

Approximating Wardrop Equilibria with Finitely Many Agents

Simon Fischer*, Lars Olbrich**, and Berthold Vöcking***

Dept. of Computer Science, RWTH Aachen, Germany
{fischer,lars,voecking}@cs.rwth-aachen.de

Abstract. We study adaptive routing algorithms in a round-based model. Suppose we are given a network equipped with load-dependent latency functions on the edges and a set of commodities each of which is defined by a collection of paths (represented by a DAG) and a flow rate. Each commodity is controlled by an agent which aims at balancing its traffic among its paths such that all used paths have the same latency. Such an allocation is called a Wardrop equilibrium.

In recent work, it was shown that an infinite population of users each of which carries an infinitesimal amount of traffic can attain approximate equilibria in a distributed and concurrent fashion quickly. Interestingly, the convergence time is independent of the underlying graph and depends only mildly on the latency functions. Unfortunately, a direct simulation of this process requires to maintain an exponential number of variables, one for each path.

The focus of this work lies on the distributed and efficient computation of the adaptation rules by a finite number of agents. In order to guarantee a polynomial running time, every agent computes a randomised path decomposition in every communication round. Based on this decomposition, agents remove flow from paths with high latency and reassign it proportionally to all paths. This way, our algorithm can handle exponentially large path collections in polynomial time.

1 Introduction

We consider routing problems in the Wardrop model. We are given a network equipped with non-decreasing latency functions mapping flow on the edges to latency. For each of several commodities a fixed flow rate has to be routed from a source to a sink via a collection of paths. A flow vector is said to be at Wardrop equilibrium if for all commodities the latencies of all used paths are minimal with respect to this commodity. Whereas such equilibria can be formulated as

* Supported by DFG grant Vo889/1-3 and by DFG through German excellence cluster UMIC at RWTH Aachen.

** Supported by the DFG GK/1298 “AlgoSyn”

*** Supported in part by the the EU within the 6th Framework Programme under contract 001907 (DELIS) and by DFG through German excellence cluster UMIC at RWTH Aachen.

convex programs (under some mild assumptions on the latency functions) and can thus be solved by centralised algorithms in polynomial time, in this work, we study *distributed* algorithms to compute Wardrop equilibria.

A common interpretation of the Wardrop model is that flow is controlled by an infinite number of selfish agents each of which carries an infinitesimal amount of flow. In [12] it was shown that in this setting such a population approaches Wardrop equilibria quickly by following a simple round-based load-adaptive rerouting policy. This policy, called the *replication policy*, is executed by all agents in parallel and proceeds in the following way. Each agent samples another agent at random and, if this improves the latency, migrates to this agent's path with a probability that increases with the latency gain. In this setting, a natural goal is to reach approximate equilibria in the following bicriterial sense. We say that a flow is at δ - ϵ -equilibrium if at most an ϵ -fraction of the flow utilises paths whose latency exceeds the average latency of their commodity by more than a δ -fraction of the overall average latency. Remarkably, the number of rounds to reach an approximate equilibrium in this sense is independent of the size and the topology of the underlying graph and chiefly depends on the approximation parameters and the elasticity of the latency functions.

In this work, we consider a different setting, in which the flow is controlled by a finite number of agents only, each of which is responsible for the entire flow of one commodity. Each agent has a set of admissible paths among which it may distribute its flow. To be able to represent exponentially large collections of paths we assume that these are represented by an arbitrary DAG connecting the source and the sink of the agent. Each agent aims at balancing its own flow such that the jointly computed allocation will be at Wardrop equilibrium. Let us remark that agents do not aim at minimising the overall latency of their flow, but seek to minimise the maximum latency of their commodity. Unfortunately, the replication policy does not yield a feasible distributed algorithm in this setting directly. Simulating an infinite number of agents each of which chooses one out of the given collection of paths would require maintaining one variable for each path and computing a quadratic number of migration rates between pairs of paths. As the number of paths may be exponential in the size of the network this approach is rendered computationally infeasible.

We present two approaches to circumvent this problem. Our first approach exploits the fact that, for a simplified variant of the replication policy, the updates of the edge flows can be expressed in a way that merely uses the edge flow variables themselves (rather than the path flow variables). Thus, the updates can be computed in polynomial time. Unfortunately, the convergence time of this variant is only pseudopolynomial in the latency functions since it depends on the maximum slope of the latency functions.

Since the original replication policy cannot be expressed in this compact way, we consider a second approach to achieve convergence in a polynomial number of communication rounds. Consider a collection of paths for one of the commodities. In a first step, our algorithm samples a polynomial number of paths with probability proportional to their flow. We thus obtain a randomised

path decomposition. We consider paths in this decomposition with above-average latency. From such paths, a fraction of the flow is removed and reallocated proportionally among all admissible paths. If this is done carefully, oscillations can be avoided, and a potential function argument ensures convergence towards Wardrop equilibria. Thus, we achieve essentially the same convergence rates as in the setting with an infinite number of agents and keep the computation time of one communication round polynomial. Altogether, we can compute approximate Wardrop equilibria in polynomial time.

1.1 Related Work

The game theoretic traffic model considered in this paper was introduced by Wardrop [19]. Many aspects of Wardrop equilibria have been studied, the most prominent being the degradation of performance due to the selfish behaviour, called the price of anarchy [18] as well as the inverse, the increase of the maximum latency incurred to an agent due to optimal routing [17]. It has also been shown that the price of anarchy can be decreased by imposing taxes on the edges [9, 14]. Cominetti *et al.* consider the price of anarchy in a model with finitely many agents aiming at minimising their average latency [10].

