The zero-one principle for switching networks [*]

Yossi Azar [†] Yossi Richter [‡]

Abstract

Recently, approximation analysis has been extensively used to study algorithms for routing weighted packets in various network settings. Although different techniques were applied in the analysis of diverse models, one common property was evident: The analysis of input sequences composed solely of two different values is always substantially easier, and many results are known only for restricted value sequences. Motivated by this, we introduce our zero-one principle for switching networks which characterizes a wide family of algorithms for which achieving c-approximation (as well as c-competitiveness) with respect to sequences composed of 0's and 1's implies achieving c-approximation. The zero-one principle proves to be very efficient in the design of switching algorithms, and substantially facilitates their analysis. We present three applications. First, we consider the Multi-Queue Switching model and design a 3-competitive algorithm, improving the result from [12]. Second, we study the Dynamic Routing problem on a line topology of length k and present a k-competitive algorithm, which improves and generalizes the results from [2, 21]. As a third application, we consider the work of [20], that compares the performance of local algorithms to the global optimum in various network topologies, and generalize their results from 2-value sequences to arbitrary value sequences.

1 Introduction

Overview: Packet routing networks, most notably the Internet, have become the preferred platform for carrying data of all kinds. Due to the steady increase of network traffic, and the fact that Internet traffic tends to constantly fluctuate, Quality of Service (QoS) networks, which allow prioritization between different traffic streams have gained considerable attention within the networking community. As network overloads become frequent, intermediate switches have to cope with increasing amounts of traffic, while attempting to pass forward more "valuable" packets, where values correspond to the required quality of service for each packet. We can measure the quality of the decisions made within a network by considering the total value of packets that were delivered to their destination.

Traditionally, network routing algorithms were studied within the stability analysis framework, either by a probabilistic model for packet injection (queuing theory, see e.g. [15, 28]) or an adversarial model (adversarial queuing theory, see e.g. [8, 16]). In stability analysis we usually consider

[*]A preliminary version of this paper appears in the proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC), 2004, pp. 64-71.

[†]**Corresponding author.** Phone: +972-3-6406354, Fax:+972-3-6409357, azar@tau.ac.il. School of Computer Science, Tel Aviv University, Tel Aviv, 69978, Israel. Research supported in part by the Israeli Ministry of Industry and Trade and by the Israel Science Foundation.

[‡]yo@tau.ac.il. School of Computer Science, Tel-Aviv University, Tel-Aviv, 69978, Israel. Research supported in part by the Israeli Ministry of industry and trade.

networks with unit-value packets, and the goal is to measure the largest amount of packets ever waiting for transmission on a link, as a function of the network topology and the packet injection model, thereby bounding the buffer size needed to prevent packet drop. Since it seems inevitable to drop packets in real-world networks, approximation analysis framework, which avoids any assumptions on the input sequence and compares the performance of algorithms to the optimal solution, has been adopted recently for studying throughput maximization problems. In particular, competitive analysis has been applied lately to investigate online routing algorithms. Researchers have investigated single-queue switches in various settings [1, 5, 6, 7, 14, 21, 22, 24, 27], multi-queue switches [4, 10, 12, 11, 19, 23, 25, 29] and multiple-node networks [2, 9, 13, 18, 20].

The zero-one principle: While different techniques were used to analyze algorithms in various switching models, there was one common property: Analysis of 2-value sequences, in which packets can take only 2 distinct values, was always substantially easier compared with arbitrary packet sequences. Moreover, many results are known only for restricted value sequences, due to the fact that handling the state of a system containing packets with arbitrary values is significantly more involved. Motivated by this, we introduce the zero-one principle for switching networks. This principle applies to all *comparison-based* switching algorithms, that base their decisions on the relative order between packet values. The principle says that in order to prove that an algorithm achieves c-approximation it is sufficient to prove that it achieves c-approximation with respect to sequences composed solely of 0's and 1's, where ties between packets with equal values may be broken arbitrarily. We note that one can assume that without loss of generality there are no 0-value packets in the input sequence since such packets could have been dropped. Indeed, the optimal solution may ignore all the 0-value packets, however, the comparison-based algorithm may not, since it only regards the relative order between values.

A zero-one principle has been already introduced for sorting networks in the comparison model [3, 17, 26], and it has been proved that a sorting network, that sorts correctly all sequences that are composed of 0's and 1's, is guaranteed to sort correctly an arbitrary sequence. In a similar manner to the zero-one principle for sorting networks, our principle turns out to be very useful in the design of a wide range of approximation algorithms for different switching models, and considerably facilitates their analysis, due to the fact that it allows to focus on 0/1 sequences. We note that in contrast with sorting networks where sorting 0's and 1's is equivalent to sorting any two distinct values, the analysis of 0/1 sequences in switching networks is easier compared with arbitrary 2-value sequences. In addition, we note that there is a myriad of research papers on throughput maximization problems in networks transmitting unit-value packets. Our paper provides a linking step between previous analysis of unit-value networks and QoS networks, by allowing to modify previous analysis only for 0/1 sequences, which are in many cases not very different from uniform sequences. We present three applications to the zero-one principle.

Applications: We first consider the preemptive Multi-Queue QoS Switching model that was originally introduced in [12]. In this model we have a switch with m incoming FIFO queues with bounded capacities and one output port. At each time step new packets arrive to the queues, each associated with a value. Additionally, at each time step the switch selects one non-empty queue and transmits the packet at its head through the output port. The goal is to maximize the total value of transmitted packets. We present a 3-competitive algorithm for this problem, improving upon the 3.5-competitive algorithm that was shown in [12].

Our second application is the Dynamic Routing problem on a line. We consider a network with a topology of a line of length k , i.e. node i ($i = 1, \ldots, k-1$) is connected to node $i+1$ by a unidirectional link, and contains a fixed-size FIFO queue to store the packets waiting to be transmitted. At each time step new packets may arrive online to the network nodes, each packet is associated with a value and a destination node. Additionally, each node can transmit the packet at the head of its queue to the next node. The goal is to maximize the total value of packets that were delivered to their destination. Special cases of this model were studied in [2, 21]. Kesselman *et al.* [21] studied the single-queue model, that corresponds to the $k = 2$ case in our formulation, and proved that the natural greedy algorithm is asymptotically 2-competitive. The unweighted version of our model, in which all packets have unit value, was investigated by Aiello *et al.* [2] who proved that the greedy algorithm is $O(k)$-competitive. We prove that the natural greedy algorithm is k-competitive for the weighted problem, therefore generalizing the results in [2, 21].

