as the flow on the top edge and f_2 as the flow on the bottom edge. Clearly, this implies $C(f) = f_1 \cdot 1 + f_2 \cdot 1 = 1$ by using the constraint that $f_1 + f_2 = 1$. Thus, we can assign any flow and we will always find that $C(f) = 1$. As an example both the flow $f = (1, 0)$ and the flow $f = (\frac{1}{3}, \frac{2}{3})$ would lead to system optimal solutions, but are not the same flow. $\triangle$

This behaviour is to be expected. First note the set of feasible flows is convex. Take two feasible flows $f, \tilde{f}$. Then, any convex combination is also feasible. Namely,

$$\sum_{P \in \mathcal{P}_i} (\lambda f_P + (1-\lambda)\tilde{f}_P) = \lambda \sum_{P \in \mathcal{P}_i} f_P + (1-\lambda) \sum_{P \in \mathcal{P}_i} \tilde{f}_P = \lambda r_i + (1-\lambda) r_i = r_i.$$

Moreover, the new flow will only consist of positive entries, since $\lambda \in [0, 1]$ and thus the signs are not altered when making a convex combination. The cost function we obtained in the example is however not strictly convex, but only convex. Therefore, by proposition A.13 we can only guarantee a unique local minimum. Summing up this discussion, we get the following proposition.

Proposition 2.12. *The problem* (SO) *has a solution. If the function $c_e(f_e)f_e$ is convex for all $e \in E$ and if $f, \tilde{f}$ are two solutions then we have that $C(f) = C(\tilde{f})$. Moreover, if the function $c_e(f_e)f_e$ is strictly convex then the problem has a unique global minimum.*

The convexity requirement is for example satisfied for the class of polynomials with non-negative constants, as we will see in our discussion on the price of anarchy.
A nice characterization of an optimal flow can be formulated as a variational inequality involving the marginal cost functions [Nisan et al., 2007, Chapter 18].

Proposition 2.13. *Let I be an instance of a routing game with commodities $\mathcal{D} = \{d_1, d_2, \dots, d_N\}$, so that each function $c_e(f_e)f_e$ is convex. Then, the following are equivalent:*

1. *The flow f^* is optimal;*

2. *For every commodity d_i and every pair of paths $P, \tilde{P} \in \mathcal{P}_i$ with $f^*_P > 0$: $c^*_P(f^*) \leq c^*_{\tilde{P}}(f^*)$*

Proof. Before proving the statement, we take a closer look at the problem (SO).To find the system optimum, we solve the problem

$$\min_f C(f) = \sum_{P\in\mathcal{P}} c_P(f) f_P$$

$$\text{subject to: } g_P(f) = -f_P \leq 0 \quad \forall P \in \mathcal{P},$$

$$h_i(f) = \sum_{P\in\mathcal{P}_i} f_P - r_i = 0 \quad \forall 1 \leq i \leq N.$$

Therefore, the Lagrangian function related to this problem is:

$$\begin{aligned} L(f,\lambda,\mu) &= \sum_{P\in\mathcal{P}} c_P(f) f_P + \sum_{i=1}^{N} \lambda_i h_i(f) + \sum_{P\in\mathcal{P}} -\mu_P f_P \\ &= \sum_{e\in E} c_e(f_e) f_e + \sum_{i=1}^{N} \lambda_i h_i(f) + \sum_{P\in\mathcal{P}} -\mu_P f_P. \end{aligned}$$

In particular we see that h_i and g_P are linear functions in f and by assumption the functions $c_e(f_e)f_e$ are convex. By proposition A.16 a feasible flow f^* is then optimal if and only if it satisfies the KKT conditions. We compute the gradients of the functions C, g_P and h_i. First, by the chain rule we have that:

$$(\nabla C(f))_P = \sum_{e\in E} \frac{\partial C}{\partial f_e} \frac{\partial f_e}{\partial f_P}.$$

By using the relation between an edge flow and a path flow as in equation $(2.1.5)$:

$$\frac{\partial f_e}{\partial f_P} = \begin{cases} 1 & \text{if } e \in P \\ 0 & \text{if } e \notin P. \end{cases}$$

Therefore, the sum collapses to just the edges in our path P. The gradient of C then becomes:

$$(\nabla C(f))_P = \sum_{e\in P} \frac{\partial C}{\partial f_e} = \sum_{e\in P} \frac{\mathrm{d}}{\mathrm{d}f_e} \left(c_e(f_e) f_e\right) = \sum_{e\in P} c_e^*(f_e) := c_P^*(f).$$

For the function $h_i(f)$ we have that:

$$(\nabla h_i)_P = \begin{cases} 1 & \text{if } P \in \mathcal{P}_i \\ 0 & \text{if } P \notin \mathcal{P}_i, \end{cases}$$

and finally for $g_P(f)$ we compute:

$$(\nabla g_P)_{\tilde{P}} = \begin{cases} -1 & \text{if } P = \tilde{P} \\ 0 & \text{if } P \neq \tilde{P}, \end{cases}$$

for any $\tilde{P} \in \mathcal{P}$.

$(1 \implies 2)$ Let f^* be optimal. Therefore, there exist Lagrange multiplier vectors μ^* and λ^* so that

$$\nabla C(f^*) + \sum_{i=1}^{N} \lambda_i^* \nabla h_i(f^*) + \sum_{P \in \mathcal{P}} \mu_P^* \nabla g_P(f^*) = 0,$$

with $\mu_P \geq 0$ and $-f_P \mu_P = 0$. Consider a path $P \in \mathcal{P}_i$ so that $f_P^* > 0$. By the condition above this implies that $\mu_P = 0$. For this P we must then have that:

$$c_P^*(f) + \lambda_i^* = 0 \iff c_P^*(f) = -\lambda_i^*.$$

For any other path $\tilde{P} \in \mathcal{P}_i$, we must have that:

$$c_{\tilde{P}}^*(f^*) + \lambda_i^* - \mu_{\tilde{P}}^* = 0.$$

But then,

$$c_{\tilde{P}}^*(f^*) = c_P^*(f^*) + \mu_{\tilde{P}}^* \geq c_P^*(f^*),$$

since $\mu_{\tilde{P}}^* \geq 0$.

$(2 \implies 1)$ Suppose now that for every commodity d_i and every pair of paths $P, \tilde{P} \in \mathcal{P}_i$ with $f_{P_1} > 0$ we have that $c_P^*(f^*) \leq c_{\tilde{P}}^*(f^*)$. It then suffices to find Lagrange multiplier vectors λ^*, μ^* so that:

$$\nabla_f L(f^*, \lambda^* \mu^*) = 0$$
$$\mu_P^* g_P(f^*) = 0.$$

Let $P \in P_i$ be so that $f_P > 0$. Drawing inspiration from the previous implication, for any path $\tilde{P} \in \mathcal{P}_i$ we set:

$$\lambda_i^* = c_P^*(f^*), \quad \mu_{\tilde{P}}^* = c_P^*(f^*) - c_{\tilde{P}}^*(f^*).$$

Then, by the same computation we have that:

$$(\nabla_f L(f^*, \lambda^* \mu^*))_{\tilde{P}} = c_{\tilde{P}}^*(f^*) - c_P^*(f^*) + c_P^*(f^*) - c_{\tilde{P}}^*(f^*) = 0.$$

It remains to check that the multipliers μ are non-negative and $\mu_{\tilde{P}} g_{\tilde{P}}(f^*) = 0$. The non-negativity of μ is guaranteed by the assumption that $c_P^*(f^*) \leq c_{\tilde{P}}^*$ whenever $f_P > 0$. For the other condition, note that if $\tilde{P} = P$ we have $\mu_{\tilde{P}}^* = 0$. If this does not hold, we distinguish two possibilities:

1. If $f_{\tilde{P}} = 0$, then $\mu_{\tilde{P}}^* g_{\tilde{P}}(f^*) = -\mu_{\tilde{P}}^* f_{\tilde{P}}^* = 0$;
2. If $f_{\tilde{P}} \neq 0$, by assumption we find that $c_P^*(f^*) = c_{\tilde{P}}^*(f^*)$. Specifically, we have that $c_P^*(f^*) \leq c_{\tilde{P}}^*(f^*)$ and $c_P^*(f^*) \geq c_{\tilde{P}}^*(f^*)$ by using the fact that for every pair of paths $P, \tilde{P}$ with $f_P > 0$ we have $c_{\tilde{P}}^*(f^*) \leq c_P^*(f^*) \leq c_{\tilde{P}}^*(f^*)$. But then, we have that $\mu_{\tilde{P}} f_{\tilde{P}}^* = (c_P^*(f^*) - c_{\tilde{P}}^*(f^*)) f_{\tilde{P}}^* = 0$

Therefore, these choices of λ^*, μ^* satisfy the KKT conditions and therefore f^* is optimal. $\square$

2.3 User equilibria

While the previous section gives a good mathematical way of analysing the problem it certainly does not give a satisfying way to look at this problem. Clearly, it is impossible to assign all traffic over certain routes in real life. Instead it is more realistic to look at Wardrop's first principle. Namely, that the journey times on all paths are equal, and less than what a single vehicle would experience if switching to another route. Formalizing this mathematically leads to the following definition [Cominetti et al., 2012].

Definition 2.14. Let I be an instance of a routing game and let f be a feasible flow. The flow f satisfies Wardrop's first principle if:

$$f_P > 0 \implies c_P(f) = \min_{P \in \mathcal{P}_i} c_P(f),$$

for all paths P and commodities i. That is, a path is only taken if it gives the least cost.

Intuitively, the idea behind this is to simulate selfish behaviour in drivers. Each driver gets knowledge of how much traffic is on each path, and chooses the path that is cheapest to take for them. They do not take in to account how to make the overall cost lower for everyone. As a small aside, the first principle might remind the reader of a Nash Equilibrium [Petrosyan and Zenkevich, 1996].

Definition 2.15 (Nash Equilibrium). Let G be a game with a payoff function H depending on a strategy $s = (x_1, \dots, x_i, \dots, x_n)$. Then, the game is at *Nash Equilibrium* for the strategy s if for any deviation of a single player, which gives the strategy $s' = (x_1, \dots, x_i', \dots, x_n)$, we have that:

$$H(s) \geq H(s'). \tag{2.3.1}$$

In our context, the payoff function for each player might be the negative of the cost they experience when travelling between the origin destination pairs.

In this section we wish to find a way to compute the equilibrium flow in this situation, like we did for the system optimal flow. We start by looking back at proposition 2.13.If we look at the second statement, this embodies exactly what we define a Wardrop equilibrium to be, except we look at the marginal cost function instead of the normal cost function on an edge. This leads to the following condition for a flow to be a Wardrop equilibrium [Roughgarden, 2005].

Proposition 2.16. *A flow f is optimal for the instance $I = (G, r, c)$ if and only it is a user equilibrium for the instance $\tilde{I} = (G, r, c^*)$.*

From this, we can look for a formulation which helps us compute these equilibria. Notice that proposition 2.13 holds regardless of function in the sum, but still subject to the constraints. Therefore, we look for a function so that its derivative is equal to the cost function on an edge. That is, we wish to find h_e so that:

$$h'_e(f_e) = c_e(f_e),$$

for all e. Using the fundamental theorem of calculus, the function

$$h_e(f_e) = \int_0^{f_e} c_e(t)\,\mathrm{d}t,$$

works here. We begin by proving a simple lemma that gives us a guarantee that each of the functions h_e is convex.

Lemma 2.17. *Let $g(\cdot)$ be an integrable, non-decreasing function. Then, the function*

$$f(x) = \int_0^x g(t)\mathrm{d}t, \tag{2.3.2}$$

is convex. Moreover, if $g(\cdot)$ is strictly increasing, then $f(x)$ is strictly convex. Furthermore, if $x = (x_1, \dots, x_N) \in \mathbb{R}_+^N$ (that is, a vector in $\mathbb{R}^N$ with non-negative entries) and $h(x) = \sum_{i=0}^N x_i$ then the function

$$f(x) = \int_0^{h(x)} g(t)\mathrm{d}t, \tag{2.3.3}$$

is convex.

Proof. Let $f(x) = \int_0^x g(t)\mathrm{d}t$ where $g(t)$ is non-decreasing. Then, $f'(x) = g(x)$ and $f''(x) = g'(x) \geq 0$. Therefore, $f(x)$ is convex.
For the second statement, define

$$f(x) = f(x_1, \dots, x_N) = \int_0^{\sum_{0\leq i\leq N} x_i} g(t)\mathrm{d}t.$$

Then,

$$(\nabla f(x))_j = g\left(\sum_{0\leq i\leq N} x_i\right), \quad 0 \leq j \leq N$$
$$(\nabla^2 f(x))_{j,k} = g'\left(\sum_{0\leq i\leq N} x_i\right) \geq 0, \quad 0 \leq j,k \leq N.$$

Thus, the Hessian is a positively scaled version of the matrix with entry 1 everywhere:

$$\nabla^2 f(x) = g'\left(\sum_{0\leq i\leq N} x_i\right) \mathbb{1}_{N\times N}.$$

To check for positive (semi)-definiteness, we compute the eigenvalues of the matrix $M = \mathbb{1}_{N\times N}$.Since this matrix has $\mathrm{rank}(M) = 1$, we must have that the matrix has eigenvalue $\lambda = 0$ with multiplicity $N-1$, since the eigenspace corresponding to the 0 eigenvalue is exactly the kernel. By the rank-nullity theorem, this space is $n-1$ dimensional. Therefore, the multiplicity of this eigenvalue will be $n-1$. Since $\mathrm{tr}(M) = \sum_{i=0}^{N} \lambda_i = 0 + \cdots + 0 + \lambda_N$ and clearly $\mathrm{tr}(M) = N$, we have $\lambda_n = N$. Combining this with the fact that $g'(x) \geq 0$, Thus, $\lambda_i \geq 0$ for all i and so the matrix is positive semi-definite. By proposition A.7 the function f is thus convex.

We stress that even if the functions g are strictly increasing, the function f will not be strictly convex because the zero eigenvalues persist. □

The following theorem then gives us a way to compute user equilibria [Cominetti et al., 2012, Beckmann et al., 1955].

Theorem 2.18. *Consider an instance $I = (G, r, c)$. The vector $f \in \mathbb{R}^{|\mathcal{P}|}$ is a Wardrop equilibrium if and only if it solves the problem:*

$$\begin{aligned} \min_f Z(f) &:= \sum_{e\in E} \int_0^{f_e} c_e(t)\mathrm{d}t \\ \textit{subject to} \sum_{P\in\mathcal{P}_i} f_P &= r_i \quad \forall i \\ f_e &= \sum_{\substack{P\in\mathcal{P} \\ \text{s.t. } e\in P}} f_P \\ f_P &\geq 0 \end{aligned} \tag{UE}$$

Proof. Similar to the system optimum case, we note that the set of feasible flows is compact, and the function $Z(f)$ is continuous under the assumption that all cost functions are continuous. Therefore, by Weierstrass's theorem A.12 the function $Z(f)$ achieves its minimum on the set of feasible flows. By lemma 2.17, $Z(f)$ is convex in f_e since the sum of convex functions is convex. Since f_e is defined by a sum of path flows, it is also convex in f_P by using the second statement of the same lemma. By construction we can now apply the same logic as in the proof of theorem 2.13. Note that instead now we have:

$$(\nabla Z(f))_P = \sum_{e\in E} \frac{\partial Z}{\partial f_e}\frac{\partial f_e}{\partial f_P} = \sum_{e\in E} c_e(f_e) = c_P(f).$$

We do not perform all steps of the proof here, since they are identical to the ones in the proof of theorem 2.13. We find that the flow f satisfies the problem (UE) equilibrium if and only if it satisfies the fact that if $f_P > 0$, then $c_P(f) \leq c_{\tilde{P}}(f)$ for all $P, \tilde{P} \in \mathcal{P}_i$ and all i. This however is exactly equivalent to the definition of a Wardrop equilibrium.

□

Having established existence of solutions, uniqueness follows as well [Cominetti et al., 2012].

Proposition 2.19. *Let $I = (G, r, c)$ be an instance. Then, the edge flow $f^* = (f_e)_{e \in E}$ corresponding to a user equilibrium is unique, while a path flow assignment need not be unique.*

Proof. By lemma 2.17 the function $Z(f)$ as in (UE) is strictly convex in f_e, which implies there is a unique global minimum $f^* = (f_e)_{e \in E}$ by proposition A.13. However, by the same lemma the function is only convex in the path flows, which does not yield a unique minimum, but still gives a global minimum. □

To reassure the reader, even though the path flows are not unique, the cost associated with them is in fact unique.

Proposition 2.20. *Let $f, \tilde{f}$ be two path flow vectors that satisfy the problem* (UE)*. Then, $C(f) = C(\tilde{f})$.*

Proof. First, we show that if f and $\tilde{f}$ are two flows satisfying (UE) then $c_e(f_e) = c_e(\tilde{f}_e)$ [Nisan et al., 2007]. First, notice that since f and $\tilde{f}$ minimize $Z(f)$, they are both global minima by proposition 2.19. Thus, $Z(f) = Z(\tilde{f})$. Moreover, for any convex combination $\lambda f + (1 - \lambda \tilde{f})$ we must also have that:

$$Z(\lambda(f) + (1 - \lambda)\tilde{f}) \leq \lambda Z(f) + (1 - \lambda) Z(\tilde{f}),$$

since $Z(f)$ is convex in the path flows. But f and $\tilde{f}$ are global minima, so we also have the reverse inequality:

$$Z(\lambda f + (1 - \lambda)\tilde{f}) \geq Z(\tilde{f}) = \lambda Z(f) + (1 - \lambda) Z(\tilde{f}),$$

since $Z(f) = Z(\tilde{f})$. Thus, $Z(\lambda f + (1 - \lambda)\tilde{f}) = \lambda Z(f) + (1 - \lambda) Z(\tilde{f})$. Therefore, the function Z is in fact linear between f and $\tilde{f}$, but this means that each of the integrals has to be linear between f_e and $\tilde{f}_e$ for all $e \in E$. However, if the integral is linear between f_e and $\tilde{f}_e$, this must mean that the function is constant between f_e and $\tilde{f}_e$. Therefore, $c_e(f_e) = c_e(\tilde{f}_e)$.
The proposition now follows by writing out the definition fo the cost of a flow:

$$\begin{aligned} C(f) = \sum_{P \in \mathcal{P}} c_P(f) f_P &= \sum_i \sum_{P \in \mathcal{P}_i} C_i f_P \\ &= \sum_i C_i \sum_{P \in \mathcal{P}_i} f_P = \sum_i C_i r_i \\ &= \sum_i C_i \sum_{P \in \mathcal{P}_i} \tilde{f}_P = \sum_{P \in \mathcal{P}} c_P(\tilde{f}) \tilde{f}_P = C(\tilde{f}). \end{aligned}$$

The first equality follows by the optimality condition derived in theorem 2.18. By this condition the path cost of a flow will only depend on the OD-pair we are looking at. The fourth equality follows by the flow constraint f satisfies, and the final equality follows since $c_P(\tilde{f}) = c_P(f)$ by the argument above. If the cost on edges is the same, then so is the cost on paths by summing over all edges in the path. □

Example 2.21 (Braess' Paradox). Consider the instances I_1 on the left and I_2 on the right of figure 4. We investigate the user equilibria for both of these instances.

