Competitive FIB Aggregation without Update Churn

Marcin Bienkowski
University of Wrocław
Wrocław, Poland
mbi@ii.uni.wroc.pl

Nadi Sarrar
TU Berlin
Berlin, Germany
nadi@inet.tu-berlin.de

Stefan Schmid
T-Labs & TU Berlin
Berlin, Germany
stefan@net.t-labs.tu-berlin.de

Steve Uhlig
Queen Mary University
London, UK
steve.uhlig@qmul.ac.uk

***Abstract*—This paper attends to the well-known problem of compressing the Forwarding Information Base of a router or switch, while preserving a correct forwarding. In contrast to related work, we study an online variant of the problem where BGP routes can change over time, and where the number of updates to the FIB are taken into account explicitly. Minimizing the number of FIB updates is important, especially when they are sent across the network (e.g., from the network-attached SDN controller).**

This paper pursues a competitive analysis approach and introduces a formal model which is an interesting generalization of several classic online aggregation problems. The main contribution is a *O(w)*-competitive algorithm, where *w* is the length of an IP address. We also derive a lower bound which shows that our result is asymptotically optimal within a natural class of algorithms.

***Keywords*-Networking, Prefix Aggregation, Competitive Analysis, Ski Rental**

I. Introduction

At the heart of any Internet router lies a so-called *Forwarding Information Base* (FIB) containing the router's *forwarding rules*. A routing decision for a given packet is made on the basis of these rules and the destination IP address of a packet. A fast rule lookup requires the FIB to be stored in a fast (and expensive) memory on the line cards. Nowadays, the number of these rules at the Internet core routers is growing at an alarming rate. New forwarding rules emerge primarily because of the growth of the Internet itself, trends for advertising more specific routes, or the increasing demand for virtual networks [4], [12]. The migration to IPv6 is not expected to mitigate the address space disaggregation problem [5]. The increasing memory requirement comes at a significant cost for ISPs, as this memory is expensive and power-hungry.

A natural and local solution to mitigate the problem — before possible long-term solutions are deployed — is the *aggregation/compression of the FIB*, i.e., the replacement of the existing set of rules by an *equivalent but smaller* set. The aggregation of FIB rules has the appealing property that it is a purely local solution in the sense that it does not affect neighboring routers and it can be done by a simple software update [21].

While the compression of the FIB is beneficial in terms of memory, it also entails a potential overhead: As the FIB of a router changes dynamically over time — typically several thousands rules are modified each second [7] — the rule compression may lead to a situation where already aggregated FIB entries need to be disaggregated again, resulting in a larger number of rule updates. There is a certain cost associated with each such update. For example, the route processor (or, in the context of Software-Defined Networks (SDN): *the SDN controller* [14]), may need to send updates to the memory on the line card (or to the OpenFlow switch [14]); such additional transmissions of control messages is problematic as the communication channel between route processor and line card (resp. between the controller and the switch) can become a bottleneck [15]. Moreover, upon each update, the internal FIB structures have to be rebuilt.

In this paper, we present and analyze FIB aggregation algorithms which simultaneously try to maximize the compression ratio and minimize the number of updates to the compressed FIB.

A. The Model

An (IP) *address* is a binary string of length w (e.g., $w = 32$ for IPv4 and $w = 128$ for IPv6) or equivalently an integer from $[0, 2^w - 1]$. An (IP) *prefix* is a binary string of length at most w; we denote the empty prefix by ε. A prefix *contains* all addresses that start with it, i.e., it corresponds to a *range* of addresses of the form $[k \cdot 2^i, (k+1) \cdot 2^i - 1]$, where $w - i$ is the prefix length and $k \geq 0$ is an integer.

Forwarding Information Base (FIB). We consider a packet forwarding router with a set of *ports* (also known as the *next-hops*). A *Forwarding Information Base (FIB)* is a set of *forwarding rules* used by the router; each rule is a *(prefix,port)* pair (p, c). For the presentation, we will refer to the ports by *colors*, i.e., assume a unique color for each port. For any packet processed by the router, a decision is made on the basis of its destination IP address x using the *longest prefix match* policy [15]: among the FIB rules $\{(p_i, c_i)\}_i$, the router chooses the longest p_i being a prefix of x, and forwards the packet to port c_i. (We assume that there are no two rules with the same prefixes and different ports.) If no rule matches, the packet is dropped.

For instance, consider a FIB containing four rules $\{(\varepsilon, a), (00, b), (1, c), (11, a)\}$, where a, b, and c are ports. It could

be replaced by an equivalent FIB containing the rules $\{(\varepsilon, a), (00, b), (10, c)\}$. In this compression process, we require *strong forwarding correctness* [21], i.e., we require that the forwarding and dropping behavior remain the same.