Kesselman *et al.* [20] analyzed the performance of local off-line and on-line algorithms in some network topologies compared with the performance of the global optimal solution, with the additional restriction that algorithms are work-conserving. Their strongest results applied only to 2-value sequences composed of 1 and $\alpha > 1$ values. As a third application, we employ our zero-one principle to generalize their results to arbitrary value sequences.

Related results: Initially, online algorithms for single-queue switches were studied, for both the preemptive case (in which packets stored in the queue can be discarded) and the non-preemptive case (where an accepted packet can no longer be discarded). Aiello *et al.* [1] initiated the study of different queuing policies for the 2-value non-preemptive model, in which each packet has a value of either 1 or $\alpha > 1$. Andelman *et al.* [6, 7] later showed a tight bound (up to an additive factor) of $\ln \alpha$ (where $\alpha > 1$ is the ratio between the largest value to the smallest one) for the general non-preemptive model. The preemptive 2-value model was initially studied by Kesselman and Mansour [22], followed by Lotker and Patt-Shamir [27] who presented a 1.3-competitive deterministic algorithm, which almost matches the known lower bound of 1.28. Finally, Andelman [5] introduced a randomized 1.25-competitive algorithm that beats the deterministic lower bound. The general preemptive single-queue model, where packets can take arbitrary values, was investigated by Kesselman *et al.* [21], who proved that the natural greedy algorithm is 2-competitive (more specifically $2\alpha/(1+\alpha)$-competitive). Later on, Bansal *et al.* [14] presented a 1.75-competitive algorithm (which is an improvement over an algorithm previously suggested by Kesselman *et al.* [24]), that achieves the best ratio known today. The best lower bound for the general preemptive model is currently 1.419 ([24]).

In the multi-queue model the switch contains multiple incoming queues but only a single output port. This model was introduced in [12] where it is shown how it can be reduced to the single queue model. Specifically, a 3.5-competitive algorithm was presented in [12] for the general preemptive setting. Albers and Schmidt [4] introduced a deterministic 1.89-competitive algorithm for the unit-value model (improving upon the natural 2-competitive algorithm), followed by Azar and Litichevskey [10] that showed a 1.58-competitive algorithm for switches with large buffers. A randomized 1.58-competitive algorithm was presented in [12] for the unit-value model, and later improved by Schmidt [29] to a 1.5-competitive randomized algorithm. An alternative model to the multi-queue switch is the shared memory switch, in which memory is shared among all queues. Hahne *et al.* [19] studied buffer management policies in this model while focusing on deriving upper and lower bounds for the natural Longest Queue Drop policy. Improved results were later obtained by Kesselman and Mansour [23].

Aiello *et al.* [2] studied the Dynamic Routing problem with unit packets and investigated the performance of various greedy algorithms with comparison to their stability guarantees. In particular, they showed that the natural greedy algorithm is $O(k)$-competitive for a line topology. A different version of the problem assumes that the queues are non-FIFO (note that this makes sense even for the unit-value case since packets can be transmitted out of order according to their distance from destination for example). For the unit-value non-FIFO version of the line problem, Aiello *et al.* [2] showed an $O(k^{2/3})$-competitive algorithm. Recently, this result was improved to $O(\log k)$ independently in [9] and [13].

Paper structure: The paper is organized as follows. Section 2 includes formal definitions and notations. In section 3 we prove our zero-one principle and its extensions. We apply our zero-one principle to three different switching models in sections 4, 5 and 6.

2 Definitions and notations

In this section we formally define the three problems we consider throughout the paper and introduce some notations.

2.1 Multi-Queue Switch

In the online Multi-Queue Switch problem we are given a switch with m FIFO queues, where queue i has size B_i, and one output port. Packets arrive online, each packet is destined to one of the queues and is associated with a non-negative value. Initially, the m queues are empty. We assume that time is discrete, and each time step $t \geq 0$ is divided into two phases: At the beginning of the first phase of time t a set of packets arrive to the queues. Packets can be inserted to each queue without exceeding its capacity. Remaining packets must be discarded. In the second phase of time t, the switching algorithm may select one of the non-empty queues and transmit the packet at its head. The goal is to maximize the total value of transmitted packets. We consider the preemptive case, in which packets that are stored in the queues may be discarded.

2.2 Dynamic Routing

In the problem of Dynamic Routing on a *line* we have a network organized in a line topology of length $k \geq 2$, i.e. node $i = 1, \ldots, k-1$ is connected to node $i+1$ via a unidirectional link with unit capacity. Node $i = 1, \ldots, k-1$ contains a FIFO queue of size B_i, which is initially empty, to buffer the packets waiting to be transmitted via its outgoing link. We assume time proceeds in discrete steps, and each time step $t \geq 0$ is divided into two phases: At the first phase new packets may arrive to nodes $i = 1, \ldots, k-1$, each packet p is associated with a value $v(p)$ and a destination node $d(p)$. Note that packets cannot originate at the last node, which can only serve as a destination. Therefore, the single queue model is a special case obtained for $k = 2$. During the second phase of time step t, node $i = 1, \ldots, k-1$ may transmit the packet at the head of its queue to node $i+1$. If a packet reaches its destination it is absorbed. Otherwise, arriving packets (from both phase 1 and phase 2) can be enqueued without exceeding the queues capacities. Remaining packets must be discarded. We allow preemption, i.e. packets that are stored in the queues may be discarded. The goal is to maximize the total value of packets that reach their destination.

The Dynamic Routing problem can be generalized to an arbitrary directed graph $G = (V, E)$, where nodes represent the network switches, and edges represent unidirectional links. In the general case each packet arrives with a predefined path from its source to destination, and each directed link $e \in E$ is associated with a capacity $c(e)$, and a FIFO queue Q_e to store the packets waiting for transmission on e.