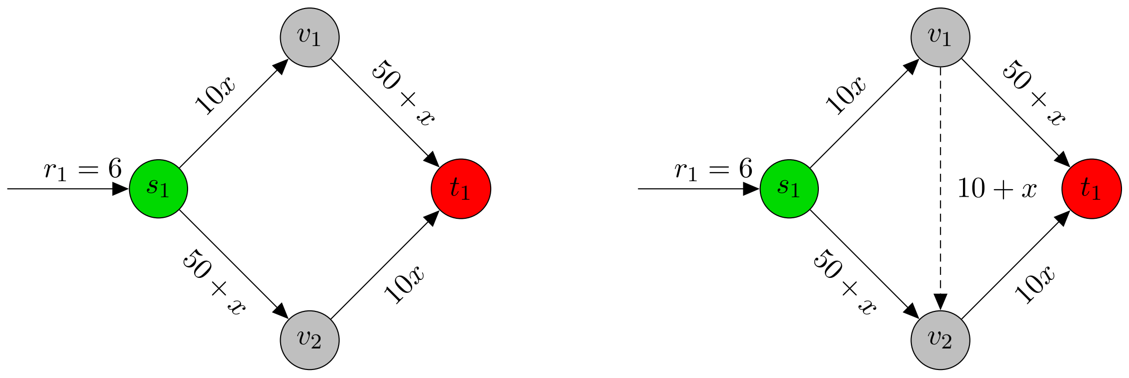


Figure 4: Braess's paradox

First, let us consider the left network. In this case, we have two different paths we can take to go from s_1 to t_1. Namely, $P_1 = \{s_1, v_1, t_1\}$ and $P_2 = \{s_1, v_2, t_1\}$. However, notice that the cost functions on the top path and the bottom path are the same. Since this is the case, we would expect the flow corresponding to a user equilibrium to split equally. Recalling that in a user equilibrium the cost along each path should be the same, the only way for a flow to achieve this in this system is to split equally.
To check this computationally we compute the solution to the minimization problem as follows. Set the flow on the top path equal to x and the flow on the bottom edge equal to y. Then, we wish to solve:

$$\min_{(x,y)} Z(x,y) := \int_0^x 50 + 11t \, \mathrm{d}t + \int_0^y 50 + 11t \, \mathrm{d}t = 50x + \frac{11}{2}x^2 + 50y + \frac{11}{2}y^2$$
$$\text{subject to } x + y = 6, \quad x, y \geq 0.$$

Using the constraint, we find that $y = 6 - x$ and so the objective function can be simplified to:

$$\tilde{Z}(x) = 50(x+y) + \frac{11}{2}\left(x^2 + (6-x)^2\right) = 300 + \frac{11}{2}(2x^2 - 12x + 36).$$

Computing the first and second derivative,

$$\tilde{Z}'(x) = 11(2x - 6), \quad \tilde{Z}''(x) = 22 > 0$$

Since $\tilde{Z}'' > 0$, any critical point will be a local minimum of the function. Clearly, the root of the derivative is at $x = 3$, and so the flow on the top path will be 3. Since $x + y = 6$, we find that $x = y = 3$ on this network.

For the second network, we have three paths. First, P_1 and P_2 still exist, but we add another path $P_3 = \{s_1, v_1, v_2, t_1\}$. One could view this as a local authority building a new road to try and help traffic flow. Set the flow on this route equal to z. First, let us build the objective function. In figure 5 the flows on each edge are drawn. Combining this with the cost functions on each edge the cost function will be given by:

$$\begin{aligned}F(x,y,z) &= \int_0^{x+z} 10t\,\mathrm{d}t + \int_0^{x} 50+t\,\mathrm{d}t + \int_0^{z} 10+t\,\mathrm{d}t + \int_0^{y} 50+t\,\mathrm{d}t + \int_0^{y+z} 10t\,\mathrm{d}t \\ &= 5(x+z)^2 + 50x + \frac{x^2}{2} + 10z + \frac{z^2}{2} + 50y + \frac{y^2}{2} + 5(y+z)^2 \\ &= 5(x^2+2xz+z^2) + 50x + \frac{x^2}{2} + 10z + \frac{z^2}{2} + 50y + \frac{y^2}{2} + 5(y^2+2yz+z^2) \\ &= \frac{11}{2}x^2 + \frac{11}{2}y^2 + \frac{21}{2}z^2 + 10xz + 10yz + 50y + 50x + 10z.\end{aligned}$$

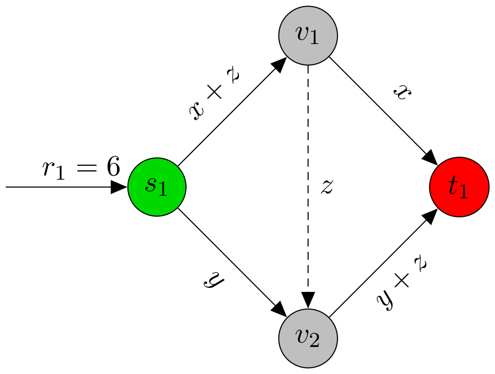


Figure 5: Flows on edges with $f = (x, y, z)$

The gradient of this function is then equal to:

$$\nabla F(x,y,z) = \begin{bmatrix} 11x + 10x + 50 \\ 11y + 10z + 50 \\ 21z + 10x + 10y + 10 \end{bmatrix}.$$

Using the Lagrange multiplier method, we compute the minimum of F subject to $g(x,y,z) = 0$, with $g(x,y,z) = x+y+z-6$, and check that the solution is non-negative. Indeed, solving the system:

$$\nabla F(x,y,z) + \lambda \nabla g(x,y,z) = 0,\ g(x,y,z) = 0 \iff \begin{bmatrix} 11 & 0 & 10 & 1 \\ 0 & 11 & 10 & 1 \\ 10 & 10 & 21 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \lambda \end{bmatrix} = \begin{bmatrix} -50 \\ -50 \\ -10 \\ 6 \end{bmatrix}.$$

gives the solution $x = y = z = 2$ and $\lambda = -92$.
Having computed both equilibrium flows, let us now compare the social cost of each of these flows. For the first network, we find:

$$C_1(x,y) = (50 + 11x)x + (50 + 11y)y \implies C_1(3,3) = 498.$$

For the second network, we use proposition 2.8 to compute that:

$$C_2(x,y,z) = 10(x+z)^2 + (50+x)x + (10+z)z + (50+y)y + 10(y+z)^2 \implies C_2(2,2,2) = 552.$$

What we can see is that adding an edge makes traffic worse, in the sense that the new cost is higher than the original cost. Making this even more extreme, if we were to make the middle edge have cost $c(z) = 0$ instead of $c(z) = 10 + z$, the problem still persists [Roughgarden, 2005]. $\triangle$

The above example is not just a theoretical exercise however. Several examples of this happening in urban planning exist [Bagloee et al., 2019]. For example, in New York the very congested 42^{nd} street was closed off for earth day in 1990. Remarkably, the closure of this street not only did not lead to any major traffic jams, but even improved traffic flow on streets surrounding the blocked off road [Kolata, 1990].

3 Price of anarchy

3.1 Introduction

The term price of anarchy was first used by Koutsoupias and Papadimitriou to describe the inefficiency of a selfish equilibrium in a network game [Koutsoupias and Papadimitriou, 2009]. The term has a wider use in game theory and economics, extending beyond the use we have in this paper. In this section we study the price of anarchy in network routing games. We follow the theory proposed by Roughgarden [Roughgarden, 2005]. Whenever a result from Roughgarden is used, we give its location in brackets for the reader to find in the literature. We first give a definition of the price of anarchy:

Definition 3.1 (Price of anarchy, 2.3.1). Let $I = (G, r, c)$ be an instance of a routing game. Consider a user equilibrium f and a system optimum f^*. Then, *the price of anarchy* of I is

$$\rho(I) = \frac{C(f)}{C(f^*)}, \tag{3.1.1}$$

with the understanding that $0/0 = 1$.

Note that by proposition 2.20, if f and $\tilde{f}$ are user equilibria then $C(f) = C(\tilde{f})$ and so the price of anarchy is well-defined.

In this section, we prove the following theorem:

Theorem 3.2 (Bound on price of anarchy with polynomial cost functions, 3.5.1). *Let I be an instance with cost functions from the set*

$$\mathfrak{C}_p = \left\{ f \in \mathcal{C}^2(\mathbb{R}, \mathbb{R}) \text{ such that } f(x) = \sum_{i=0}^{p} a_i x^p, \quad a_i \geq 0 \right\}. \tag{3.1.2}$$

Then,

$$\rho(I) \leq \left(1 - p(p+1)^{-\frac{p+1}{p}}\right)^{-1}. \tag{3.1.3}$$

We tackle this problem in two steps. First, we prove that if $p = 1$, then $\rho(I) \leq \frac{4}{3}$. After this, we expand the built up theory and prove the more general result.

3.2 Price of anarchy for linear cost functions

Consider an instance $I = (G, r, c)$, with each c_e having the form $c_e(f_e) = a_e f_e + b_e$ with $a_e, b_e \geq 0$. First, we note that the cost function corresponding to these edge costs is convex. Indeed, we have that:

$$C(f) = \sum_{e \in E} c_e(f_e) f_e = \sum_{e \in E} a_e f_e^2 + b_e f_e. \tag{3.2.1}$$

Convexity of the cost function now follows from the following proposition:

Proposition 3.3. *Let f be a polynomial of degree n, with non-negative coefficients. Then, the function f is convex on the interval $[0, \infty)$.*

Proof. Let $f(x) = \sum_{i=0}^{n} a_i x^i$. Then $f'(x) = \sum_{i=1}^{n} i a_i x^{i-1}$ and $f''(x) = \sum_{i=2}^{n} i(i-1) a_i x^{i-2}$. Since $x \geq 0$, and $a_i \geq 0$ for all i, we have that $f''(x) \geq 0$. Thus, by applying proposition A.7 in the one dimensional case, we find that f is convex. $\square$

First, we provide a result that we will use extensively.

Proposition 3.4 (Adapted from 2.4.4). *Let f^* be a feasible flow for the instance $I = (G, r, c)$ where each of the functions $c_e(x)x$ is convex. Then, the following are equivalent:*

1. *The flow f^* is optimal;*
2. *For all i and paths $P_1, P_2 \in \mathcal{P}_i$ with $f^*_{P_1} > 0$:*
$$c^*_{P_1}(f^*) \leq c^*_{P_2}(f^*); \tag{3.2.2}$$
3. *For any feasible flow f:*
$$\sum_{P \in \mathcal{P}} c^*_P(f^*) f^*_P \leq \sum_{P \in \mathcal{P}} c^*_P(f^*) f_P; \tag{3.2.3}$$
4. *For all feasible flows f:*
$$\sum_{e \in E} c^*_e(f^*_e) f^*_e \leq \sum_{e \in E} c^*_e(f^*_e) f_e. \tag{3.2.4}$$

Proof. We prove that $1 \iff 2 \implies 3 \iff 4 \implies 1$.
($1 \iff 2$) This statement is exactly proposition 2.13.
($2 \implies 3$): Define the function $H(f) = \sum_{P \in \mathcal{P}} c^*_P(f^*) f_P$, which can be interpreted as the cost of a path on a network with paths having cost functions $c^*_P(f^*)$ (which are independent of the flow). Note that H is minimized by the flow that routes all traffic over the cheapest path. We see this since $c^*_P \geq 0$. However, (assuming 2) the flow that achieves this is precisely f^*.
($3 \iff 4$): This is the same statement as proposition 2.8.
($4 \implies 1$) : Let f and f^* be feasible flows. Then, since each function $c^*_e(x)x$ is assumed to be convex, we find that:
$$C(f) = \sum_{e \in E} c_e(f_e) f_e \geq \sum_{e \in E} c_e(f^*_e) f^*_e + \sum_{e \in E} c^*(f_e)(f_e - f^*_e).$$
But we assumed that $\sum_{e \in E} c^*_e(f^*_e) f^*_e \leq \sum_{e \in E} c^*_e(f^*_e) f_e$ so the last term will be positive. Thus,
$$C(f) \geq C(f^*) + \sum_{e \in E} c^*(f_e)(f_e - f^*_e) \geq C(f^*),$$
which implies that the flow f^* is optimal. $\square$

In fact, this proposition holds more generally. The implications $2 \iff 3 \iff 4$ hold for any convex function $h_e(f_e)$. Instead of c^* we would then find $h'_e(f_e)$. For the specific choice $h_e(f_e) = c_e(f_e)f_e$ we can add the first implication as well. By this line of reasoning, if we set $h_e(f_e) = \int_0^{f_e} c_e(x)\mathrm{d}x$, we would find the same proposition, except now $h'_e(f_e) = c_e(f_e)$ and the first implication would be that f^* is a user equilibrium. If we apply the above theorem to our instance with linear cost functions, we find the following result.

Proposition 3.5 (Adapted from 3.2.1). *Let I be an instance with only linear cost functions. Then*

1. *A feasible flow f is a user equilibrium for I if, and only if for all i and $s_i - t_i$ paths $P_1, P_2 \in \mathcal{P}_i$, with $f_{P_1} > 0$*
$$\sum_{e \in P_1} a_e f_e + b_e \leq \sum_{e \in P_2} a_e f_e + b_e; \tag{3.2.5}$$

2. *A feasible flow f^* is optimal for I if, and only if for all i and $s_i - t_i$ paths $P_1, P_2 \in \mathcal{P}_i$ with $f_{P_1} > 0$*
$$\sum_{e \in P_1} 2a_e f_e^* + b_e \leq \sum_{e \in P_2} 2a_e f_e^* + b_e. \tag{3.2.6}$$

Proof.

1. This is the result of theorem 2.18 (or a direct reformulation of the definition of a user equilibrium).

2. This is the relation $(1 \iff 2)$ in proposition 3.4 after summing. Let f^* be optimal. Then for all i and paths $P_1, P_2 \in \mathcal{P}_i$ ith $f^*_{P_1} > 0$ the marginal cost of f^* on P_1 is less than or equal to the marginal cost on P_2. In our case, we have that:
$$c_e^*(x) = \frac{\mathrm{d}}{\mathrm{d}x}\left(a_e x^2 + b_e x\right) = 2a_e x + b_e.$$
Thus, we have that:
$$c^*_{P_i}(f^*) = \sum_{e \in P_i} c_e^*(f^*) = \sum_{e \in P_i} 2a_e f_e^* + b_e.$$
Applying part 2 of proposition 3.4 gives the desired inequality.

□

After establishing this result, we proceed as follows:

1. We form a relation between the marginal cost and the cost of a flow, and we show that if f is optimal for $I = (G, r, c)$, then $\frac{f}{2}$ is optimal for the instance $\tilde{I} = (G, \frac{r}{2}, c)$;

2. We then establish lower bounds on cost functions for instances with altered rate demand vectors;

3. Combining these results, we establish the desired result in theorem 3.2 for $p = 1$.

First, we have the following simple lemma:

Lemma 3.6 (3.2.3). *Suppose I is an instance with only linear cost functions and f is a user equilibrium. Then,*

1. $c_e^*\left(\frac{f_e}{2}\right) = c_e(f_e)$ *for all edges e;*
2. *The flow $\frac{f}{2}$ is optimal for the instance $\tilde{I} = (G, \frac{r}{2}, c)$.*

Proof.

1. Note that $c_e^*(x) = c_e(x) + c_e'(x)x = 2a_e x + b_e$. Therefore, we have that

$$c_e^*\left(\frac{f_e}{2}\right) = 2a_e\left(\frac{f_e}{2}\right) + b_e = a_e f_e + b_e = c_e(f_e).$$

2. Note that since f is a user equilibrium, we know by theorem 3.5 that we must have that for all i and $s_i - t_i$ paths P_1 and $P_2 \in \mathcal{P}_i$ with the flow on P_1 being nonzero

$$\sum_{e \in P_1} a_e f_e + b_e \leq \sum_{e \in P_2} a_e f_e + b_e.$$

Moreover, we have that the flow $\frac{f}{2}$ is feasible for the instance $\tilde{I}$, since we only changed the demand rates. But then we see that we have

$$\begin{aligned}\sum_{e \in P_1} 2a_e\left(\frac{f_e}{2}\right) + b_e &= \sum_{e \in P_1} a_e f_e + b_e \\ &\leq \sum_{e \in P_2} a_e f_e + b_e = \sum_{e \in P_2} 2a_e\left(\frac{f_e}{2}\right) + b_e.\end{aligned}$$

And thus the flow $\frac{f}{2}$ is optimal by part 2 of theorem 3.5.

□

Another straightforward result is the following.

Lemma 3.7 (3.2.4). *The cost of the flow $\frac{f}{2}$ is at least $\frac{1}{4}C(f)$*

Proof. We compute

$$\begin{aligned}C\left(\frac{f}{2}\right) &= \sum_{e \in E}\left(a_e\left(\frac{f_e}{2}\right) + b_e\right)\frac{f_e}{2} \geq \sum_{e \in E}\left(a_e\left(\frac{f_e}{2}\right) + \frac{b_e}{2}\right)\frac{f_e}{2} \\ &= \sum_{e \in E}\frac{1}{2}\left(a_e f_e + b_e\right)\frac{f_e}{2} = \frac{1}{4}\sum_{e \in E}\left(a_e f_e + b_e\right) f_e = \frac{1}{4}C(f),\end{aligned}$$

where the inequality holds since $b_e \geq 0$. □

The main result that we will use is somewhat similar in nature to the previous lemma, except we now make the route demand larger.

Proposition 3.8 (3.2.5). *Let I be an instance with linear cost functions and let f^* be optimal for I. Then, for all $\delta > 0$, if f is feasible for the instance $\tilde{I} = (G, (1+\delta)r, c)$ then,*

$$C(f) \geq C(f^*) + \delta \sum_{e \in E} c^*(f_e^*) f_e^*. \tag{3.2.7}$$

Proof. Let $I = (G, r, c)$ be an instance with optimal flow f^* and fix $\delta > 0$. Suppose that f is feasible for the instance $\tilde{I} = (G, (1+\delta)r, c)$. By proposition 3.3, we know that for all edges e the function $c_e(f_e)f_e = a_e f_e^2 + b_e f_e$ is convex in f_e. Therefore,

$$c_e(f_e)f_e \geq c_e(f_e^*)f_e^* + c_e^*(f_e^*)(f_e - f_e^*),$$

since $c_e^*(f_e) = \frac{\mathrm{d}}{\mathrm{d}f_e}\left(c_e(f_e)f_e\right)$. Summing over all edges gives us that

$$C(f) = \sum_{e \in E} c_e(f_e)f_e \geq \sum_{e \in E} c_e(f_e^*)f_e^* + \sum_{e \in E} c_e^*(f_e^*)(f_e - f_e^*)$$

We focus on the second sum on the right hand side of the inequality. Since f^* is optimal for I, for any other feasible flow f we have by propoition A.14 that

$$\sum_{e \in E} c_e^*(f_e^*)f_e^* \leq \sum_{e \in E} c_e^*(f_e^*)f_e.$$

Moreover, the flow $\frac{f}{1+\delta}$ is feasible for I, since f is feasible for the instance $\tilde{I} = (G, (1+\delta)r, c)$. The above inequality then gives that

$$\sum_{e \in E} c_e^*(f_e^*)f_e^* \leq \frac{1}{1+\delta} \sum_{e \in E} c_e^*(f_e^*)f_e.$$

Rewriting this inequality gives that:

$$\begin{aligned}
\frac{1}{1+\delta} \sum_{e \in E} c_e^*(f_e^*)f_e - \sum_{e \in E} c_e^*(f_e^*)f_e^* &\geq 0 \\
\frac{1}{1+\delta} \left(\sum_{e \in E} c_e^*(f_e^*)f_e - (1+\delta) \sum_{e \in E} c_e^*(f_e^*)f_e^* \right) &\geq 0 \\
\frac{1}{1+\delta} \left(\sum_{e \in E} c_e^*(f_e^*)(f_e - f_e^*) - \delta \sum_{e \in E} c_e^*(f_e^*)f_e^* \right) &\geq 0 \\
\sum_{e \in E} c_e^*(f_e^*)(f_e - f_e^*) &\geq \delta \sum_{e \in E} c_e^*(f_e^*)f_e^*
\end{aligned}$$

Combing with 3.2 we find that

$$\begin{aligned}C(f) &\geq \sum_{e\in E} c_e(f_e^*)f_e^* + \sum_{e\in E} c_e^*(f_e^*)(f_e - f_e^*)\\ &= C(f^*) + \sum_{e\in E} c_e^*(f_e^*)(f_e - f_e^*)\\ &\geq C(f^*) + \delta \sum_{e\in E} c_e^*(f_e^*)f_e^*\end{aligned}$$

□

With all of these results, we can now prove the first main result of this section.

