Minimum Delay Load-Balancing via Nonparametric Regression and No-Regret Algorithms

Federico Larroca[a,1,*], Jean-Louis Rougier[a]

[a] *Telecom ParisTech. 46 rue Barrault, F-75634 Paris Cedex 13, Paris, France*

Abstract

In the current network scenario, where traffic is increasingly dynamic and resource demanding, Dynamic Load-Balancing (DLB) has been shown to be an excellent Traffic Engineering tool. In particular, we are interested in the problem of minimum delay load-balancing. That is to say, we assume that the queueing delay of a link is given by a function of its load. The objective is then to adjust the traffic distribution over paths so that, for the current traffic demand, the addition of these functions times the load is minimized. The contribution of our article is twofold. Firstly, we analyze the possibility of using so-called *no-regret* algorithms to perform the load balancing. As opposed to other distributed optimization algorithms (such as the classical gradient descent) the algorithm we discuss requires no fine-tuning of any speed-controlling parameter. Secondly, we present a framework that does not assume any particular model for the queueing delay function, and instead learns it from measurements. This way, the resulting mean delay of optimizing with this learnt function is an excellent approximation of the real minimum delay traffic distribution. The whole framework is illustrated by several packet and flow level simulations.

Keywords: Wardrop Equilibrium, Convex Nonparametric Least Squares, Weighted Least Squares, No-Regret

1. Introduction

Two aspects of the current networking scenario call for new and more effective ways of handling traffic. First of all, service convergence in the same network, in addition to the ever increasing access rates available for end-users, have led to very dynamic and unpredictable traffic patterns. Secondly, the assumption that overprovisioning is the panacea for all problems is being increasingly reconsidered. This is due on the one hand to the emergence of new architectures with intrinsically scarce resources (e.g. Wireless Mesh Networks), and on the other hand to the fact that the assumption that core capacities are orders of magnitude higher than access rates may no longer hold in the near future.

Robust Routing (RR) [1, 2, 3] has emerged in recent years as a possible answer to the above problems. In RR, traffic uncertainty is taken into account directly within the routing optimization, computing a single routing configuration for all traffic demands within an *uncertainty set* where traffic is assumed to vary. This uncertainty set can be defined in different ways, depending on the available information; e.g. largest values of link loads previously seen, a set of previously observed demands (previous day, same day of the previous week, etc.). Although relatively simple and naturally stable, RR presents some important drawbacks. Firstly, the definition of the set of traffic demands presents a clear tradeoff: larger sets are able to handle a broader group of traffic demands, but at the cost of routing inefficiency; conversely, tighter sets produce more efficient routing schemes, but are subject to poor performance guarantees [3]. Secondly, optimization under uncertainty is generally more complex than classical optimization, which forces the use of simpler optimization criteria. In this sense, the typical objective is to minimize the maximum link utilization (MLU) in the network. Other more complex objective functions (such as the one considered in this article) generally result in a marginally bigger MLU and an improvement in other performance indicators (such as the mean link utilization) that may be significant (see for instance [4, 5] or the results we present here).

To deal with these drawbacks, *Dynamic Load-Balancing* (DLB) has been proposed [5, 6, 7, 8, 9]. In these schemes each Origin-Destination (OD) pair is connected by several, established a priori, paths. Based on feedback from the network, each OD pair dynamically adjusts the portion of its traffic sent through each path. The objective is, given the configured paths and the current traffic demand, to minimize a certain network-wide cost function. Although this "always optimized" characteristic is very attractive, the deployment of DLB has been, to say the least, limited. Network operators are reluctant to use dynamic

*Corresponding author. Telephone: +598 2 711 09 74 - Fax: +598 2 711 74 35

Email addresses: flarroca@fing.edu.uy (Federico Larroca), rougier@telecom-paristech.fr (Jean-Louis Rougier)

[1] Present Address: Facultad de Ingeniería, Universidad de la República. Julio Herrera y Reissig 565, C.P. 11.300, Montevideo, Uruguay

mechanisms mainly because they are afraid of a possible oscillatory behavior of the algorithm used by each OD pair to adjust their traffic distribution (as the early experiences in ArpaNet has proved [10], these concerns are not without reason). Indeed, previously proposed DLB algorithms always include a parameter that controls the convergence speed, which is very tricky to assign. Although for each algorithm there exists a range for this parameter in which it is stable, these values result in unresponsiveness in certain situations.

Our first contribution is to present a Dynamic Load-Balancing algorithm based on so-called *no-regret* algorithms. The authors of [11] proved that OD pairs using algorithms of this kind converge to a greedy equilibrium which may be induced to coincide with the optimum we are looking for (see the following sections for a more detailed discussion). In particular, we will use a variation of the *Incrementally Adaptive Weighted Majority Algorithm* [12], which presents the advantage of being completely self-regulated, thus avoiding any tricky speed-controlling parameter setting and the reactivity-stability tradeoff we mentioned before. Furthermore, we will present certain adaptations of the algorithm that are necessary to enable its use in the presence of non-stationary traffic.

In most DLB schemes the objective function is the addition over all links of a certain link-cost function. The idea is that this function should measure the congestion on the link, for which the queueing delay is generally used. The choice is justified by its versatility (big queueing delays mean bad performance for all types of traffic) and simple algebra (the total delay of a path is the addition of the delay at each link). However, most DLB schemes require an analytical expression of this delay, for which classic and oversimplistic models (e.g. *M/M/1*) are used [6], resulting in an actual total delay that is significantly bigger than the optimum.

As a second contribution of this article, we propose a framework that makes very few assumptions on the delay function. Except for some natural hypothesis on its shape (e.g. monotonicity) we will only assume that the queueing delay on link l is of the form $f_l(\rho_l)$ (i.e. depends only on the mean load of the link). The actual form of $f_l(\rho_l)$ will be obtained (or learned) from past measurements. To achieve this we present two different regression methods. The first one is a variation of the nonparametric regression method presented in [15], and finds the regression function that best fits the measurements. However, it presents scalability issues as the number of available measurements increases. We consider then the algorithm presented in [16]. This heuristic finds a parametric function that reasonably adjusts the measurements in a very short time, although its precision is not as good as the one obtained by the first method.

The complete framework is illustrated by several flow as well as packet level simulations using a real topology and several real traffic demands. For instance, our study indicates that using the *M/M/1* model instead of our learned function results in an increase in the total queueing delay that may easily exceed 10%, and can go as high as more than 80%. Moreover, the comparison with previous proposals in terms of link utilization shows that our framework either outperforms them, or the difference is not significant.

2. Greedy Load-Balancing

2.1. Network Model

The network is defined as a graph $G = (V, E)$. In it there are a number of so-called *commodities* (or OD pairs), indexed by $s = 1, .., S$, specified in terms of the triplet o_s, q_s and d_s; i.e. origin node, destination node and a certain fixed demand of traffic from the former to the latter. Commodity s can use any path from set $\mathcal{P}_s$, where each of its elements (noted as P_{si} with $i = 1, .., n_s$) is a subset of E connecting o_s to q_s. In practice, paths are chosen to be no less than two per commodity, and to differ as much as possible for resiliency reasons. In particular, we used the method presented in [26] to choose these paths.

All commodities can distribute their total demand arbitrarily along their paths. In particular, commodity s sends an amount $d_{P_{si}}$ of its traffic through path P_{si}, where $d_{P_{si}} \geq 0$ and $\sum d_{P_{si}} = d_s$. This distribution of traffic induces the demand vector $d = (d_{P_{si}})$.

Given the demand vector, the total load on link l is then $\rho_l = \sum_s \sum_{P \in \mathcal{P}_s : l \in P} d_P$. The presence of this traffic on the link induces a certain mean queueing delay given by the non-decreasing function $D_l(\rho_l)$. The total delay of path P is defined as $D_P = \sum_{l : l \in P} D_l(\rho_l)$. As the measure of the network total congestion we shall use the *mean end-to-end queueing delay* (or mean total delay, which is the term generally used) $D(d)$, defined as:

$$D(d) = \sum_{s=1}^{S} \sum_{P \in \mathcal{P}_s} d_P D_P = \sum_{l=1}^{L} D_l(\rho_l)\rho_l := \sum_{l=1}^{L} f_l(\rho_l)$$

that is to say, a weighted mean delay, where the weight for each path is how much traffic is sent through it, or in terms of the links, the weight of each link is how much traffic is traversing it. We prefer this weighted mean to a simple total delay because it reflects more precisely performance as perceived by traffic. Two situations where the total delay is the same, but in one of them most of the traffic is traversing heavily delayed links, should not be considered as equivalent. Note that, by Little's law, $f_l(\rho_l) := D_l(\rho_l)\rho_l$ is proportional to the average number of bytes in the queue of link l. We will then use this last value as $f_l(\rho_l)$ which (like the mean load) is readily available in most routers.

We can now write the problem explicitly:

$$\underset{d}{\text{minimize}} \sum_{l=1}^{L} f_l(\rho_l) \qquad \text{s.t.} \quad d_{P_{si}} \geq 0 \quad \sum_{P \in \mathcal{P}_s} d_P = d_s \quad (1)$$

Note that no explicit constraint on ρ_l was made. This is assumed to be implicitly included in the delay function.

For instance, if there exists a capacity constraint, $f_l(\rho_l)$ should go to infinity (or a big value) as ρ_l reaches c_l (the capacity of link l). It should also be noted that in the framework described above the destination for a commodity is not necessarily a single node (e.g. two gateways to the internet may be seen as a single destination).