Finally, we call two (different) rules *dependent* if the ranges represented by them overlap (i.e., one of these ranges is contained in the other) and *independent* otherwise.

Costs and Competitive Analysis. In our simplified setting, the router contains two parts: the *controller* (typically implemented on the route processor) and the *(compressed) FIB* (stored in a fast and expensive memory), cf. Figure 1. The controller keeps a copy of the *uncompressed FIB (U-FIB)* and receives dynamic updates to this structure (e.g., due to various events from the *Border Gateway Protocol, BGP*). More precisely, we assume continuous time; at any time t, a single forwarding rule may change its color (port). In particular, we do not allow new rules to be inserted to U-FIB nor old rules to be deleted from it. Thus, the input is a sequence of such color changes called *events*. Right after a change occurs, the controller must ensure that the U-FIB and the FIB are equivalent. To this end, the controller may insert, delete or update (change color) individual rules in the FIB. The controller can also issue these commands at any point of time (e.g., for a delayed compression of the FIB). We associate a fixed cost α to any such change of a single rule. (We emphasize that α is a fixed parameter but not necessarily a constant.)

Note that we use a fixed parameter α independently of the change type issued by the controller (insert, delete, color update) to keep the model general: α is not specific for any particular FIB data structure (e.g., trie or cache), but may also model the cost of transmitting a control packet between an SDN controller and the OpenFlow switch. (See also [10], [18].) We will refer to the total cost paid this way as *update cost*, and the amount paid by an algorithm ALG in a time interval I is denoted by $\text{U-COST}_I(\text{ALG})$. More generally, we ignore IP lookup costs in our model, as they depend on the data structure and are often negligible in practice [15, chapter 15].

The second type of cost we want to optimize is the size of the FIB, which — following [6] — is defined as the number of FIB forwarding rules. This modeling is justified by state-of-the-art approaches (see, e.g., [15, chapter 15]), where the size of such a structure is usually proportional to the number of entries in the FIB. For an algorithm ALG and time t, we denote the number of FIB rules at time t by $\text{SIZE}_t(\text{ALG})$. The total memory cost in a time interval I is then defined as $\text{M-COST}_I(\text{ALG}) = \int_I \text{SIZE}_t(\text{ALG})\, dt$.

In both objective functions (U-COST and M-COST), we drop time interval subscripts when referring to the total cost during the runtime of an algorithm. This paper focuses on minimizing the sum of these two costs, i.e., $\text{COST}(\text{ALG}) = \text{U-COST}(\text{ALG}) + \text{M-COST}(\text{ALG})$. Note that the parameter α can be used to put more emphasis on either of the two costs.

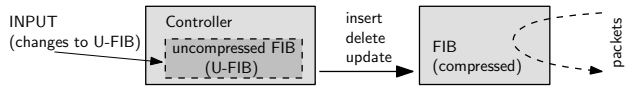

Figure 1. Controller and FIB: the controller updates the rules in the FIB. This paper focuses on online algorithms for the controller.

We assume a conservative standpoint and study algorithms that do not have any knowledge of future prefix changes, and need to decide *online* on where and when to aggregate. Not relying on predictions seems to be a reasonable assumption considering the chaotic behavior of the route updates in the modern Internet [9]. We use the standard yard-stick of online analysis [3], i.e., we compare the cost of the online algorithm to the cost of an optimal offline algorithm OPT which knows the whole input sequence in advance. We call an online algorithm ALG *ρ-competitive* if there exists a constant γ, such that for any input sequence it holds that $\text{COST}(\text{ALG}) \leq \rho \cdot \text{COST}(\text{OPT}) + \gamma$. The competitive ratio of an algorithm is the *infimum* over all possible ρ, such that the algorithm is ρ-competitive.

A Note about IP Lookup. In our modeling, we do not take into account the impact a FIB compression may have on IP lookup times, because they are affected only to a very limited extent. The state-of-the-art data structures used for IP lookup (see, [15, chapter 15] and the references therein) use a large variety of tree-like constructs augmented with additional information. This allows for lookup times of order $O(\log w)$, with practical implementations using 2-3 memory lookups on average. Unfortunately, little is known about proprietary data structures actually used in the routers of different vendors.

B. Related Work

There are known fast algorithms for optimal FIB aggregation of table snapshots, for example the *Optimal Routing Table Constructor (ORTC)* [6] and others [16], [19]. However, as these algorithms are static and do not support the efficient handling of incremental updates, a re-computation of the optimally aggregated FIB on each forwarding rule change is needed. This is computationally expensive and can lead to high churn.