2.3 Work-Conserving Dynamic Routing

We also consider the dynamic routing problem described in the previous paragraph under the *work-conserving* restriction. A schedule is called *work-conserving* if for every time t and link e, the number of packets transmitted over e at time t is the minimum between $c(e)$ and the number of packets in Q_e. Following [20], we consider two network topologies: A directed line topology, identical to the one defined in the previous paragraph, and a directed tree topology, where packets are only injected at the leaves and are destined to the root. An algorithm is called *local on-line* if its action at time t at node v depends only on the packets arriving to v until time t. An algorithm is called *local off-line* if its action at time t at node v depends only on the sequence of packets arriving at node v (possibly after time t). We denote by OptL_v the local off-line optimal solution, that maximizes the total value transmitted out of v. For any acyclic network we denote by OptL the algorithm that exercises OptL_v in topological order. We denote by Opt the global work-conserving optimal algorithm that knows all the information in the system (all nodes, all times).

2.4 Additional notations

We denote by $\sigma = \{p_1, \ldots, p_n\}$ the finite sequence of incoming packets, and by $v : \sigma \to \mathbb{R}^+$ the packets value function. Let $\mathcal{A}$ be a switching algorithm, we denote by $\mathcal{A}(\sigma)$ the set of packets delivered by $\mathcal{A}$ given the sequence σ (also referred to as the output sequence). Given a packet sequence σ, we denote by $V(\sigma)$ the total value of packets in the sequence, i.e. $V(\sigma) = \sum_{p\in\sigma} v(p)$. Given a function $f : \mathbb{R}^+ \to \mathbb{R}^+$ and a packet sequence σ, we denote by $f(\sigma)$ the packet sequence σ with the modified value function $f \circ v$. In particular, for a given switching algorithm $\mathcal{A}$ we use the notation $f(\mathcal{A}(\sigma))$ to denote the set of transmitted packets with modified values.

A deterministic (resp. randomized) approximation algorithm $\mathcal{A}$ achieves c-approximation ($c \geq 1$) for a maximization problem iff for every packet sequence σ we have: $V(\mathsf{Opt}(\sigma)) \leq c \cdot V(\mathcal{A}(\sigma))$ (resp. $V(\mathsf{Opt}(\sigma)) \leq c \cdot E[V(\mathcal{A}(\sigma))]$), where Opt denotes the optimal solution. The definition of the competitive ratio with respect to online algorithms is the same.

3 The zero-one principle and its extensions

In this section we introduce our zero-one principle for switching networks and its extensions. We focus on comparison-based algorithms which are defined as follows.

Definition 3.1 *A switching algorithm $\mathcal{A}$ (on-line or off-line) is called* comparison-based *if it bases its decisions on the relative order between packet values (by performing only comparisons), with no regard to the actual values.*

The following theorem shows that in order to prove a certain approximation ratio for a comparison-based algorithm, it is sufficient to consider only 0/1 sequences, that are composed of packets whose values are either 0 or 1.

Theorem 3.1 *[**zero-one principle**] Let $\mathcal{A}$ be a comparison-based switching algorithm (deterministic or randomized). $\mathcal{A}$ is a c-approximation algorithm if and only if $\mathcal{A}$ achieves c-approximation* *for all packet sequences whose values are restricted to* $\{0, 1\}$ *for every possible way of breaking ties between equal values.*

Proof: The "only if" direction is straightforward (even if $\mathcal{A}$ breaks ties between equal packet values in a specific way), thus it remains to prove the other direction. We define the monotonically increasing step function $f_t(x)$ by: $f_t(x) = 1$ if $x \geq t$, and otherwise $f_t(x) = 0$. Given a packet sequence σ, we denote by $\mathcal{A}(\sigma)$ the set of packets delivered by $\mathcal{A}$. We further denote by $\mathcal{A}(f_t(\sigma))$ the set of packets delivered by $\mathcal{A}$ when given the 0/1 sequence $f_t(\sigma)$, where we break ties between packets with equal values according to their original values in σ. Therefore, since $\mathcal{A}$ is comparison-based, the sets $\mathcal{A}(\sigma)$ and $\mathcal{A}(f_t(\sigma))$ contain the same packets, but with different values.

In the following claims we first show that the value of a set of packets can be broken into a sum of 0/1 sequences by using f_t. We then show that when $\mathcal{A}$ is comparison-based, a monotonic transformation can be applied to either the output or input sequence, yielding the same result.

Claim 3.2 *Let* σ *be any packet sequence. Then the following holds:* $V(\sigma) = \int_{t=0}^{\infty} V(f_t(\sigma))dt$.

Proof:

$$V(\sigma) = \sum_{p\in\sigma} v(p) = \sum_{p\in\sigma} \int_{t=0}^{\infty} f_t(v(p))dt = \int_{t=0}^{\infty} \sum_{p\in\sigma} f_t(v(p))dt = \int_{t=0}^{\infty} V(f_t(\sigma))dt,$$

where the second equality follows from $\int_{t=0}^{\infty} f_t(x)dt = x$. ■

Claim 3.3 *Let* $\mathcal{A}$ *be a comparison-based switching algorithm and let* f *be any monotonically increasing function. Then the following holds for any packet sequence* σ: $V(f(\mathcal{A}(\sigma))) = V(\mathcal{A}(f(\sigma)))$.