2.2. Wardrop Equilibrium

In this section we present and discuss how to solve problem (1) in a distributed fashion. In particular, we will consider greedy mechanisms since they require minimum coordination among commodities. In this kind of mechanism, a path cost ϕ_P is defined, and each commodity greedily minimizes the cost it obtains from each of its paths. This context constitutes an ideal case study for game theory, and is known as *Routing Game* in its lingo [17].

In a routing game like ours, where the traffic generated by a commodity may be arbitrarily distributed among paths, commodities are assumed to be constituted of an infinite number of agents. These agents control an infinitesimal amount of traffic, and decide along which path to send their traffic ($d_{P_{si}}/d_s$ represents then the fraction that have P_{si} as their choice for commodity s). If each of these agents acts selfishly, then the system will be at equilibrium when no agent may decrease its cost by unilaterally changing its path decision. This situation constitutes what is known as a *Wardrop Equilibrium* (WE) [18], which is formally defined as follows:

Definition 1. A demand vector is a Wardrop Equilibrium if for each commodity $s = 1 \dots S$ and for each path P_{si} with $d_{P_{si}} > 0$ it holds that $\phi_{P_{si}} \leq \phi_{P_{sj}}$ for all $P_{sj} \in \mathcal{P}_s$.

It is easy to see that in a WE, and for any given commodity s, all paths with $d_{P_{si}} > 0$ have the same cost ϕ_{P_s}, namely the minimum among all paths of the corresponding commodity.

Depending on the definition of ϕ_P there are roughly two types of routing games. The first one, known as *Bottleneck Routing Game* [19], defines $\phi_P = \max_{l \in P}\{\phi_l(\rho_l)\}$, where $\phi_l(\rho_l)$ is known as the link cost function. It can be proved that, under reasonable conditions, the resulting WE minimizes the maximum $\phi_l(\rho_l)$ over all links. The second type does not take the maximum $\phi_l(\rho_l)$ as the path cost ϕ_P, but rather the sum (i.e. $\phi_P = \sum_{l:l \in P} \phi_l(\rho_l)$), and is known as *Congestion Routing Game* [17]. It can be proved that, if $\phi_l(\rho_l)$ is positive and nondecreasing, the corresponding WE results in a local minimum of the so-called potential function:

$$\Phi(d) = \sum_{l=1}^{L} \int_0^{\rho_l} \phi_l(x)dx$$

Given an optimization problem like (1), we may then find a link cost function $\phi_l(\rho_l)$ such that the resulting WE is the optimum demand vector. In this case, let us consider the following cost function:

$$\phi_l(\rho_l) = \frac{\partial f_l(\rho_l)}{\partial \rho_l} \Rightarrow \tag{2}$$

$$\Phi(d) = \sum_{l=1}^{L} \int_0^{\rho_l} \frac{\partial f_l(x)}{\partial x} dx = \sum_{l=1}^{L} \left(f_l(\rho_l) - f_l(0)\right)$$

This means that the WE of a congestion routing game where the link cost $\phi_l(\rho_l)$ is the derivative of $f_l(\rho_l)$ results in a local minimum of (1) (note that since $f_l(0)$ is a constant, the optimum d is the same with or without its addition). However, since $\phi_l(\rho_l)$ should be positive and nondecreasing, $f_l(\rho_l)$ should be increasing and convex (meaning that the local minimum is then the unique global minimum demand vector).

3. No-Regret Algorithms

3.1. Definition and Results

In this section we will present an algorithm that, given the link cost $\phi_l(\rho_l)$, converges to the corresponding WE. As we mentioned in Sec. 1 we will consider so-called no-regret algorithms. This kind of algorithm is iteratively applied over time, and as such we shall note the demand vector at time-step t as d^t. In the context of a routing congestion game, the *per-time-step regret* incurred by a commodity is defined as the difference between its average path cost and the cost of the best fixed path at hindsight. That is to say, for a total time T, regret for commodity s is defined as follows:

$$\frac{1}{Td_s} \sum_{t=1}^{T} \sum_{P \in \mathcal{P}_s} d_P^t \phi_P^t - \frac{1}{T} \min_{P \in \mathcal{P}_s} \sum_{t=1}^{T} \phi_P^t \tag{3}$$

No-Regret algorithms are those for which the per-time-step regret may be upper bounded by zero as T goes to infinity. The authors of [11] proved that if all commodities applied no-regret algorithms, the resulting demand vector will converge towards the WE. Let us formally present this result, for which we will first define an ϵ-Wardrop Equilibrium (ϵ-WE) [11][2]:

Definition 2. A demand vector d is a ϵ-WE if its total average path cost is within ϵ of the weighted mean cost of the minimum cost path available to each commodity. That is to say:

$$\sum_{s=1}^{S} \sum_{P \in \mathcal{P}_s} d_P \phi_P - \sum_{s=1}^{S} d_s \min_{P \in \mathcal{P}_s} \phi_P \leq \epsilon \sum_{s=1}^{S} d_s \tag{4}$$

[2] Although [11] refers to ϵ-Nash Equilibria (ϵ-NE), in this context, with infinite agents, ϵ-NE and ϵ-WE are equivalent.

Note that, since $\phi_P \geq 0$, a 0-WE is simply a normal WE. Let us define as T_ϵ the number of time steps required to bound (3) by ϵ. We may now formally present the convergence result as obtained in [11]:

Theorem 1. *For a link-cost function $\phi_l(\rho_l)$ with maximum derivative equal to δ, for all but a ϵ' fraction of steps up to time T_ϵ, d^t is a ϵ'-WE, where $\epsilon = \Omega\left(\frac{\epsilon'^4}{\delta|E|^4+\delta^2|V|^2}\right)$. Moreover, T_ϵ depends quadratically on δ.*

In the theorem above, $|E|$ and $|V|$ are the number of edges and vertices in the network respectively, and $\Omega(\cdot)$ indicates the big omega notation. Intuitively, this theorem means that for most time steps up to T_ϵ (i.e. choosing a small ϵ') the instantaneous demand vector d^t is very near the WE (i.e. with a total average path cost within ϵ' of the weighted mean cost of the minimum cost path available to each commodity, cf. Eq. (4)). Moreover, T_ϵ is not arbitrarily large since ϵ is bounded from below. This means that the difference between d^t and the WE vanishes with time. However, it should be noted that the bigger the maximum derivative of $\phi_l(\rho_l)$, the slower the convergence speed.

3.2. A No-Regret Algorithm: iAWM

The result presented in the previous subsection is very general, in the sense that it does not specify any algorithm in particular. Its only requirement is the use of no-regret algorithms by all OD pairs. In particular, we will consider the *Incrementally Adaptive Weighted Majority* (iAWM) algorithm [12], which originated in the context of online learning [13]. The algorithm does not only obtain a very good upper bound on the regret, but also presents the desirable characteristic of not requiring any parameter tuning. The complete pseudo-code for commodity s is described in Fig. 1.

Each path has a corresponding regret $L^t_{P_{si}}$ that measures its performance up to time-step t. Depending on the minimum regret among all n_s paths (L^{t-1}_s), λ^t_s is calculated. This parameter, called the *learning rate*, controls the speed of the algorithm: the bigger the learning rate, the more reactive the demand vector adjustment. In this sense, we shall note as $\omega^t_{P_{si}}$ the portion of traffic sent through path P_{si} at time-step t. The interesting aspect of iAWM is that, as mentioned above, λ^t_s is automatically tuned. As the regret of the best path increases, the learning rate decreases accordingly. Regarding the formulae used to calculate λ^t_s and $\omega^t_{P_{si}}$, although they may appear arbitrary at first, they are a dynamic version of the classic Weighted Majority Algorithm [14]. The interested reader may consult reference [13] for a very complete overview of this topic.

Note the presence of the input y^t_s, called *outcome*, that is used to update the regret of the paths. Let us define $\widehat{y}^t_s = \sum_{i=1}^{n_s} \omega^t_{P_{si}} \phi^t_{P_{si}}$. It may be proved that for any arbitrary sequence of y^t_s of length T, the following inequality

---

$L^0_{P_{si}} \leftarrow 0 \ \forall i = 1, \ldots, n_s$
for $t = 1, \ldots, \infty$ **do**
  $L^{t-1}_s \leftarrow \min_{i=1,..,n_s} L^{t-1}_{P_{si}}$
  $\varepsilon^t_s \leftarrow \min\left\{\frac{1}{4}, \sqrt{2\frac{\log n_s}{L^{t-1}_s}}\right\}$
  $\lambda^t_s \leftarrow \frac{1}{1-\varepsilon^t_s}$
  $W^t_s \leftarrow \sum_{i=1}^{n_s} (\lambda^t_s)^{-L^{t-1}_{P_{si}}}$
  $\omega^t_{P_{si}} \leftarrow (\lambda^t_s)^{-L^{t-1}_{P_{si}}} / W^t_s \ \forall i$
  Receive the path cost $\phi^t_{P_{si}} \in [0,1] \ \forall i$
  Adjust the demand vector: $d^t_{P_{si}}/d_s = \omega^t_{P_{si}} \ \forall i$
  Receive the outcome $y^t_s \in [0,1]$
  Update the path regret $L^t_{P_{si}} \leftarrow L^{t-1}_{P_{si}} + |y^t_s - \phi^t_{P_{si}}| \ \forall i$
end for

Figure 1: Incrementally Adaptive Weighted Majority (iAWM) Algorithm

is verified [12]:

$$\sum_{t=1}^{T} \left|\widehat{y}^t_s - y^t_s\right| - L^T_s \leq (2.83 + o(1))\sqrt{L^T_s \log n_s} \qquad (5)$$

The outcome y^t_s is a value that we consider as the objective. In our case, any value smaller than or equal to all path costs will serve our purposes. Finally, note that the path costs should be in the interval $[0,1]$. We may easily address this issue by using the alternative path cost $\widehat{\phi}^t_{P_{si}} = \phi^t_{P_{si}} / \max_{P \in \mathcal{P}_s} \phi^t_P$. Note that a routing congestion game that uses $\widehat{\phi}^t_{P_{si}}$ instead of the original $\phi^t_{P_{si}}$ still has the same WE. We will then prove the no-regret property in this modified game, which will mean convergence to the WE on the original one.