There are several papers that deal with this problem by proposing heuristics that simultaneously try to limit the number of updates to the FIB while maintaining a good compression rate, including SMALTA [20], FIFA [10], and others [8], [11], [13], [21]. Moreover, some authors even proposed to only store a *subset* of rules in the FIB, leveraging Zipf's law [18]. However, none of these works give a formal bound on the achievable performance over time neither with respect to the number of updates to the aggregated FIB, nor

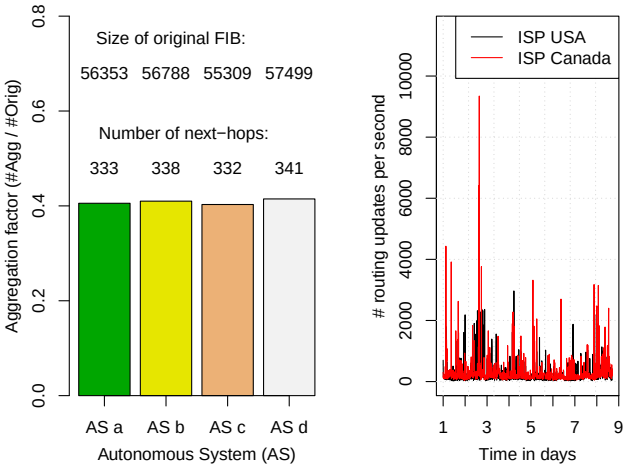

Figure 2. *Left:* Benefit of aggregation for four different AS: The ratio of aggregated FIB size divided by original FIB size is typically around 40% only. *Right:* Routing updates per second over a week.

to the aggregation gain. They also do not consider to exploit the (temporal and spatial) locality of churn for their benefit.

The paper closest to ours is [2]. The authors study online algorithms for FIB compression under the assumption of independent prefixes (both in U-FIB and in FIB). Without prefix dependencies, the nature of the problem is more related to online ski rental and technically different: achieving a constant competitive ratio is simple, but what the optimal constant is remains an open question. The authors present a 3.603-competitive solution.

C. Our Contribution

This paper presents the first formal study of the tradeoff between FIB compression and update churn under dependent prefixes. In particular, we present the online algorithm HIMS (HIDE INVISIBLE AND MERGE SIBLING), which is based on the concept of *sticks*. Sticks capture the subset of prefixes that are subject to optimization without violating forwarding correctness. HIMS (1) removes unnecessary and "invisible" prefixes from the FIB, and (2) merges FIB prefixes that are forwarded to the same port and describe adjacent IP address spaces.

We rigorously prove that HIMS achieves a competitive ratio of $O(w)$; the performance is hence independent of the update cost α (which need not to be a constant). Furthermore, we derive a tight lower bound of $\Omega(w)$ on the competitive (resp. approximation) ratio of a natural class of online (resp. offline) algorithms.

D. Empirical Motivation

To motivate the churn-aware compression of router forwarding tables, we gathered some empirical data. Precisely speaking, we have collected snapshots of the FIBs of seven different *Autonomous Systems (AS)* peering at a large European IXP (Internet Exchange Point). Figure 2 (*left*) shows the potential of aggregation: the ratio between the aggregated FIB size and the original FIB size. (In order to compute the optimally aggregated FIBs, i.e., the FIBs of minimal size with equivalent forwarding, we used the ORTC [6] dynamic programming algorithm.) The gain is typically larger than a factor of two, even for routers with a large number of ports (i.e., colors). This benefit is mostly due to the specific prefix distribution (over the ports) in the Internet; if the ports for an actual routing table were chosen randomly, the compression factor would be much worse.

At the same time, routers also have to deal with a high degree of update churn [7]. Our analysis of public traces from the RouteViews project [1] shows that there are very frequent routing events that cause more than a thousand routing table updates per second. Figure 2 (*right*) depicts the number of updates per second at two routers, one of an ISP in Canada and of an ISP in the USA.

For more empirical data as well as an evaluation of various heuristics under update streams, we refer the reader to the technical report [17].

II. BASIC CONCEPTS

In this section, we introduce the basic aggregation concepts and terminology on which our algorithm HIMS is based, such as *superfluous nodes* and *sticks*.

Trie Representation. Throughout this paper, we represent both the U-FIB and the FIB as one-bit tries containing all the prefixes from the forwarding rules. This affects merely the description: we do not assume anything about the actual implementation of the U-FIB/FIB structures. Each node of the tree (corresponding to some prefix p) has an associated *color* c if there is a forwarding rule (p, c); a node without any associated color is called *blank*. We identify nodes with the prefixes they represent and furthermore with the address ranges their prefix implies. In particular, we call two nodes *adjacent* if the address ranges they cover are adjacent.