For solving the corresponding classical goal of finding a minimum cost multi-commodity flow, several algorithms are known. For an overview see, e.g., [1] and [7]. An efficient distributed steepest-descent algorithm for solving multi-commodity flow problems with linear latency functions has been presented recently in [3]. In [2], a stateless algorithm for this problem is presented.

It is also known that the problems of finding an optimal allocation and finding a Wardrop equilibrium are essentially equivalent. Under mild conditions on the latency functions, a flow at Wardrop equilibrium with respect to so-called *marginal-cost* latency functions is optimal with respect to the original latency functions, see e.g. [5] and [18].

Several authors (e.g. [4, 8]) consider dynamic routing from an online-learning perspective. Awerbuch and Kleinberg [4] present an algorithm for the online shortest path problem in an end-to-end feedback model. Blum *et al.* [8] show that approximate Wardrop equilibria defined in a similar way can be attained if the agents follow no-regret algorithms. Their bounds on the convergence time depend polynomially on the regret bounds and network size and depend pseudopolynomially on the maximum slope of the latency functions.

The problem of load-balancing has also been studied in various discrete settings for networks of parallel links. For the case of identical links, both sequential [15] and concurrent distributed algorithms were considered [6]. Even-Dar *et al.* [11] consider distributed algorithms for load balancing on links with speeds using sampling rules which depend pseudopolynomially on the speed of the links.

The rerouting policy upon which our algorithms are based was introduced in [13] and [12]. It was shown that an infinite number of agents executing this policy can attain a Wardrop equilibrium quickly in a concurrent setting.

2 Model and Problem Statement

2.1 Wardrop's Traffic Model

We consider Wardrop's traffic model originally introduced in [19]. We are given a graph $G = (V, E)$ with non-decreasing differentiable latency functions $\ell_e : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$. Furthermore, we are given a set of commodities $[k] = \{1, \dots, k\}$ specified by source-sink-pairs $(s_i, t_i) \in V \times V$, a directed acyclic subgraph G_i of G connecting s_i and t_i and flow demands $r_i \in \mathbb{R}_{\geq 0}$. The total demand is $r = \sum_{i \in [k]} r_i$, and we normalise $r = 1$ for simplicity. Let \mathcal{P}_i denote the admissible paths of commodity i , i.e., all paths connecting s_i and t_i in G_i , and let $\mathcal{P} = \bigcup_{i \in [k]} \mathcal{P}_i$. We may assume that the sets \mathcal{P}_i are disjoint and define i_P to be the unique commodity to which path P belongs.

A non-negative path flow vector $(f_P)_{P \in \mathcal{P}}$ is *feasible* if it satisfies the flow demands $\sum_{P \in \mathcal{P}_i} f_P = r_i$ for all $i \in [k]$. We denote the set of all feasible flow vectors by \mathcal{F} . A path flow vector $(f_P)_{P \in \mathcal{P}}$ induces an edge flow vector $f = (f_{e,i})_{e \in E, i \in [k]}$ with $f_{e,i} = \sum_{P \in \mathcal{P}_i: e \in P} f_P$. The total flow on edge e is $f_e = \sum_{i \in [k]} f_{e,i}$. Furthermore, for $v \in V$ and $i \in [k]$, the total flow of commodity i through node v is $f_{v,i} = \sum_{(u,v) \in E} f_{(u,v),i} = \sum_{(v,w) \in E} f_{(v,w),i}$ for $v \notin \{s_i, t_i\}$ and $f_{s_i,i} = f_{t_i,i} = r_i$. The latency of an edge $e \in E$ is given by $\ell_e(f_e)$ and the latency of a path P is given by the sum of the edge latencies $\ell_P(f) = \sum_{e \in P} \ell_e(f_e)$. Finally, the weighted average latency of commodity $i \in [k]$ is given by $L_i(f) = \sum_{e \in E} \ell_e(f) \cdot (f_{e,i}/r_i)$ and the overall average latency is $L(f) = \sum_{e \in E} \ell_e(f) \cdot f_e/r$. We drop the argument f of $\ell(\cdot)$ and $L(\cdot)$ whenever it is clear from the context.

A flow vector in this model is considered stable when no fraction of the flow can improve its sustained latency by moving unilaterally to another path. This implies that all used paths must have the same minimal latency. Unused paths may have larger latency.

Definition 1 (Wardrop equilibrium). *A feasible flow vector f is at Wardrop equilibrium if for every commodity $i \in [k]$ and paths $P_1, P_2 \in \mathcal{P}_i$ with $f_{P_1} > 0$ it holds that $\ell_{P_1}(f) \leq \ell_{P_2}(f)$.*

It is well-known that Wardrop equilibria are exactly those allocations that minimise the following potential function introduced in [5]:

$$\Phi(f) = \sum_{e \in E} \int_0^{f_e} \ell_e(u) du .$$

This potential precisely absorbs progress: If an infinitesimal amount of flow dx is shifted from path ℓ_P to ℓ_Q , thus improving its latency by $(\ell_P - \ell_Q)$, the potential decreases by $(\ell_P - \ell_Q) dx$. We will make use of this fact frequently. The minimum potential is denoted by $\Phi^* = \min_{f \in \mathcal{F}} \Phi(f)$. Every flow vector f with $\Phi(f) = \Phi^*$ is then at Wardrop equilibrium. We assume that Φ^* is positive. The case that $\Phi^* = 0$ can be treated by adding virtual offsets to the latency functions. For a detailed treatment see [12].