Proof:

$$V(f(\mathcal{A}(\sigma))) = \sum_{p\in\mathcal{A}(\sigma)} f(v(p)) = \sum_{p\in\mathcal{A}(f(\sigma))} f(v(p)) = V(\mathcal{A}(f(\sigma))).$$

■

We can now show for every sequence σ:

$$\begin{aligned} E[V(\mathcal{A}(\sigma))] &= E\left[\int_{t=0}^{\infty} V(f_t(\mathcal{A}(\sigma)))dt\right] = \int_{t=0}^{\infty} E[V(f_t(\mathcal{A}(\sigma)))]dt \\ &= \int_{t=0}^{\infty} E[V(\mathcal{A}(f_t(\sigma)))]dt \geq \int_{t=0}^{\infty} \frac{1}{c} \cdot V(\mathsf{Opt}(f_t(\sigma)))dt \\ &\geq \frac{1}{c} \cdot \int_{t=0}^{\infty} V(f_t(\mathsf{Opt}(\sigma)))dt = \frac{1}{c} \cdot V(\mathsf{Opt}(\sigma)), \end{aligned}$$

where the first and last steps follow from claim 3.2, the second step follows from linearity of expectation, the third step follows from claim 3.3, the fourth step follows since $f_t(\sigma)$ is a 0/1 sequence, and the fifth step follows since $\mathsf{Opt}(f_t(\sigma))$ maximizes the number of delivered packets with original value at least t. ■

Given Theorem 3.1, if we wish to prove that a comparison-based algorithm $\mathcal{A}$ achieves c-approximation, it is enough to show that for every 0/1 packet sequence σ, and for every possible way of breaking ties between equal values, we have: $V(\mathsf{Opt}(\sigma)) \leq c \cdot E[V(\mathcal{A}(\sigma))]$.

In many cases, the approximation ratio of a switching algorithm is given in terms of the ratio between the largest to smallest packet values, denoted by α. In the following theorem we extend the zero-one principle to this case, and prove that it is sufficient to consider packet sequences composed solely of two values: 1 and α.

Theorem 3.4 *[$1/\alpha$ **principle**] Let $\mathcal{A}$ be a comparison-based switching algorithm (deterministic or randomized). $\mathcal{A}$ is a $c(\alpha)$-approximation algorithm if and only if $\mathcal{A}$ achieves $c(\alpha)$-approximation for all packet sequences whose values are restricted to $\{1, \alpha\}$ for every possible way of breaking ties between equal values.*

Proof: The "only if" direction is straightforward, thus it remains to prove the other direction. We define the monotonically increasing step function $g_t(x) : [1, \alpha] \to \{1, \alpha\}$ for $1 \leq t \leq \alpha$ as follows: $g_t(x) = \alpha$ if $1 \leq t \leq x \leq \alpha$, otherwise $g_t(x) = 1$. As in the proof of Theorem 3.1, we denote by $\mathcal{A}(g_t(\sigma))$ the set of packets delivered by $\mathcal{A}$ when given the $1/\alpha$ sequence $g_t(\sigma)$, where we break ties between packets with equal values according to their original values in σ. We begin by restating claim 3.2.

Claim 3.5 *Let σ be any packet sequence whose values lie in the interval $[1, \alpha]$. Then the following holds: $V(\sigma) = \frac{1}{\alpha-1}\int_{t=1}^{\alpha} V(g_t(\sigma))dt$.*

Proof:

$$\begin{aligned} V(\sigma) &= \sum_{p\in\sigma} v(p) = \sum_{p\in\sigma} \frac{1}{\alpha-1}\int_{t=1}^{\alpha} g_t(v(p))dt = \frac{1}{\alpha-1}\int_{t=1}^{\alpha} \sum_{p\in\sigma} g_t(v(p))dt \\ &= \frac{1}{\alpha-1}\int_{t=1}^{\alpha} V(g_t(\sigma))dt, \end{aligned}$$

where the second equality follows since $\int_{t=1}^{\alpha} g_t(x)dt = \alpha(x-1) + (\alpha - x) = (\alpha-1)x$, for $x \in [1, \alpha]$.
■

We can now show for every sequence σ whose values lie in $[1, \alpha]$:

$$\begin{aligned} E[V(\mathcal{A}(\sigma))] &= E\left[\frac{1}{\alpha-1}\int_{t=1}^{\alpha} V(g_t(\mathcal{A}(\sigma)))dt\right] = \frac{1}{\alpha-1}\int_{t=1}^{\alpha} E[V(\mathcal{A}(g_t(\sigma)))]dt \\ &\geq \frac{1}{\alpha-1}\int_{t=1}^{\alpha} \frac{1}{c}\cdot V(\mathsf{Opt}(g_t(\sigma)))dt \geq \frac{1}{\alpha-1}\cdot\frac{1}{c}\cdot\int_{t=1}^{\alpha} V(g_t(\mathsf{Opt}(\sigma)))dt \\ &= \frac{1}{c}\cdot V(\mathsf{Opt}(\sigma)), \end{aligned}$$

where the first and last steps follow from claim 3.5, the second step follows from claim 3.3 and linearity of expectation, the third step follows since $g_t(\sigma)$ is a $1/\alpha$ sequence, and the fourth step follows since $\mathsf{Opt}(g_t(\sigma))$ maximizes the total value (with respect to g_t) of the packets transmitted.
■

As a second extension to the zero-one principle, we define a set of switching algorithms that achieve c-approximation if and only if they achieve c-approximation for all unit-value sequence, in which all packet have a value of 1.

Definition 3.2 *A switching algorithm $\mathcal{A}$ is called* work-conserving *if it always transmits a packet whenever a packet is available for transmission (i.e. it does not hold back packets).*

Definition 3.3 *A switching algorithm $\mathcal{A}$ is called* greedy *if it exercises a greedy admission control and a greedy transmission policy in all queues (i.e. it always accept the largest packets, and always transmits the largest packet).*

In particular, note that Definition 3.3 is valid only in non-FIFO models, in which packets can be transmitted in any order out of the queues.

Theorem 3.6 *[One principle] Let $\mathcal{A}$ be a comparison-based work-conserving greedy switching algorithm. $\mathcal{A}$ is a c-approximation algorithm if and only if $\mathcal{A}$ achieves c-approximation for all unit-value packet sequences for every possible way of breaking ties between equal values.*

Proof: The "only if" direction is straightforward, thus it remains to prove the other direction. Since $\mathcal{A}$ is comparison-based, by Theorem 3.1 it suffices to show that it achieves c-approximation for every 0/1 sequence, under every possible tie-breaking. Let us denote by $\Sigma_{0/1}$ the set of all finite 0/1 sequences, and by Σ_1 the set of all finite unit-value sequences. We define the operator $\pi : \Sigma_{0/1} \to \Sigma_1$ that deletes all the packets with value 0 from a given 0/1 sequence. The theorem will follow immediately from the next claim.