Theorem 2. *The algorithm iAWM using $y^t_s = \min_{i=1,..,N} \widehat{\phi}^t_{P_{si}}$ is no-regret in the context of a congestion routing game.*

Proof. We need to prove that (3) is bounded, and that this bound goes to zero with T. In this case, it may be written as:

$$\frac{1}{T}\left(\sum_{t=1}^{T}\sum_{i=1}^{n_s} \frac{d^t_{P_{si}}}{d_s} \widehat{\phi}^t_{P_{si}} - \min_{i=1,..,n_s} \sum_{t=1}^{T} \widehat{\phi}^t_{P_{si}}\right) =$$

$$\frac{1}{T}\left(\sum_{t=1}^{T}\sum_{i=1}^{n_s} \omega^t_{P_{si}} \widehat{\phi}^t_{P_{si}} - \sum_{t=1}^{T}(y^t_s - y^t_s) - \min_{i=1,..,n_s} \sum_{t=1}^{T} \widehat{\phi}^t_{P_{si}}\right) =$$

$$\frac{1}{T}\left(\sum_{t=1}^{T}\left(\widehat{y}^t_s - y^t_s\right) - \min_{i=1,..,n_s} \sum_{t=1}^{T}\left(\widehat{\phi}^t_{P_{si}} - y^t_s\right)\right) =$$

$$\frac{1}{T}\left(\sum_{t=1}^{T}\left|\widehat{y}^t_s - y^t_s\right| - \min_{i=1,..,n_s} \sum_{t=1}^{T}\left|\widehat{\phi}^t_{P_{si}} - y^t_s\right|\right) =$$

$$\frac{1}{T}\left(\sum_{t=1}^{T}\left|\widehat{y}^t_s - y^t_s\right| - L^T_s\right) \leq \frac{1}{T}(2.83 + o(1))\sqrt{L^T_s \log n_s} \leq$$

$$\frac{1}{T}(2.83 + o(1))\sqrt{T \log n_s} \underset{T}{\to} 0$$

Where in the last step we have used the fact that both y_s^t and $\widehat{\phi}_{P_{si}}^t$ belong to $[0, 1]$, and thus their accumulated absolute difference is never more than T (i.e. $L_s^T \leq T$). As mentioned above, it is important that $y_s^t \leq \widehat{\phi}_{P_{si}}^t \, \forall i = 1, .., n_s$. This condition allows us to take the absolute value in the fourth step. Using $y_s^t = \min_{i=1,..,n_s} \widehat{\phi}_{P_{si}}^t$ means that when the algorithm converges the regret of the best path does not increase. This fact will prove useful, as we will discuss later, when the algorithm is used with real, time-changing demands.

4. Learning the Delay Function

So far, we have shown that commodities that apply iAWM in a congestion game with a link cost $\phi_l(\rho_l)$ equal to the derivative of the mean queue size $f_l(\rho_l)$ will converge to the minimum total mean delay demand vector. We now address the problem of obtaining a good estimation of $\phi_l(\rho_l) = f_l'(\rho_l)$ from previous measurements on queue size and load. Actually, since it is the observable quantity, we shall first estimate $f_l(\rho_l)$ and then simply derivate this estimation. For the sake of clarity, and since the procedure is the same for every link, in this section we shall omit the subindex l.

Assume we have a set of N measurements $\{(\rho_1, Y_1) \ldots (\rho_N, Y_N)\}$ (also called *training set*), and assume that the response variable Y (the measured mean queue size) is related to the covariate ρ (the mean load) by the following equation:

$$Y_i = f(\rho_i) + \epsilon_i \qquad i = 1, \ldots, n \tag{6}$$

The measurement error ϵ_i is a random variable such that $\mathbb{E}\{\epsilon_i\} = 0$ and $\mathrm{Var}\{\epsilon_i\} = \sigma_i < \infty$. The *Weighted Least Squares* (WLS) problem consists in finding the function $\widehat{f}(\rho)$ that minimizes the weighted sum of quadratic errors, assuming that $\widehat{f}(\rho)$ belongs to a given family of functions $\mathcal{F}$:

$$\min_f \sum_{i=1}^{N} w_i \left(Y_i - f(\rho_i)\right)^2 \quad \text{s.t. } f \in \mathcal{F} \tag{7}$$

where the weight $w_i \geq 0$ represents the relative importance of measurement point i with respect to the rest of the measurements in the training set.

The following subsections present two different methods to solve (7), and differ mainly on the assumed $\mathcal{F}$. We will start with the most general case, and based on the results obtained by it, derive the other method. How to assign the weights w_i will be discussed after that. We shall finish the section by presenting an example.

4.1. Convex Nonparametric Weighted Least Squares

In this subsection we present a method that keeps the assumptions on $\widehat{f}(\rho)$ to the minimum. Regarding its shape, we have only two necessary assumptions. Firstly, $\widehat{f}(\rho)$ should be non-decreasing, since more load may never mean less queue size. Secondly, $\widehat{f}(\rho)$ should be convex in order to guarantee the existence and uniqueness of the optimum demand vector, and that this optimum is also the WE.

We then consider $\mathcal{F}$ as the family of continuous, monotonous increasing and convex functions. We shall call Problem (7) with such $\mathcal{F}$ *Convex Nonparametric Weighted Least Squares* (CNWLS), a variation of the original unweighted Convex Nonparametric Least Squares (CNLS) [15]. The size of $\mathcal{F}$ makes this problem very difficult to solve in such general form. Consider instead the following alternative family of piecewise linear functions $\mathcal{G}_1$:

$$\mathcal{G}_1(P) = \Big\{ g : \mathbb{R} \to \mathbb{R} \mid g(\rho) = \max_{i=1,\ldots,N} \alpha_i + \beta_i \rho; \\ \beta_i \geq 0 \;\; \forall i = 1, \ldots, N; \\ \alpha_i + \beta_i \rho_i \geq \alpha_j + \beta_j \rho_i \;\; \forall j, i = 1, \ldots, N \Big\}$$

It is clear that $\mathcal{G}_1(P) \subseteq \mathcal{F}$ for all $P = \{\rho_i\}_{i=1,..,N}$. Moreover, consider the following theorem:

Theorem 3. *Let s_f^2 be the minimum of problem (7). Let s_{g1}^2 be also the minimum of problem (7), except that we substitute the constraint by $f \in \mathcal{G}_1(P)$. Then $s_f^2 = s_{g1}^2$.*

Proof. The proof is exactly the same as in the original CNLS [15].

This result allows us to transform the infinite dimensional problem (7) into the following standard finite dimensional Quadratic Programming (QP) problem:

$$\min_{\epsilon,\alpha,\beta} \sum_{i=1}^{N} w_i \epsilon_i^2 \tag{8}$$
$$\text{subject to} \quad Y_i = \alpha_i + \beta_i \rho_i + \epsilon_i \; \forall i = 1, \ldots, N$$
$$\alpha_i + \beta_i \rho_i \geq \alpha_j + \beta_j \rho_i \; \forall j, i = 1, \ldots, N$$
$$\beta_i \geq 0 \; \forall i = 1, \ldots, N$$

Although each observation has its own associated pair (α_i, β_i), as we shall illustrate later and already presented in the original CNLS, the actual number of significantly different values (which we shall note N^*) generally results in a small fraction of N. However, note that there are a total of $3N$ variables and $2N + N(N-1)$ constraints. In particular, the second set of constraints, which are the key to enforce the convexity of $\widehat{f}(\rho)$, is quadratic in the number of observations, which will represent a problem as their number increases.