Let us remark, that the problems of computing a Wardrop equilibrium and computing a flow minimising L are equivalent. It is sufficient to replace the latency functions ℓ_e by so-called *marginal-cost* latency functions $h_e(x) = (x \cdot \ell_e(x))' = \ell_e(x) + x \cdot \ell_e'(x)$. If for all $e \in E$, $x \cdot \ell_e(x)$ is convex, then Wardrop equilibria with respect to $(h_e)_{e \in E}$ minimise L [5, 18].

The algorithms presented in this paper will compute approximate equilibria in the following bicriterial sense.

Definition 2 (δ - ϵ -equilibrium). Consider a flow vector f and let $\mathcal{P}_i^\delta = \{P \in \mathcal{P}_i \mid \ell_P(f) > L_i(f) + \delta L(f)\}$ denote the set of δ -expensive paths. A flow vector is at a δ - ϵ -equilibrium if $\sum_{i \in [k]} \sum_{P \in \mathcal{P}_i^\delta} f_P \leq \epsilon$.

This definition of approximate Wardrop equilibria requires that almost all flow utilises paths with a latency that is close to the average of their own commodity. A similar definition of approximate Nash equilibria is used, e. g., in [8].

2.2 Elasticity of Latency Functions

Our algorithms take the steepness of the latency functions into account when deciding how much flow to shift from one path to another. In [12] it was shown that the critical parameter in this setting is not the slope but the elasticity.

Definition 3. For any positive differentiable function $\ell : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$, the elasticity of ℓ at x is $d(x) = \frac{x \cdot \ell'(x)}{\ell(x)}$.

In other words, the elasticity of a function is bounded from above by d if the (absolute) slope at any point is at most by a factor of d larger than the slope of the line connecting the origin and the point $(x, \ell(x))$. Note that a polynomial with positive coefficients and degree d has elasticity at most d , hence, elasticity can be considered as a generalisation of the degree of such a polynomial. The function $a \cdot \exp(\lambda x)$, $x \in [0, 1]$ has maximum elasticity λ .

2.3 Implicit Path Decomposition

Wardrop equilibria are defined with respect to path flows. Our algorithms, however, will make use only of the edge flow vectors, which do not determine a vector of path flows uniquely. However, in a DAG, an edge flow vector $(f_e)_{e \in E}$ induces a natural vector of path flows by starting with the flow injected at the source, and splitting the flow at each node v such that the set of paths containing the outgoing edge e receives a flow proportional to f_e . Since the decomposition for one commodity $i \in [k]$ is independent of the flow of other commodities, we can omit the index i for simplicity.

Definition 4. Consider any edge flow vector $(f_e)_{e \in E}$ (for some commodity i). For any path $P = (v_1, \dots, v_l)$ let

$$\tilde{f}_P = f_{v_1} \cdot \prod_{j=1}^{l-1} \frac{f_{(v_j, v_{j+1})}}{f_{v_j}}.$$

It is easily verified by induction on the distance from the source that this is actually a valid flow decomposition of $(f_e)_{e \in E}$, i. e., $f_e = \sum_{P \ni e} \tilde{f}_P$.

2.4 Distributed Computation Model

Our algorithms operate in the following setting. Agents operate in a synchronous, round-based fashion. We assume that there is a billboard via which the agents are able to share information. On this billboard, each agent can observe the edge flows of its own commodity and the latency values of the paths it uses. Agents know an upper bound d on the elasticity of the latency functions, but they do not know the latency functions themselves. However, it is easily possible to extend our algorithm such that it does not rely on the knowledge of a bound on the elasticity.

In every round an agent can update the edge flows of its own commodity on the billboard. These updates become visible to all agents only in the next round. All agents execute the same algorithm in parallel. Therefore, in the descriptions of our algorithms, we may omit the index for the commodity, i. e., f_e refers to the flow $f_{e,i}$ of commodity i on edge e .

3 A Pseudopolynomial Algorithm

Our first approach works by simulating the replication policy presented in [12]. We will see that this can be done in polynomial time although this policy operates on an exponential number of paths.

3.1 The Replication Policy

Let us start by introducing the replication policy formally. We consider an infinite population of agents each of which controls an infinitesimal amount of flow which it assigns to a path. In each round agents may migrate their flow from the current path to another one. Consider an agent in commodity $i \in [k]$ currently using path $P \in \mathcal{P}_i$. Whenever activated, it performs two steps.

1. *Sampling.* Sample another path Q where the probability to sample any path Q' equals $f_{Q'}/r_i$.
2. *Migration.* There are two cases:
 - (a) $\ell_Q \geq \ell_P$. In this case, the agent stays with its old path.
 - (b) $\ell_Q < \ell_P$. The agent migrates to the sampled path Q with probability $\lambda \cdot (\ell_P - \ell_Q)$ for some constant $\lambda > 0$ to be determined later.

Altogether, we can characterise our policy by specifying the rate of agents migrating from one path $P \in \mathcal{P}_i$ to another path $Q \in \mathcal{P}_i$ with $\ell_Q(f) < \ell_P(f)$ within one round. This rate can be obtained by multiplying the probabilities specified in steps (1) and (2) with the volume of agents using path P . For this rate we obtain

$$\rho_{PQ} = \lambda \cdot f_P \cdot \frac{f_Q}{r_i} \cdot (\ell_P - \ell_Q)$$

if $\ell_Q < \ell_P$ and $\rho_{PQ} = 0$ otherwise. Thus, we can compute a sequence of flow vectors $(f_P(t))_{P \in \mathcal{P}}$ generated by this policy by summing over all paths Q :

$$\begin{aligned} f_P(t+1) &= f_P(t) + \sum_{Q \in \mathcal{P}_i} \rho_{QP} - \sum_{Q \in \mathcal{P}_i} \rho_{PQ} \\ &= f_P(t) + \lambda f_P \sum_{Q \in \mathcal{P}_i} \frac{f_Q}{r_i} (\ell_Q - \ell_P) \\ &= f_P(t) + \lambda f_P (L_i - \ell_P) . \end{aligned} \tag{1}$$