Claim 3.7 *Let σ be any 0/1 sequence, and apply any tie-breaking between equal values. Then $\pi(\mathcal{A}(\sigma)) = \mathcal{A}(\pi(\sigma))$.*

Proof: By a simple induction on the time steps. For each time step, each queue will hold the same 1-packets (since it is greedy) and will transmit the same 1-packet whenever a 1-packet is transmitted (since it is work-conserving), when $\mathcal{A}$ is given σ or $\pi(\sigma)$ as inputs. Therefore, the same set of 1-packets will be delivered for σ and $\pi(\sigma)$ and the claim follows. ■

To complete the proof of Theorem 3.6 we notice that for every 0/1 sequence σ under any tie-breaking, we obtain using Claim 3.7:

$$V(\mathcal{A}(\sigma)) = V(\pi(\mathcal{A}(\sigma))) = V(\mathcal{A}(\pi(\sigma))) \geq \frac{1}{c}V(\mathsf{Opt}(\pi(\sigma))) = \frac{1}{c}V(\mathsf{Opt}(\sigma)),$$

where the second step follows from Claim 3.7, the third step follows since $\pi(\sigma)$ is a unit-value sequence, and the last step follows since we may assume w.l.o.g that Opt discards all 0-packets from the input sequence. ■

4 Application 1: Multi-queue switch

In this section we present the first application of our zero-one principle. We consider the online problem of a multi-queue switch (see Section 2.1 for a formal definition) and design a 3-competitive algorithm for the problem, improving upon the 3.5-competitive algorithm presented in [12]. Our algorithm, named **TransmitLargestHead** (abbreviated TLH), is given in figure 1.

Algorithm TLH **[Multi-Queue]**

1. **Admission control:** Use algorithm GR for admission control in all m incoming queues.
2. **Scheduling:** At each time step, transmit the packet with the largest value among all packets at the head of the queues.

Figure 1: Algorithm TLH.

Before we proceed to prove the competitive ratio of algorithm TLH, we state the following well-known observation, which allows us to assume without loss of generality that the optimal solution does not use preemption.

Observation 4.1 *The optimal solution is the same for the preemptive and non-preemptive models and for the FIFO and non-FIFO models.*

Proof: The optimal (off-line) algorithm knows in advance all the packets that will be eventually transmitted. Therefore, only transmitted packets will be enqueued, hence preemption is redundant. Similarly, since only transmitted packets are enqueued, transmitting in FIFO order compared to non-FIFO order can only change the relative order of packets, but not their accumulated value. ■

Theorem 4.1 *Algorithm* TLH *is 3-competitive for the Multi-Queue QoS Switching problem.*

Proof: Clearly, algorithm TLH bases its admission control and scheduling decisions solely on relative order between values, therefore is it comparison-based. According to our zero-one principle (Theorem 3.1) we need only show that the algorithm is 3-competitive for 0/1 sequences, under every possible tie-breaking. Let σ be any 0/1 sequence. With slight abuse of notation we denote by $\mathsf{TLH}(\sigma)$ the set of packets with value 1 (1-packets in short) transmitted by the algorithm, and by $\mathsf{Opt}(\sigma)$ the set of packets transmitted by the optimal solution (note that we may assume that the optimal solution discards all 0-packets). Theorem 4.1 directly follows from the next lemma.

Lemma 4.2 *For every 0/1 sequence σ under every possible tie-breaking we have:* $|\mathsf{Opt}(\sigma)\backslash\mathsf{TLH}(\sigma)| \leq 2 \cdot |\mathsf{TLH}(\sigma)|$.

Proof: We first note that preemption of a 1-packet can only occur when the entire queue is full with 1-packets, hence the contents of the queue remain unchanged. We can therefore analyze the TLH algorithm while assuming without loss of generality that preemption of 1-packets does not occur (i.e. we exchange preemption of a 1-packet with rejection of a 1-packet). We prove the lemma by providing a matching from $(\mathsf{Opt}(\sigma) \setminus \mathsf{TLH}(\sigma))$ to $\mathsf{TLH}(\sigma)$ in which each 1-packet from $\mathsf{TLH}(\sigma)$ is matched at most twice. This is done by a marking scheme that marks one of the 1-packets stored in the queues according to TLH operation, whenever a 1-packet is accepted by Opt but rejected by TLH, and when transmissions take place. A description of the marking scheme follows (figure 2).

Marking scheme

For each time step t do:

1. For each incoming 1-packet to queue i do:
   (a) If the packet is accepted by TLH, consider it as an unmarked packet.
   (b) Otherwise, if the packet is accepted by Opt, look for the first unmarked 1-packet, starting from the head of the queue and moving to its tail, and mark it.
2. In the transmission phase, whenever Opt transmits a packet do:
   (a) If Opt and TLH use the same queue for transmission, do nothing.
   (b) Otherwise, let $i \neq j$ be the queues used by Opt and TLH, respectively. If queue i contains marked packets, unmark the marked 1-packet closest to the tail in queue i and mark the packet transmitted from queue j.

Figure 2: Marking scheme for TLH.

Clearly, each 1-packet in $\mathsf{TLH}(\sigma)$ can be marked at most twice, once while it resides in the queue and once while it is transmitted. The following claims prove that each 1-packet transmitted by Opt but not by TLH is matched through the marking scheme to a 1-packet in TLH(σ). We begin by proving that whenever we change a marking (step 2b) a 1-packet is transmitted, and therefore the change is possible.

Claim 4.3 *For each time step t, if the queues hold any marked packets, then a 1-packet is transmitted.*

Proof: Let queue i hold a marked packet p at time t. When p was marked, queue i contained no 0-packets, hence all the packets closer to the head than p are 1-packets. In particular, since FIFO order is maintained, the packet at the head of queue i at time t is a 1-packet. Therefore, a 0-packet can not be transmitted at time t. ■

Before we proceed we introduce some notations. For a time step t, and a queue $i = 1, \ldots, m$ we denote by U_i^t the number of unmarked packets in queue i at time t. We further denote by TLH_i^t and Opt_i^t the number of 1-packets in queue i at time t in TLH and Opt, respectively. For simplicity of notation we drop the superscript t whenever the time is clear from context. To complete the proof of Lemma 4.2, it remains to show that whenever a 1-packet destined to queue i is rejected by TLH but accepted by Opt, queue i contains unmarked packets (step 1b).