We call a node v a U-FIB (FIB) *rule* if it is colored in the U-FIB (in the FIB). For any node v (also a blank one), we denote its *least colored ancestor* (the ancestor farthest from the root) in the U-FIB and in the FIB by $\text{lca}_\text{U}(v)$ and $\text{lca}_\text{F}(v)$, respectively.

We assume that each non-leaf node has exactly two children. A non-root node we call *left (right)* if it is a left (respectively right) son of its parent. For U-FIB, we assume minimal tries, that is, tries without blank sibling leaves (they may contain blank leaves, though). In the trie representation of the FIB, minimality is not assumed.

Color Determination and Superfluous Nodes. Observe that the coloring of the U-FIB (FIB) implies the coloring of the whole address space $[0, 2^w - 1]$: each address has the color of the prefix that would be applied as a forwarding rule.

We say that such a node v *determines* the color of address j in the U-FIB (FIB). Unlike in the U-FIB, the node that determines the color of a given address in the FIB may change with time. For succinctness of the description, we slightly extend the address space, incorporating two blank addresses -1 and 2^w.

We call a U-FIB rule that does not determine the color of any address *superfluous*. As the color changes of superfluous nodes can be ignored, w.l.o.g., we may assume that the U-FIB does not contain any such nodes: they could be removed from the FIB by an algorithm at the very beginning and at constant cost. We hence have the following property.

Observation II.1. *Any input event (i.e., a color change) changes the coloring of the address space, and hence any algorithm has to react by modifying the* FIB *and paying at least* α.

The necessity of dependencies in the FIB. To illustrate our definitions, we now explain why keeping dependent prefixes in the FIB is crucial for obtaining a good competitive ratio. It would be tempting to consider a simple online algorithm that just observes the coloring of the address space induced by the current U-FIB rules and tries to reflect that state using only independent prefixes in the FIB. It appears that no such algorithm can achieve a non-trivial competitive ratio.

Lemma II.2. *Let* ALG *be a (possibly offline) algorithm that never keeps any dependent prefixes in its* FIB. *Then, there exists an input sequence, for which* $\textsc{Cost}(\textsc{Alg}) = \Omega(2^w) \cdot \textsc{Cost}(\textsc{Opt})$.

Proof: Consider a U-FIB that is represented by a full binary tree of height w, with root colored initially green and each second leaf colored black. All other nodes are blank. Now the input sequence contains changes of the root color from green to red and back to green. By simply copying the state of the U-FIB to its FIB, OPT pays α for each change to the U-FIB. On the other hand, each change of the U-FIB induces $\Omega(2^w)$ changes to the coloring of the address space. To reflect these changes in its FIB, each time ALG has to pay $\Omega(2^w) \cdot \alpha$. Hence, $\text{U-}\textsc{Cost}(\textsc{Alg}) = \Omega(2^w) \cdot \text{U-}\textsc{Cost}(\textsc{Opt})$. This relation directly implies the desired bound as by frequently changing the U-FIB, the adversary renders the memory costs negligible. ∎

Sticks and Active Nodes. After removing superfluous nodes, we decompose the U-FIB into multiple groups called *sticks*. This decomposition remains invariant throughout the runtime of the algorithm. Informally speaking, each stick is a maximal subtree (where leaves of this subtree may be U-FIB internal nodes), such that all subtree leaves are colored and all its internal nodes are blank, cf. Figure 3.

To this end, we first group all rules of the U-FIB into sets $L_1, L_2, L_3, \ldots$. Each L_i is a maximal (w.r.t. to cardinality) set of colored nodes corresponding to adjacent address

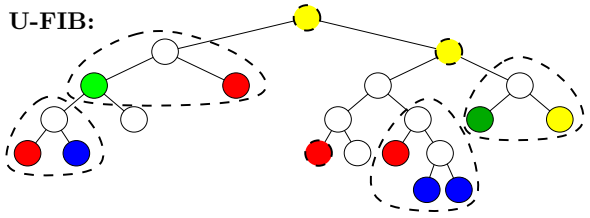

Figure 3. Partition of the U-FIB into sticks. Superfluous nodes are already removed. Stick boundaries are marked with dashed lines.

ranges whose union is a range that can be represented by a single node v_i. In other words, if all these nodes were of the same color c, they could be compressed to a single node v_i of color c. Note that this partitioning does not depend on the order in which we gather nodes into sets L_i.

A stick S_i is then defined to contain all nodes "between v_i and L_i inclusively": all nodes in the tree rooted at v_i that are either in L_i or are ancestors of L_i. L_i, $S_i \setminus L_i$, and v_i are called the leaves, the internal nodes and the root of stick S_i, respectively. Note that the stick leaves are not necessarily tree leaves. When L_i is a singleton, S_i is also a singleton and is called a *trivial* stick. As U-FIB does not contain superfluous nodes, all sticks are disjoint and all internal nodes of a stick are blank in the U-FIB. We call nodes that belong to any stick *active*.