3.2 Convergence Towards Equilibria

For the time being assume that agents are migrating in a continuous fashion as described by the above rules. Then, an infinitesimal amount of flow dx migrating from a path P to another path Q improving its latency from ℓ_P to ℓ_Q causes the potential Φ to reduce by $(\ell_P - \ell_Q) dx$. Since we only accept migrations that improve the latency, this implies that the potential always decreases which in turn implies convergence towards a Wardrop equilibrium by Lyapunov's direct method if all paths are used in the initial flow. However, in our concurrent round-based model, flow is not shifted continuously, but in finite chunks. Thus, if these chunks are chosen too large, overshooting and oscillation effects may occur. This issue can be resolved by choosing the migration rate in step 2b of the replication policy carefully. In [13] it was shown that if we choose $\lambda = \Theta(1/\ell'_{\max})$ small enough with

$$\ell'_{\max} = \max_{P \in \mathcal{P}} \max_{f \in \mathcal{F}} \sum_{e \in P} \ell'_e(f) ,$$

convergence towards Wardrop equilibria can be guaranteed. We may assume that $\ell'_{\max} > 0$ since otherwise all latency functions are constant and our problem can be solved trivially by assigning the entire flow to the path with lowest latency.

Theorem 1 ([13, 12]). *If $\lambda = \Theta(1/\ell'_{\max})$ sufficiently small, the replication policy given by Equation (1) with initial flow $f(0) = f^0$ converges towards a Wardrop equilibrium if $f_P^0 > 0$ for all $P \in \mathcal{P}$. Furthermore, the number of rounds in which the flow is not at a δ - ϵ -equilibrium is*

$$\mathcal{O} \left(\frac{1}{\epsilon^2 \delta^2} \cdot \frac{\ell'_{\max}}{\ell_{\min}} \cdot \ln \left(\frac{\Phi(f^0)}{\Phi^*} \right) \right) .$$

One may observe that the ratio between maximum slope and minimum latency used in this theorem depends on the scale by which we measure flow. This scale, however, is fixed since we have normalised the total flow demand to be $r = 1$.

3.3 Simulating the Replication Policy

By a naive application of Theorem 1 we can compute a sequence of flow vectors $(f(t))_{t \geq 0}$ according to Equation (1) to obtain approximate Wardrop equilibria.

However, this approach is rendered computationally intractable by the fact that there may be an exponential number of variables f_P .

In the following, we describe an algorithm that computes the iterative change rates of the edge flows according to the implicit flow decomposition \tilde{f} described in the preceding section. To that end, we show that the change rates of the edge flows f_e can be expressed solely in terms of edge flows and edge latencies (i. e., without explicit reference to the f_P variables). It suffices to know the weighted average latencies of all paths containing e defined as

$$L_e = \sum_{P \ni e} \frac{f_P}{f_e} \cdot \ell_P$$

Recall that we have fixed a commodity here, so we may drop the index i .

Lemma 1. *Consider an edge flow vector $(f_e(t))_{e \in E}$ and its path decomposition $\tilde{f}(t)$, and let $\tilde{f}(t+1)$ denote the flow generated by the replication policy in Equation (1) from $\tilde{f}(t)$. Finally, let $f_e(t+1) = \sum_{P \ni e} \tilde{f}_P(t+1)$. Then,*

$$f_e(t+1) = f_e(t) + \lambda \cdot f_e \cdot (L - L_e) \ .$$

Proof. Let $f = f(t)$ and $f' = f(t+1)$. By definition of f_e ,

$$f'_e - f_e = \sum_{P \ni e} f'_P - f_P = \lambda \cdot \sum_{P \ni e} f_P \cdot (L - \ell_P) = \lambda \cdot f_e \cdot \left(L - \frac{\sum_{P \ni e} f_P \ell_P}{f_e} \right) \ ,$$

where the last term equals L_e . \square

In order to obtain the value of L_e , we implicitly compute the path decomposition \tilde{f} , i. e., for every edge e' we compute the flow caused by paths containing e on edge e' . This is done by Algorithm SIMULATEDREPLICATION (Algorithm 1) in time $\mathcal{O}(m)$ for every edge $e \in E$. Since there are m edges, each iteration can be performed in time $\mathcal{O}(m^2)$.

Algorithm 1 SIMULATEDREPLICATION() (executed by all commodities in parallel; $(f_e)_{e \in E}$ denotes the edge flows vector of commodity i)

- 1: **for** all edges $e \in E$ **do**
 - 2: sort all edges (v, w) in the subgraph reachable from e topologically
 - 3: compute total flow of all paths containing e and (v, w) $\tilde{f}_e^{(v,w)} = \sum_{(u,v) \in E} \tilde{f}_e^{(u,v)} \cdot \frac{f_{(v,w)}}{f_v}$
 - 4: reverse all edges and repeat steps 2 and 3 for edges between e and s
 - 5: compute $L_e = \sum_{e'} \frac{f_{e'}}{f_e} \ell_{e'}$
 - 6: $f'_e \leftarrow f_e + \lambda \cdot f_e \cdot (L - L_e)$ with $\lambda = 1/\ell'_{\max}$
 - 7: **end for**
 - 8: replace $(f_e)_{e \in E}$ on the billboard with $(f'_e)_{e \in E}$
-

Corollary 1. *The sequence of flow vectors computed by Algorithm SIMULATEDREPLICATION converges towards the set of Wardrop equilibria. Furthermore, the number of rounds in which the flow is not at a δ - ϵ -equilibrium with respect to \tilde{f} , is bounded by*

$$\mathcal{O}\left(\frac{1}{\epsilon^2 \delta^2} \cdot \frac{\ell'_{\max}}{\ell_{\min}} \cdot \ln\left(\frac{\Phi(f^0)}{\Phi^*}\right)\right),$$

where f^0 is the initial flow vector. Each iteration takes time $\mathcal{O}(m^2)$.