Theorem 3.9 (Price of anarchy for linear cost functions, 3.2.6). *Let I be an instance with linear cost functions. Then,*

$$\rho(I) \leq \frac{4}{3} \tag{3.2.8}$$

Proof. Let $I = (G, r, c)$ be an instance with only linear cost functions and let f be a user equilibrium for this instance. Then, by part 2 of lemma 3.6 we have that the flow $\frac{f}{2}$ is optimal for the instance $\tilde{I} = (G, \frac{r}{2}, c)$. Notice that $r = (1+1)\cdot\frac{r}{2}$ and thus if f^* is feasible for the instance I by proposition 3.8 with $\delta = 1$ we find that for the cost of f^* we have

$$C(f^*) \geq C\left(\frac{f}{2}\right) + \sum_{e\in E} c_e^*\left(\frac{f_e}{2}\right)\frac{f_e}{2}$$

Combining this with 3.7 we find that

$$C(f^*) \geq \frac{1}{4}C(f) + \frac{1}{2}\sum_{e\in E} c_e^*\left(\frac{f_e}{2}\right) f_e$$

And finally, applying part 1 of 3.6 we arrive at the fact that:

$$\begin{aligned}C(f^*) &\geq \frac{1}{4}C(f) + \frac{1}{2}\sum_{e\in E} c_e^*\left(\frac{f_e}{2}\right) f_e\\ &= \frac{1}{4}C(f) + \frac{1}{2}\sum_{e\in E} c_e(f_e)f_e = \frac{3}{4}C(f)\end{aligned}$$

And so,

$$\frac{C(f)}{C(f^*)} \leq \frac{4}{3}$$

Since, we assumed that f^* is any feasible for I the above inequalities must also hold for the system optimal flow, which proves the theorem. □

3.3 Price of anarchy for general polynomials

Now that we have established an upper bound for affine cost functions, we wish to do the same for general polynomial functions. As mentioned before we assume to be working with functions from the set:

$$\mathfrak{C}_p = \left\{ f \in \mathcal{C}^2(\mathbb{R}, \mathbb{R}) \text{ such that } f(x) = \sum_{i=0}^{p} a_i x^p, \quad a_i \geq 0 \right\}. \tag{3.3.1}$$

To start our discussion, we give the following definition motivated by the Pigou example, which we will see later.

Definition 3.10 (Anarchy value, 3.3.1). Let I be an instance with cost functions from the set $\mathcal{C}$. The *anarchy value* of the function $c \in \mathfrak{C}$:

$$\alpha(c) = \sup_{x,r \geq 0} \frac{rc(r)}{xc(x) + (r-x)c(r)}. \tag{3.3.2}$$

with the understanding that $0/0 = 1$. The anarchy value of the set $\mathfrak{C}$ is the supremum of the anarchy values of all costs in $\mathfrak{C}$. That is:

$$\alpha\left(\mathfrak{C}\right) = \sup_{0 \neq c \in \mathfrak{C}} \sup_{x,r \geq 0} \frac{rc(r)}{xc(x) + (r-x)c(r)}. \tag{3.3.3}$$

The anarchy value can be interpreted as the price of anarchy of a specific instance. Consider a single commodity instance with two edges. On the top edge, we take the cost to be constant 1 while on the bottom we take some function $c(\cdot)$. Then, take the traffic rate to be the positive real number r so that $c(r) = 1$.[1] Then, the user equilibrium will have cost $rc(r)$, since all traffic will flow over the bottom edge. Thus, we have $f = (0, r)$. The optimum case however, will have cost:

$$\min_x xc(x) + (r - x)$$

We see this as follows. Let the flow split into two variables y and x flowing over the top and bottom edge respectively. These variables must satisfy the constraint $x + y = r$. But then, we must have $y = r - x$. Therefore, the cost function will become

$$C(f) = \sum_{e \in E} c_e(f_e) f_e = (r - x) \cdot 1 + xc(x).$$

Suppose that x minimizes this cost. The flow in this case is then $f^* = (r - x, x)$ Then, the price of anarchy of this instance is precisely the fraction in equation 3.3.2. Namely,

$$\rho(I) = \frac{C(f)}{C(f^*)} = \frac{rc(r)}{(r-x)c(r) + x \cdot c(x)} \tag{3.3.4}$$

The anarchy value is then the least upper bound on these values, for any x and r. To prove the upper bound from the theorem we proceed as follows. First, we find a simple way to compute the anarchy bound for any

[1] If such a number does not exist, we can always scale the function c by a factor to ensure existence.

function. Then, we show that the anarchy value in fact is an upper bound for the price of anarchy. After this, we argue that for cost functions in $\mathfrak{C}_p$ we find the upper bound as in theorem 3.2 and that this gives a tight bound by means of an example.

If we look at the expression for the anarchy value of even a single value, we are dealing with a supremum over 2 variables. We could make this work, however it is not really a practical way to compute the anarchy value. Thankfully, to aid us in computing the anarchy value we have the following proposition:

Proposition 3.11 (3.3.2). *Let $c(\cdot)$ be a C^1 cost function and assume that the associated marginal cost function is convex. Then,*

$$\alpha(c) = \sup_{r\geq 0} \left(\lambda\mu + (1-\lambda)\right)^{-1} \tag{3.3.5}$$

where $\lambda \in [0,1]$ is a solution to $c^*(\lambda r) = c(r)$ and $\mu = \frac{c(\lambda r)}{c(r)}$.

Proof. Fix $r \geq 0$ and define the function $f(\lambda) = c^*(\lambda r) - c(r)$. Notice that

$$f(0) = c^*(0) - c(r) = c(0) - c(r) \leq 0$$
$$f(1) = c^*(r) - c(r) = rc'(r) \geq 0$$

Since c is assumed to be non-decreasing and continuous. By the intermediate value theorem there exists $\lambda \in [0,1]$ so that $f(\lambda) = 0$ [Abbott, 2015, Chapter 4]. Thus, there exists a λ so that $c^*(\lambda r) = c(r)$. As we outlined above the anarchy value is the price of anarchy in a two link network. Setting $x = \lambda r \leq r$ in the example attached to equation 3.3.4 we obtain a feasible flow $f = (r - \lambda r, \lambda r)$, which is optimal by the equivalence 1 $\iff$ 2 of proposition 3.4. The price of anarchy of this instance will then be:

$$\begin{aligned}\rho(I) &= \frac{rc(r)}{\lambda r \cdot c(\lambda r) + (r-\lambda r)c(r)} \\ &= \frac{1}{\lambda\frac{c(\lambda r)}{c(r)} + (1-\lambda)\frac{c(r)}{c(r)}} = \left(\lambda\mu + (1-\lambda)\right)^{-1}\end{aligned}$$

The price of anarchy can then be obtained by taking the supremum over all r, since the variable x is now replaced by λr. That is,

$$\alpha(c) = \sup_{r\geq 0} \frac{rc(r)}{\lambda r \cdot c(\lambda r) + (r-\lambda r)c(r)} = \sup_{r\geq 0} \left(\lambda\mu + (1-\lambda)\right)^{-1}$$

□

The only difficulty we still have is computing a value of λ that works here. As we will see later however, this process is not too difficult when considering polynomial functions. To finally put together the connection between the anarchy value and the price of anarchy we show that the anarchy value provides an upper bound on the price of anarchy.

Theorem 3.12 (Adapted from 3.3.6, 3.3.7). *Let $\mathfrak{C}$ be a set of cost functions and $I = (G, r, c)$ an instance where the cost functions come from the set $\mathfrak{C}$. Then,*

$$\rho(I) \leq \alpha(\mathfrak{C}) \tag{3.3.6}$$

Proof. We proceed like in the proof of theorem 3.9. That is, take f^* to be an optimal flow and f a user equilibrium. First, we note that we can establish the following relation:

$$\alpha(\mathfrak{C}) = \sup_{0 \neq c \in \mathfrak{C}} \sup_{x,r \geq 0} \frac{rc(r)}{xc(x) + (r-x)c(r)} \geq \frac{rc(r)}{xc(x) + (r-x)c(r)} \forall x, r \geq 0, \forall c \in \mathfrak{C}$$

Therefore,

$$\begin{aligned} \alpha(\mathfrak{C}) &\geq \frac{rc(r)}{xc(x) + (r-x)c(r)} \\ \iff xc(x) + (r-x)c(r) &\geq \frac{rc(r)}{\alpha(\mathfrak{C})} \\ \iff xc(x) &\geq (x-r)c(r) + \frac{rc(r)}{\alpha(\mathfrak{C})} \end{aligned}$$

for all $c \in \mathfrak{C}$ and $x, r \geq 0$. In particular, setting $x = f_e^*$ and $r = f_e$ we find that

$$\begin{aligned} C(f^*) &= \sum_{e \in E} f_e c_e(f_e) \\ &\geq \sum_{e \in E} (f_e^* - f_e) c_e(f_e) + \sum_{e \in E} \frac{c_e(f_e) f_e}{\alpha(\mathfrak{C})} \geq \sum_{e \in E} \frac{c_e(f_e) f_e}{\alpha(\mathfrak{C})} = \frac{C(f)}{\alpha(\mathfrak{C})}. \end{aligned}$$

The second inequality can be found as follows. By using the note after proposition 3.4, we found that f is a user equilibrium if and only if:

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

for any feasible flow $f^* \in \mathcal{F}$. Therefore, if f is a user equilibrium, we must have that:

$$(f_e^* - f_e) c_e(f_e) \geq 0$$

for some feasible flow f^*, since all terms here are assumed to be positive. In particular, this flow f^* could be the system optimal flow. Moving around the terms, we obtain that:

$$\frac{C(f)}{C(f^*)} = \rho(I) \leq \alpha(\mathfrak{C})$$

□

Now that we know that the anarchy value upper bounds the price of anarchy and we have a way to compute it we move on to proving the main theorem of this section.

Theorem 3.2 (Bound on price of anarchy with polynomial cost functions, 3.5.1). *Let I be an instance with cost functions from the set*

$$\mathfrak{C}_p = \left\{ f \in \mathcal{C}^2(\mathbb{R}, \mathbb{R}) \text{ such that } f(x) = \sum_{i=0}^{p} a_i x^p, \quad a_i \geq 0 \right\}. \tag{3.1.2}$$

Then,

$$\rho(I) \leq \left(1 - p(p+1)^{-\frac{p+1}{p}}\right)^{-1}. \tag{3.1.3}$$

Proof. First, we note that it suffices to show that $\alpha(\mathfrak{C}) = \left(1 - p(p+1)^{-\frac{p+1}{p}}\right)^{-1}$. After this we show that this bound cannot be improved upon by means of an example.
To compute the anarchy value, we note that we can consider just the cost functions of the form $c(x) = ax^i$ for some $a > 0$ and $i \leq p$. We see this, since we can simply split the cost function over a single edge into multiple edges, so that the cost of these edges adds up to the original edge cost as in figure 6.

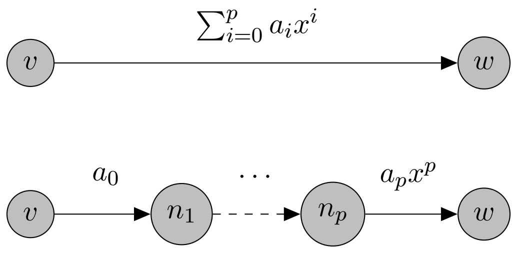


Figure 6: Splitting a single edge into multiple with the same total cost

Therefore, we assume that $c(x) = ax^i$ and we will see that the result we obtain does not depend on a and will be largest when $i = p$. As this function is convex on $[0, \infty)$ and differentiable, we can compute the value of $\alpha(c)$ using proposition 3.11. We compute

$$c^*(x) = c(x) + xc'(x) = ax^i + x \cdot aix^{i-1} = a(i+1)x^i$$

Fixing $r \geq 0$ we wish to solve $c^*(\lambda r) = c(r)$ for some $\lambda \in [0, 1]$. Therefore,

$$\begin{aligned} c^*(\lambda r) = c(r) &\iff a(i+1)(\lambda r)^i = ar^i \\ &\iff (i+1)\lambda^i = 1 \iff \lambda = (i+1)^{-1/i} \end{aligned}$$

Moreover, we have that $\lambda \in [0, 1]$ for $i \in \mathbb{Z}^+$ (that is, if i is a positive integer) since:

$$\lambda = \frac{1}{\sqrt[i]{i+1}} \leq \frac{1}{i+1} \leq 1,$$

and clearly $\lambda > 0$. If $i = 0$, then any value of $\lambda \in [0, 1]$ satisfies $c^*(\lambda r) = c(r)$. Having found this value for

λ we compute that:

$$\mu = \frac{c(\lambda r)}{c(r)} = \frac{a\lambda^i r^i}{ar^i} = \lambda^i = \frac{1}{1+i}$$

Both λ and μ are independent of both r and a. This gives that:

$$\begin{aligned}\alpha(c) &= (\mu\lambda + (1-\lambda))^{-1} = \left(\lambda^{i+1} + (1-\lambda)\right)^{-1} \\ &= \left((i+1)^{-(i+1)/i} + (i+1)^{-(i+1)/i}\left((i+1)^{(i+1)/i} - (i+1)\right)\right)^{-1} \\ &= \left((i+1)^{-(i+1)/i}\left(1 + (i+1)^{(i+1)/i)} - i - 1\right)\right) = \left(1 - i(i+1)^{-\frac{i+1}{i}}\right)^{-1}\end{aligned}$$

Finally, we note that $\alpha(c)$ is increasing in i. Setting $y = i(i+1)^{-\frac{i+1}{i}}$ and defining

$$f(i) = (1 - y(i))^{-1}$$

we can use the chain rule to compute that:

$$f'(i) = \frac{\mathrm{d}f}{\mathrm{d}y}\frac{\mathrm{d}y}{\mathrm{d}i} = (1-y)^{-2}\frac{\mathrm{d}y}{\mathrm{d}i}.$$

Since the first term is positive (since $y > 0$) we focus on the second.

$$\begin{aligned} y = i(i+1)^{-\frac{i+1}{i}} &\iff \ln(y) = \ln(i) - \left(1 + \frac{1}{i}\right)\ln(i+1) \\ \frac{1}{y}\frac{\mathrm{d}y}{\mathrm{d}i} &= \frac{1}{i} + \frac{\ln(i+1)}{i^2} - \left(\frac{i+1}{i}\frac{1}{i+1}\right) \\ \frac{\mathrm{d}y}{\mathrm{d}i} &= y\left(\frac{1}{i} + \frac{\ln(i+1)}{i^2}\right) > 0\end{aligned}$$

Since $i \geq 1$. Therefore $f'(i) > 0$ and $\alpha(c)$ will be largest if i is largest. Noting that $i \leq p$ we get our desired result:

$$\alpha(\mathfrak{C}) = \left(1 - p(p+1)^{-\frac{p+1}{p}}\right)^{-1}$$

□

The nonlinear Pigou network shows us that we cannot improve on this bound.

Example 3.13 (Nonlinear Pigou example, 2.5.4). Consider the networks given in figure 7. The left network was first studied (albeit not directly) by economist Pigou in 1920 [Pigou, 1920]. However, we discuss a nonlinear variant (where $p \geq 1$) of this problem like in the right network. We claim that this network has price of anarchy exactly $\left(1 - p(p+1)^{-\frac{p+1}{p}}\right)^{-1}$.

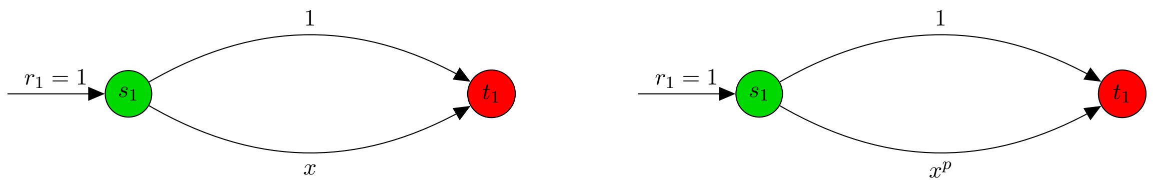


Figure 7: Pigou's example of traffic flow

First let us compute the system optimal solution. Denote the flow as $f = (x, y)$ where there are x units of traffic on the lower edge, and y units of traffic on the top edge. Therefore, we wish to find a solution f to the problem:

$$\min C(x,y) = x^{p+1} + y$$
$$\text{subject to } x, y \geq 0, x + y = 1$$

From the second constraint, we find that $y = 1 - x$. Therefore, we want to minimize $C(x) = x^{p+1} + 1 - x$ with respect to x. Using the first order optimality condition, we find that $C'(x) = (p+1)x^p - 1$ is zero when $x = (p+1)^{-1/p}$. It follows $y = 1 - x = 1 - (p+1)^{-1/p}$. The total cost of the system optimum flow is thus:

$$\begin{aligned} C(x,y) &= (p+1)^{-(p+1)/p} + 1 - (p+1)^{-1/p} \\ &= 1 - (p+1)^{-1/p}\left(1 - (p+1)^{-1}\right) \\ &= 1 - (p+1)^{-1/p}\left(\frac{p}{p+1}\right) \\ &= 1 - p(p+1)^{-1/p-1} = 1 - p(p+1)^{-\frac{p+1}{p}} \end{aligned}$$

Like we found before.

To find the user equilibrium, we use the problem UE. That is, we wish to solve

$$\min z(x,y) = \int_0^x t^p \mathrm{d}t + \int_0^y \mathrm{d}t = \frac{1}{p+1}x^{p+1} + y$$
$$\text{subject to } x, y \geq 0, x + y = 1$$

Again, noting $y = 1 - x$ the objective function can be minimized using the first order optimality condition.

$$z(x) = \frac{1}{p+1}x^{p+1} + 1 - x \implies z'(x) = x^p - 1$$
$$z'(x) = 0 \iff x = 1$$

And therefore $f^* = (x, y) = (1, 0)$. We could have argued this directly from the graph as well. Notice that since we only have to transport 1 total unit of traffic, and $p \geq 1$ the bottom edge will always be cheaper to choose for this unit of traffic than the top edge. Therefore, we can also expect this to give us a flow of 1 unit on the lower edge.

Combining these two results, the price of anarchy of this instance will be

$$\rho(I) = \frac{1}{1 - p(p+1)^{-(p+1)/p}} = \alpha(\mathfrak{C})$$

where $\mathfrak{C}$ is the set of polynomials of degree p. Thus, since we have shown this instance has price of anarchy equal to $\alpha(\mathfrak{C})$, the upper bound given in theorem 3.2 is tight. $\triangle$

4 Numerical computation of equilibria

In this section, we discuss two numerical algorithms to find user equilibria for instances of network congestion games. We first discuss the Frank-Wolfe algorithm, which is a simple algorithm to compute minima for convex problems. We discuss the implementation, convergence rate and drawbacks of using this method. After this, we consider an alternative method that adapts the plain Frank-Wolfe method to increase convergence speed. This adaptation still uses the fundamental ideas of the Frank-Wolfe algorithm, but combines it with a conjugate based method.

4.1 The Frank-Wolfe Algorithm

A commonly used algorithm to numerically compute the user equilibria of an instance of a routing game is the Frank-Wolfe algorithm [Gonzalez-Calderon et al., 2011]. Suppose we wish to compute the solution x to the problem:

$$\min_{x \in \mathcal{F}} f(x),$$

where $f(x)$ is assumed to be convex and $L-$smooth, and the feasible set $\mathcal{F}$ is a compact and convex subset of an inner product space $\mathcal{X}$, with inner product $\langle \cdot, \cdot \rangle$ [Jaggi, 2013, Frank and Wolfe, 1956]. We can use algorithm 1 to compute a solution x^*.

Algorithm 1 Frank-Wolfe algorithm with line search

```
Input: $x_0 \in \mathcal{F}, \epsilon$
  $k \leftarrow 0$
  while convergence criterion $> \epsilon$ do
    $s_k \leftarrow \arg\min_{y \in \mathcal{F}} \langle y, \nabla f(x_k) \rangle$
    $\alpha_k \leftarrow C$, with $C \in [0, 1]$
    $x_{k+1} \leftarrow x_k + \alpha_k (s_k - x_k)$
    $k \leftarrow k + 1$
  end while
  $x^* \leftarrow x_k$
Output: $x^*$
```

We go over the algorithm step by step.