Starting with an initial partition $P_1^{(0)} \dots P_k^{(0)}$
for $l = 0, \dots, l_{max}$ **do**
  $(\alpha_j, \beta_j)^{(l+1)} \leftarrow \underset{\beta_j \geq 0;\, \alpha_j}{\operatorname{argmin}} \sum_{i \in P_j^{(l)}} w_i \left(Y_i - \alpha_j - \rho_i \beta_j\right)^2$
  $P_j^{(l+1)}$ contains i if $j = \underset{s=1\dots k}{\operatorname{argmax}} \{\alpha_s^{(l+1)} + \rho_i \beta_s^{(l+1)}\}$
  Finish if $P_j^{(l+1)}$ is equal to $P_j^{(l)}$ $\forall j = 1 \dots k$
end for

Figure 2: Convex Piecewise-Linear Fitting (CPLF) Heuristic

Starting with the initial weights $w_i^{(0)} = 1$ $\forall i = 1..N$
for $l = 0, \dots, l_{max}$ **do**
  Solve (7) with $w_i^{(l)}$ to compute $(\alpha_j, \beta_j)^{(l+1)}$, resulting in the regression function $\widehat{f}^{(l+1)}(\rho)$
  Update $w_i^{(l+1)}$ as the inverse of $\left|\widehat{f}^{(l+1)}(\rho_i) - Y_i\right|$
  Finish if convergence is reached
end for

Figure 3: Iteratively Reweighted Least Squares Algorithm

4.2. Convex Piecewise-Linear Fitting

We have seen that problem (7) with the biggest possible family $\mathcal{F}$ may be solved by means of a QP problem. The resulting solution $\widehat{f}(\rho)$ is a piecewise function, where the partition of the linear segments is not fixed a priori; i.e. the number and location of the segments are endogenously determined to minimize the weighted squared residual. However, the resulting QP problem presents scalability issues as the number of observations increases. In this subsection we will try to solve this issue by fixing the number of segments to an arbitrary k, thus considering the following family of functions:

$$\mathcal{G}_2 = \left\{ g : \mathbb{R} \to \mathbb{R} \mid g(\rho) = \max_{j=1,\dots,k} \alpha_j + \beta_j \rho; \; \beta_j \geq 0 \;\; \forall j = 1, \dots, k \right\}$$

Substituting $\mathcal{F}$ by $\mathcal{G}_2$ in (7) results in the problem known as *Convex Piecewise-Linear Fitting* (CPLF). If k is bigger or equal than N^* (cf. previous subsection), then the optimum for CPLF and CNWLS will be the same. Considering that N^* is generally a small fraction of N, the possibility of solving the former instead of the latter seems interesting. Unfortunately, CPLF is not globally convex, meaning that, contrary to CNWLS, an exact solution cannot be found. The authors of [16] present an heuristic that approximately solves the unweighted version of the resulting problem, which may be easily adapted to solve the weighted one.

The algorithm is relatively simple, and it alternates between partitioning the measurements and performing a constrained ($\beta_j \geq 0$) linear WLS fitting to update the (α_j, β_j) pairs. Let $P_j^{(l)}$ for $j = 1 \dots k$ be a partition of the measurements indices at iteration l, so that $\bigcup P_j^{(l)} = \{1 \dots N\}$ and $P_{j_1}^{(l)} \bigcap P_{j_2}^{(l)} = \emptyset$. The heuristic is described in Fig. 2.

During the iterations, certain partitions may be emptied. In such case, their (α_j, β_j) should be eliminated. This means that the input k actually indicates the *maximum* number of pairs in the solution. Regarding complexity, the core of the iteration, solving a constrained linear WLS, is a much simpler problem than CNWLS. It is still a QP problem, but the number of variables and restrictions are now $3k$ and k respectively. Finally, trying different random initial partitions $P_1^{(0)} \dots P_k^{(0)}$ and keeping the best solution generally results in a relatively precise estimation.

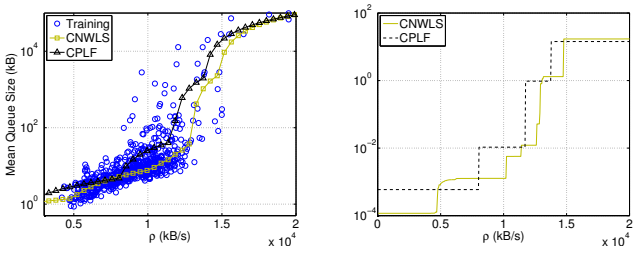
4.3. Choosing the Weights

In this subsection we discuss a possible way of choosing the values of w_i in (7). Since they are the result of a process whose characteristics may change over time, measurements present heteroscedasticity and important outliers. We will then set the weights so that (7) results in the Least Absolute Deviations problem (i.e. minimize the sum of the absolute, instead of the squared, errors), which is known to be more robust to these problems. The classic way of calculating such weights is to use the *Iteratively Reweighted Least Squares* [20] described in Fig. 3.

There are several possible criteria to decide when convergence is reached in the above algorithm. In our particular case, we stop if the ratio in absolute mean error between two consecutive iterations is more than 0.95. To avoid numerical problems, as suggested in [20], if at any given iteration an error is less than 10^{-5} the corresponding weight should be set to 10^5. Anyway, note that the algorithm requires a solution of (7) at each iteration. This does not represent a problem for CPLF, although, as we mentioned before, the time required by CNWLS may be considerable, and applying it several time may result in a prohibitive amount of time. In this case then, we shall proceed as follows. We perform an initial simple estimation $\widehat{f}^{(0)}(\rho_i)$, and calculate the corresponding weight as:

$$w_i = \frac{1}{\left|\widehat{f}^{(0)}(\rho_i) - Y_i\right|} \tag{9}$$

As the initial $\widehat{f}^{(0)}(\rho_i)$ we used the *m-nearest neighbors* algorithm (with $m = 10$), which simply estimates $f(\rho)$ as the median of the m measurements Y_i corresponding to the ρ_i's nearest to ρ. In this way, we seek to find a curve that fits the bulk of the data, and minimize the effect of outliers. Moreover, convergence is not guaranteed for the reweighting algorithm. We deal with this situation (which may be detected by checking if the total absolute error has



(a) The estimation of the mean queue size function (b) The estimation of the derivative

Figure 4: An example of a regression for the two methods

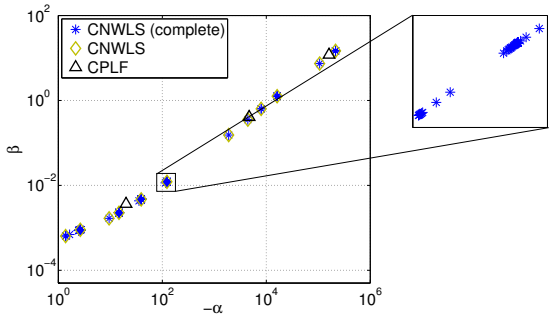


Figure 5: The (α_j, β_j) pairs in the plane

increased from one iteration to the next) by falling back to these default weights.

4.4. A Regression Example

We finish this section by presenting a typical regression. To obtain the measurements we injected a 72 hour long packet trace (obtained from [21]) to a simple queue emulator we developed. In the absence of information we assumed a relatively big buffer size of 100 MB. For each measurement, we took the mean load and queue size in a 60 second period.

The resulting regression function for the two methods using $N = 720$ observations may be seen in Fig. 4(a) (for CPLF we used $k = 10$). We may verify that CNWLS obtains an approximative function that represents more precisely the measurements than CPLF. In particular, on ρ between 0.8 MB/s and 12.5 MB/s we may appreciate that it follows more closely the bulk of data. The same happens at the lower values of ρ. Although not included due to space limitations, we have conducted a study that confirms that CNWLS is more precise than CPLF. In fact, after $k \approx 5$ the precision of CPLF does not increase, and is approximately half that obtained by CNWLS. However, as we shall see in the next section, the difference between them in terms of delay is not as significant. In terms of computation time, for a training set size bigger than $N \approx 500$ CNWLS is significantly slower than CPLF. The interested reader may consult the complete study at [22].

Figure 4(b) shows the estimation of the derivative $\widehat{\phi}(\rho)$ for the two methods. It should be noted that the resulting estimation for both methods becomes constant after, at the latest, no more observations are available. This aspect, which is due to the shape of the regression function, could be problematic and shall be further discussed later. Moreover, in Fig. 5 we show the pairs (α_j, β_j) in the plane. Regarding CNWLS, we verify that the number of significantly different pairs is a small fraction of N (in this case $N^* = 14$). A very simple clustering algorithm may be used to estimate the final set of pairs. Concerning CPLF, the final number of pairs was reduced from 10 to 3.

Another important aspect is that, since $\widehat{f}(\rho)$ is estimated as a piecewise linear function, the estimated $\widehat{\phi}(\rho)$ is piecewise constant. As we mentioned in Sec. 3.1, the convergence time of the no-regret algorithms depends on the maximum derivative of $\phi(\rho)$, which for our approximation is unbounded[3]. To address this issue we took a very simple approach, and approximate this piecewise constant function by a continuous piecewise linear one. The idea is the following. The piecewise constant function $\widehat{\phi}(\rho)$ may be characterized by the intervals in which its value is constant, and the corresponding value. If we have N^* pairs, we will have N^* of these intervals. We will assume that the first of these intervals starts at 0 and the last ends at the last available observation in the training set. Let us note the center of these intervals as ρ_j, where we will assume that $\rho_{j_1} < \rho_{j_2}\ \forall j_1 < j_2$. If we have to estimate the cost at a load ρ between ρ_j and ρ_{j+1}, then our continuous estimation will be:

$$\widehat{\phi}^*(\rho) = \frac{\rho_{j+1} - \rho}{\rho_{j+1} - \rho_j}\widehat{\phi}(\rho_j) + \frac{\rho - \rho_j}{\rho_{j+1} - \rho_j}\widehat{\phi}(\rho_{j+1}) \qquad (10)$$

If we have to estimate the cost for $\rho < \rho_1$, we will then assume that there is a $\rho_0 = 0$ and that its cost is $\widehat{\phi}(\rho_0) = 0$. If we are on the other end and have to estimate the cost for $\rho > \rho_{N^*}$, we will simply return $\widehat{\phi}(\rho_{N^*})$. To characterize our continuous approximation we then require only the interval centers $\{\rho_j\}_{j=1,..,N^*}$ and their associated cost $\widehat{\phi}(\rho_j)$. To clarify the explanation, we included in Fig. 6 this continuous approximation.