Proof. Lemma 1 implies that the edge flow vector computed by the algorithm equals the edge flow vector obtained by applying the replication policy given by Equation (1) to the path decomposition $(\tilde{f})_{P \in \mathcal{P}}$. Combining this with the upper bounds on the convergence time given in [12], the claim follows. \square

4 The Polynomial Time Algorithm

The migration probability specified for step 2b of the replication policy can get very small since the latency difference $\ell_P - \ell_Q$ may become small in relation to λ if λ is chosen constant. This causes the algorithm to obtain only a pseudopolynomial convergence time depending on the maximum slope of the latency functions. In this section we present an approach that gets rid of this dependence.

To this end, we choose the amount of flow removed from a path proportional to its *relative* deviation $(\ell_P - L_{i_P})/\ell_P$ from the average and the reciprocal of the elasticity d to obtain a polynomial number of communication rounds. Whereas in the preceding section the amount of flow removed or added to a path within one round could be expressed in a nice closed form as $\lambda \cdot f_P \cdot (L - \ell_P)$ (Equation (1)), this is now no longer possible.

To compute flow updates in polynomial time we use a randomised flow decomposition. First we sample a path at random according to the implicit path decomposition \tilde{f} , i. e., the probability to sample path P is \tilde{f}_P/r_{i_P} . Since the length of a path is bounded by n this is possible in time $n \log n$ by representing adjacent nodes and their flows in a binary tree. Now, the path is assigned a certain flow volume f_P . For the time being, assume that we assign the entire bottleneck flow to P . Then, if P has latency above L_{i_P} , we remove a portion of

$$x = \Theta\left(f_P \cdot \frac{\ell_P - L_{i_P}}{d \ell_P}\right)$$

of its flow and distribute it proportionally among all admissible paths, i. e., after removing a flow of x from path P , the flow on every edge $e \in E$ is increased by $(f_{e,i}/r_i) \cdot x$.

Why does this process decrease the potential quickly? As long as we are not at a δ - ϵ -equilibrium, the probability of sampling a δ -expensive path is at least ϵ . In this case, the latency gain and thus the potential gain *per flow unit* will be large and proportional to \tilde{f}_P . If we sample only a single path, we may in fact

assign the entire bottleneck flow to it. We can lower bound the probability that this bottleneck flow is not too small (Lemma 2). To increase the potential gain we repeat this process several times. Doing this, we can no longer assign the entire bottleneck to a path since it may happen that an edge is sampled several times. Hence, we assign at most a $\Theta(1/\log m)$ fraction of the bottleneck while at the same time sampling $T = m \log m$ paths rather than only a single one. It thus becomes unlikely that an edge becomes empty along the way if its flow is $\mathcal{O}(1/m)$. In order to achieve the same result for edges with larger flow, we limit the amount of flow consumed in one step to $\mathcal{O}(1/(m \log m))$. More precisely, let

$$\Delta_e = \min \left\{ \frac{1}{7 m \log m}, \frac{f_e}{7 \log m} \right\} .$$

We start with an empty decomposition. In a round in which path P is sampled we increase f_P by Δ_{e^*} where e^* is a bottleneck edge in P . We say that an edge is *alive* if the overall flow assigned to paths containing e is at most $f_e - \Delta_e$ (i. e. it can be sampled one more time without having our decomposition exceeding the flow of e). Our algorithm terminates as soon as there are any edges that are not alive. The final algorithm `RANDOMISEDBALANCING(d)` is described in Algorithm 2.

Under the assumption that the latency functions are constant, we can thus show that the potential decreases in every round by a factor that only depends on ϵ and δ , and the elasticity d (Lemma 5). We furthermore show that due to our careful migration rate the potential gain with respect to the true latency functions is still at least half of the potential gain with respect to constant latencies (Lemma 4). Finally, we show that the expected potential gain implies a bound on the time to reach a minimum potential (Lemma 6). Altogether, this yields the following upper bound for our algorithm.

Theorem 2. *The sequence of flow vectors computed by Algorithm `RANDOMISEDBALANCING` converges towards the set of Wardrop equilibria. Furthermore, the expected number of rounds in which the flow is not at a δ - ϵ -equilibrium with respect to \tilde{f} , is bounded by*

$$\mathcal{O} \left(\frac{d}{\epsilon^3 \delta^2} \log \left(\frac{\Phi(f_0)}{\Phi^*} \right) \right) ,$$

if d is an upper bound on the elasticity of the latency functions. The computation time of each round is bounded by $\mathcal{O}(n \log n \cdot m \log m)$.

We present the proof after establishing the necessary lemmas.

Note that our algorithm can be easily modified for the case that the elasticity of the latency functions is not known to the algorithm in advance.

4.1 Randomised Decomposition

Our algorithm generates a randomised flow decomposition using a sampling process based on f . In this section, we lower bound the probability that the bottleneck flows of the sampled paths are not too small. Furthermore, we show that the flow removed from every edge is at most f_e with high probability.