III. Aggregation Algorithm HIMS

Our algorithm tries to merge nodes of the same color within a single stick. That is, if all the leaves of a single non-trivial stick have the same color (in the U-FIB) for some period of time, then in the FIB they should become blank and the root of the stick should have that color assigned. Furthermore, we should perform also partial intra-stick optimizations when possible (e.g., if some adjacent nodes of a single stick are of the same color and can be replaced by a single colored node). However, optimizing the sticks alone may still yield a poor performance. Consider, for example, a U-FIB containing a non-superfluous red root (being a trivial stick) and many non-adjacent red leaves, all being trivial sticks. In this example, the red nodes below the root are "invisible", i.e., they could be deleted without changing the meaning of the forwarding table. The optimally compressed FIB contains only the root.

Counters. For our algorithm, we define, for any active node, two counters that are functions of time and depend on the coloring of the U-FIB. Fix any (active) node u belonging to some stick S. If u is a leaf of S, then let $L(u) = \{u\}$, otherwise let $L(u)$ contain all leaves of S that are descendants of u. Furthermore, if u is not a root of a stick, $p(u)$ denotes its parent in the trie, otherwise $p(u)$ is undefined.

1) For any node u, the *counter* $C_u(t)$ measures how long, until time t (uninterruptedly), all nodes of $L(u)$ have

the same color. Hence, for a stick leaf u, $C_u(t)$ simply measures the time since the last change of u's color.

2) The second counter is used to hide invisible nodes. Assume that $\text{lca}_\text{U}(u)$ exists. The *counter* $H_u(t)$ measures how long, until time t (uninterruptedly), all nodes of $L(u) \cup \{\text{lca}_\text{U}(u)\}$ have the same color. When $\text{lca}_\text{U}(u)$ does not exist, $H_u(t) = 0$ for any time t.

Since multiple nodes cannot change colors simultaneously, any color change of a stick leaf u causes the resetting of the C and H counters on the path from u to the root of a stick containing u. Similarly, the color change of $\text{lca}_\text{U}(u)$ resets all H counters from the stick containing u. Note also that $C_u(t) \geq H_u(t)$ and, if $p(u)$ is defined, $C_{p(u)}(t) \leq C_u(t)$ and $H_{p(u)}(t) \leq H_u(t)$.

Algorithm Definition. We are now ready to present our algorithm HIMS (HIDE INVISIBLE AND MERGE SIBLING). In the FIB of HIMS, the inactive nodes are always blank. For any active node u, HIMS decides whether it should be colored and, if so, with which color.

An active node u is a FIB rule at time t if and only if all the following three conditions hold:

1) $H_u(t) < \alpha$,
2) $C_u(t) \geq \alpha$ or u is a stick leaf,
3) $C_{p(u)}(t) < \alpha$ or u is a stick root.

If u is decided to be FIB rule, then its color is the color of the nodes in $L(u)$. Note that this color is well defined (either u is a leaf and then $L(u)$ is a singleton, or $C_u(t) > 0$).

Example HIMS Execution. To get some understanding of the behavior of HIMS, let us take a look at two extreme cases. For a trivial stick consisting of a single node u, the second and the third condition always hold as u is both a stick leaf and a stick root. Therefore, the algorithm simply waits till $H_u(t)$ reaches α and then removes (the invisible) u from the FIB. On the other hand, for a stick S that has no colored ancestors in the U-FIB, $H_u(t) = 0$ for any $u \in S$. Thus, the actions of HIMS on S depend only on the C counters inside S. For example, if all the stick leaves in the U-FIB are unicolor for time α, then only the root of S remains present in the FIB of HIMS.

IV. ANALYSIS OF HIMS

We start by presenting the framework of our analysis, while most of the technical details are given in the subsequent sections. To explain the rationale behind bounding the memory cost of HIMS, let us consider a special adversarial strategy: given the state of the U-FIB at time t, the adversary does not change anything for a certain time period. Then, in the time interval $(t, t + \alpha]$, HIMS may perform some optimizations, but after time $t+\alpha$, HIMS will not introduce any further changes to the FIB. Furthermore, it is possible to show that at time $t + \alpha$, the size of the algorithm's FIB is an $O(w)$-approximation of the optimal FIB size.

Clearly, we cannot expect the adversary to behave as described above, as it has many more options. Nevertheless, we can show that if we take the U-FIB and FIB snapshots at any particular time t, then either some compressions were already performed by HIMS or the changes to the U-FIB are quite recent, i.e., they occurred in time interval $(t-\alpha, t]$. In either case, we are able to construct a lower bound on OPT's cost, and hence relate M-COST(HIMS) to COST(OPT).