5. Flow-Level Simulations

In this section we will present several flow-level simulations that will help us gain insight into the framework. We will first analyze the performance of iAWM as it faces unforeseen and abrupt changes in the traffic demand. This will justify the final version of the load-balancing algorithm. Another aspect that we shall analyze is the resulting performance of the framework when compared with other load-balancing mechanisms. We are particularly interested in the gain in terms of total delay that we may

[3]This is not a particular characteristic of no-regret algorithms, since all distributed optimization methods require a finite derivative to converge.

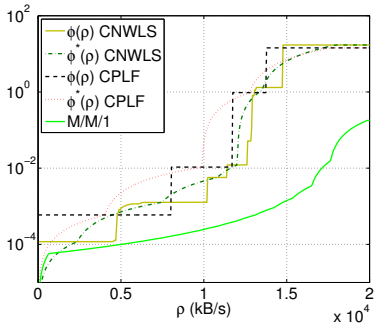
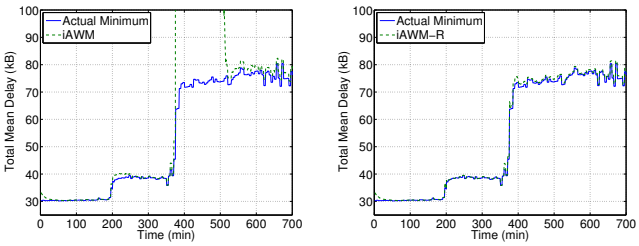


Figure 6: The estimated link cost used on the comparison



(a) iAWM (b) iAWM-R

Figure 7: The total mean delay as a function of time for the two load-balancing mechanisms.

achieve by using our regression over simplistic models such as the *M/M/1*. We will also try to answer the important question of how often we need to update the regression cost function $\widehat{\phi}(\rho)$.

As the reference topology we used Abilene [23]. This network consists of 12 nodes and 15 bidirectional links all with the same capacity. The topology comes as an example in the TOTEM toolbox [24] and we used the traffic demands obtained from [25], which consists of 6 months of traffic matrices collected every 5 minutes via Netflow. Paths were constructed as discussed in [26]. We will assume the same $\widehat{f}_l(\rho_l)$ for all links in the network, namely the one obtained in Sec. 4.4 and shown in Fig. 6.

5.1. Reactivity of iAWM

In this subsection we shall analyze the temporal performance of iAWM. We are particularly interested in its behavior as it faces abrupt changes in the traffic demand. We will then consider 200 TMs from dataset X06 in [25], whose main characteristic is the anomalous increase of the traffic demand for two commodities. The simulation will be performed as follows. The TMs are fed to the mechanism in consecutive temporal order. The demand vector d is initiated at an arbitrary value d^0, which will be updated using iAWM (cf. Fig. 1) as new link load measurements arrive. We will assume that each OD pair receives these measurements every minute, meaning that for each new TM the OD pairs will perform five updates of the portions of traffic sent through each of its paths. We will then calculate the total mean delay ($D(d)$) corresponding to each of these minutes. As a reference, we also computed the optimum value of $D(d)$ for every TM on the dataset.

In Fig. 7(a) we show the results corresponding to this simulation. Note how iAWM rapidly converges to the optimum, and even when at $t = 200$ the first anomaly starts, there is only a small difference between the obtained delay and the optimum. However, the behavior is quite the opposite for the second anomaly, which starts at $t \approx 400$. In this case, the overshoot is very important, and the convergence time is more than 100 minutes (or iterations in our case).

The reason behind this slow convergence is relatively simple. At the moment of the anomaly, for the commodity in question, the path through which some of the traffic is routed at optimality has a very bad history (i.e. a very big $L^t_{P_{si}}$). This means that for iAWM, this path has to be the cheapest for several iterations to revert its bad image and start being able to route traffic. When an anomaly occurs, conditions severely change and history should not be so relevant. If we consider that we are in such a situation, we could for instance completely ignore history and restart iAWM by setting $L^t_{P_{si}} = 0 \ \forall i$ for the commodity in question.

Regarding the decision of restarting iAWM, one may imagine several possibilities. In particular, we will consider that if $|y^t_s - \widehat{y}^t_s|$ is abnormally big, it constitutes a "suspicious" situation. By abnormally big, we mean that this difference exceeds the mean in previous iterations by a certain threshold $\widehat{L}^{th}_s$; in particular, we used $\widehat{L}^{th}_s = 0.1 \, \forall s$ (remember that y^t_s and $\widehat{y}^t_s$ are in the interval $[0, 1]$). Such a situation should indicate to us that the current situation does not correspond to what we have seen so far. However, it could also be due to noisy measurements. We will then require that this "suspicious" situation is repeated a certain amount m^{th}_s of consecutive times (in our case we used $m^{th}_s = 5 \ \forall s$). The complete pseudo-code is described in Fig. 8. The results obtained with this new algorithm may be seen in Fig. 7(b). Note how the overshoot has now disappeared almost completely.

Adapting on-line learning algorithms to non-stationary environments, as we have just discussed for iAWM, is an active research area. The alternative to restarting, is the more "continuous" framework of *tracking the best expert* [27]. The objective is the same as in the no-regret framework, except that we consider that time is divided into segments, and that each of them has a best path. Performance of the algorithm is then compared to the performance obtained by this sequence of paths. Although it would be interesting to adapt such algorithms to be used for load-balancing, several aspects remain to be addressed. Firstly, convergence to the Wardrop Equilibrium has not been proved. Secondly, these algorithms are not self-configured which, as we discussed, is a very important property.

Initialize $\tau_s^0 \leftarrow 0$, $\widehat{L}_s^0 \leftarrow 0$ and $m_s^0 \leftarrow 0$
for $t = 1, \ldots, \infty$ **do**
  Perform a normal iteration of iAWM.
  $\tau_s^t \leftarrow \tau_s^{t-1} + 1$
  $\widehat{L}_s^t \leftarrow \widehat{L}_s^{t-1} + |y_s^t - \widehat{y}_s^t|$
  if $|y_s^t - \widehat{y}_s^t| > \widehat{L}_s^t/\tau_s^t + \widehat{L}_s^{th}$ **then**
    $m_s^t \leftarrow m_s^{t-1} + 1$
  else
    $m_s^t \leftarrow 0$
  end if
  if $m_s^t > m_s^{th}$ **then**
    $L_{P_{si}}^t \leftarrow 0 \ \forall i = 1, .., n_s$
    $\tau_s^t \leftarrow 0$, $\widehat{L}_s^t \leftarrow 0$ and $m_s^t \leftarrow 0$
  end if
end for

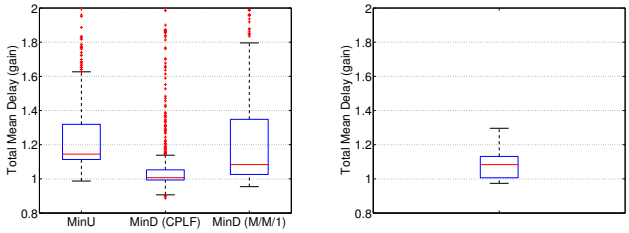
Figure 8: Incrementally Adaptive Weighted Majority with Restart (iAWM-R) Algorithm

5.2. Assessing the Performance Gain

In addition to the $\widehat{\phi}(\rho)$ we estimated through measurements, in Fig. 6 we compare it with the *M/M/1* model. If we assume this model instead of the real $f_l(\rho_l)$ we would incur an increase of the total mean delay ($D(d)$) with respect to the optimum that may be important. To quantify this increase more precisely, we will consider 672 traffic demands spanning a complete week from dataset X01 in [25]. For each of these traffic demands, we apply iAWM-R using the *M/M/1* model and the $\widehat{\phi}_l^*(\rho_l)$ resulting from CNWLS and CPLF, and measure the difference in $D(d)$ assuming the $\widehat{f}_l(\rho_l)$ obtained by CNWLS as the true $f_l(\rho_l)$. We shall note these schemes as MinD (as in Minimum Delay), where the cost function will be indicated between parenthesis (e.g. MinD(CNWLS)).

For the sake of completeness, we will also make the comparison in terms of the link utilization ($u_l = \rho_l/c_l$). A link with a u_l close to one is operating near its capacity, and in order to be able to support sudden increases in traffic and link/node failures, network operators prefer to keep link utilization relatively low. It would not make much sense to minimize the total mean delay if it meant highly utilized links. As a reference for the comparison we will use the results obtained by a greedy load-balancing mechanism whose path cost is the maximum link utilization ($\phi_P = \max_{l \in P}\{u_l\}$), which converges to a demand vector that minimizes the maximum utilization over all links [7, 8]. This scheme shall be noted MinU (as in Minimum Utilization). We will measure two network-wide performance indicators: the 90% quantile and maximum link utilization.