As a first step, we need to find a descent direction. To find a descent direction, we solve a linear subproblem: We find the minimum of the linearization of f at the current iterate. By Taylor's theorem, we have that:

$$f(x) \approx f(x_k) + \nabla f(x_k) \cdot (x - x_k) = f(x_k) + \langle \nabla f(x_k), x - x_k \rangle,$$

up to a quadratic error term. To find the search direction, we minimize both sides of this relation to get an estimate of where the minimum of f is:

$$\begin{aligned}\arg\min_{x\in\mathcal{F}} f(x) &\approx \arg\min_{x\in\mathcal{F}} f(x_k) + \langle \nabla f(x_k), x - x_k\rangle \\ &= \arg\min_{x\in\mathcal{F}} f(x_k) + \langle x, \nabla f(x_k)\rangle - \langle x_k, \nabla f(x_k)\rangle \\ &= \arg\min_{x\in\mathcal{F}} \langle x, \nabla f(x_k)\rangle\,,\end{aligned}$$

where the last equality follows since $f(x_k)$ and $\langle x_k, \nabla f(x_k)\rangle$ do not depend on x. Therefore, these terms cannot influence where the minimum occurs. Setting $s_k := \arg\min_{x\in\mathcal{F}}\langle x, \nabla f(x_k)\rangle$ the descent direction is then $s_k - x_k$. Figure 8 shows this process [Jaggi, 2013].

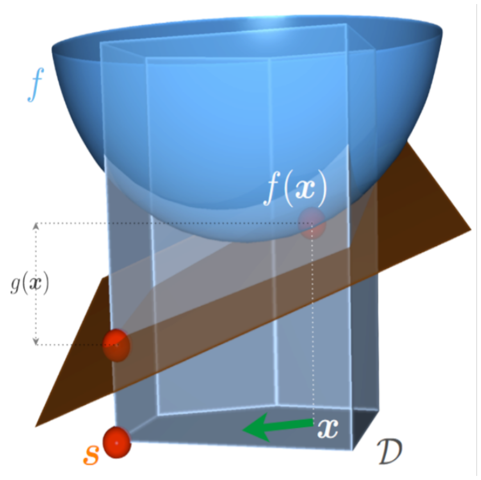


Figure 8: Visualization of the direction finding problem

After finding the desired search direction, we need to determine the step size to use. There is multiple ways to do this, that have slightly different speeds of convergence [Braun et al., 2022]

1. Line search: Find the step size α_k so that $\alpha_k = \arg\min_{t\in[0,1]} f(x_k + t(s - x_k))$. This guarantees we take the step size so we decrease our function the most. This method is in general more expensive to use, since an auxiliary problem needs to be solved.

2. Constant step size: Setting $\alpha_k = \frac{2}{k+3}$. This step might not give as good of a convergence speed the first option, however does have the advantage of a simpler implementation.

It is important to note that both of these methods enjoy the same convergence rate. Further algorithms for finding step sizes do exist, however the two methods above are simple and have an intuitive motivation on why to use them (see for example [Bertsekas, 2016, appendix C]).

After having determined both a search direction and a step size, we update via $x_{k+1} = x_k + \alpha(s_k - x_k)$. Notice that since $s_k, x_k \in \mathcal{F}$, and $\alpha_k \in [0, 1]$, we know that x_{k+1} is also in F since it is a convex combination of two elements in the convex set $\mathcal{F}$. By induction $x_n \in \mathcal{F}$ for all $n \geq 0$.

As for the convergence rate of this algorithm, we have the following [Braun et al., 2022].

Theorem 4.1 (Convergence rate of the FW-algorithm). *Let $f : C \subseteq \mathcal{X} \to \mathbb{R}$ be a convex and $L-$smooth function, where the set C is a compact and convex subset of an inner product space $\mathcal{X}$, with $\operatorname{diam}(C) = D$. Then for any iterate x_k of algorithm 1 using step size $\alpha_0 = 1, \alpha_k = \frac{2}{k+3}$ for $k \geq 1$ or $\alpha_k = \frac{2}{k+2}$,*

$$f(x_k) - f(x^*) \leq \frac{2LD^2}{k+3}, \tag{4.1.1}$$

for all $k \geq 1$. That is, to achieve an accuracy of $\epsilon > 0$, the algorithm requires at least $\frac{2LD^2}{\epsilon} - 3$ iterations.

Proof. We only discuss the case $\alpha_0 = 1, \alpha_k = \frac{2}{k+3}$ here, since the other case is similar. By the L-smoothness of f, we have that:

$$f(x_{k+1}) - f(x_k) \leq \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{L}{2}\|x_{k+1} - x_k\|^2.$$

Using that $x_{k+1} = x_k + \alpha_k(s_k - x_k)$ this reduces to:

$$\begin{aligned} f(x_{k+1}) - f(x_k) &\leq \langle \nabla f(x_k), x_k + \alpha_k(s_k - x_k) - x_k \rangle + \frac{L}{2}\|x_k + \alpha_k(s_k - x_k) - x_k\|^2 \\ &= \alpha_k \langle \nabla f(x_k), s_k - x_k \rangle + \frac{\alpha_k^2 L}{2}\|s_k - x_k\|^2 \\ &\leq \alpha_k \langle \nabla f(x_k), s_k - x_k \rangle + \frac{\alpha_k^2 LD^2}{2}. \end{aligned}$$

Recalling that s_k minimizes $\langle \nabla f(x_k), x \rangle$ over all $x \in \mathcal{F}$, it must in hold that:

$$\langle \nabla f(x_k), s_k \rangle \leq \langle \nabla f(x_k), x^* \rangle$$

Combining this with the fact that $f(x^*) - f(x_k) \geq \langle \nabla f(x_k), x^* - x_k \rangle$ by the convexity of f, the chain of inequalities ultimately becomes:

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \alpha_k \langle \nabla f(x_k), s_k - x_k \rangle + \frac{\alpha_k^2 LD^2}{2} \\ f(x_{k+1}) &\leq f(x_k) + \alpha_k \langle \nabla f(x_k), x^* - x_k \rangle + \frac{\alpha_k^2 LD^2}{2} \\ f(x_{k+1}) - f(x^*) &\leq f(x_k) - f(x^*) + \alpha_k(f(x^*) - f(x_k)) + \frac{\alpha_k^2 LD^2}{2} \end{aligned}$$

Or equivalently:

$$\epsilon_{k+1} \leq (1 - \alpha_k)\epsilon_k + \frac{\alpha_k^2 LD^2}{2},$$

where $\epsilon_n = f(x_n) - f(x^*)$. Note that the above holds for any $\alpha_k \in [0, 1]$. We now prove the theorem by induction on k, using the above inequality.

For the base case when $k = 0$, taking $\alpha_0 = 1$ suffices. In this case. the above inequality gives that:

$$\epsilon_1 \leq \frac{LD^2}{2} = \frac{2}{3+1} LD^2$$

as the theorem suggests. Assume that the statement holds for some $k = t$. Then for $k = t+1$, the established inequality gives reads:

$$\epsilon_{t+1} \leq \left(1 - \frac{2}{t+3}\right) \epsilon_t + \frac{2}{(t+3)^2} LD^2,$$

By using the induction hypothesis, we find that:

$$\begin{aligned} \epsilon_{t+1} &\leq \left(1 - \frac{2}{t+3}\right) \frac{2LD^2}{t+3} + \frac{2}{(t+3)^2} LD^2 \\ &= \frac{2LD^2}{t+3} - \frac{4LD^2}{(t+3)^2} + \frac{2LD^2}{(t+3)^2} \\ &= 2LD^2 \left(\frac{t+2}{(t+3)^2}\right) = \frac{2LD^2}{t+4} \left(\frac{(t+2)(t+4)}{(t+3)^2}\right) \end{aligned}$$

It remains to show that the factor on the right is less than one. We have:

$$(t+2)(t+4) = t^2 + 6t + 8 < t^2 + 6t + 9 = (t+3)^2,$$

and so the fraction is indeed less than one. Therefore,

$$\epsilon_{t+1} \leq \frac{2LD^2}{(t+1)+3},$$

and so the statement also holds for $k = t+1$. By induction, the statement thus holds for all $k \geq 1$. Suppose we wish that ϵ_k is less than some $\epsilon > 0$. Then, we find that:

$$\epsilon \leq \frac{2LD^2}{k+3} \iff k \geq \frac{2LD^2}{\epsilon} - 3$$

Thus, to achieve an accuracy of ϵ, we require at least $\frac{2LD^2}{\epsilon} - 3$ iterations. □

Having established that the algorithm indeed converges to the desired minimum function value, let us now investigate how to solve the linear sub-problem to find the search direction. Recall from the proof of theorem 2.18 that if $Z(f)$ is the objective function to find a user equilibrium and f is indexed by the paths $(P_1, \ldots, P_k)$, then:

$$\nabla Z(f) = (c_{P_1}(f), \ldots, c_{P_k}(f)).$$

We can now state the following.

Proposition 4.2. *Let $Z(f)$ be the target function to find a user equilibrium as in theorem 2.18, corresponding* *to an instance $I = (G, r, c)$. Consider the problem:*

$$\min_{f \in \mathcal{F}} \langle f, \nabla Z(f_k) \rangle, \tag{4.1.2}$$

with the standard inner product on $\mathbb{R}^n$. Here $\mathcal{F}$ is the set of feasible flows, and f_k is an iterate of the Frank-Wolfe algorithm. Then, the solution to this problem is given by the all-or-nothing assignment based on the cost of the previous flow.

Proof. Writing out the full inner product, we find that:

$$\langle f, \nabla Z(f_k) \rangle = \sum_{P \in \mathcal{P}} \nabla Z(f_k) f_P = \sum_{P \in \mathcal{P}} c_P(f_k) f_P.$$

Notice now that this is the cost of a flow f for an instance where the cost of each path is fixed as $c_P(f_k)$. In this case, since each cost function is constant, the marginal cost is the same as the normal cost. Moreover, since the cost functions on each path are constant, so are the costs on each edge, and thus the costs $c_e(f_e)$ are constant[2]. This gives that each function $c_e(f_e)f_e$ is convex. By proposition 2.13, we find that a flow f^* is optimal for this instance if and only if for every commodity i and every pair of paths $P, \tilde{P} \in \mathcal{P}_i$ with the flow on $f^*_P > 0$, we have that $c^*_P(f^*) \leq c^*_{\tilde{P}}(f^*) \iff c_P(f_k) \leq c_{\tilde{P}}(f_k)$. Therefore, if f^* is optimal and routes *any* flow on the edge P it must strictly be cheaper to do this than take any other path. Noting that all costs are constant however, this completely rules out taking two different path to find an optimal flow (as long as their cost is different). Indeed, take two different paths P and $\tilde{P}$ with costs $c_P(f_k) < c_{\tilde{P}}(f_k)$ and suppose for contradiction that $f_P > 0$ *and* $f_{\tilde{P}} > 0$. By the above logic, this means that:

$$\begin{aligned} c_P(f_k) &\leq c_{P^*}(f_k), \\ c_{\tilde{P}}(f_k) &\leq c_{P^*}(f_k), \end{aligned}$$

for *all* paths $P^* \in \mathcal{P}_i$. However these two inequalities cannot both hold assuming that $c_P(f_k) < c_{\tilde{P}}(f_k)$, since both $P, \tilde{P}$ are elements of $\mathcal{P}_i$. Therefore, we cannot have two paths routing flow at the same time. Thus, the optimal flow will be the one that routes all traffic over the cheapest path based on the flow f_k. If we have two paths with the same cost, it does not matter which path we choose. Therefore, the solution to the problem in (4.1.2) is given as an all-or-nothing assignment for the cheapest path. □

After finding a suitable search direction we now need to find an optimal step size. Given an iterate x_k and a search direction s_k we wish to find α_k so that:

$$\alpha_k = \arg\min_{t \in [0,1]} Z\left(x_k + t(s_k - x_k)\right). \tag{4.1.3}$$

That is, the step size we compute gives the best decrease in our target function Z. Let $x_k \in \mathbb{R}^{|V| \times |V|}$ be the

[2]This implicitly uses the fact that the cost functions are increasing in f_e and so cannot even cancel out any contribution of any other edge.

matrix that contains the flow on each edge at iteration k of the Frank-Wolfe algorithm[3]. If f_e is the flow on edge $(i,j) \in E$, then $(x_k)_{(i,j)} = f_e$. Since $[0,1]$ is a compact set, by the same argument as given in theorem 2.18 the function $Z(x_k + t(s_k - x_k))$ achieves its minimum with respect to t. In this case, the function in equation $(4.1.3)$ can be minimized using the first order optimality condition (assuming $\alpha \in (0,1)$). Define $g(t) = Z(x_k + t(s_k - x_k))$. Then:

$$\begin{aligned} g'(t) = \frac{\mathrm{d}}{\mathrm{d}t}\left[Z(x_k + t(s_k - x_k))\right] &:= \frac{\mathrm{d}}{\mathrm{d}t}\left[Z(X_k(t))\right] = \sum_{(i,j)\in E} \frac{\mathrm{d}}{\mathrm{d}t}\int_0^{X_k(t)_{(i,j)}} c_{(i,j)}(z)\mathrm{d}z \\ &= \sum_{(i,j)\in E} c_{(i,j)}(X_k(t)_{(i,j)})(s_k - x_k)_{(i,j)}, \end{aligned}$$

where we denote the edges by their matrix entries (i,j). Using the bisection method, we can then find the minimum of this function $g'(t)$, to find the minimum of g. Of course, this may fail if there are multiple minima, or maxima. In our case however, this will not happen since we shall generally use standard polynomial functions for which we can guarantee a unique minimum. If the bisection method fails to find a root, the minimum will occur oat $\alpha = 0$ or $\alpha = 1$. This is a simple extra test case that only requires two function evaluations. In particular, if $\alpha = 0$, we have that the current iterate x_k is already the minimum value of Z.

Another question to ask ourselves is what an appropriate convergence criterion might be. For this, we consider the *relative gap* at each iteration. The relative gap is defined to be the following [Mitradjieva and Lindberg, 2013]:

$$RG_k = \frac{UB_k - BLB_k}{|BLB_k|}, \tag{4.1.4}$$

where

$$\begin{aligned} UB_k &= Z(x_k + \alpha_k s_k), \\ BLB_k &= \max\{BLB_{k-1}, Z(x_k) + \langle \nabla Z(x_k), s_k - x_k\rangle\} := \max\{BLB_{k-1}, L_k(s_k)\} \end{aligned}$$

where L_k is the linearization of Z at the current iterate x_k. Suppose that x^* is a minimum of the function Z. Then, we have that $Z(x^*) \le Z(x_k + \alpha_k s_k) = Z(x_{k+1})$ for any iterate x_{k+1}. Therefore, the function values at the iterates x_k will always form an upper bound to the minimal function value. Furthermore, we see that since s_k is a solution to $(4.1.2)$, it follows that:

$$\begin{aligned} \langle \nabla Z(x_k), s_k - x_k\rangle &= \langle \nabla Z(x_k), s_k\rangle - \langle \nabla Z(x_k), x_k\rangle \\ &\le \langle \nabla Z(x_k), x_k\rangle - \langle \nabla Z(x_k), x_k\rangle = 0. \end{aligned}$$

[3]We transform our problem into computing edge flows, since by proposition 2.19 the assignment of a user equilibrium is unique with respect to the edge flows. We store the edge flows in a matrix, similar to a standard adjacency matrix of a graph.

This then implies that for the linearization used in the BLB_k term,

$$L_k(s_k) = Z(x_k) + \langle \nabla Z(x_k), s_k - x_k \rangle \leq Z(x_k) \leq Z(x^*).$$

Therefore, $L_k(s_k)$ forms a lower bound for the optimal function value. The relative gap thus represents the relative difference between the best found linearization value and the function value at the current iterate. In particular, by multiplying the relative gap by 100%, we find the percentage increase from the best lower bound to the current upper bound. As we move towards the minimum value, we therefore expect the relative gap to vanish. A sufficient criterion for convergence when comparing networks is the requirement that $RG_k < 10^{-4}$ [Boyce et al., 2004].
The full algorithm can now be outlined in a the following steps [Gonzalez-Calderon et al., 2011]

1. Import the instance; Import the graph with the cost functions on each edge, the origin-destination pairs and the traffic rates. The cost functions are stored in a cost matrix. This matrix is similar to the adjacency matrix of a graph, except we have the cost on each edge as its entries. That is:

$$C_{(i,j)} = \begin{cases} c_{(i,j)}(x) & \text{if } (i,j) \in E \\ 0 & \text{else,} \end{cases}$$

   where C is the cost matrix.

2. Find all paths between the given OD-pairs.

3. Start the Frank-Wolfe algorithm by finding a first feasible vector based on an all-or-nothing assignment based on zero flow. That is, x_0 is formed by taking all the cheapest path assuming there is no flow on each edge.

4. Update a separate cost matrix that stores the current cost on each edge.

5. While the relative gap is more than a given tolerance we do the following steps:

6. Perform an all-or-nothing assignment based on the cost of the previous iteration to find a search direction s_k

7. Find the step size that minimizes $Z(x_k + \alpha(s_k - x_k))$, by using the bisection method. Else, use a static step size.

8. Update $x_{k+1} = x_k + \alpha_k(s_k - x_k)$ and update the cost matrix to correspond with the new edge flow. Increase the iteration counter and return to step 6.

To compute the system optimal flow, we recall an earlier result. We can use proposition 2.16 and compute the user equilibrium of the instance (G, r, c^*), which corresponds to the system optimum of the original instance. Using both the user equilibrium and the system optimal flow, we can then compute the price of anarchy of an instance as well.

To check the correctness of our implementation, we compute the price of anarchy for the nonlinear Pigou instance from example 3.13. The code is implemented on Python and can be viewed at the link given here [4].

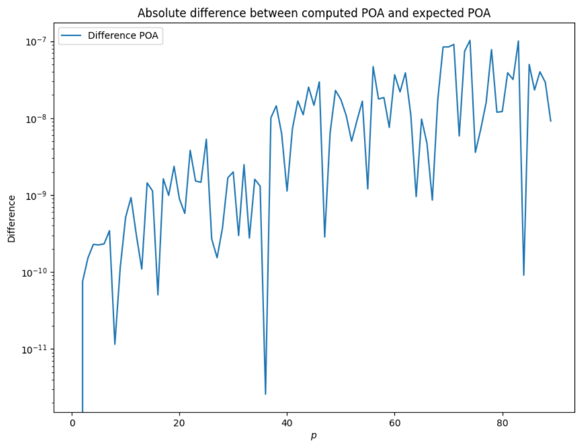


Figure 9: Error of the numerically computed price of anarchy and the analytically computed price of anarchy

We see that the difference is very small for most values of p, which gives us assurance our implementation of the algorithm is correct.

4.2 Conjugate Frank-Wolfe

While the above gives promising results, there is a drawback to using the above algorithm. The following example highlights this issue.

Example 4.3. Consider the least squares problem of finding a vector $x \in \mathbb{R}^N$ so that the distance $\|w - x\|^2$ is minimal, for some other point $w \in \mathbb{R}^N$ [Lawphongpanich et al., 2009]. Suppose we further constrain the problem so that $x \in X \subset \mathbb{R}^N$ for some convex polytope X. That is, X is the finite intersection of half-spaces. In particular, this implies that for any $x \in X$, we must satisfy some matrix inequality $Ax \leq b$ and, without loss of generality we can assume that $x \geq 0$ (otherwise we can flip signs in A and b).

Since we have that X is convex and $\nabla f(x) = -2(w - x)$ and $\nabla^2 f(x) = 2I \geq 0$, we have a convex problem. Suppose we wish to find the solution using the above proposed Frank-Wolfe algorithm. At iteration k, we have found an approximate solution x_k. To find x_{x+1} we need to find the next search direction. To do this, we find the solutions s_k to the problem:

$$\min_y \nabla f(x_k) \cdot y = -2(w - x_k) \cdot y$$
$$\text{subject to } Ay \leq b, y \geq 0.$$

Noticing that this is the standard form for a linear program, we know the solution has to occur at a vertex of our polytope [Pardalos, 2009]. Geometrically, the solution vertex is the vector that gives the smallest dot

[4]https://colab.research.google.com/drive/1MYsnkSm3Ky2zB3k03fViSCRhwVZyc2fL?usp=sharing

product with the vector $-2(w-x)$. By noting $u \cdot v = ||u||\,||v||\cos(\theta)$, this means we wish to have the cosine of the angles between these two vectors as negative as possible, which occurs if the vectors are pointing in opposite directions and the norm is as small as possible.