Claim 4.4 *For each time step t and queue i, the following properties hold:*

- $U_i^t \geq \mathsf{TLH}_i^t - \mathsf{Opt}_i^t$.
- *There are available unmarked packets if required by the marking scheme (step 1b).*

Proof: First, recall that we assume w.l.o.g that preemption of 1-packets does not occur. Additionally, our marking scheme is not concerned with 0-packets, hence we may ignore any operation that involves 0-packets.

We prove the claim by induction on the time steps. For time step $t = 0$ the properties clearly hold. We assume correctness by time step t and prove that the properties hold for the beginning of time step $t + 1$. Consider a 1-packet p arriving to queue i. If p is accepted by TLH, no marking is required, U_i increases and the inequality holds. Otherwise, if p is rejected by TLH and accepted by Opt, then by the induction hypothesis $U_i \geq \mathsf{TLH}_i - \mathsf{Opt}_i$ and since $\mathsf{TLH}_i = B_i$ and $\mathsf{Opt}_i < B_i$ (we assume w.l.o.g that Opt does not preempt packets, see Observation 4.1) we have an unmarked packet in queue i. After we mark a 1-packet both sides of the inequality decrease by 1. Now, consider the transmission phase. Let i and j be the queues used for transmission by Opt and TLH, respectively. The inequality clearly holds for queue j if it does not hold any marked packets. Otherwise, a marked packet is transmitted from queue j (since we mark packets in the direction from the head to the tail), U_j is unchanged while the right hand side of the inequality can only decrease. If $i \neq j$, we increase U_i by unmarking a packet (step 2b), and the inequality continues to hold for queue i. ■

This concludes the proofs of Lemma 4.2 and Theorem 4.1. ■

The following lemma proves that our analysis of TLH is asymptotically tight.

Lemma 4.5 *Algorithm* TLH *is at least* $(3 - \frac{1}{m} - \frac{m}{\sum_{i=1}^{m} B_i})$*-competitive.*

Proof: We construct the following sequence σ:

1. At $t = 0$, B_i packets with value ε arrive to queue i $(i = 1, \ldots, m)$.
2. For $1 \leq i \leq m-1$, during time steps $\sum_{j=1}^{i-1}(B_j - 1), \ldots, \sum_{j=1}^{i}(B_j - 1) - 1$, $B_i - 1$ packets with value $1 + \delta$ arrive to queue i, one packet at a time. Then, during the next $B_m - 1$ time steps, $B_m - 1$ packets with value 1 arrive to queue m, one packet at a time.
3. When phase 2 is completed, a set of $\sum_{i=1}^{m} B_i$ packets arrives at the same time, B_i packets to queue i $(i = 1, \ldots, m)$. All packets destined to queues $\{1, \ldots, m-1\}$ have value $1 + \delta$, while packets destined to queue m have value 1.
4. During the next $\sum_{i=1}^{m-1} B_i$ time steps, a packet with value 1 arrives at each time step to queue m.
5. No more packets arrive.

We now analyze the competitive ratio of TLH while we assume w.l.o.g $B_m = \min_{1\leq i\leq m} B_i$ and we take $\varepsilon, \delta \to 0$, thereby considering σ to consist of 0's and 1's. Algorithm TLH accepts all the packets that arrive at step 1. During phase 2, $B_i - 1$ ε-packets are transmitted from queue i $(i = 1, \ldots, m)$ while the queues are filled with 1-packets. We may assume w.l.o.g that all the packets that arrive in step 3 are enqueued while all the 1-packets from phase 2 are discarded. During phase 4 TLH empties queues $\{1, \ldots, m-1\}$ first, therefore rejecting all the packets that arrive during this phase to queue m. In contrast, Opt discards all the ε-packets that arrive at the beginning, and can therefore transmit all subsequent packets. Hence:

$$\frac{\mathsf{Opt}(\sigma)}{\mathsf{TLH}(\sigma)} = \frac{3 \cdot \sum_{i=1}^{m} B_i - B_m - m}{\sum_{i=1}^{m} B_i} \geq 3 - \frac{1}{m} - \frac{m}{\sum_{i=1}^{m} B_i}$$

■

5 Application 2: Dynamic Routing

In this section we present our second application of the zero-one principle. We consider the online problem of weighted dynamic routing on a line (see section 2.2 for a formal definition) and present a k-competitive algorithm called PF for the problem (figure 3).

Algorithm PF (PushForward)

For each node $i = 1, \ldots, k-1$ do:

1. **Arrival phase:** Use the GR algorithm to accept packets into the queue.
2. **Transmission phase:** If the queue is not empty, transmit the packet at the head of the queue to the next node.

Figure 3: Algorithm PF.

Theorem 5.1 *Algorithm PF is k-competitive for the Dynamic Routing problem on a line of length k.*

Proof: Clearly, algorithm PF bases its admission control decisions solely on relative order between values, therefore it is comparison-based. According to our zero-one principle (Theorem 3.1) we need only show that the algorithm is k-competitive for 0/1 sequence, given any possible tie-breaking. Let σ be a 0/1 sequence. With slight abuse of notation we denote by $\mathsf{PF}(\sigma)$ the set of 1-packets that were delivered by PF, and by $\mathsf{Opt}(\sigma)$ the set of packets delivered by Opt. Theorem 5.1 directly follows from the next lemma.

Lemma 5.2 *For every 0/1 sequence σ under every possible tie-breaking we have:* $|\mathsf{Opt}(\sigma)\backslash\mathsf{PF}(\sigma)| \leq (k-1)\cdot|\mathsf{PF}(\sigma)|$.