In Fig. 9(a) we can see the boxplot of the results on the total mean delay. In particular, for each traffic demand we calculated $D(d)$ for the considered schemes: MinU and



(a) M/M/1 and MinU with respect to MinD (b) Using the $\widehat{f}_l(\rho_l)$ of the day before

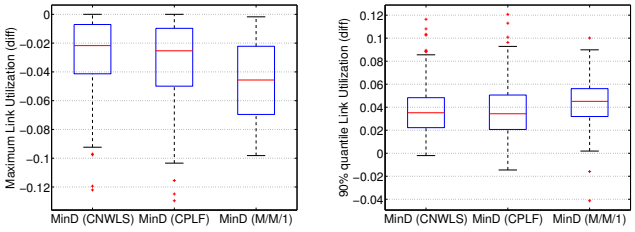
Figure 9: Increase in Total Mean Delay in the Abilene network

MinD for the three possible $\widehat{\phi}^*(\rho)$. We present the division between the value obtained by each scheme and MinD(CNWLS).

We can see that the total mean delay obtained by MinD(M/M/1) is generally between 5 and 35% bigger than the one obtained by MinD(CNWLS). This difference may actually go as high as a 80%, and in some cases even more (although not shown for the sake of clarity of the graph, the actual maximum was 560%). If we look carefully at Fig. 6 we can see that this difference originates in the fact that the *M/M/1* model underestimates $\phi_l(\rho_l)$. In particular, the abrupt increase in queue size that occurs at $\rho \approx 12\,\text{MB/s}$, is also present in the *M/M/1* estimation, but at a much higher load of $\rho \approx 17.5\,\text{MB/s}$. This leads iAWM-R to "believe" that links are operating at a low queueing delay load, when it is actually the opposite.

Regarding MinD(CPLF), we may verify that the loss in precision does not seriously impact the obtained performance. Except for some isolated cases, the increase in total mean delay is between 0% and 5%. On the other hand, the difference in total mean delay obtained by MinU is similar to the one obtained by MinD(M/M/1), generally with an increase between 10 and 30% with respect to MinD(CNWLS), with a maximum of 60%. Although MinU tries to avoid loaded links (thus obtaining slightly better results than MinD(M/M/1)), it is so conservative in its objective that it ends up unnecessarily increasing the total mean delay.

As far as the link utilization is concerned, we calculated the results obtained by all the mechanisms, and present the difference between the reference (MinU) and the other schemes, which we show in Fig. 10. It should be noted that the results for all versions of MinD are very similar, in agreement with what we mentioned in Sec. 1. Quite surprisingly, the 90% quantile is bigger in MinU. The argument remains the same as before. MinU is so conservative in its objective, that, although it minimizes the maximum link utilization (where the difference with MinD is always less than 10%), it overlooks the less loaded links. These results confirm that minimizing $D(d)$ is a good objective, since it does not neglect links utilization. On the contrary, although it obtains a somewhat bigger maximum utilization than MinU, the rest of the links are more lightly



(a) Maximum Link Utilization (b) 90% quantile of the Link Utilization

Figure 10: Difference in link utilization between MinU and MinD in the Abilene network

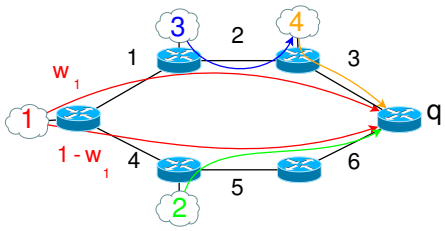


Figure 11: The network for the packet-level simulations.

loaded.

5.3. Temporal Behavior

A natural question that arises in our framework is how often links need to be characterized. In other words, how long can $\widehat{\phi}_l^*(\rho_l)$ be used as a good approximation of $\phi_l(\rho_l)$? Although more frequent updates of the links characterization will mean a more optimal or fine-tuned network, it will also mean greater computational expenses. This tradeoff between the optimality of the network and computational burden should be addressed.

Here we will give a partial answer to this question, and, as a reference, provide a lower bound to the validity of the link characterization used in the previous subsection (which we will note as $(\alpha_i, \beta_i)_{\text{PREV}}$). The idea is the following. From the same 72 hours long packet trace used before, we take the same 12 hours worth of measurements, but from the next day. We will note the characterization resulting from this new training set as $(\alpha_i, \beta_i)_{\text{NEXT}}$. We now assume that the correct $f_l(\rho_l)$ for all links in Abilene is $\widehat{f}_l(\rho_l)_{\text{NEXT}}$, and measure the increase in the total mean delay if we were to apply the iAWM-R algorithm using $\widehat{\phi}_l^*(\rho_l)_{\text{PREV}}$ instead.

In Fig. 9(b) we show the results obtained in this case. We can see that although in some few cases the increase due to the misspecification may be as much as 30%, it is generally under 13%. These results are to be compared to those obtained by MinD(M/M/1), which obtained an excess in the total mean delay of more than 10% in half of the cases, and a maximum difference of more than 80%.

Our partial answer is then that the characterization of a link obtained from the measurements of any given day, is also valid the next day. This validates our implicit assumption that $\phi_l(\rho_l)$ remains relatively stable over time. Another positive conclusion is that regression need not be performed very frequently. It should be noted that the trace used in this study contained only working days. Our conjecture is that the characterization obtained from any working day holds for the rest of the working days in the same week. The traffic mix generally changes on weekends, which will probably result in a different $f_l(\rho_l)$ than that of the working days, thus requiring its own characterization.

6. Packet-Level Simulations

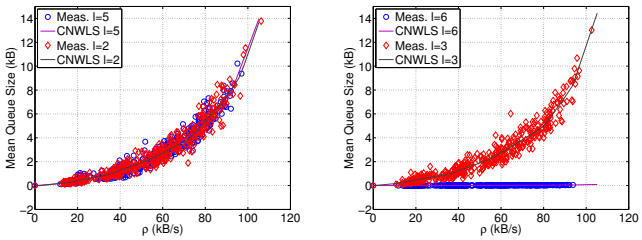
In this section we will consider a relatively simple example we implemented in ns-2 [28] that will verify the correct performance of iAWM-R in the presence of delayed and noisy measurements. It is important to highlight that in all the simulations load balancing is performed at the granularity of flows (i.e. once a flow is routed through a path, it stays there throughout its lifetime) and is random; i.e. commodity s will route new incoming flows through path P_{si} with probability $\omega_{P_{si}}^t$ (cf. Sec. 3.2).

The network may be seen in Fig. 11. Its six "core" links have a capacity of 125 kB/s, while the "access" ones 250 kB/s. There are a total of four commodities, all with the same destination node q, except for commodity 3, whose destination node is origin 4. Moreover, only commodity 1 has more than one path available. We will note ω_1 the portion of traffic of commodity 1 routed through the upper path. The traffic in the network is a mixture of elastic and streaming flows. The elastic ones (whose size is exponentially distributed with mean 20 kB) are generated as a Poisson process of intensity λ_e. The streaming traffic is constituted of Constant Bit Rate flows (at a bitrate of 10 kbps and an exponentially distributed duration with mean 20 s) also arriving as a Poisson process (of intensity λ_s). The corresponding intensities were calculated so that streaming represents 10% of the total traffic.

The training set has been constructed by fixing ω_1 at 0.5 and varying the demand generated by each commodity. The training set and the corresponding regression for links 2, 3, 5 and 6 may be seen in Fig. 12. Note that links 2, 3 and 5 have almost the same characterization. On the contrary, link 6 presents little or no link delay. This is because traffic on this link is already shaped by the queue of link 5.

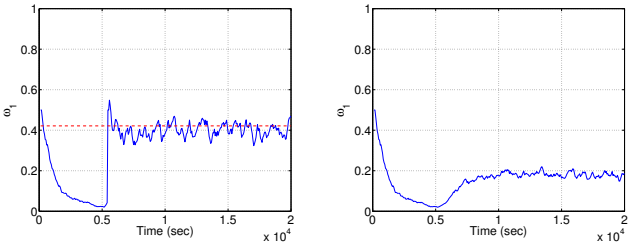
Links with little or no delay clearly represent a problem for our framework. This small delay may be due to the traffic characteristics as in this case, or simply because the link has a small buffer. Such links will have a constant insignificant cost, and the converged demand vector could overload them. This could also be the case for a characterization resulting from an incomplete training set. If the maximum measured load is relatively small, as observed in Sec. 4.4, the resulting cost function $\widehat{\phi}_l^*(\rho_l)$ will become a small constant.

To illustrate the above mentioned problems, we have



(a) Links $l = 2$ and $l = 5$ (b) Links $l = 6$ and $l = 3$

Figure 12: The training sets and the corresponding regressions



(a) A complete link characterization (b) An incomplete characterization of link 5

Figure 13: ω_1 as a function of time for two link characterizations

conducted two simulations. In both of them, the total traffic intensity for commodities 2, 3 and 4 is 50 kB/s throughout the simulation, while for commodity 1 it changes from 12.5 kB/s to 75 kB/s at $t = 5000$ s of the simulation (in a total of 20000 seconds). The difference between the two simulations lies in the characterization used for link 5. In the first simulation, we used the complete characterization, as it appears in Fig. 12(a), while in the second one we have removed the (α_j, β_j) corresponding to the biggest β_j. This means that after $\rho \approx 75$ kB/s the link cost function becomes constant. In Fig. 13 we show the evolution of ω_1 over time for the two simulations. Notice how the ω_1 corresponding to the complete link characterization converges to a value very near the optimum we calculated offline (which is marked by a horizontal line). On the other hand, the incomplete link characterization makes iAWM-R believe that link 5 is cheaper than it really is, resulting in ω_1 converging to 0.2. In the first case, it is interesting to verify how the restart in iAWM-R is only applied when the abrupt change in demand occurs. For the second training set, the difference in cost is not enough to trigger a restart.