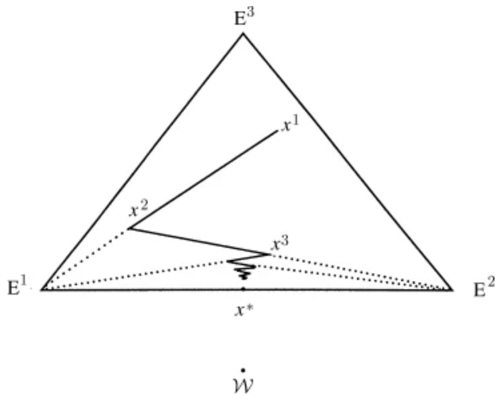


Figure 10: Search directions for the sketched problem, leading to zigzagging.

In figure 10 this process is drawn out for a polygon with extreme points E_1, E_2 and E_3. As we can see, the solution x^* is the projection of w onto the polytope. This leads to excessive zigzagging since we choose the search direction in opposite directions successively. This can lead to very slow convergence when the solution is near the boundary. $\triangle$

To remedy this problem, one can implement many different variations on the original algorithm from the previous section. Examples can be found in [GuéLat and Marcotte, 1986, Arezki and Van Vliet, 1990]. The first paper proposes a method that is based on allowing negative backtracking steps, by generating a direction opposite to the Frank-Wolfe step normally generated, while the second uses the corners of the trajectory like in a momentum based method.

In this paper we discuss a more recent method based on a conjugate gradient-like iteration [Mitradjieva and Lindberg, 2013]. First, we define what it means for two vectors to be conjugate.

Definition 4.4 (Conjugate vectors). Two vectors $u, v \in \mathbb{R}^N$ are *conjugate* with respect to the matrix $A \in \mathbb{R}^{N\times N}$ if $u^T Av$ = 0. If A is positive definite, the map $\langle \cdot, \cdot \rangle_A : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}, (u, v) \mapsto u\,^T Av$, forms an inner product on $\mathbb{R}^N$.

When performing a conjugate gradient method, we generate a set of search directions $\{v_1, \dots, v_n\}$, so that each of these directions is conjugate with respect to the last. That is, we look for constants β_k so that $v_k = \nabla f(x) + \beta_k v_{k-1}$. The constant β_k gives us that v_k is conjugate to v_{k-1} with respect to the Hessian matrix of f. If the conjugate gradient method is applied to a quadratic form $x^T Ax - b^T x$ for some positive definite matrix $A \in \mathbb{R}^{N\times N}$, the method will converge in at most N steps. We therefore make a quadratic approximation to the general function f, which leads to the choice of making the vectors conjugate with respect to the Hessian H [Luenberger and Ye, 2016].

In our case, we get the added benefit that our Hessian is a diagonal matrix with respect to the edge flow variables:

$$Z(f_1,\ldots,f_n) = \sum_{i=1}^{n} \int_0^{f_i} c_i(t)\mathrm{d}t.$$

$$(\nabla Z)_j = c_j(f_j), \quad (\nabla^2 Z)_{(j,k)} = \begin{cases} c_j^{'}(f_j) & \text{if } j = k \\ 0 & \text{if } j \neq k. \end{cases}$$

The simplicity of the Hessian can help with efficient computation of vectors Hv by exploiting the diagonal structure of H.

To apply this method to our problem we follow the scheme proposed by Mitradjieva and Lindberg [Mitradjieva and Lindberg, 2013]. Suppose we have found the current iterate x_k, using the rule

$$x_k = x_{k-1} + \tau_{k-1} d_{k-1}^{CFW},$$

where $d_{k-1}^{CFW} = s_{k-1}^{CFW} - x_{k-1}$ is the descent direction and τ_{k-1} is the step size. Moreover, suppose that at each iteration we still solve the linear sub-problem to find the "Frank-Wolfe direction" vector y_k^{FW}. For this method, we wish to find a new descent direction

$$d_k^{CFW} = d_k^{FW} + \beta_k d_{k-1}^{CFW}. \tag{4.2.1}$$

Here the constant β_k makes the new direction conjugate to the previous with respect to the Hessian of the objective function evaluated at the current iterate x_k. Let $\bar{d}_{k-1}$ denote the residual search direction: $\bar{d}_{k-1} = (1-\tau_{k-1})d_{k-1}^{CFW}$. Up to changing our step size τ_k, we may choose $d_k^{CFW} = s_k^{CFW} - x_k$, for some $s_k^{CFW} = \alpha_k s_{k-1}^{CFW} + (1-\alpha_k)y_k^{FW}$. We can see this in figure 11 by constructing d_k^{CFW} as in equation $(4.2.1)$. This leads to a vector in the same direction as s_k^{CFW}. By appropriately scaling τ_k, we can choose s_K^{CFW} to be on the line between the previous search direction and the Frank-Wolfe point.

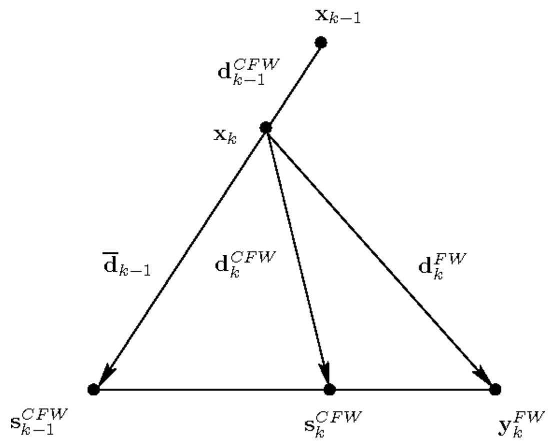


Figure 11: Construction of the descent direction for the conjugate Frank-Wolfe method

This choice leads to a descent direction as follows:

$$\begin{aligned}
d_k^{CFW} &= s_k^{CFW} - x_k = \alpha_k s_{k-1}^{CFW} + (1-\alpha_k) y_k^{FW} - x \\
&= \alpha(s_{k-1}^{CFW} - y_k^{FW}) + y_k^{FW} - x_k = \alpha(s_{k-1}^{CFW} - y_k^{FW}) + d_k^{FW} \\
&= \alpha_k(s_{k-1}^{CFW} - x_k) + (1-\alpha) d_k^{FW} \\
&= \alpha_k \bar{d}_{k-1} + (1-\alpha_k) d_k^{FW} = d_k^{FW} + \alpha_k(\bar{d}_{k-1} - d_k^{FW})
\end{aligned}$$

It now remains to find a suitable α_k. Recall that we wish to make d_k^{CFW} and d_{k-1}^{CFW} conjugate. From figure 11 it suffices to make d_k^{CFW} and $\bar{d}_{k-1}$ conjugate, since d_{k-1}^{CFW} is parallel to d_{k-1}^{CFW}. Thus we get that:

$$0 = \bar{d}_{k-1}^T H_k d_k^{CFW}.$$

By using the derived expression for d_k^{CFW} we can isolate the conjugate step size α_k:

$$\begin{aligned}
0 &= \bar{d}_{k-1}^T H_k d_k^{CFW} \\
&= \bar{d}_{k-1}^T H_k (d_k^{FW} + \alpha_k(\bar{d}_{k-1} - d_k^{FW})) \\
\alpha_k \bar{d}_{k-1}^T H_k (d_k^{FW} - \bar{d}_{k-1}) &= \bar{d}_{k-1}^T H_k d_k^{FW} \\
\alpha_k &= \frac{\bar{d}_{k-1}^T H_k d_k^{FW}}{\bar{d}_{k-1}^T H_k (d_k^{FW} - \bar{d}_{k-1})} := \frac{N_k}{D_k}
\end{aligned}$$

This however does not tell the full story. For one that it is possible for $\bar{d}_{k-1} = 0$ vanish, which occurs when $\tau_{k-1} = 1$. Then, then $x_k = s_{k-1}^{CFW}$, and thus $\bar{d}_{k-1} = 0$. In this case, α_k would be undefined. To get around this problem, we slightly alter the definition of α_k. Let $0 < \delta < 1$ be some small constant. We set:

$$\alpha_k = \begin{cases} \frac{N_k}{D_k} & \text{if } D_k \neq 0, \frac{N_k}{D_k} \in [0, 1-\delta] \\ 1-\delta & \text{if } D_k \neq 0, \frac{N_k}{D_k} > 1-\delta \\ 0 & \text{else.} \end{cases} \tag{4.2.2}$$

If $D_k = 0$, we find that either $\bar{d}_{k-1} = 0$ or that $d_k^{FW} = \bar{d}_{k-1}$. In both of these cases, we do not need an extra conjugate step. In the first case, the direction is trivially conjugate to the previous direction. In the other case, we have that by the descent direction derivation $d_k^{CFW} = d_k^{FW}$, which we would like to avoid in order to avoid the same zigzagging as before. Similarly, we wish to avoid $\alpha = 1$, since then $d_k^{CFW} = \bar{d}_{k-1}$ which gives us no new descent direction.

After determining the step size for the conjugate direction, the step size τ_k will still be determined by means of the bisection method. Implementing the method using $\delta = 0.01$, we apply the same test as we did for the normal Frank-Wolfe method. We compute the price of anarchy for the nonlinear Pigou example for different powers of p. The results are given in figure 12. Similar to the normal method, the difference is small and thus we can conclude that the method converges to the right solutions.

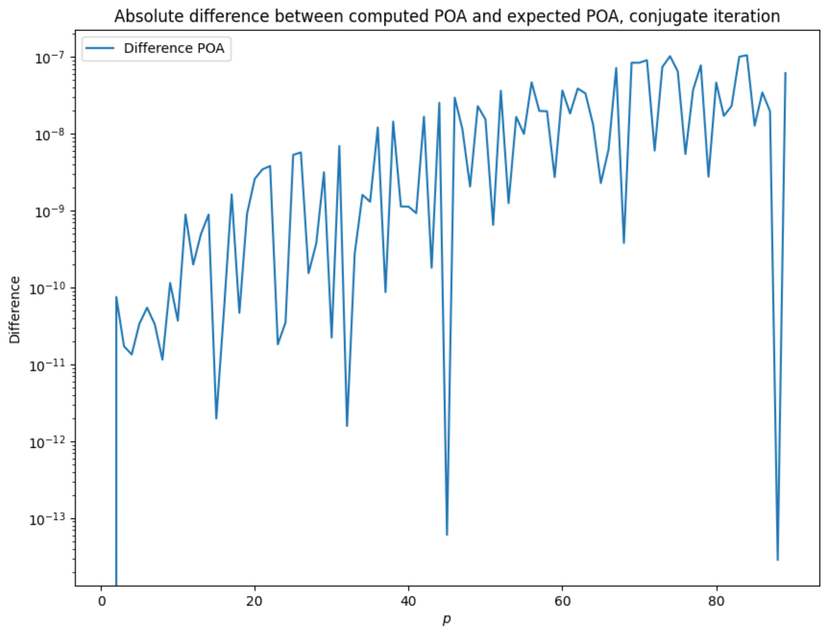


Figure 12: Difference between analytically computed price of anarchy and numerically found price of anarchy using the conjugate iteration

5 Case study: Morning traffic in Groningen

In this section, we want to apply all our gathered knowledge to analyse a model of Groningen, which simulates morning traffic going into the city and surrounding areas. First, we compare the convergence speed of the normal Frank-Wolfe algorithm and the conjugate Frank-Wolfe iterations. After this, we briefly analyse the results obtained and compute the price of anarchy of this network.

5.1 Description of the model

Suppose we take a graph network as shown in figure 13 (a large version of the network is given in appendix B along with full results of the simulations). The red nodes indicate either an origin node or a destination node, while the gray nodes indicate intersections. Based on the real life situation we distinguish three types of edges; a good edge with fast flowing traffic, a neutral edge, and a bad edge with slow traffic that can jam easily.

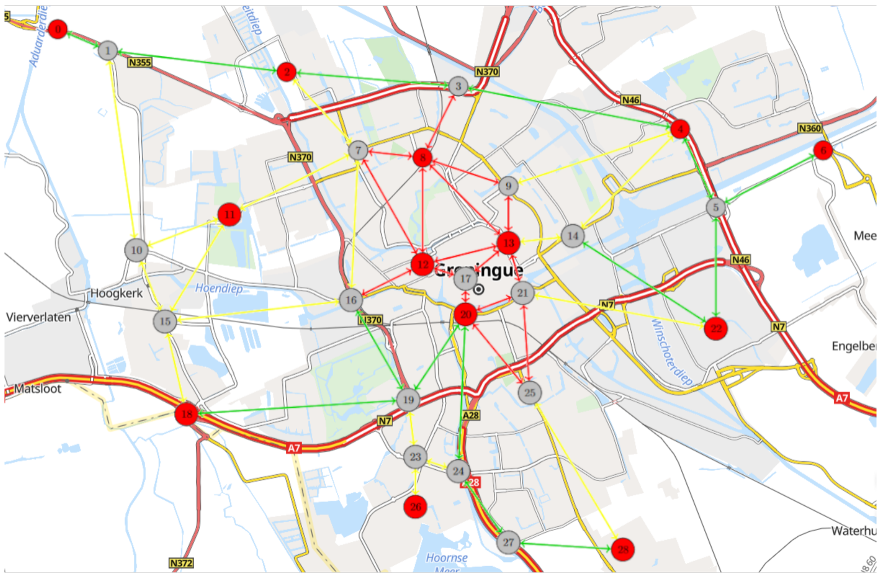


Figure 13: Graph network laid over map of Groningen [Michelin, 2023]

To find a suitable model for the roads in these edges we use a standard function that describes traffic flow as given by the United States Bureau of Public Roads [United States. Bureau of Public Roads, 1964]:

$$c(x) = c_0 \left(1 + \alpha \left(\frac{x}{\rho}\right)^{\beta}\right). \tag{5.1.1}$$

As a standard, we take $\alpha = 0.15$ and $\beta = 4$. In this equation, the constants c_0 and ρ represent the base travel time at zero flow and the practical capacity of the road respectively. At flow $x = \rho$ the cost sees an uptick after which the cost gets higher and higher. This point can be seen as the point at which traffic jams might start forming, or where people have to slow down to let traffic merge for example. A schematic plot can be found in figure 14. For each road, we adapt the above cost function slightly to represent the flow on each type of edge. To determine the cost at zero flow, we look at the speed limit prescribed on each of the roads and compute how long it would take to traverse a distance of 5 kilometres. This distance was chosen according to the approximate road length of each edge in figure 13. After this, we determine a suitable practical capacity, which we scale to the traffic rates we use. Of course, the outer ring road should not have as many traffic jams as the inner city roads do and therefore will have a differing practical capacity.

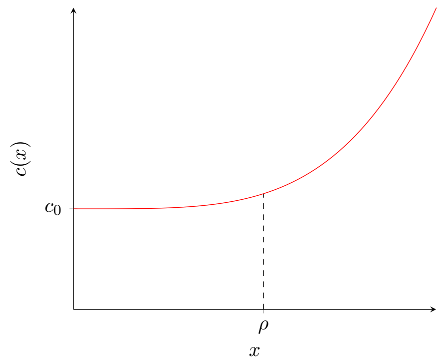


Figure 14: The cost function as in equation (5.1.1)

If we assume the good edge has an average speed of $100\,\mathrm{km\,h^{-1}}$, the neutral edge an average speed of $60\,\mathrm{km\,h^{-1}}$ and the bad edge an average speed of $30\,\mathrm{km\,h^{-1}}$, we can derive the cost at no flow from this. Of course, not every road is the exact same. To compensate for this we introduce two normally distributed random variables C_1 and C_2 that will act as multiplicative factors for both the base travel time and the practical capacity. In particular, we will take $C_1, C_2 \sim \mathcal{N}(\mu, \sigma) = \mathcal{N}(1, 0.05)$. That is, we assume that C_1 and C_2 follow a normal distribution centered at 1, with standard deviation 0.05. These constants give that $\mathbb{P}(0.9 \leq C \leq 1.1) \approx 0.95$. Therefore, the cost functions we use are:

$$\begin{aligned} c_{\text{good}}(x) &= 3C_1\left(1 + 0.15\left(\frac{x}{700C_2}\right)^4\right) \\ c_{\text{medium}}(x) &= 5C_1\left(1 + 0.15\left(\frac{x}{650C_2}\right)^4\right) \\ c_{\text{bad}}(x) &= 10C_1\left(1 + 0.15\left(\frac{x}{600C_2}\right)^4\right). \end{aligned} \tag{5.1.2}$$

To arrive at workable model we make a few assumptions. First, we assume that the maximum length of a path between origin-destination pairs is 8 vertices long (including the beginning and end). We do this, since otherwise we would get paths that give unrealistic shapes. One would never cross the city two or three times just to end up at a neighbouring vertex.

We generate the rate demand vectors as follows: Define the vertices $0, 4, 6, 11, 18$ and 28 to be origin vertices and vertices 2, 8, 12, 13, 20, 22 and 26 to be destination vertices. From these two sets we make all possible pairs, which forms the set of OD-pairs:

$$D = \{(0, 2), (4, 2), \ldots, (18, 26), (28, 26)\} \tag{5.1.3}$$

The rate between assigned to each of these pairs is based on traffic data gathered by the municipality of Groningen [Team OpenData Groningen, 2023]. We use the locations given in the data file, and assume one third of all traffic listed visits in the morning. That is, (in order of origin vertex):

$$r = (1200, 1600, 1400, 600, 4000, 3000) \tag{5.1.4}$$

Each origin destination pair then receives an equal amount of traffic. Since there are 7 destination vertices, we divide all above traffic numbers by 7 and distribute them equally across each pair. After computing all the paths, we have that we are looking at a model with 94 variables (the edge flows), subject to 94 non-negativity constraints and 42 flow conservation constraints, which have 1653 path flow variables.

5.2 Numerical results

First, let us compare the convergence speeds of the normal Frank-Wolfe method to that of the conjugate Frank-Wolfe method. According to [Mitradjieva and Lindberg, 2013], when using standard test networks the conjugate iteration performed better by a factor of about 10. When comparing this to the network described above, we find a similar result. The progression of the relative gap is given in figure 15. The conjugate iteration finds an equilibrium in 200 iterations, while the normal iteration reaches the same relative gap in 4226 iterations. This is a factor of approximately 21 better. When comparing the maximum absolute difference between the two flows we find an error of approximately 7.9.

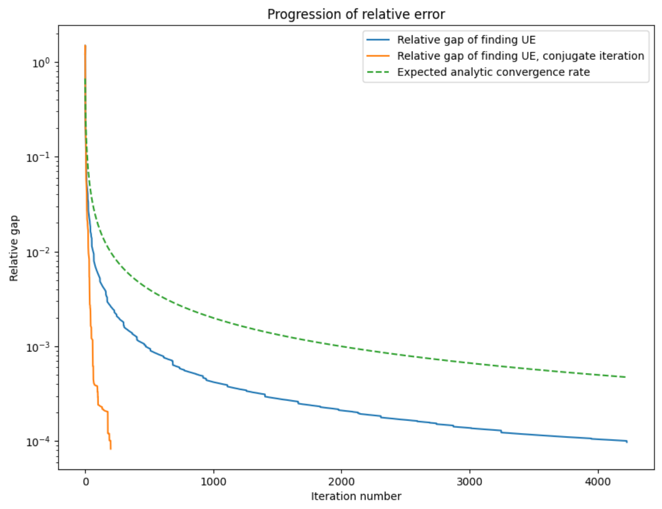


Figure 15: Error comparison CFW and FW method

Concluding that the conjugate method is indeed much faster, while not being very different from the standard iteration we proceed the analysis with this method. Zooming in on the center of town, we visualize the results in figure 16. We draw the edge flows with a linear color gradient with a fully green edge being the lowest edge flow and a fully red having the highest edge flow. The thickness is similar except here we display the edge cost. Therefore, the best case scenario is a thin green arrow, while the worst case scenario is a big red arrow.