Proof: By the same considerations given in the proof of Lemma 4.2 we assume w.l.o.g that preemption of 1-packets does not occur. We prove the lemma by providing a matching from $(\mathsf{Opt}(\sigma)\backslash\mathsf{PF}(\sigma))$ to $\mathsf{PF}(\sigma)$ in which each packet from $\mathsf{PF}(\sigma)$ is matched at most $k-1$ times. This is done by a marking scheme that marks one of the 1-packets in PF queues whenever a 1-packet is accepted by Opt but rejected by PF. A description of the marking scheme follows (figure 4).

Marking scheme

For each node $i = 1, \ldots, k-1$ do:

1. For each incoming 1-packet do:
   (a) If the packet is accepted by PF, consider it as an unmarked packet.
   (b) Otherwise, if the packet is accepted by Opt, look for the first unmarked 1-packet, starting from the head of queue i and moving to its tail, and mark it.
2. In the transmission phase:
   If a 1-packet arrives from the previous node, it is considered to be unmarked.

Figure 4: Marking scheme for PF.

We observe that each 1-packet is marked at most $k-1$ times (at most once in each of the intermediate nodes on its path), and that packets are not dropped during the transmission phase since all nodes that store packets push a packet forward. Therefore, it remains to prove that we always have an available unmarked packet when we reject a packet that is accepted by Opt (step 1b in the marking scheme). In the following we use the notations U_i^t, PF_i^t and Opt_i^t $(i = 1, \ldots, k-1)$ with similar meanings to those used in the proof of Claim 4.4.

Claim 5.3 *For each time step t and queue i, the following properties hold:*

- $U_i^t \geq \mathsf{PF}_i^t - \mathsf{Opt}_i^t$.
- *There are available unmarked packets if required by the marking scheme (step 1b).*

Proof: First, recall that we assume w.l.o.g that preemption of 1-packets does not occur. Additionally, our marking scheme is not concerned with 0-packets, hence we may ignore any operation that involves 0-packets.

We prove the claim by induction on the time steps. At the beginning of time step $t = 0$ the queues are empty, hence the inequality clearly holds. We assume correctness for time step t and prove that the inequality continues to hold for at $t + 1$. We first consider the packet arrival phase.

Consider a 1-packet p arriving to queue i. If p is accepted by PF, U_i increases and the inequality holds. Otherwise, if p is rejected by PF and is accepted by Opt, then by the induction hypothesis $U_i \geq \mathsf{PF}_i - \mathsf{Opt}_i$ and since $\mathsf{PF}_i = B_i$ and $\mathsf{Opt}_i < B_i$ (we again assume that Opt does not preempt packets, see Observation 4.1) we have an unmarked 1-packet in queue i. After we mark a packet both sides of the inequality decrease by 1. Next, consider the transmission phase. Recall that no packets are dropped during this phase. We examine the changes in both the transmitting end and the receiving end. At the transmitting end, note that if queue i does not contain marked packets then the inequality clearly holds. Otherwise, U_i remains unchanged (since a marked packet is transmitted) while the right hand side of the inequality can only decrease (if Opt chooses not to transmit). At the receiving end, we first observe that a packet that reaches its destination has no effect. A 1-packet enqueued into queue i in PF causes U_i to increase; A packet enqueued by Opt causes the right hand side of the inequality to decrease. In either case, the inequality continues to hold. ■

This concludes the proofs of Lemma 5.2 and Theorem 5.1. ■

Theorem 5.1 can be generalized as follows. The proof is obtained using the same marking scheme.

Theorem 5.4 *Let $G = (V, E)$ be a directed graph with uniform edge capacities such that for each vertex v, $d_{in}(v) = 1$. The competitive ratio of algorithm PF on networks with topology G is $\ell(G) + 1$ (where $\ell(G)$ is the length of the longest acyclic directed path in G).*

Observe that in particular Theorem 5.4 implies that algorithm PF is k-competitive for a cycle topology of length k. The following lemma shows that our analysis of PF is tight up to a factor of 2.

Lemma 5.5 *Algorithm PF is at least $k/2$-competitive on a line topology of length k.*

Proof: Consider the following packet sequence σ arriving to a network with queues of uniform size B:

1. At time $t = 0$, $2B$ packets are injected at node i ($i = 1, \ldots, k-1$), the first B packets have value $1 + \varepsilon$ and are destined to node k, the last B packets have value 1 and are destined to node $i + 1$.
2. Packets with value 1 and destination $i + 1$ are injected at node i ($i = 2, \ldots, k-1$) during time steps $1, \ldots, (i-1) \cdot B - 1$.

Algorithm PF accepts the B more valuables packets injected at step 1 at each node and rejects all other packets in the sequence. In contrast, Opt rejects the valuable packets injected at step 1, and can therefore deliver all other packets. Taking $\varepsilon \to 0$ we obtain:

$$\frac{\mathsf{Opt}(\sigma)}{\mathsf{PF}(\sigma)} = \frac{\sum_{i=1}^{k-1} i \cdot B}{(k-1)B} = \frac{k}{2}.$$

■

6 Application 3: Work-Conserving Dynamic Routing

Kesselman *et al.* [20] studied the dynamic routing problem under the *work-conserving* restriction. Specifically, they compared the performance of the local on-line GR algorithm and the local off-line OptL algorithm to the global work-conserving optimum, in networks with line and tree topologies (see section 2.3 for exact definitions). The strongest results in [20] are shown only for sequences restricted to two values: 1 and $\alpha > 1$, while the bounds for arbitrary sequences remain as open questions. Using our zero-one principle we can generalize their results to arbitrary sequences as follows. First, we observe that the local off-line algorithm, OptL_v, is in fact a comparison-based algorithm that sorts the sequence of packets according to their values and then attempts to schedule the packets starting from the largest value (see e.g. [27]). The local on-line GR algorithm is clearly comparison-based. In the proofs given in [20] the number of valuable packets (of value $\alpha > 1$) that were delivered is compared against the global optimum. In fact, the proofs follow as-is for 0/1 sequences and arbitrary tie-breaking. Before we proceed to restate the main results from [20] in their general form, we need to introduce some additional definitions and notations from [20].