For a formal proof, we introduce a concept of *rainbow points*. A rainbow point is an address-time pair (a, t), denoting that at time t address $a \in [-1, 2^w - 1]$ has a different color than address $a + 1$ (where blank is treated as an additional color). We call two rainbow points *different* if their addresses are different. Rainbow points measure the spatial-temporal complexity of the coloring of the address space: even OPT has to represent this coloring by its own FIB.

Lemma IV.1. *If there are k pairwise different rainbow points in some time interval I of length α, then* $\text{COST}_I(\text{OPT}) \geq \lceil k/2 \rceil \cdot \alpha$.

Proof: A rainbow point (a, t) is a witness for a rule that had to be present at time t in the OPT's FIB and whose range either ended with address a (at the right) or $a + 1$ (at the left). Therefore, k different rainbow points are the witnesses of at least $\lceil k/2 \rceil$ distinct rules that were present in the FIB at some times from I. Any such rule was either present in the FIB throughout the whole interval I or it was inserted or deleted at some time of I. In either case, such a rule contributes α to $\text{COST}_I(\text{OPT})$. ■

It remains to show how to find sufficiently many rainbow points: We just need to consider the snapshots of HIMS's FIB at certain times. Precisely speaking, in Section IV-A, we will show the following result.

Lemma IV.2. *Fix any time t at which the* FIB *of* HIMS *does not change. There are $\Omega(\text{SIZE}_t(\text{HIMS})/w)$ pairwise different rainbow points in the interval $(t - \alpha, t]$.*

Finally, we need to bound the number of updates HIMS performs in the FIB. Using a potential function argument, we charge each FIB update either to a change of the U-FIB or to a time period of length at least α this rule spent in the FIB. By assuring that no U-FIB update is charged more than $O(w)$ times, we obtain the following result (proven formally in Section IV-B).

Lemma IV.3. *For any input sequence with m color changes,* $\text{U-COST}(\text{HIMS}) = O(\text{M-COST}(\text{HIMS})) + O(w) \cdot m \cdot \alpha + O(\alpha) \cdot \text{SIZE}(\text{U-FIB})$.

Theorem IV.4. HIMS *is $O(w)$-competitive.*

Proof: First, we bound M-COST(HIMS). We partition

the entire runtime of the algorithm into disjoint intervals $I_1, I_2, \ldots, I_\ell$ of length α. At any such interval I_j, we identify a time $t_j \in I_j$ at which there is no change in the FIB and the size of the FIB of HIMS is the greatest; let $k_j = \text{SIZE}_{t_j}(\text{HIMS})$. Clearly, $\text{M-COST}_{I_j}(\text{HIMS}) \leq k_j \cdot \alpha$.

For any $j \geq 2$, let $r_j = (t_j - \alpha, t_j]$. The number of rainbow points in interval r_j is $\Omega(k_j/w)$ by Lemma IV.2, and thus, by Lemma IV.1, $\text{COST}_{r_j}(\text{OPT}) = \Omega(k_j \cdot \alpha/w)$. The intervals r_j may overlap, however any time belongs to at most two such intervals (as the distance between every second t_j is at least α). Thus, $\sum_{j=2}^{\ell} \text{M-COST}_{I_j}(\text{HIMS}) \leq \sum_{j=2}^{\ell} k_j \cdot \alpha = O(w) \cdot \sum_{j=2}^{\ell} \text{COST}_{r_j}(\text{OPT}) = O(w) \cdot \frac{1}{2} \cdot \text{COST}(\text{OPT})$. Finally, the memory cost in the first interval is at most $k_1 \cdot \alpha \leq \alpha \cdot \text{SIZE}(\text{U-FIB})$, and hence $\text{M-COST}(\text{HIMS}) = O(w) \cdot \text{COST}(\text{OPT}) + \alpha \cdot \text{SIZE}(\text{U-FIB})$.

Now by Lemma IV.13 and Observation II.1, $\text{U-COST}(\text{HIMS}) = O(\text{M-COST}(\text{HIMS})) + O(w) \cdot \text{COST}(\text{OPT}) + O(\alpha) \cdot \text{SIZE}(\text{U-FIB})$. Thus, in total, $\text{COST}(\text{HIMS}) = \text{M-COST}(\text{HIMS}) + \text{U-COST}(\text{HIMS}) = O(w) \cdot \text{COST}(\text{OPT}) + O(\alpha) \cdot \text{SIZE}(\text{U-FIB})$. As the term $O(\alpha) \cdot \text{SIZE}(\text{U-FIB})$ is a constant independent of the input sequence, HIMS is $O(w)$-competitive. ∎

A. Finding Rainbow Points (Proof of Lemma IV.2)

We start with some basic properties of the algorithm HIMS. We call a node u that satisfies the second and the third condition given in the description of HIMS a (FIB) *semi-rule*. If a semi-rule u satisfies also the first condition, it is clearly a FIB rule, otherwise we call it (FIB) *hidden rule*.