In Fig. 14 we present the load for all "core" links on the simulation corresponding to the incomplete link characterization. Naturally, links 5 and 6 are overloaded in the second part of the simulation. It is interesting to notice how the total demand generated by commodity 1 (approximately 120 kB/s from Fig. 14) exceeds the traffic intensity of 75 kB/s. This is because at overload, due to retransmissions, the mean traffic generated by each TCP flow exceeds the 20 kB. This means that, although the implicit assumption that demands do not depend on network

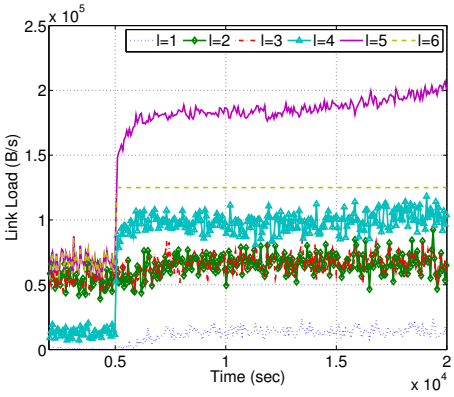


Figure 14: The links load as a function of time for the incomplete link characterization

condition or routing is not fulfilled in this simulation, the load-balancing algorithm still converges. Another interesting aspect to note is that although the level of noise in the load measurements is important, its impact on the converge of ω_1 is very small.

We will now present a possible solution to the above problems. The idea is to avoid overloaded links due to unobserved link loads or little mean queue size. We propose to substitute $\widehat{\phi}_l^*(\rho_l)$ by a certain function $\psi_l(\rho_l)$ for $\rho_l > \rho_l^{\max}$, where $\psi(\rho_l)$ has the following characteristics:

- So that the resulting cost function is still continuous, $\psi_l(\rho_l^{\max}) = \widehat{\phi}_l^*(\rho_l^{\max})$.
- $\psi(\rho_l)$ should be increasing on ρ_l.
- As load exceeds $\rho_l^{\max}$, all links should have a big and similar cost. Else, the algorithm would still prefer to overload the cheapest links.

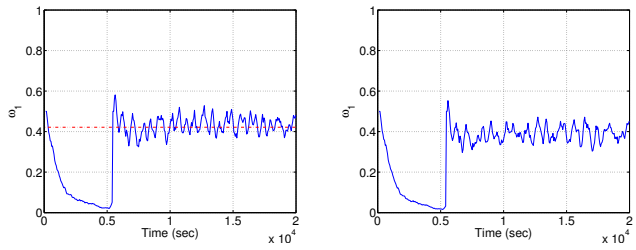
Let us temporarily define $\psi_l(\rho_l)$ as:

$$\psi_l(\rho_l) = e^{b_l \rho_l / c_l} - 1 \qquad (11)$$
$$\text{with } b_l = \frac{c_l}{\rho_l^{\max}} \log\left(1 + \widehat{\phi}_l^*(\rho_l^{\max})\right)$$

We suggest an exponential because, in our experience, it is a rough approximation of the shape of the $\widehat{\phi}_l^*(\rho_l)$ obtained from measurements. Function (11) goes through the origin and at $\rho_l^{\max}$ is exactly $\widehat{\phi}_l^*(\rho_l^{\max})$. For each link, $\psi_l(\rho_l)$ then fulfills the two first characteristics of the above list. To fulfill the third one, we will take the maximum b_l among all links ($b_{\max}$), and use the resulting function as the definitive substitute cost $\psi_l(\rho_l)$. The final link cost is then:

$$\phi_l(\rho_l) = \begin{cases} \widehat{\phi}_l^*(\rho_l) & \text{if } \rho_l < \rho_l^{\max}, \\ e^{b_{\max}\rho_l/c_l} - e^{b_{\max}\rho_l^{\max}/c_l} + \widehat{\phi}_l^*(\rho_l^{\max}) & \text{else} \end{cases} \qquad (12)$$

As the threshold $\rho_l^{\max}$ we could use the biggest ρ_l in the training set, a certain fraction of c_l, or the minimum of both. There are many possibilities. In particular, in



(a) A complete link characterization (b) An incomplete characterization of link 5

Figure 15: ω_1 as a function of time for two link characterizations using the corrective function

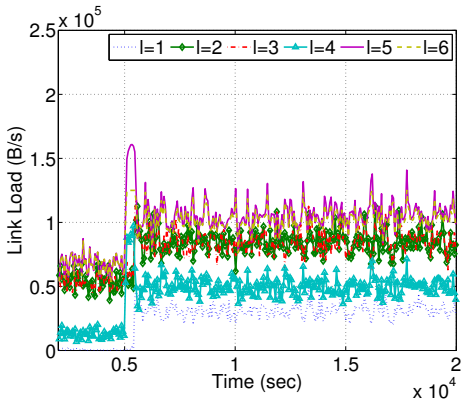


Figure 16: The links load as a function of time for the incomplete link characterization using the corrective function $\psi_l(\rho_l)$

our simulations we used the last option, with a fraction of 80%. It could happen that the link capacity was not known, in which case we should then substitute it by a load that we consider critical for the operation of the link. The exact value of c_l in (12) is not so important and an order-of-magnitude value should be enough. Finally, note that as long as the load at all links is smaller than the corresponding $\rho_l^{\max}$, the presence of $\psi_l(\rho_l)$ has no influence.

We will now repeat the two previous simulations, but using the corrected link cost function (12). In Fig. 15(a) we may see that the influence of $\psi_l(\rho_l)$ when using the complete characterization is negligible. The differences with Fig. 13(a) are due to a relatively incomplete training set on links 1 and 4. In Fig. 15(b) we may see how the presence of $\psi_l(\rho_l)$ when using the incomplete link characterization plays a fundamental role in the convergence of ω_1. In Fig. 16 we may verify that no link is overloaded now.

6.1. Implementation Issues

The application of our framework in a real-world network is relatively simple. Once all links have been characterized, each OD pair receives ρ_l from the links it uses (for this purpose, a Traffic Engineering enabled routing protocol such as OSPF-TE [29] may be used), calculates its paths' cost with (12), and applies iAWM-R to update the portion of traffic routed through each of them (i.e. one iteration in Fig. 8). This process is repeated regularly every few seconds. This update period should be long enough so that the quality of the measurements obtained is reasonable, but not too long to avoid unresponsiveness (in particular, we suggest 60 sec).

Regarding the learning phase (i.e. gathering the training set and performing the regression) we envisage several possibilities, differing in the degree of distribution of the resulting architecture. One possibility is that a central entity gathers the measurements, performs the regression and communicates the parameters obtained to all ingress routers (we assume that these routers, through which commodities inject traffic to the network, distribute this traffic). This first possibility presents the advantage that the new functionalities required on the router are minimal. However, as all centralized schemes, it may not be possible to implement in some network scenarios, and handling the failure of this central entity could be very complicated. An alternative is that links (or rather, the router at the origin of the link) perform the regression. Links keep the mean queue size measurements for themselves, perform the regression and communicate the result to ingress routers. The regression could be done once a day, in periods of low intensity (i.e. the night) so that normal operation is not affected. Recall that, as discussed in Sec. 5.3, frequent updates in the regression function are not necessary.

An important aspect that should be analyzed is which measurements to keep for the training set. It is clear that newer measurements should be given priority over older ones. However, those corresponding to bigger loads will give us more information on $\phi_l(\rho_l)$ (and are more rare). A possible structure for the measurements database could be to partition the load into intervals, and keep the same amount of measurements per interval, each of which will work as a FIFO (first-in-first-out) queue. As mentioned before, those intervals corresponding to bigger loads should be smaller. How to assign this intervals exactly and how many measurements to use is left for future work.

With respect to the final form of $\widehat{\phi}_l^*(\rho_l)$ (cf. (12)) the main issue is finding the maximum b_l among all links. In the centralized architecture we described above, the problem is straightforward, since the central entity has all the information, and it need only communicate this value along with the links' characterization to all routers. In the distributed scenario it is somewhat more difficult. If routers have information of only some of the links, they cannot calculate $b_{\max}$ and have to communicate with the rest to find it. Fortunately, distributed and efficient algorithms to do this exist [30].

Regarding the choice of the regression method, we have shown in Sec. 5.2 that the difference in the obtained total mean delay between them is generally small, except in some rare cases. However, whether the extra computational burden incurred by CNWLS is worth the performance improvement depends on the given situation. If the size of the training set we are working with is significant, or the computation capacity available limited, we could fall back on CPLF. If not, we recommend using CNWLS.

Not only does it obtain a better precision, but its solution does not rely on a heuristic and may be calculated exactly.

7. Related Work

The problem of minimizing the total mean delay trying to assume as little as possible on $f_l(\rho_l)$ has been studied in the past. For instance, the authors of [31] present a dynamic load-balancing method that, differently to our work, does not assume *any* model for the mean queue size function (i.e. this function is now $f_l(W)$, where the argument is the whole packet input process during the time period). The load-balancing approach is however relatively similar, in the sense that, based on the results discussed in Sec. 2.2, they use a greedy algorithm on the path cost $\phi_P = \sum_{l \in P} \partial f_l / \partial \rho_l$. To estimate this derivative in such general case, they use the so-called Perturbation Analysis, which requires measuring the moment each packet enters and leaves the queue of link l.