Algorithm 2 RANDOMISEDBALANCING(d) (executed by all commodities in parallel; $(f_e)_{e \in E}$ denotes the edge flows vector of commodity i)

```

1: for  $T = m \log m$  times do
2:   sample a path  $P$  where  $\mathbb{P}[P] = \frac{\tilde{f}_P}{r_i}$ 
3:   let  $e^*$  denote the bottleneck edge of  $P$ ; let  $f_P = \Delta_{e^*}$ 
4:   if  $\ell_P > L_i$  then
5:     reduce the flow on all edges  $e \in P$  by  $\Delta f_P = f_P \cdot \frac{\ell_P - L_i}{4d\ell_P}$ 
6:     if for any  $e \in P$ ,  $e$  is not alive then
7:       abort loop and continue in line 11
8:     end if
9:   end if
10: end for
11: increase the flow on all edges  $e \in E$  proportionally by  $\frac{f_e}{r_i} \cdot \sum_{P \in \mathcal{P}_i} \Delta f_P$ 

```

Lemma 2. Consider a flow vector f of volume 1 and a set of paths \mathcal{P}_ϵ with $\sum_{P \in \mathcal{P}_\epsilon} \tilde{f}_P = \epsilon$. Then, $\mathbb{P}_{P \sim \tilde{f}} [P \in \mathcal{P}_\epsilon \wedge \min_{e \in P} f_e \geq \frac{\epsilon}{2m}] \geq \frac{\epsilon}{2}$.

Proof. We consider a scaled flow vector which supports only paths in \mathcal{P}_ϵ .

$$f'_P = \begin{cases} \frac{\tilde{f}_P}{\epsilon} & P \in \mathcal{P}_\epsilon \\ 0 & P \notin \mathcal{P}_\epsilon \end{cases}.$$

Observe that the total volume of f' is 1 again, hence $\mathbb{P}_{P \sim f'} [P = Q] = \mathbb{P}_{P \sim \tilde{f}} [P = Q \mid P \in \mathcal{P}_\epsilon]$. Now,

$$\begin{aligned} \mathbb{P}_{P \sim \tilde{f}} \left[P \in \mathcal{P}_\epsilon \wedge \min_{e \in P} f_e \geq \frac{\epsilon}{2m} \right] &= \mathbb{P}_{P \sim \tilde{f}} [P \in \mathcal{P}_\epsilon] \cdot \mathbb{P}_{P \sim \tilde{f}} \left[\min_{e \in P} f'_e \geq \frac{1}{2m} \mid P \in \mathcal{P}_\epsilon \right] \\ &= \epsilon \cdot \mathbb{P}_{P \sim f'} \left[\min_{e \in P} f'_e \geq \frac{1}{2m} \right] \geq \frac{\epsilon}{2}, \end{aligned} \quad (2)$$

where the first equality uses the definition of f' and the second one uses the above observation.

Now, let $d(x, y)$ denote the number of edges of a shortest path connecting x and y . We can show that $\mathbb{P}[e = (v, w) \in P] = f_e$ by induction on $d(s, v)$. This holds for $d(s, v) = 0$ by definition of \tilde{f} . Now, assume that the statement holds for all edges (u, v) with $d(s, u) = k$ and consider an edge $e = (v, w)$ with $d(s, v) = k + 1$.

$$\begin{aligned} \mathbb{P}[e \in P] &= \mathbb{P}[v \in P] \cdot \mathbb{P}[e \in P \mid v \in P] = \sum_{(u, v)} \mathbb{P}[(u, v) \in P] \cdot \frac{f_e}{f_v} \\ &= \sum_{(u, v)} f_{(u, v)} \cdot \frac{f_e}{f_v} = f_e. \end{aligned}$$

With $E' = \{e \in E \mid f_e \leq 1/(2m)\}$,

$$\mathbb{P}[P \ni e : e \in E'] \leq \sum_{e \in E'} \mathbb{P}[e \in P] \leq \sum_{e \in E'} f_e \leq \frac{|E'|}{2m} \leq \frac{1}{2}.$$

Substituting this into Equation (2) yields our desired bound. \square

We now consider a sequence of $T = m \log m$ rounds. Observe that Δ_e is an upper bound on the flow removed from a path containing e by our algorithm, since for the bottleneck edge e^* , $\Delta_{e^*} = \min_{e \in P} \{\Delta_e\}$. The flow on e may decrease to below zero only if it is contained in the sampled path at least f_e/Δ_e times. In the following we show that this is unlikely.

Lemma 3. *With probability $1 - o(1)$, after a sequence of $T = m \log m$ iterations, all edges are still alive.*

Proof. In the proof of Lemma 2 we have seen that the probability to hit edge e in one round equals f_e . Let the random variable X denote the number of hits in T rounds. We have $\mathbb{E}[X] = T f_e$. An edge is alive if $X \leq f_e/\Delta_e - 1$. There are two cases:

1. $f_e < \frac{1}{m}$ implying $\Delta_e = f_e/(7 \log m)$. Then,

$$\begin{aligned} \mathbb{P}\left[X > \frac{f_e}{\Delta_e} - 1\right] &= \mathbb{P}\left[X > \mathbb{E}[X] \cdot \left(\frac{7}{f_e m} - \frac{1}{T f_e}\right)\right] \\ &\leq \mathbb{P}\left[X > \mathbb{E}[X] \cdot \left(\frac{6}{f_e m}\right)\right] \\ &\leq 2^{-\mathbb{E}[X] \cdot \frac{6}{f_e m}} = m^{-6}. \end{aligned}$$

The first inequality is the definition of T and Δ_e and uses our assumption that $f_e \cdot m < 1$, and the second inequality is Chernoff's inequality (which asserts that $\mathbb{P}[X \geq r \cdot \mathbb{E}[X]] \leq 2^{-r \cdot \mathbb{E}[X]}$ for $r \geq 6$ for a random variable X that is the sum of 0-1 random variables, see [16]).