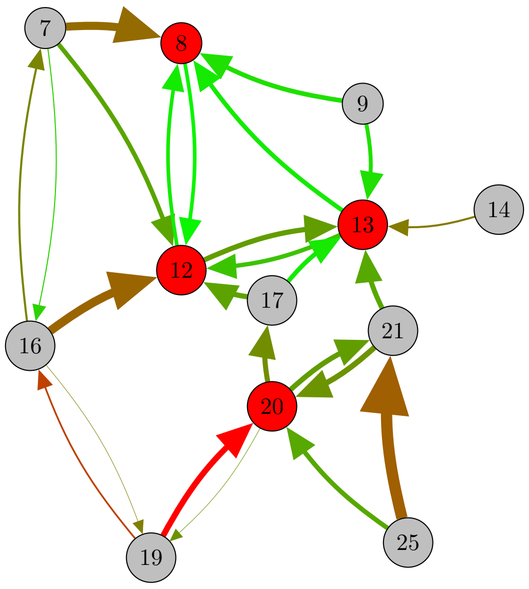


Figure 16: User equilibrium for the center of town

From this, we can see that (relative to the other edges in the center of town) especially the edges between vertices 25 and 21, 16 and 12, and 7 and 8 are pretty congested. We note that not much can be done about the edges to vertices 8 and 12, since these are destination vertices. One could think about altering the network a little so that some traffic from node 25 is diverted.

Computing the total cost of the user equilibrium flow, we have that:

$$C(f) = \sum_{e\in E} c_e(f_e)f_e \approx 655,579.23$$

To compute the system optimal flow, we apply proposition 2.16. To this end, we compute the marginal cost functions. For the general cost function as given in (5.1.1), we have that:

$$\begin{aligned} c^*(x) &= \frac{\mathrm{d}}{\mathrm{d}x}\left(xc(x)\right) = \frac{\mathrm{d}}{\mathrm{d}x}\left(c_0\left(x + \alpha\frac{x^{\beta+1}}{\rho^\beta}\right)\right) \\ &= c_0\left(1 + \alpha(\beta+1)\left(\frac{x}{\rho}\right)^\beta\right) \end{aligned}$$

We then apply the same algorithm to the new instance $\tilde{I} = (G, r, c^*)$. Again, the results can be found in appendix B. If we look at the same image of the city center as in figure 16 we see that there are slight differences in the distribution of traffic.

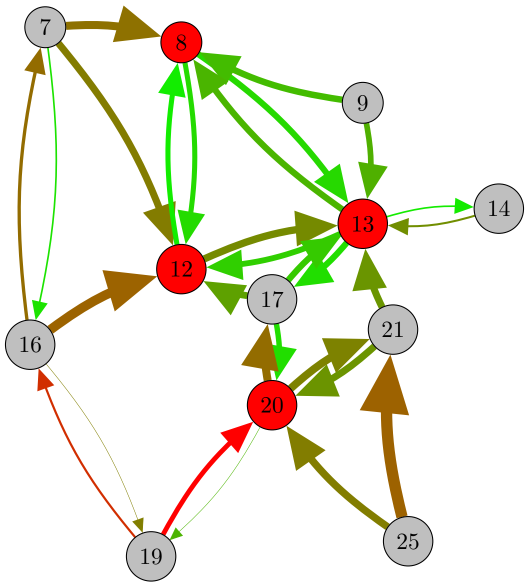


Figure 17: System optimum for the center of town

For one, the system optimal solution depicted in figure 17 uses two more edges that the user equilibrium. Moreover, the arrows are generally larger than in the UE case, which could imply that there is a move even distribution of traffic. Indeed, looking at the results in table 2 compared to table 1 we can see that some of the edge flows in the UE case are much larger, while some do not differ that much. If we compute the net flow change for this subgraph, we find that this comes down to a difference of approximately -595.12. That is, the flow is a net 595 higher in the system optimal case, when we use the convention that a change towards the user equilibrium is positive and a shift towards the system optimal case is negative. Intuitively, this makes sense. In the user equilibrium case, it is beneficial to try to avoid high cost edges as much as possible. In the center of town there are a lot of small, tight roads which have a "worse" cost function in the sense that the cost increases more for a small increase in traffic compared to a better cost function. The system optimal case will likely involve spreading traffic more equally, which would also need to put more traffic on paths a selfish player would like to avoid.

Computing the cost of the optimal flow like before we find that:

$$C(f^*) = \sum_{e \in E} c_e(f_e^*) f_e^* \approx 640,538.16$$

which is about 15,000 lower than in the user equilibrium case. Computing the price of anarchy for the network, we find that:

$$\rho = \frac{C(f)}{C(f^*)} = \frac{655,579.23}{640,538.16} \approx 1.02.$$

That is, the cost of a user equilibrium is approximately 2% worse than that of a system optimal flow. We notice that this is in fact a lot better than the worst-case bound provided by equation

Finally, we compare the error progression of computing the system optimum and the user equilibrium.

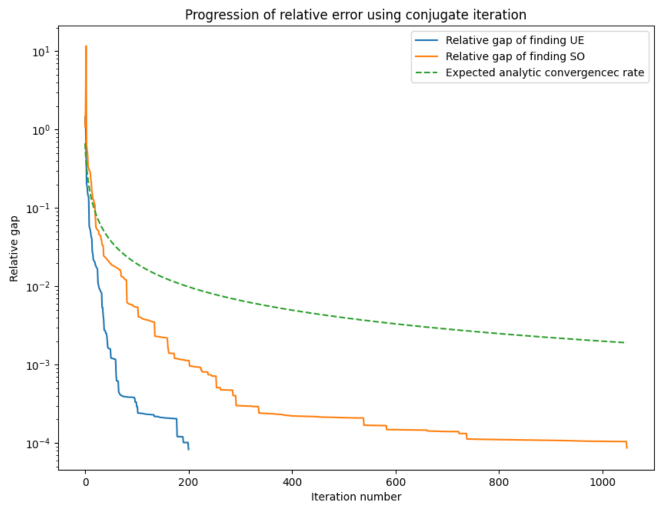


Figure 18: Error progression when computing a user equilibrium

We see that the same convergence rate $\mathcal{O}\left(\frac{1}{k}\right)$ persists, like we found in theorem 4.1 for the system optimal case. This is further backed up by the results found in the paper by Mitradjieva and Lindberg [Mitradjieva and Lindberg, 2013].

A possible reason for the large difference in iteration count could be the following. Recall from theorem 4.1 that we have that:

$$f(x_k) - f(x^*) \leq \frac{2LD^2}{k+3}.$$

Note that D does not change between the two problems, since the constraints are the exact same. The only thing we changed is the cost function. Therefore, we expect the difference to come from an increase in the smoothness constant L. Indeed, recalling that the gradient of the objective function is just the vector of cost functions, we find that:

$$\begin{aligned}(\nabla Z(x) - \nabla Z(y))_i &= c_{0,i}\left[1 + \alpha_i\left(\frac{x_i}{\rho_i}\right)^{\beta_i}\right] - c_{0,i}\left[1 + \alpha_i\left(\frac{y_i}{\rho_i}\right)^{\beta_i}\right] \\ &= c_{0,i}\alpha_i\frac{x_i^{\beta_i} - y_i^{\beta_i}}{\rho_i}.\end{aligned}$$

Notice that this function is L-smooth for some L, as the feasible set $\mathcal{F}$ is compact and the objective function Z is of smoothness class C^∞. By theorem A.10 there exists some L so that the function is L-smooth on $\mathcal{F}$

Furthermore, notice that:

$$
\begin{aligned}
(\nabla Z^*(x) - \nabla Z^*(y))_i &= c_{0,i}\left[1 + \alpha_i(\beta_i+1)\left(\frac{x_i}{\rho_i}\right)^{\beta_i}\right] - c_{0,i}\left[1 + \alpha_i(\beta_i+1)\left(\frac{y_i}{\rho_i}\right)^{\beta_i}\right] \\
&= c_{0,i}\alpha_i(\beta_i+1)\frac{x_i^{\beta_i} - y_i^{\beta_i}}{\rho_i}.
\end{aligned}
$$

Therefore,

$$\|\nabla Z^*(x) - \nabla Z^*(y)\| = (\beta_i+1)\|Z(x) - Z(y)\| \leq (\beta_i+1)L.$$

By the same argument as for implies that the marginal objective function is $(\beta_i+1)L$-smooth. By numerically checking this by taking random vectors $x, y \in \mathcal{F}$, and computing

$$\frac{\|\nabla Z(x) - \nabla Z(y)\|}{\|x - y\|}$$

for both Z and Z^*. We can then take the largest value of this, and this should give a good approximation of L. If we compute the ratio between these factors, we indeed find that this ratio is approximately equal to $4 + 1 = \beta + 1$. By applying our error bound, we can expect that the convergence could be 5 times as slow. Comparing the number of iterations, we find that this ratio is about the same, namely, $\frac{1047}{200} \approx 5.2$. Of course, the error bound only gives an upper bound, and thus in fact the actual result might be better. The above analysis does however give a plausible reason as to why finding a system optimal flow might be slower than a user equilibrium when using the same network and demand vectors.

6 Conclusion

In this thesis, we developed a broad mathematical foundation for the field of non-atomic congestion games. We did this by answering three fundamental questions: How to compute selfish and optimal flows, how bad are selfish flows compared to optimal flows and how can we numerically compute these flows? After introducing basic notation, we focused on the computation of both user equilibria and system optimal flows. We combined well known results to give a condensed, yet detailed derivations of the optimization problems related to computing these equilibria. Furthermore, existence and uniqueness of these problems was established.

Having determined how to compute both the optimal and user equilibrium flows, we discussed the results of Roughgarden on upper bounds on the price of anarchy in polynomial instances. Along the way, a few technical lemmas were established relating the costs of optimal flows and the costs of user equilibria.

Finally, we analysed a standard numerical method to compute selfish flows in network congestion games. The Frank-Wolfe algorithm was shown to have a linear rate of convergence. This was then compared to the conjugate version of the same algorithm. Consistent with the literature, we found that this gives a large improvement over the standard method, while maintaining the same convergence rate. To test this, a model of Groningen was implemented and briefly analyzed.

6.1 Further research suggestions

As noted, this thesis only gives a broad foundation in the topic of network congestion games. Therefore, there are a lot of different topics that we did not discuss here.

A first topic that could be looked at further is the extension of the work presented here to atomic congestion games. Like mentioned in section 2, in an atomic congestion game we consider a finite number of players which each have a larger share of traffic they control. This is in contrast to the case we consider in this thesis, where we have an infinite number of players. It is known that there exist similar optimization problems for finding a user equilibrium in these games. One can then also look at finding upper bounds on the inefficiency of these optima [Nisan et al., 2007].

Building on this, in both atomic and non-atomic congestion games, there is still current research going on in probability based assignment and more generally in probability based instances. One extension of the work proposed in this thesis to this framework is that of a stochastic user equilibrium [Cominetti et al., 2012]. Here, we consider the cost of an edge to be modelled as $\tilde{c}_e = c_e + \varepsilon_e$ where ε_e is a random variable such that $\mathbb{E}(\epsilon_e) = 0$. The path flow for on a path $P \in \mathcal{P}_i$ corresponding to some origin-destination pair d_i is then defined as:

$$f_P = r_i \, \mathbb{P}(\tilde{c}_P \leq \tilde{c}_{\tilde{P}}, \, \forall \tilde{P} \in \mathcal{P}_i).$$

One can then again examine existence and uniqueness in these games. It could be of interest to look at the price of anarchy in these instances. Another further extension of this can be by looking at Bernoulli distributed rate demands as done in [Cominetti et al., 2023]. In these types of games, we consider a standard atomic congestion game, where each player participates with a given probability p_i. In the paper it is shown that we can still extract information about the price of anarchy. One could then consider extending this to inhomogeneous probabilities.

After this brief discussion on atomic congestion games, there is also a lot of work that can still be done in other fields. One of which is that of taxation and subsidizing. As was seen in section 3, the price of anarchy can be unbounded for general instances of polynomial routing games. Currently, research is done in applying extra costs, or subsidizing players when using certain edges. The goal of this is to guarantee the price of anarchy of the actual network does not tend towards the upper bound, but more towards the optimal factor of 1. In [Brown and Marden, 2018] it is shown that it is theoretically possible to steer a user equilibrium arbitrarily close towards the cost of a system optimal flow. One could then investigate how to find a robust method to find an optimal toll design, while also taking into account the willingness of people to pay these tolls.

As with any topic in numerical analysis there is always a search for faster or better numerical algorithms. This is no different here. In this thesis, we discussed the (conjugate) Frank-Wolfe methods, which are examples of path-based assignment methods. There exist examples of origin-based models that might scale better (in terms of memory usage), since the number of origin-destination pairs is in general smaller than the number of paths in a network [Bar-Gera, 2002]. While adaptations of this formulation were shown to not perform better on the test networks considered in the paper establishing the conjugate Frank-Wolfe method, it is worth investigating if this formulation indeed leads to better memory usage or if there is an adaptation possible that yields better or similar performance.

References

[Abbott, 2015] Abbott, S. (2015). *Understanding Analysis*. Springer New York, New York, NY.

[Angelelli et al., 2021] Angelelli, E., Morandi, V., Savelsbergh, M., and Speranza, M. (2021). System optimal routing of traffic flows with user constraints using linear programming. *European Journal of Operational Research*, 293(3):863–879.

[Arezki and Van Vliet, 1990] Arezki, Y. and Van Vliet, D. (1990). A full analytical implementation of the PARTAN/Frank–Wolfe algorithm for equilibrium assignment. *Transportation Science*, 24(1):58–62.

[Bagloee et al., 2019] Bagloee, S. A., Ceder, A., Sarvi, M., and Asadi, M. (2019). Is it time to go for no-car zone policies? Braess paradox detection. *Transportation Research Part A: Policy and Practice*, 121:251–264.

[Bar-Gera, 2002] Bar-Gera, H. (2002). Origin-based algorithm for the traffic assignment problem. *Transportation Science*, 36(4):398–417.

[Beckmann et al., 1955] Beckmann, M. J., McGuire, C. B., and Winsten, C. B. (1955). *Studies in the Economics of Transportation*. RAND Corporation, Santa Monica, CA.

[Bertsekas, 2016] Bertsekas, D. P. (2016). *Nonlinear programming*. Athena Scientific, Belmont, Massachusetts, Third edition.

[Boyce et al., 2004] Boyce, D., Ralevic-Dekic, B., and Bar-Gera, H. (2004). Convergence of traffic assignments: How much is enough? *Journal of Transportation Engineering*, 130(1):49–55.

[Braun et al., 2022] Braun, G., Carderera, A., Combettes, C. W., Hassani, H., Karbasi, A., Mokhtari, A., and Pokutta, S. (2022). Conditional gradient methods.

[Brown and Marden, 2018] Brown, P. N. and Marden, J. R. (2018). Optimal mechanisms for robust coordination in congestion games. *IEEE Transactions on Automatic Control*, 63(8).

[Cominetti et al., 2012] Cominetti, R., Facchinei, F., and Lasserre, J. B. (2012). *Chapter 1 Wardrop and Stochastic User Equilibrium*, pages 213–220. Springer Basel, Basel.

[Cominetti et al., 2023] Cominetti, R., Scarsini, M., Schröder, M., and Stier-Moses, N. (2023). Ordinary and prophet planning under uncertainty in Bernoulli congestion games.

[Diestel, 2017] Diestel, R. (2017). *Graph Theory*. Springer Berlin Heidelberg.

[Díaz et al., 2017] Díaz, J., Giotis, I., Kirousis, L., Mourtos, I., and Serna, M. (2017). The social cost of congestion games by imposing variable delays. *ICT Express*, 3(4):155–159. SI: Intelligent Transportation Communication Systems.

[Epperson, 2013] Epperson, J. F. (2013). *An introduction to numerical methods and analysis*. John Wiley & Sons, Inc., Hoboken, New Jersey, Second edition.

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

[Frank and Wolfe, 1956] Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

[Gonzalez-Calderon et al., 2011] Gonzalez-Calderon, C., Calderón, G., and Posada-Henao, J. (2011). Solving the traffic assignment problem using real data for a segment of Medellin's transportation network. *Revista Facultad de Ingeniería Universidad de Antioquia*, pages 47–58.

[GuéLat and Marcotte, 1986] GuéLat, J. and Marcotte, P. (1986). Some comments on Wolfe's 'away step'. *Mathematical Programming*, 35(1):110–119.

[Jaggi, 2013] Jaggi, M. (2013). Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 427–435, Atlanta, Georgia, USA. PMLR.

[Kolata, 1990] Kolata, G. (1990). What if they closed 42d street and nobody noticed? *The New York Times*.

[Koutsoupias and Papadimitriou, 2009] Koutsoupias, E. and Papadimitriou, C. (2009). Worst-case equilibria. *Computer Science Review*, 3(2):65–69.

[Law et al., 2012] Law, L. M., Huang, J., and Liu, M. (2012). Price of anarchy for congestion games in cognitive radio networks. *IEEE Transactions on Wireless Communications*, 11(10):3778–3787.

[Lawphongpanich et al., 2009] Lawphongpanich, S., Floudas, C. A., and Pardalos, P. M. (2009). *Frank–Wolfe algorithm*, pages 1094–1097. Springer US, Boston, MA.

[Liu and Wu, 2008] Liu, M. and Wu, Y. (2008). Spectum sharing as congestion games. pages 1146 – 1153.

[Luenberger and Ye, 2016] Luenberger, D. G. and Ye, Y. (2016). *Conjugate Direction Methods*, pages 263–284. Springer International Publishing, Cham.

[Michelin, 2023] Michelin (2023 (accessed May 22, 2023)). Michelin map of Groningen. https://www.viamichelin.nl/.

[Mitradjieva and Lindberg, 2013] Mitradjieva, M. and Lindberg, P. O. (2013). The stiff is moving—conjugate direction Frank-Wolfe methods with applications to traffic assignment. *Transportation Science*, 47(2):280–293.

[Moklyachuk, 2020] Moklyachuk, M. P. (2020). *Convex optimization: introductory course*. ISTE, Ltd. ;, London.

[Nisan et al., 2007] Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V. (2007). *Algorithmic Game Theory*. Cambridge University Press.

[Pardalos, 2009] Pardalos, P. M. (2009). *Linear programmingLinear Programming*, pages 1883–1886. Springer US, Boston, MA.

[Petrosyan and Zenkevich, 1996] Petrosyan, L. A. and Zenkevich, N. A. (1996). *Game Theory.*, volume 3 of *Series on Optimization*. World Scientific.

[Pigou, 1920] Pigou, A. (1920). *The Economics of Welfare*. Macmillan, London.

[Pugh, 2015] Pugh, C. C. (2015). *Real Mathematical Analysis*. Springer Cham.

[Rosenthal, 1973] Rosenthal, R. W. (1973). The network equilibrium problem in integers. *Networks*, 3(1):53–59.

[Roughgarden, 2005] Roughgarden, T. (2005). *Selfish Routing and the Price of Anarchy*. MIT Press.

[Skiena, 2008] Skiena, S. S. (2008). *Graph Traversal*, pages 145–190. Springer London, London.

[Stewart et al., 2021] Stewart, J., Clegg, D., and Watson, S. (2021). *Calculus: Early Transcendentals*. Cengage Learning, Ninth edition.

[Team OpenData Groningen, 2023] Team OpenData Groningen (2022 (accessed June 12, 2023)). Telcijfers autoverkeer gemeente Groningen [Car traffic figures municipality of Groningen]. `https://data.groningen.nl/dataset/telcijfers-autoverkeer-gemeente-groningen`.

[United States. Bureau of Public Roads, 1964] United States. Bureau of Public Roads (1964). *Traffic Assignment Manual for Application with a Large, High Speed Computer*. Traffic Assignment Manual for Application with a Large, High Speed Computer. U.S. Department of Commerce, Bureau of Public Roads, Office of Planning, Urban Planning Division, 2 edition.

[Wardrop, 1952] Wardrop, J. G. (1952). Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, 1(3):325–362.

[Zhou and Yang, 2019] Zhou, B. and Yang, H. (2019). Rendering scheduling framework in edge computing: A congestion game-based approach. In *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, pages 1–6.