Definition 6.1 *For a given link e in a given network, define the* delay *of e, denoted $D(e)$, to be the ratio $\lceil size(Q_e)/c(e) \rceil$. The delay of a given path is the sum of the edge delays on that path.*

Definition 6.2 *Let $e = (u, v)$ be any directed link in a given tree topology. The* height *of e, denoted $h(e)$, is the maximum path delay, over all paths starting at a leaf and ending at v. The* weakness *of e, denoted $\lambda(e)$, is defined to be $\lambda(e) = \frac{h(e)}{D(e)}$.*

We are now ready to restate the main results from [20] in their generalized form, for arbitrary sequences. We emphasize that all other results in [20] restricted to sequences with two values can be generalized as well. As mentioned before, proofs follow as-is, given the zero-one principle.

Theorem 6.1 *Under the work-conserving restriction, the competitive ratio of* GR *for any given tree topology $G = (V, E)$ is $O(\max\{\lambda(e) \mid e \in E\})$.*

Theorem 6.2 *Under the work-conserving restriction, the approximation ratio of* OptL *for a complete binary tree of depth h is at most $O(h/\log h)$.*

Theorem 6.3 *Under the work-conserving restriction, the approximation ratio of* OptL *for a line of length h is $O(\sqrt{h})$.*

References

[1] W. Aiello, Y. Mansour, S. Rajagopolan, and A. Rosén. Competitive queue policies for differentiated services. In *Proc. of IEEE INFOCOM*, pages 431–440, 2000.

[2] W. Aiello, R. Ostrovsky, E. Kushilevitz, and A. Rosén. Dynamic routing on networks with fixed-size buffers. In *Proc. 14th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 771–780, 2003.

[3] M. Ajtai, J. Komlós, and E. Szemerédi. An $o(n \log n)$ sorting network. In *Proc. 15th ACM Symp. on Theory of Computing (STOC)*, pages 1–9, 1983.

[4] S. Albers and M. Schmidt. On the performance of greedy algorithms in packet buffering. In *Proc. 36th ACM Symp. on Theory of Computing (STOC)*, pages 35–44, 2004.

[5] N. Andelman. Randomized queue management for diffserv. In *Proc. of the 17th Annual ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 1–10, 2005.

[6] N. Andelman and Y. Mansour. Competitive management of non-preemptive queues with multiple values. In *Proc. 17th International Symp. on Distributed Computing (DISC)*, pages 166–180, 2003.

[7] N. Andelman, Y. Mansour, and A. Zhu. Competitive queueing policies for QoS switches. In *Proc. 14th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 761–770, 2003.

[8] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results for greedy contention-resolution protocols. *J. of the ACM*, 48(1):39–69, 2001.

[9] S. Angelov, S. Khanna, and K. Kunal. The network as a storage device: Dynamic routing with bounded buffers. In *Proc. 8th. International Workshop on Approx. Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 1–13, 2005.

[10] Y. Azar and M. Litichevskey. Maximizing throughput in multi-queue switches. In *Proc. 12th Annual European Symp. on Algorithms (ESA)*, pages 53–64, 2004.

[11] Y. Azar and Y. Richter. An improved algorithm for CIOQ switches. In *Proc. 12th Annual European Symp. on Algorithms (ESA)*, pages 65–76, 2004.

[12] Y. Azar and Y. Richter. Management of multi-queue switches in QoS networks. *Algorithmica*, 43(1–2):81–96, 2005.

[13] Y. Azar and R. Zachut. Packet routing and information gathering in lines, rings and trees. In *Proc. 13th Annual European Symp. on Algorithms (ESA)*, pages 484–495, 2005.

[14] N. Bansal, L. Fleischer, T. Kimbrel, M. Mahdian, B. Schieber, and M. Sviridenko. Further improvements in competitive guarantees for QoS buffering. In *Proc. 31st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 196–207, 2004.

[15] A. Birman, H. R. Gail, S. L. Hantler, Z. Rosberg, and M. Sidi. An optimal service policy for buffer systems. *J. of the ACM*, 42(3):641–657, 1995.

[16] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. Williamson. Adversarial queuing theory. *J. of the ACM*, 48(1):13–38, 2001.

[17] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, 1990.

[18] E. Gordon and A. Rosén. Competitive weighted throughput analysis of greedy protocols on dags. In *Proc. 24th ACM Symp. on Principles of Distrib. Computing (PODC)*, pages 227–236, 2005.

[19] E. L. Hahne, A. Kesselman, and Y. Mansour. Competitive buffer management for shared-memory switches. In *Proc. of the 13th Annual ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 53–58, 2001.

[20] A. Kesselman, Z. Lotker, Y. Mansour, and B. Patt-Shamir. Buffer overflows of merging streams. In *Proc. 11th Annual European Symp. on Algorithms (ESA)*, pages 349–360, 2003.

[21] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in QoS switches. *SIAM J. on Comput.*, 33(3):563–583, 2004.

[22] A. Kesselman and Y. Mansour. Loss-bounded analysis for differentiated services. *J. of Algorithms*, 46(1):79–95, 2003.

[23] A. Kesselman and Y. Mansour. Harmonic buffer management for shared-memory switches. *Theoret. Computer Sci.*, 324(2–3):161–182, 2004.

[24] A. Kesselman, Y. Mansour, and R. van Stee. Improved competitive guarantees for QoS buffering. *Algorithmica*, 43(1–2):63–80.

[25] A. Kesselman and A. Rosén. Scheduling policies for CIOQ switches. In *Proc. of the 15th Annual ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 353–362, 2003.

[26] D. E. Knuth. *The Art of Computer Programming, Sorting and Searching*, volume 3. Addison-Wesley, Reading, MA, 1973.

[27] Z. Lotker and B. Patt-Shamir. Nearly optimal fifo buffer management for two packet classes. *Computer Networks*, 42(4):481–492, 2003.

[28] M. May, J. C. Bolot, A. Jean-Marie, and C. Diot. Simple performance models of differentiated services for the internet. In *Proc. of IEEE INFOCOM*, pages 1385–1394, 1999.

[29] M. Schmidt. Packet buffering - randomization beats deterministic algorithms. In *Proc. 22nd Symp. on Theoretical Aspects of Comp. Science (STACS)*, pages 293–304, 2005.