2. $f_e \geq \frac{1}{m}$ implying $\Delta_e = 1/(7T)$. This case can be treated similarly.

In both cases, the probability that edge e is not alive at the end of a sequence of T iterations is bounded by m^{-6} . Using a union bound, the probability that at least one edge does not survive is at most m^{-5} and consequently the probability that all edges survive the sequence is at least $1 - m^{-5}$. \square

4.2 Lower Bounding the Potential Gain

We use a potential function argument to prove convergence. In order to show that our algorithm avoids oscillations, we consider the potential gain achieved within one round. We show that this potential gain is at least half of the potential gain that would occur if latencies values were fixed at the beginning of a round. A second lemma shows that, in expectation, the potential decreases by a factor in every round, as long as we are not yet at an approximate equilibrium.

Lemma 4. *Let d denote an upper bound on the elasticity of the latency functions. For a flow vector f consider a flow vector f' generated by Algorithm RANDOMISEDBALANCING(d) (Algorithm 2) with positive probability. For any $P \in \mathcal{P}$ let Δf_P denote the amount of flow removed from path P . Then, $\Phi(f) - \Phi(f') \geq \frac{1}{2} \cdot \sum_{P \in \mathcal{P}} (\ell_P(f) - L_{i_P}) \cdot \Delta f_P$.*

Due to space limitations, we defer the proof to the full version.

Lemma 5. *Assume that f is a flow that is not at δ - ϵ -equilibrium and let the random variable f' denote a flow generated by our algorithm. Then, $\mathbb{E}[\Phi(f')] \leq \Phi(f) \cdot \left(1 - \Omega\left(\frac{\epsilon^3 \delta^2}{d}\right)\right)$.*

Proof. For the time being, assume that the latency functions are constant. By Markov's inequality, the total volume of flow in commodities with $L_i > 2 \cdot L/\epsilon$ is at most $\epsilon/2$. We consider only commodities with $L_i \leq 2 \cdot L/\epsilon$. In total, at least a flow volume of ϵ utilises δ -expensive paths and there is still at least a volume of $\epsilon/2$ left in the commodities we consider. Consider such a commodity $i \in [k]$ and denote the flow volume using δ -expensive paths in this commodity by ϵ_i .

Consider any iteration satisfying the precondition that all edges are alive. Let P denote the path sampled by the algorithm. By Lemma 2, the probability that $\ell_P \geq L_i + \delta L$ and the minimum edge flow along P is at least $\epsilon_i/(2m)$ is at least $\epsilon_i/(2r_i)$ (we have to scale the flow of this commodity by a factor $1/r_i$ to make it a unit flow). The amount of flow removed from this path by our algorithm is

$$\frac{\epsilon_i}{2m} \cdot \frac{1}{7 \log m} \cdot \frac{\ell_P - L_i}{4d\ell_P} \geq \frac{\epsilon_i \epsilon \delta}{113dm \log m}$$

where we have used that $\ell_P \geq L_i + \delta L$ and $L_i \leq 2L/\epsilon$. The latency gain of this path is then at least δL and since this event happens with probability $\epsilon_i/(2r_i)$ the expected virtual potential gain of such a path is then at least

$$\frac{\epsilon_i^2 \epsilon \delta^2}{226dr_i m \log m} L .$$

By Lemma 3 the probability that in this iteration all edges are alive is $1 - o(1)$ and the expected potential gain computed above is independent of this event. Summing up over all $T = m \log m$ iterations and all commodities, the total expected virtual potential gain of one round is at least

$$(1 - o(1)) \cdot \sum_{i \in [k]} \frac{\epsilon_i^2 \epsilon \delta^2}{226dr_i} L \geq (1 - o(1)) \cdot \frac{\epsilon^3 \delta^2}{226d} L .$$

For the last inequality we have used the Cauchy-Schwarz Inequality which asserts that for two vectors (a_i) and (b_i) , $\sum_i a_i^2 \geq (\sum_i a_i b_i)^2 / \sum_i b_i^2$. Using $a_i = \epsilon_i / \sqrt{r_i}$ and $b_i = \sqrt{r_i}$ yields the result. This implies the claim since L is an upper bound on Φ and Lemma 4 ensures that the true potential gain with respect to the real latency functions is at least half of the potential gain with respect to the constant latency functions. \square

4.3 From Expected Potential Gain to Expected Stopping Time

The preceding section has shown that in every round the potential decreases by a factor in expectation. Intuitively, this implies an expected running time that is logarithmic in this factor and the initial values. This intuition is made precise by the following lemma.

Lemma 6. *Let X_0, X_1, \dots denote a sequence of non-negative random variables and assume that for all $i \geq 0$ $\mathbb{E}[X_i \mid X_{i-1} = x_{i-1}] \leq x_{i-1} \cdot \alpha$ for some constant $\alpha \in (0, 1)$. Furthermore, fix some constant $x^* \in (0, x_0]$ and let τ be the random variable that describes the smallest t such that $X_t \leq x^*$. Then, $\mathbb{E}[\tau \mid X_0 = x_0] \leq \frac{2}{\log(1/\alpha)} \cdot \log\left(\frac{x_0}{x^*}\right)$.*

We defer the proof to the full version. Finally, we can proof our main result.