Appendices

A Preliminaries

A.1 Convex analysis and constrained optimization

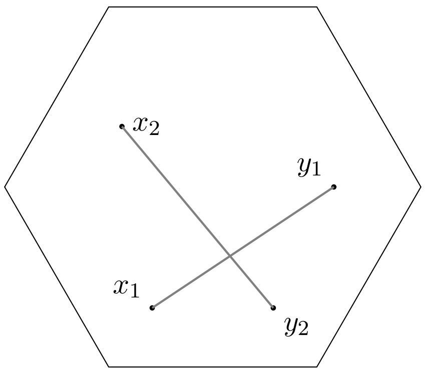
We first go over some definitions from convex analysis and constrained optimization theory that we are reused throughout the paper. The theory here is adapted from [Moklyachuk, 2020] and [Bertsekas, 2016].
We start with recalling some basic notions from convex analysis. After this we move on to discussing the existence and uniqueness of minimizers of functions. Finally, we discuss the Lagrange multiplier method to find these minimizers. To begin, let us define what a convex set is.

Definition A.1 (Convex set). A set $X \subseteq \mathbb{R}^N$ is *convex* if for all $x, y \in X$ we have that

$$\lambda x + (1-\lambda)y \in X \qquad \forall\, \lambda \in (0,1). \tag{A.1.1}$$

This definition says that a set X is convex if for any two points we can draw a straight line between them so that all points on this line lie in X.

Example A.2. The left set drawn here is convex, while the one on the right is not. $\triangle$



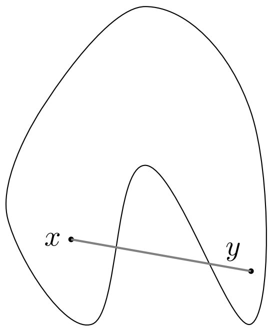


Figure 19: Two examples of sets, one of which is convex while the other is not

As an example of a standard proof to show a set is convex, we have the following proposition which we will see reappear later on.

Proposition A.3. *The set $X = \{x \mid Ax \le b\}$ is a convex set. Here, the inequality signifies that for all i we have that $(Ax)_i \le b_i$.*

Proof. Take $x, y \in X$ arbitrary and let λ be a constant in $(0,1)$. Consider the element $\lambda x + (1-\lambda)y$. We check that this term satisfies the condition to be an element of X.

$$\begin{aligned} A(\lambda x + (1-\lambda)y) &= \lambda Ax + (1-\lambda)Ay \\ &\leq \lambda b + (1-\lambda)b = b \end{aligned}$$

Since $x, y \in X$. Therefore $\lambda x + (1-\lambda)y \in X$ as well, and thus the set X is convex. □

Similar to a convex set, we can define what it means for a function to be convex.

Definition A.4 (Convex function). Let $X \subseteq \mathbb{R}^N$ be a convex set. A function $f : X \to \mathbb{R}$ is *convex* if for all $x, y \in X$ we have that

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \qquad \forall\, \lambda \in (0,1) \tag{A.1.2}$$

If the inequality above is strict, we say that f is strictly convex.

Just like in the definition of a convex set, we have a visual interpretation for this definition as well. Namely a function f is convex over a set X if its graph between any $x, y \in X$ is below the chord connecting x and y.

Example A.5. Consider the function $f(x) = x^2$. It can easily be shown that f is convex on all of $\mathbb{R}$, for example using some of the methods we will describe later. If we draw the graph of f, we can see that indeed it also satisfies the geometric interpretation given above. Since f is a convex function, the chord will always be above the graph for any $x \neq y$. △

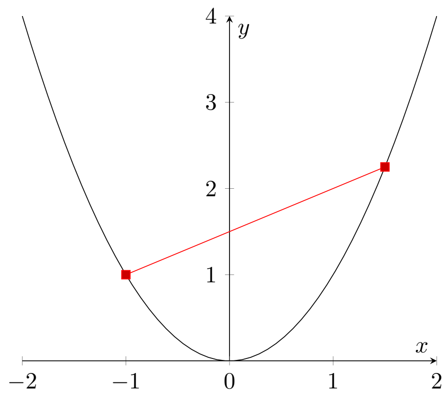


Figure 20: The function $f(x) = x^2$ satisfies the geometric interpretation of definition A.4

It is easy to see that the sum of convex functions is yet another convex function. Another useful characterization we will use often is the following.

Proposition A.6. *Let $X \subseteq \mathbb{R}^N$ be a convex set. Consider a function $f : X \to \mathbb{R}$ that is differentiable over X. Then, f is convex over X if and only if for all $x, y \in X$, $f(x) \geq f(y) + \nabla f(y) \cdot (x - y)$.*

We only prove the left to right direction since we mainly use this direction.

Proof. Since f is convex, we have that:

$$\begin{aligned} & f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda) f(y) \\ \iff & f(\lambda x + (1-\lambda)y) - f(y) \le \lambda(f(x) - f(y)) \\ \iff & \frac{f(\lambda(x-y)+y) - f(y)}{\lambda} \le f(x) - f(y) \end{aligned}$$

When we take $\lambda \to 0$, we find that the left hand side is exactly the directional derivative of f in the direction of the vector $x - y$, evaluated at y [Stewart et al., 2021]. Therefore,

$$f(y) + \nabla f(y) \cdot (x-y) \le f(x)$$

Rearranging then gives the desired inequality. □

Finally, another characterization is given by the following proposition.

Proposition A.7. *Let f be twice continuously differentiable. If the Hessian of f, $\nabla^2 f(x)$, is positive semi-definite for all $x \in X$, then f is convex. Similarly, if $\nabla^2 f(x)$ is positive definite for all $x \in X$ then the function f is strictly convex.*

Proof. By Taylor's theorem we have that for any $x, y \in X$ there exists a point $\xi \in X$ so that:

$$f(x) = f(y) + \nabla f(y) \cdot (x-y) + \frac{1}{2}(x-y) \cdot \nabla^2 f(\xi)(x-y)$$

But since the Hessian is positive semi-definite, the last term is a non-negative constant. Therefore,

$$f(x) \ge f(y) + \nabla f(y) \cdot (x-y)$$

Applying proposition A.6 we find that f is convex. Similarly, proposition A.6 works for strictly convex functions, as long as the inequality there is strict (and $x \neq y$). Therefore, the second statement follows similarly. □

Another standard definition in optimization theory is that of $L-$smoothness.

Definition A.8. Let $f : X \subseteq \mathbb{R}^N \to \mathbb{R}$ be continuously differentiable, and let $x, y \in X$. Then f is $L-$*smooth* if

$$\|\nabla f(x) - \nabla f(y)\| \le L\|x-y\| \tag{A.1.3}$$

In fact, this is equivalent to the following well known result.

Theorem A.9 (Descent lemma). *A continuously differentiable function $f : X \to \mathbb{R}$ is L-smooth if and only if for all $x, y \in$*

$$f(y) - f(x) \leq \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2. \tag{A.1.4}$$

A nice test for smoothness can be formalized in the following:

Theorem A.10. *Let $f : X \subset \mathbb{R}^N \to \mathbb{R}$ be a twice continuously differentiable function over a compact, convex set X. Then, f is $L-$smooth for some constant $L > 0$.*

Proof. First, we generalize the mean value theorem to functions with target space $\mathbb{R}^M$ [Pugh, 2015, p. 288]. To this end, we consider a function $g : X \to \mathbb{R}^M$. Define $h : [0, 1] \to \mathbb{R}$ via $t \mapsto \langle u, g(x + t(y - x)) \rangle$ for any unit vector $u \in \mathbb{R}^M$. Here, we assume we use the standard inner product on $\mathbb{R}^M$. Then, by the single variable mean value theorem, there exists an $s \in (0, 1)$ so that $h(1) - h(0) = h'(s)$. Therefore,

$$\begin{aligned}\langle u, g(y) - g(x) \rangle = h(1) - h(0) = h'(s) &= \langle u, Dg_{x+t(y-x)}(y - x) \rangle \\ &\leq \|u\| \left\| Dg_{x+t(y-x)} \right\| \|y - x\| = \left\| Dg_{x+t(y-x)} \right\| \|y - x\|,\end{aligned}$$

where $Dg_{x+t(y-x)}$ is the Jacobian of g evaluated at $x+t(y-x)$ and the inequality follows by using the Cauchy-Schwarz inequality. To establish the final result, we note that since the inequality holds for any arbitrary unit vector u, we can take the unit vector in the same direction as $g(y) - g(x)$, so that $\langle u, g(y) - g(x) \rangle = ||u||||g(y) - g(x)|| \cos(0) = \|g(y) - g(x)\|$, so that finally,

$$\|g(y) - g(x)\| \leq \|Dg_\xi\| \|y - x\|$$

Taking $g = \nabla f : \mathbb{R}^N \to \mathbb{R}^N$, we find that over the set X,

$$\|\nabla f(y) - \nabla f(x)\| \leq \|H_\xi\| \|y - x\|,$$

where H is the Hessian of the map f. Recalling that the set X is compact, we notice that $\|H_\xi\|$ maps a point ξ to a real number. Furthermore, this map is continuous, since the function f is twice continuously differentiable. By theorem A.12, which we discuss momentarily, it must therefore attain a maximum and minimum on the set, and thus can be uniformly bounded by a constant L. Combining all inequalities we find that for all $x, y \in X$:

$$\|\nabla f(y) - \nabla f(x)\| \leq \|H_\xi\| \|y - x\| \leq L\|y - x\|$$

□

Next, we formalize what it means to achieve a minimum value.

Definition A.11. A vector x^* is a *local minimum* of the function $f : X \subseteq \mathbb{R}^N \to \mathbb{R}$ if there exists an $\epsilon > 0$ such that:

$$f(x^*) \leq f(x) \quad \forall\, x \in X \text{ with } \|x - x^*\| < \epsilon \tag{A.1.5}$$

A vector x^* is a *global minimum* of f over X if $f(x^*) \leq f(x)$ for all $x \in X$.

Having defined both a local and global minima, we now ask ourselves when a function has a minimum. The following theorem gives us an answer to this question.

Theorem A.12 (Weierstrass' Theorem). *Let X be a nonempty and compact subset of $\mathbb{R}^N$ and let $f : X \to \mathbb{R}$ be continuous at all points of X. Then, the set of minima and the set of maxima of f over X are nonempty and compact.*

In fact, we only need lower-subcontinuity for the above result to hold. If we assume that the function f is (strongly) convex over a convex set X, we can say a little more.

Proposition A.13. *Let X be a compact convex subset of $\mathbb{R}^N$. If $f : X \to \mathbb{R}$ is convex then any local minimum over X is also a global minimum over X. Moreover, if f is strongly convex then f has a unique global minimum over X.*

The above proposition follows quite easily by directly applying the definition of a convex function.

Finally, we briefly go over optimality conditions for constrained optimization problems. First, consider the problem:

$$\min_{x \in X} f(x)$$

where X is a convex constraint set and f is convex over X. Then, the following proposition gives a necessary and sufficient condition for a point x^* to be a global minimum over X.

Proposition A.14. *Let X be a convex set and let $f : \mathbb{R}^N \to \mathbb{R}$ be convex over X. If f is continuously differentiable, then $\nabla f(x^*)(x - x^*) \leq 0$ for all $x \in X$ is a necessary and sufficient condition for a vector $x^* \in X$ to be a global minimum for f over X.*

Proof. Since f is convex, it holds that $f(x) \geq f(x^*) + \nabla f(x^*) \cdot (x - x^*) \geq f(x^*)$, since the second term was assumed to be positive for all x. Therefore, $f(x^*)$ is a global minimum. For the other direction, assume for contradiction that $\nabla f(x^*) \cdot (x - x^*) < 0$. Then, by recalling the definition of the directional derivative we have that:

$$\lim_{h \downarrow 0} \frac{f(x^* + h(x - x^*)) - f(x^*)}{h} < 0$$

For h small enough, we then must have that $f(x^* + h(x - x^*)) \leq f(x^*)$. But that contradicts the fact that $f(x^*)$ was assumed to be a global minimum. Therefore $\nabla f(x^*) \cdot (x - x^*) \geq 0$. $\square$

Consider now the following problem:

$$\begin{aligned} &\min f(x) \\ \text{subject to } &h_i(x) = 0, \quad i = 1, \ldots, m \\ &g_j(x) \leq 0, \quad j = 1, \ldots, r, \end{aligned} \tag{$\mathcal{P}$}$$

for functions $h_i, g_j : \mathbb{R}^N \to \mathbb{R}$. For any feasible point x (that is if x satisfies the given constraints) we denote the set of *active inequality constraints by*

$$A(x) = \{j \mid g_j(x) = 0\} \tag{A.1.6}$$

A point x^* is called regular if the set

$$V = \{\nabla h_i(x^*), \nabla g_j(x^*) \mid i = 1, \ldots, m \text{ and } j \in A(x^*)\} \tag{A.1.7}$$

is linearly independent. The *Lagrangian function* of the problem $(\mathcal{P})$ is defined as

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{j=1}^{n} \mu_i g_j(x) \tag{A.1.8}$$

The vectors λ and μ are called Lagrange multiplier vectors. A necessary condition for optimality is given by the Karush-Kuhn-Tucker conditions.

Theorem A.15 (KKT conditions for optimality). *Let x^* be a regular local minimum of $(\mathcal{P})$. Then, there exist unique vectors λ^*, μ^* so that:*

$$\begin{gathered} \nabla_x L(x^*, \lambda^*, \mu^*) = 0 \\ \mu_j^* \geq 0 \quad \textit{for } j = 1, \ldots, r \\ \mu_j^* g_j(x^*) = 0 \quad \textit{for } j = 1, \ldots, r \end{gathered} \tag{A.1.9}$$

Moreover, if $f, g, h \in C^2$, then:

$$y^T \nabla_{xx}^2 L(x^*, \lambda^*, \mu^*) y \geq 0 \tag{A.1.10}$$

for all $y \in V^\perp$, where V is the set of gradients as used in the definition of regularity.

However, if we assume that f, g_j are convex differentiable functions and that λ^*, μ^* satisfy the KKT conditions, then the condition that $\nabla_x L(x^*, \lambda^* \mu^*) = 0$ is also sufficient. Moreover, if the constraint functions h_i are linear and the functions g_j are concave (that is, $-g_j$ is convex) then we can disregard the regularity of the solution x^* [Bertsekas, 2016]. In particular, any affine transformation is both convex and concave. Namely, take $g(x) = Ax + b$ so that $g_j(x) = (Ax + b)_j$. Then, $\nabla g(x) = A^T$ and $\nabla^2 g(x) = 0$. Thus, using proposition A.7 we have that g is convex. However for $-g(x) = -Ax - b$, we have the same and so g is also concave. Summing this up, we get the following proposition.

Proposition A.16. *Let f be a convex differentiable function, and let h_i, g_j be linear functions. Consider the problem $(\mathcal{P})$. Then, a point x^* is a local minimum of $(\mathcal{P})$ if and only if there exist vectors λ^*, μ^* so that*

$$\begin{gathered} \nabla_x L(x^*, \lambda^*, \mu^*) = 0 \\ \mu_j^* \geq 0 \quad \textit{for } j = 1, \ldots, r \\ \mu_j^* g_j(x^*) = 0 \quad \textit{for } j = 1, \ldots, r \end{gathered} \tag{A.1.11}$$

A.2 Numerical algorithms

We discuss two basic algorithms that are used in the computation of equilibria as presented in section 4. First, we discuss the bisection method to find roots of single variable functions. After this, we discuss a simple path-finding algorithm to find all paths between two nodes in a graph.

A.2.1 The bisection method

Consider a function $f : \mathbb{R} \to \mathbb{R}$, with a root on the interval $[a, b]$. To find this root, we can use the bisection method [Epperson, 2013]. The bisection method, as the name suggests is based on halving the interval $[a, b]$ at each successive iteration, making sure to keep the root in the next interval. We choose to use the bisection method because of its simplicity and the fact it only relies on $f(x)$ and not any of its derivatives. If we have access to derivatives of the function f and it would not be too expensive to use these other root-finding algorithms include Newton's method or for example a fixed point based algorithm [Epperson, 2013]. An outline of the bisection method to find the root x^* of a function $f(x)$ in the interval $[a, b]$ is given in Algorithm 2.

Algorithm 2 Bisection method

```
Input: $a, b, f(x), \epsilon$
  $c \leftarrow \frac{a+b}{2}$
  if $f(a)f(b) < 0$ then
    while $\|f(c)\| > \epsilon$ do
      if $f(a)f(c) < 0$ then
        $b \leftarrow c$
      else if $f(b)f(c) < 0$ then
        $a \leftarrow c$
      end if
    end while
    $x^* \leftarrow c$
  else
    return
  end if
Output: $x^*$
```

Note we have to include the first if statement to check if the endpoints have different signs to guarantee a root. By the intermediate value theorem [Abbott, 2015], a continuous function f has a root in $[a, b]$ if $f(a) < 0 < f(b)$ or $f(a) > 0 > f(b)$. In each iteration, we half the current interval, and check in which of the two parts the root is contained. We continue with this interval and apply the same procedure.
As for the convergence rate, we can state the following theorem.

Theorem A.17 (Convergence of the bisection method). *Let $[a, b] \subset \mathbb{R}$ be an interval containing a root x^* of the function $f(x)$. Set $c_n = \frac{a_{n-1}+b_{n-1}}{2}$ as the approximate root, where a_i, b_i and c_i are generated by the* *bisection method. Then,*

$$|x^* - c_n| \leq \left(\frac{1}{2}\right)^n = (b - a) \tag{A.2.1}$$

Proof. First, notice that $b_n - a_n$ is the length of the working interval at the n^{th} iteration. Therefore, it holds that $b_n - a_n \leq \frac{1}{2}(b_{n-1} - a_{n-1})$. By recursion:

$$b_n - a_n \leq \left(\frac{1}{2}\right)^n (b_0 - a_0) = \left(\frac{1}{2}\right)^n (b - a).$$

But then,

$$|x^* - c_n| \leq \frac{1}{2}(b_{n-1} - a_{n-1}) \leq \left(\frac{1}{2}\right)^n (b - a)$$

where the first inequality follows since the root is somewhere on either the interval $[a_{n-1}, c_n]$ or $[c_n, b_{n-1}]$, and c_n is the middle of the interval $[a_{n-1}, b_{n-1}]$. □

An important example where the bisection method can fail is the following.

Example A.18 (Failure of the bisection method). Consider the function $f(x) = (x - a)^2$ for some $a \in \mathbb{R}$. Clearly the function has a root at $x = a$. However, since $f(x) > 0$ for all $x \neq a$, we cannot guarantee the existence of a root using the intermediate value theorem. Unless we pick our searching interval so that $x = a$ is exactly the middle point, the bisection method will fail to find this root. △

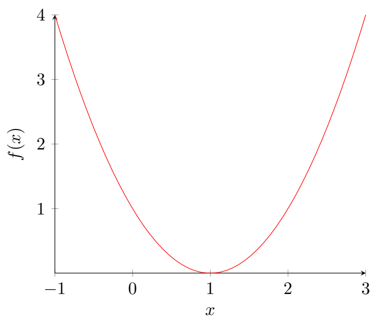


Figure 21: The function $(x - 1)^2$ has a root at $x = 1$.

Luckily, we can get around this problem in our implementation, as we shall see in section 4.1.

A.2.2 Path-finding: Breadth-first search

A natural problem when considering traffic problems is finding all possible routes between two nodes. When we model our traffic network in a graph, we can use a breadth-first search algorithm to determine all paths between two given nodes in a graph [Skiena, 2008]. Again, even though this might not be the most efficient algorithm, its use here comes from its simplicity. In essence the algorithm works as follows. Start at the first node and move to another connected node. From this node, repeat the process, until we hit either a dead end or we reach the destination node. Once we do this, we backtrack to the previous node and move to another, not yet visited node. If we keep doing this until we have exhausted all nodes in the graph, we find all paths between the two vertices. The pseudo-code can be found in algorithm 3.