Claim IV.5. *Fix any stick S. For any stick leaf u, let A_u contain all the nodes on the path from u to the stick root. At any time t, A_u contains exactly one semi-rule. If $H_u(t) < \alpha$, then A_u contains exactly one* FIB *rule.*

Proof: Consider the sequence $u_1 = u, u_2, \ldots, u_s$ of all nodes of A_u sorted from the leaf u of S to the root of S. The first part of the lemma follows in a straightforward manner by observing that the values of $C_{u_i}(t)$ are non-increasing with i. As the $H_{u_i}(t)$ values are also non-increasing with i, $H_u(t) < \alpha$ implies that $H_{u_i}(t) < \alpha$ for any i, and therefore the only semi-rule in A_u is in fact a rule. ∎

Claim IV.6. *Fix time t. Fix a stick leaf node u that changes color in the* U-FIB *at time $t' \in (t - \alpha, t)$. Fix an address a contained in the range of u. If at time t, in the* FIB *of* HIMS*, the color of a is determined by u or its ancestor, then in the* U-FIB *the color of a is determined by u.*

Proof: For the sake of contradiction, assume that in the U-FIB there exist descendants $u_1, u_2, \ldots, u_s$ of u containing a. Let S_1 be the stick containing u_1; clearly, u belongs to a different stick than S_1. As u changes color at t', $H_{u_1}(t') = 0$, and therefore $H_{u_1}(t) \leq t - t' < \alpha$. Then, by Claim IV.5, either u_1 or one of its ancestors from S_1 is present as a colored node in the FIB. As such a node lies below u in the FIB trie, neither u nor any of its ancestors can determine the color of a in the FIB at time t. ∎

Relating pairs of FIB rules to rainbow points. The following lemma captures the core properties of the optimizations performed by HIMS, essentially stating that if at some time the FIB contains two "neighboring" nodes, then either they cannot be aggregated by HIMS at all, or it was not possible to aggregate them in the nearest past. In either case, we provide a witness (a rainbow point).

Lemma IV.7. *Fix any time t at which the* FIB *of* HIMS *does not change and an address a. Assume that at time t the colors of a and $a+1$ are determined in the* FIB *by two distinct rules u and v, respectively. Assume that one of the following three cases occurs: (i) u and v are siblings; (ii) u is a left node and v is its ancestor; (iii) v is a right node and u is its ancestor. Then, there exists a rainbow point (a, t'), for some $t' \in (t - \alpha, t]$.*

Proof: We assume that addresses a and $a+1$ have the same color c at time t, as otherwise they would immediately constitute the rainbow point (a, t). We consider the three cases listed in the lemma assumptions and show that in either case U-FIB contains a stick leaf x that changes color at a time $t_c \in (t - \alpha, t)$, such that x is either u, or v, or their descendant, and contains either a or $a+1$ (but not both).

1) If u and v are siblings, then — by the way we defined sticks — they belong to the same stick. This implies that their parent p also belongs to the same stick. Hence, by the definition of HIMS, $C_p(t) < \alpha$. This means that (i) at time t, all nodes from $L(p) = L(u) \cup L(v)$ are of the same color c, (ii) at time $t_c = t - C_p(t)$ one of these nodes changed its color from c'; let x be this node. Without loss of generality, assume that $x \in L(u)$. This implies that $C_u(t) = C_p(t) < \alpha$. As u is a FIB rule at time t, u has to be a stick leaf and hence $x = u$.
2) If v is an ancestor of u, then by Claim IV.5, they belong to different sticks. Let $v' = \text{lca}_{\text{U}}(u)$. Node v' is either equal to v or is its descendant. Note that v' is a stick leaf. As u is a left node, v' contains the address $a+1$. As u is a FIB rule at time t, $H_u(t) < \alpha$, i.e., there is a node $x \in L(u) \cup \{v'\}$ that changed color at time $t_c = t - H_u(t) \in (t - \alpha, t)$. It remains to show that x fulfills our requirements. It is clearly the case when $L(u) = \{u\}$, because then x can be then either v' or u. Assume now that $L(u)$ is not a singleton set, i.e., u is not a stick leaf. As u is a FIB rule at time t, $C_u(t) \geq \alpha$, which implies that no node from $L(u)$ changed the color during the time interval $(t - \alpha, t)$. Thus, in this case $x = v'$.
3) If u is an ancestor of v, the argument is symmetric to the previous case.