Proof (Proof of Theorem 2). Let f_0, f_1, \dots denote a sequence of flow vectors generated by Algorithm 2. Lemma 5 implies that

$$\mathbb{E}[\Phi(f_{t+1}) \mid \Phi(f_t) = \phi] \leq \phi \cdot \left(1 - \Omega\left(\frac{\epsilon^3 \delta^2}{d}\right)\right).$$

Thus, the sequence $(\Phi(f_t))_{t \geq 0}$ satisfies the conditions of Lemma 6 and the expected time until $\Phi(f_t)$ reaches its minimum Φ^* implying that f_t is a δ - ϵ -equilibrium is

$$\frac{2}{\log\left(\left(1 - \Omega\left(\frac{\epsilon^3 \delta^2}{d}\right)\right)^{-1}\right)} \log\left(\frac{\Phi(f_0)}{\Phi^*}\right) = \mathcal{O}\left(\frac{d}{\epsilon^3 \delta^2} \log\left(\frac{\Phi(f_0)}{\Phi^*}\right)\right),$$

our desired bound.

One path can be sampled in time $\mathcal{O}(n \log n)$, the bottleneck edge can be found in time $\mathcal{O}(n)$, and the flow update can be computed in time $\mathcal{O}(n)$. Altogether, at most $T = m \log m$ iterations have to be computed. Finally, the removed flow can be reinserted in time $\mathcal{O}(n)$. \square

5 Open Problems

Our algorithm works by redistributing flow of overloaded paths. To identify such paths we face the subproblem of finding a flow decomposition that assigns much flow to paths with high latency. In our algorithm we have used a randomised path decomposition to achieve this goal. It is not obvious whether this randomisation can be avoided, and, in fact, naive deterministic approaches like longest path first decompositions fail.

In the long run, our algorithm converges towards the set of Wardrop equilibria. A weakness of our notion of approximate equilibria, however, is the fact that the average latency may be arbitrarily far away from the minimum latency. As an alternative, one could also consider deviations from the minimum latency rather than from the average latency. It is unclear whether convergence towards approximate equilibria in this sense can be guaranteed in polynomial time. Furthermore, it would be desirable to design specialised (not necessarily distributed) algorithms to compute (exact) Wardrop equilibria that improve upon the standard solution via convex programming.

References

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: Theory, algorithms and applications*. Prentice-Hall, 1993.
2. Baruch Awerbuch and Rohit Khandekar. Greedy distributed optimization of multi-commodity flows. In *Proc. 26th Ann. Symp. on Principles of Distributed Computing (PODC)*, 2007.
3. Baruch Awerbuch, Rohit Khandekar, and Satish Rao. Distributed algorithms for multicommodity flow problems via approximate steepest descent framework. In *Proc. 18th Ann. Symp. on Discrete Algorithms (SODA)*, 2007.
4. Baruch Awerbuch and Robert D. Kleinberg. Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches. In *Proc. 36th Ann. Symp. on Theory of Comput. (STOC)*, pages 45–53, 2004.
5. M. Beckmann, C. B. McGuire, and C. B. Winsten. *Studies in the Economics of Transportation*. Yale University Press, 1956.
6. Petra Berenbrink, Tom Friedetzky, Leslie Ann Goldberg, Paul Goldberg, Zengjian Hu, and Russel Martin. Distributed selfish load balancing. In *Proc. 17th Ann. Symp. on Discrete Algorithms (SODA)*, 2006.
7. Dimitri P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.
8. Avrim Blum, Eyal Even-Dar, and Katrina Ligett. Routing without regret: On convergence to Nash equilibria of regret-minimizing algorithms in routing games. In *Proc. 25th Ann. Symp. on Principles of Distributed Computing (PODC)*, pages 45–52, July 2006. ACM.
9. Richard Cole, Yevgeniy Dodis, and Tim Roughgarden. How much can taxes help selfish routing? In *Proc. 4th Conf. on Electronic Commerce*, pages 98–107, 2003.
10. Roberto Cominetti, José R. Correa, and Nicolás E. Stier Moses. Network games with atomic players. In *Proc. 33rd Int. EATCS Coll. on Automata, Languages and Programming (ICALP)*, pages 525–536, 2006.
11. Eyal Even-Dar and Yishay Mansour. Fast convergence of selfish rerouting. In *Proc. 16th Ann. Symp. on Discrete Algorithms (SODA)*, pages 772–781, 2005.
12. Simon Fischer, Harald Räcke, and Berthold Vöcking. Fast convergence to Wardrop equilibria by adaptive sampling methods. In *Proc. 38th Symposium on Theory of Computing (STOC)*, pages 653–662, May 2006. ACM.
13. Simon Fischer and Berthold Vöcking. Adaptive routing with stale information. In Marcos Kawazoe Aguilera and James Aspnes, editors, *Proc. 24th Symp. on Principles of Distributed Computing (PODC)*, pages 276–283, July 2005. ACM.
14. Lisa Fleischer. Linear tolls suffice: New bounds and algorithms for tolls in single source networks. In *Proc. 31st Int. EATCS Coll. on Automata, Languages and Programming (ICALP)*, pages 544–554, 2004.
15. Paul W. Goldberg. Bounds for the convergence rate of randomized local search in a multiplayer, load-balancing game. In *Proc. 23rd Symp. on Principles of Distributed Computing (PODC)*, pages 131–140. ACM, July 2004.
16. Torben Hagerup and Christine Rüb. A guided tour of Chernoff bounds. *Information Processing Letters*, 33:305–308, 1990.
17. Tim Roughgarden. How unfair is optimal routing? In *Proc. 13th Ann. Symp. on Discrete Algorithms (SODA)*, pages 203–204, 2002.
18. Tim Roughgarden and Éva Tardos. How bad is selfish routing? *J. ACM*, 49(2):236–259, 2002.
19. John Glen Wardrop. Some theoretical aspects of road traffic research. In *Proc. of the Institute of Civil Engineers, Pt. II*, volume 1, pages 325–378, 1952.