Algorithm 3 Breadth-First search between nodes $u \in V$ and $v \in V$

```
Input: $G = (V, E)$, $u \in V$, $v \in V$, curPath = [ ], visited = [ ]
  Append $u$ to curPath and visited
  if $u = v$ then
    Print curPath and remove $u$ from curPath and visited
    return
  end if
  for $x \in V$ do
    if $(u, x) \in E$ then
      BFS($G, x, v$,curPath,visited)
    end if
  end for
```

B Appendix: Model of Groningen and numerical results

In what follows, we adopt the same convention as established in section 6. We visualize the results in a graph network, where the color gradient represents the amount of flow, while the arrow size represents the cost. A gray arrow means that the flow on that edge is less than 10% of the maximum flow on the network. First, a large copy of the network is given in figure 22. After this in tables 1, 2 and 3 the full results of the computation of the user equilibrium, system optimum and the difference between the two equilibria are given. Along with this, the social costs are given. Finally, a graphical representation of these flows is given in figure 23 and 24. The first shows the user equilibrium flow, while the second shows the system optimal flow. Here we adopt the same conventions as established in section 6 when drawing the graphs. The color of the edges denote the amount of flow on the edges, while the size denotes the cost of an edge. An edge is made gray whenever it carries less than 10% of the maximal flow.

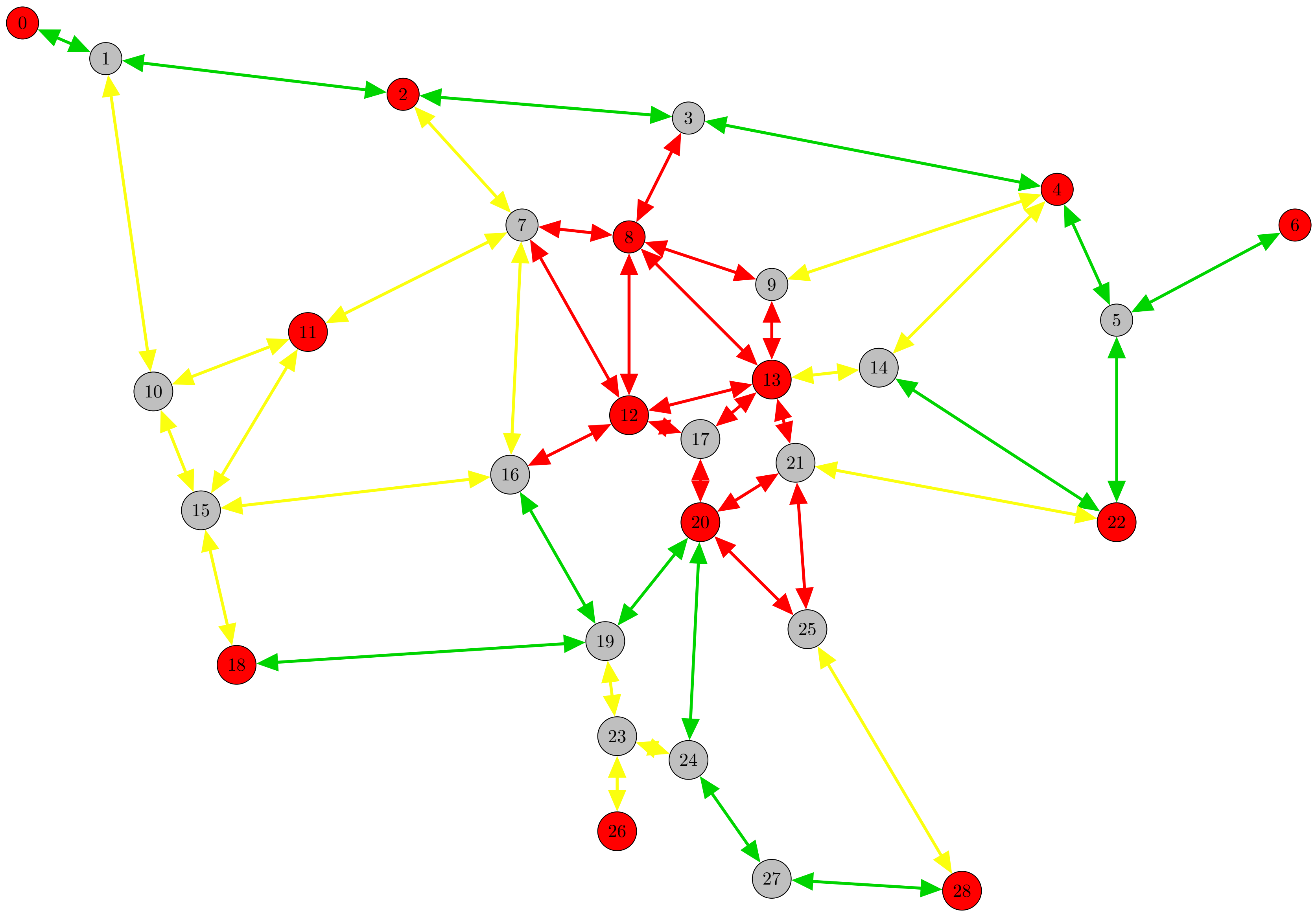


Figure 22: Network as used in section 6

Edge	Flow	Cost
(0,1)	1200.0	7.525
(1,2)	1457.947	10.105
(1,10)	311.547	4.89
(2,3)	600.0	3.133
(2,7)	448.613	5.649
(3,2)	674.447	3.265
(3,4)	428.571	2.849
(3,8)	477.373	10.214
(4,3)	980.391	4.75
(4,5)	256.383	3.06
(4,9)	365.903	4.621
(4,14)	834.292	7.57
(5,4)	408.397	3.097
(5,22)	1247.986	7.441
(6,5)	1400.0	12.053
(7,2)	601.934	5.933
(7,8)	861.83	15.764
(7,12)	522.217	10.188
(7,16)	261.304	4.903
(8,12)	60.308	8.888
(9,8)	182.935	10.343
(9,13)	182.968	9.35
(10,1)	569.495	5.332
(10,15)	311.547	5.612
(11,7)	1086.194	10.704
(11,15)	170.005	5.027
(12,8)	107.928	8.89
(12,13)	572.654	11.257
(13,8)	115.956	10.044
(13,12)	350.96	9.881
(14,13)	767.992	6.399
(14,22)	457.903	2.928
(15,10)	569.495	6.143
(15,11)	656.2	5.865
(15,16)	985.011	10.15
(16,7)	712.477	6.461
(16,12)	896.312	16.613
(16,19)	742.857	3.663
(17,12)	536.499	11.329
(17,13)	123.577	10.216
(18,15)	1729.153	44.015
(18,19)	2270.847	48.295
(19,16)	1105.331	5.884
(19,20)	1484.489	12.431
(19,23)	1057.143	12.095
(20,17)	660.076	11.115
(20,19)	633.259	3.223
(20,21)	571.429	11.306
(20,24)	200.0	3.017
(21,13)	505.44	10.939
(21,20)	628.571	11.882
(21,22)	1000.0	10.847
(22,14)	391.603	2.888
(22,21)	628.571	6.167
(23,26)	1685.714	50.51
(24,20)	1132.324	6.424
(24,23)	628.571	5.979
(25,20)	505.094	10.487
(25,21)	934.011	20.832
(27,24)	1560.895	11.601
(28,25)	1439.105	23.42
(28,27)	1560.895	15.88
Social Cost		655,579.23

Table 1: User equilibrium flow and cost for the given network

Edge	Flow	Cost
(0,1)	1200.0	7.525
(1,2)	1444.236	9.843
(1,10)	342.697	4.91
(2,3)	649.944	3.217
(2,7)	314.434	5.553
(3,2)	571.482	3.094
(3,4)	393.532	2.828
(3,8)	526.223	10.461
(4,3)	841.292	3.847
(4,5)	172.94	3.054
(4,9)	741.897	5.709
(4,14)	761.372	6.851
(5,4)	523.92	3.172
(5,22)	1049.022	5.21
(6,5)	1400.0	12.053
(7,2)	634.374	6.029
(7,8)	723.949	12.34
(7,12)	668.233	11.121
(7,16)	143.038	4.877
(8,12)	181.798	8.898
(8,13)	193.07	10.065
(9,8)	340.504	10.523
(9,13)	401.393	9.537
(10,1)	586.933	5.394
(10,11)	0.2	5.091
(10,15)	342.563	5.636
(11,7)	1105.454	11.121
(11,10)	0.284	5.091
(11,15)	170.777	5.027
(12,8)	94.008	8.889
(12,13)	595.457	11.528
(12,17)	0.804	10.369
(13,8)	375.899	10.388
(13,12)	247.022	9.711
(13,14)	99.784	4.893
(13,17)	166.311	10.224
(14,4)	0.051	5.222
(14,13)	576.716	5.372
(14,22)	560.518	3.034
(15,10)	586.715	6.216
(15,11)	676.315	5.973
(15,16)	995.576	10.364
(16,7)	749.704	6.82
(16,12)	797.872	13.998
(16,19)	656.57	3.441
(17,12)	481.061	10.99
(17,13)	268.836	10.288
(17,20)	167.115	9.336
(18,15)	1745.267	45.497
(18,19)	2254.733	47.017
(19,16)	1065.532	5.503
(19,20)	1302.859	8.558
(19,23)	971.43	10.042
(20,17)	749.897	12.304
(20,19)	383.605	2.95
(20,21)	639.391	12.011
(20,24)	285.771	3.028
(21,13)	539.256	11.158
(21,20)	547.973	11.114
(21,22)	900.278	8.944
(22,14)	276.129	2.854
(22,21)	547.973	5.801
(23,19)	44.913	4.941
(23,26)	1685.714	50.51
(24,20)	1062.639	5.659
(24,23)	759.197	6.967
(25,20)	663.791	11.572
(25,21)	800.144	16.184
(27,24)	1536.065	11.059
(28,25)	1463.935	24.752
(28,27)	1536.065	15.08
Social Cost		640,538.16

Table 2: System optimal flow and associated cost for the given network

Edge	Flow	Cost	Edge	Flow	Cost	Edge	Flow	Cost
(1,2)	13.711	0.262	**(10,15)**	-31.015	-0.024	**(18,15)**	-16.113	-1.482
(1,10)	-31.149	-0.02	**(11,7)**	-19.26	-0.417	**(18,19)**	16.113	1.278
(2,3)	-49.944	-0.083	**(11,10)**	-0.284	-5.091	**(19,16)**	39.799	0.381
(2,7)	134.18	0.096	**(11,15)**	-0.771	-0.0	**(19,20)**	181.63	3.873
(3,2)	102.965	0.171	**(12,8)**	13.919	0.001	**(19,23)**	85.713	2.053
(3,4)	35.04	0.021	**(12,13)**	-22.804	-0.271	**(20,17)**	-89.821	-1.189
(3,8)	-48.85	-0.247	**(12,17)**	-0.804	-10.369	**(20,19)**	249.654	0.273
(4,3)	139.098	0.903	**(13,8)**	-259.943	-0.344	**(20,21)**	-67.962	-0.705
(4,5)	83.443	0.005	**(13,12)**	103.938	0.171	**(20,24)**	-85.771	-0.011
(4,9)	-375.994	-1.087	**(13,14)**	-99.784	-4.893	**(21,13)**	-33.817	-0.219
(4,14)	72.919	0.719	**(13,17)**	-166.311	-10.224	**(21,20)**	80.598	0.768
(5,4)	-115.523	-0.075	**(14,4)**	-0.051	-5.222	**(21,22)**	99.722	1.903
(5,22)	198.964	2.231	**(14,13)**	191.276	1.028	**(22,14)**	115.474	0.034
(7,2)	-32.44	-0.096	**(14,22)**	-102.615	-0.106	**(22,21)**	80.598	0.366
(7,8)	137.882	3.424	**(15,10)**	-17.22	-0.074	**(23,19)**	-44.913	-4.941
(7,12)	-146.017	-0.933	**(15,11)**	-20.115	-0.108	**(24,20)**	69.685	0.765
(7,16)	118.267	0.026	**(15,16)**	-10.565	-0.215	**(24,23)**	-130.626	-0.988
(8,12)	-121.49	-0.01	**(16,7)**	-37.227	-0.359	**(25,20)**	-158.697	-1.086
(8,13)	-193.07	-10.065	**(16,12)**	98.44	2.615	**(25,21)**	133.867	4.648
(9,8)	-157.568	-0.18	**(16,19)**	86.287	0.222	**(27,24)**	24.83	0.542
(9,13)	-218.426	-0.186	**(17,12)**	55.438	0.339	**(28,25)**	-24.83	-1.332
(10,1)	-17.438	-0.062	**(17,13)**	-145.259	-0.072	**(28,27)**	24.83	0.8
(10,11)	-0.2	-5.091	**(17,20)**	-167.115	-9.336	Social Cost		15,041.07

Table 3: Difference between UE and SO flow. A positive value means the value is higher in the UE case and vice versa

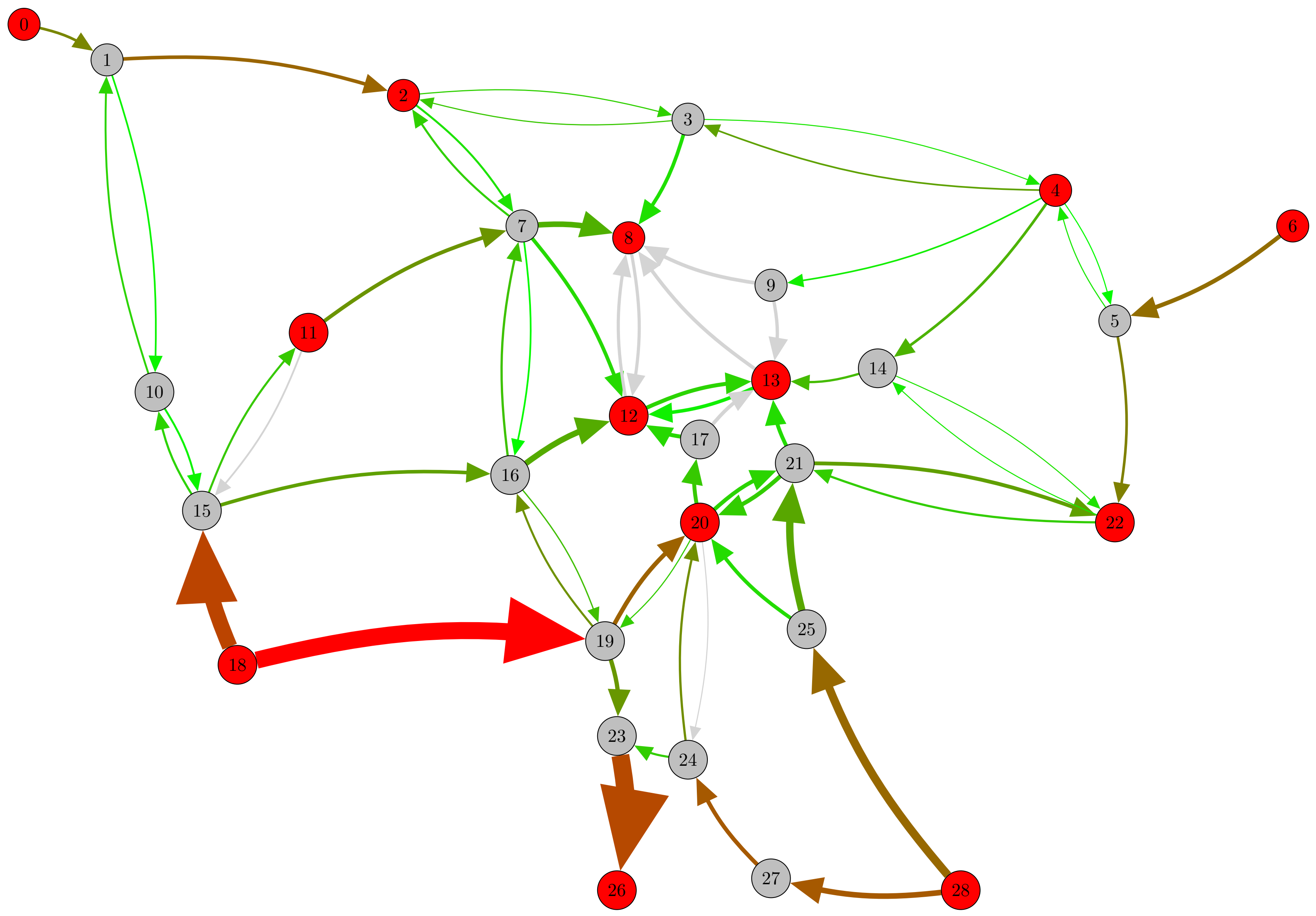


Figure 23: Model of Groningen, UE

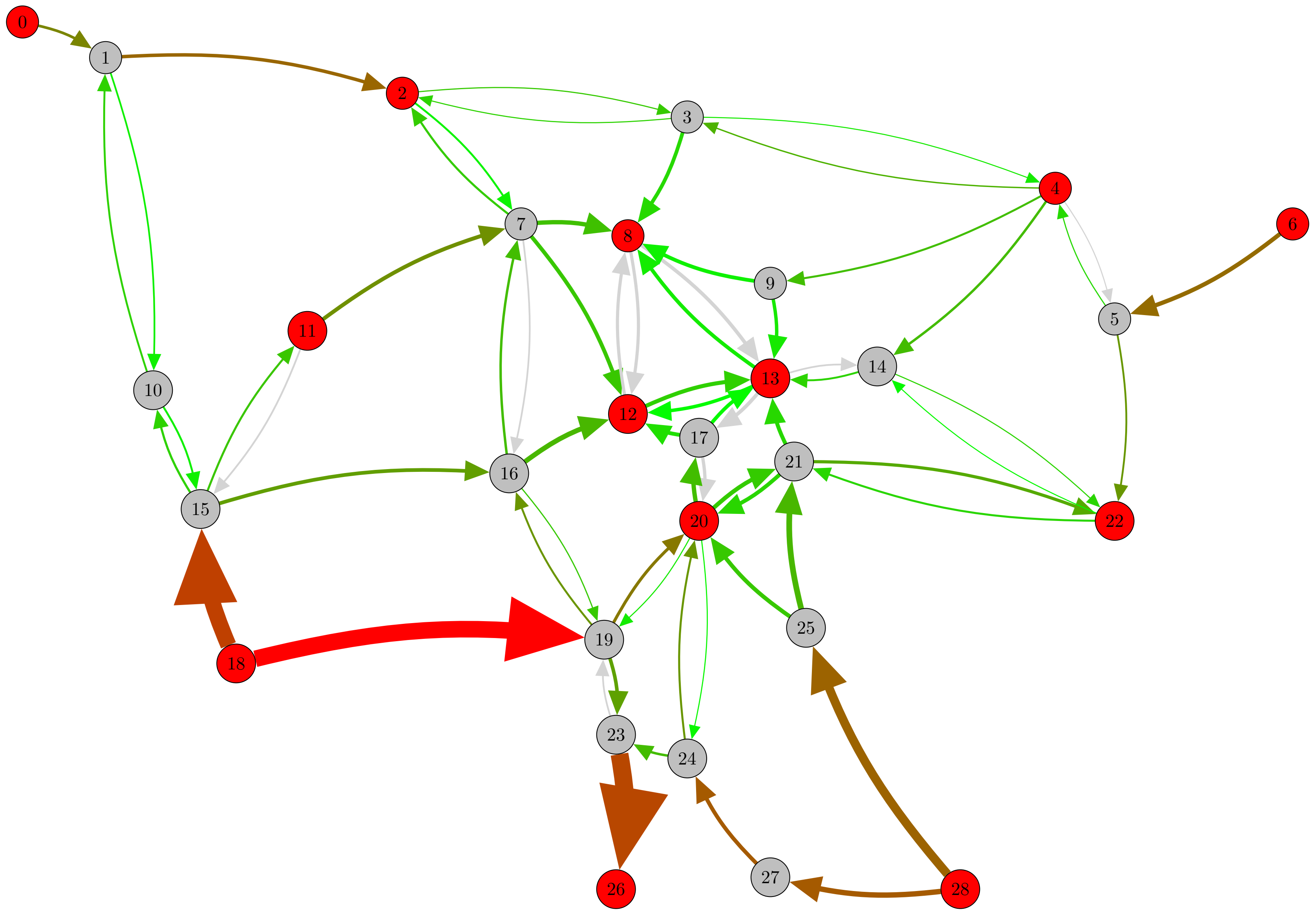


Figure 24: Model of Groningen, SO