The first disadvantage of this method is the measurements it requires. At the time of the proposal (late eighties), the assumption that the moment at which each packet of each link enters and leaves the queue was measurable, was somewhat reasonable. Nowadays, it simply cannot be done. In this sense, recent articles (e.g. [32]) propose alternative and simpler methods to estimate $\sum_{l \in P} \partial f_l / \partial \rho_l$ based on measurements. However, they all share a second disadvantage (which at first may seem like the contrary): the level of generality of their model. The optimization problem they wish to solve assumes a queueing delay function that depends solely on ρ_l (just like ours). However, their estimation of the derivative does not make such supposition, but depends on the whole packet process. This means that f_l depends on many more and unknown variables than just ρ_l. Note that since commodities may only change the portion of traffic sent through each path, these new variables are not controllable by them. This may translate into oscillations, as presented in the original paper [31].

All in all, although our framework does not use the most general model, it does assume a "controllable" one. We can then guarantee convergence and stability, and expect that the mean total delay obtained by the resulting demand vector is a good approximation of the absolute minimum. Moreover, the measurements required by our framework are available in most routers and need not be extremely precise.

8. Conclusions

In this paper we presented a Dynamic Load-Balancing (DLB) scheme that converges to the minimum *mean end-to-end queueing delay* ($D(d) = \sum_l f_l(\rho_l)$, where $f_l(\rho_l)$ is the mean queue size) demand vector. The advantage of our proposal is that we make almost no a priori assumption on the function $f_l(\rho_l)$, but learn its actual form from past measurements of mean load and queue size (both readily available in most routers), thus converging to the real minimum.

Convergence is attained by means of a no-regret algorithm, which had the desirable property of being self-regulated, thus requiring no speed parameter fine-tuning. To learn the mean queue size function we presented two alternative methods: *Weighted Constrained Nonparametric Least Squares* (WCNLS) and *Convex Piecewise Linear Fitting* (CPLF). Although the first one is more precise, as the number of available measurements increase, its computational cost could become a problem. The comparison between our framework and a simplistic model (*M/M/1*) showed that in half of the cases the increase in delay exceeded 10%, and that it could be more than 100%. In terms of link utilization, results indicate a very similar performance between the *M/M/1* model and our approximate $\phi_l(\rho_l)$. They both obtain a somewhat bigger maximum utilization than the optimum, but distribute the load among the rest of the links more evenly.

As discussed in Sec. 1, DLB (and TE in general) is usually defined in terms of a $f_l(\rho_l)$ that is arbitrarily chosen, as long as it is convex and goes to infinity when load reaches the link capacity. In terms of link utilization and TCP performance the difference between the different $f_l(\rho_l)$ may be considered as somewhat unimportant. However, in this paper we have shown that when the objective is minimizing queueing delays (the most important performance indicator for real-time traffic, an increasing amount of traffic nowadays) this choice is crucial, and a misspecification can result in significant increases of total delay with respect to the optimum.

A possible improvement to the framework has to do with the model used when defining $f_l(\rho_l)$. Although, as we saw in Sec. 4.4, the mean queue size can be reasonably modeled with such a function in wired mediums, this is not necessarily true in a wireless medium. Actually, as discussed for instance in [33], the MAC-layer interactions between routers play a significant role in determining the capacity of a link (and thus its queue size). This means that the f of any given link should include the load of all neighbor links in its collision domain, and not only itself. A deeper analysis of this non-local model represents interesting future work.

9. Acknowledgements

This work was partially funded by CELTIC project TRANS and FP7 project Euro-NF

[1] W. Ben-Ameur, H. Kerivin, Routing of uncertain traffic demands, Opt. and Eng. 6 (3) (2005) 283–313.

[2] D. Applegate, E. Cohen, Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs, in: SIGCOMM '03, Karlsruhe, Germany, 2003, pp. 313–324.

[3] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, A. Greenberg, Cope: traffic engineering in dynamic networks, SIGCOMM Comput. Commun. Rev. 36 (4) (2006) 99–110.

[4] S. Balon, F. Skivée, G. Leduc, How well do traffic engineering objective functions meet te requirements?, in: 5th International IFIP-TC6 Networking Conference, Coimbra, Portugal, 2006, pp. 75–86.
[5] F. Larroca, J.-L. Rougier, Routing games for traffic engineering, in: ICC 2009, Dresden, Germany.
[6] A. Elwalid, C. Jin, S. Low, I. Widjaja, MATE: MPLS adaptive traffic engineering, in: INFOCOM 2001, Vol. 3, Anchorage, USA, 2001, pp. 1300–1309.
[7] S. Kandula, D. Katabi, B. Davie, A. Charny, Walking the tightrope: responsive yet stable traffic engineering, in: SIGCOMM '05, Philadelphia, USA, 2005, pp. 253–264.
[8] S. Fischer, N. Kammenhuber, A. Feldmann, Replex: dynamic traffic engineering based on wardrop routing policies, in: CoNEXT '06, Lisboa, Portugal, 2006, pp. 1–12.
[9] J. He, M. Bresler, M. Chiang, J. Rexford, Towards robust multilayer traffic engineering: Optimization of congestion control and routing, IEEE JSAC 25 (5) (June 2007) 868–880.
[10] A. Khanna, J. Zinky, The revised arpanet routing metric, SIGCOMM Comput. Commun. Rev. 19 (4) (1989) 45–56.
[11] A. Blum, E. Even-Dar, K. Ligett, Routing without regret: on convergence to nash equilibria of regret-minimizing algorithms in routing games, in: PODC 2006, Denver, USA, 2006, pp. 45–52.
[12] P. Auer, N. Cesa-Bianchi, C. Gentile, Adaptive and self-confident on-line learning algorithms, Journal of Computer and System Sciences 64 (1) (2002) 48–75.
[13] N. C. Bianchi, G. Lugosi, Prediction, Learning, and Games, Cambridge University Press, 2006.
[14] N. Littlestone, P. M. Long, M. Warmuth, The Weighted Majority Algorithm, Information and Computation 108 (2) (1994), pp. 212–261.
[15] T. Kuosmanen, Representation theorem for convex nonparametric least squares, Econometrics Journal 11 (2) (2008) 308–325.
[16] A. Magnani, S. P. Boyd, Convex piecewise-linear fitting, Optimization and Engineering 10 (1) (2009) 1–17.
[17] E. Altman, T. Boulogne, R. El-Azouzi, T. Jiménez, L. Wynter, A survey on networking games in telecommunications, Comput. Oper. Res. 33 (2) (2006) 286–311.
[18] J. Wardrop, Some theoretical aspects of road traffic research, Proceedings of the Institution of Civil Engineers, Part II 1 (36) (1952) 352–362.
[19] R. Banner, A. Orda, Bottleneck routing games in communication networks, IEEE JSAC 25 (6) (2007) 1173–1179.
[20] R. C. Fair, On the robust estimation of econometric models, Annals of Economic and Social Measurement 3 (4) (1974) 117–128.
[21] K. Cho, WIDE-TRANSIT 150 Megabit Ethernet Trace 2008-03-18, http://mawi.wide.ad.jp/mawi/samplepoint-F/20080318/.
[22] F. Larroca, Techniques d'Ingénierie de Trafic Dynamique pour l'Internet, PhD thesis, Telecom ParisTech, 2009. http://iie.fing.edu.uy/investigacion/grupos/artes/publicaciones/larroca_phd.pdf
[23] The Abilene Network, http://www.internet2.edu/network/.
[24] TOTEM: TOolbox for Traffic Engineering Methods, http://totem.info.ucl.ac.be/.
[25] Yin Zhang, Abilene Dataset, http://www.cs.utexas.edu/∼yzhang/research/AbileneTM/.
[26] P. Casas, L. Fillatre, S. Vaton, Robust and reactive traffic engineering for dynamic traffic demands, in: NGI 2008, Krakow, Poland, 2008, pp. 69–76.
[27] M. Herbster, M. K. Warmuth, Tracking the best expert, Mach. Learn. 32 (2) (1998) 151–178. doi:http://dx.doi.org/10.1023/A:1007424614876.
[28] The Network Simulator - ns.
URL `http://nsnam.isi.edu/nsnam/index.php/Main_Page`
[29] D. Katz, K. Kompella, D. Yeung, Traffic Engineering (TE) Extensions to OSPF Version 2, IETF, RFC 3630, Sept. 2003. http://www.ietf.org/rfc/rfc3630.txt
[30] D. Peleg, Distributed Computing: A Locality-Sensitive Approach, SIAM, 2000.
[31] C. Cassandras, M. Abidi, D. Towsley, Distributed routing with on-line marginal delay estimation, IEEE Trans. Comm. 38 (3) (1990) 348–359.
[32] T. Guven, R. La, M. Shayman, B. Bhattacharjee, A unified framework for multipath routing for unicast and multicast traffic, Networking, IEEE/ACM Transactions on 16 (5) (2008) 1038–1051.
[33] M. Heusse, F. Rousseau, G. Berger-Sabbatel, A. Duda, Performance anomaly of 802.11b, IEEE INFOCOM 2003 2 (2003) 836–843.