By Claim IV.6, x determines the color either of a or $a+1$ in the U-FIB. Without loss of generality, assume that it determines the color of a and that at time $t_c > t - \alpha$ it changes color from c' to c. This means that there exists a sufficiently small $\epsilon > 0$, such that $(t_c - \epsilon, t_c + \epsilon) \subset (t - \alpha, t)$ where a has color c' in time interval $(t_c - \epsilon, t_c)$ and color c in $(t_c, t_c + \epsilon)$. As the color of $a + 1$ is not determined by node x and there are no simultaneous changes of colors, there is a time $t' \in (t_c - \epsilon, t + \epsilon) \subset (t - \alpha, t)$ when addresses a and $a+1$ have different colors: (a, t') is our desired rainbow point. ∎

Relating global FIB state to rainbow points. To show Lemma IV.2, we fix time t and perform the following grouping of the leaves of the FIB of HIMS. We sweep the leaves from left to right, partitioning them into groups $G_1, G_2, G_3, \ldots$ In the grouping process, we put two consecutive leaves u, v (possibly representing non-adjacent address ranges) into the same group G_i when either (i) both u and v are left nodes and v is a descendant of the right sibling of u, or (ii) both u and v are right nodes and u is a descendant of the left sibling of v. In the former case, we call a group *left*, in the latter — *right*. (Note that a group can consist of a single leaf only if the orientation of leaves change, but can also have up to w many members.)

Claim IV.8. *If there are h groups of leaves in the* FIB*, then the number of all* FIB *rules is $O(h \cdot w)$.*

Proof: For any group G_i, we denote by K_i the set of all leaves of G_i plus the union of their (not necessarily colored) ancestors. As each FIB rule is in at least one set K_i, it is sufficient to show that the number of elements of any set K_i is at most $O(w)$. Recall that, by the definition of G_i, K_i has at most one leaf on each level; let ℓ be the maximal such level. It suffices to show that there is exactly one ancestor on each of the levels $0, 1, \ldots, \ell - 1$. Such a claim follows by a simple backward induction on the levels. Level $\ell - 1$ contains exactly one ancestor being the parent of the ℓ-th level leaf of K_i. Now fix level $j < \ell - 1$. Note that level $j+1$ contains a single ancestor (by the inductive assumption) and possibly a leaf, and these nodes are siblings (by the construction of the groups). Hence these nodes of K_i have a single parent at level j, which concludes the proof of the claim. ∎

We are now ready to prove Lemma IV.2, i.e., the relation between the number of FIB entries at time t and the number of different rainbow points in the interval $(t - \alpha, t]$.

Proof of Lemma IV.2: Let $I = (t - \alpha, t]$. We group all the leaves as described above into h groups $G_1, G_2, \ldots, G_h$. For any group G_i, we denote the leftmost address covered by a leaf from G_i by a_i and its rightmost one by b_i. By Claim IV.8, it is sufficient to show that the number of rainbow points is $\Omega(h)$. If $h < 4$, then we simply consider the last colored address, b_h: as $b_h + 1$ is blank, (b_h, t) is a rainbow point and the lemma follows.

In the following, we thus assume that $h \geq 4$. We focus on a consecutive pair of groups G_i and G_{i+1}, such that at least one of the conditions hold: (i) G_i is a left group, (ii) G_{i+1} is a right group. Note that among all $h - 1$ pairs of consecutive groups, at least every second pair (i.e., at least $(h-2)/2 = \Omega(h)$ pairs) has this property. Thus, it remains to show that for such a pair of groups, we may find a unique rainbow point in I.

We denote the rightmost leaf of G_i by v_i and the leftmost leaf of G_{i+1} by v_{i+1}. Without loss of generality, we can assume that G_i is a left group, which means that v_i is a left node. (The case when G_{i+1} is a right group is symmetric, i.e., we start our construction with v_{i+1} and we reverse the roles of left and right nodes). If $b_i + 1$ is blank, then (b_i, t) is our rainbow point; otherwise let u_i be the node that determines the color of the address $b_i + 1$. We consider three cases depending on the relation between the levels (i.e., depth in the trie) of u_i and v_i, henceforth referred to by $\text{lev}(u_i)$ and $\text{lev}(v_i)$, respectively.

1) $\text{lev}(u_i) < \text{lev}(v_i)$. As v_i is a left node, the address ranges of v_i and u_i cannot be adjacent, and therefore u_i is an ancestor of v_i. By Lemma IV.7, there exists a rainbow point (b_i, t_i), where $t_i \in I$.
2) $\text{lev}(u_i) = \text{lev}(v_i)$. Then, u_i is the right sibling of v_i. By Lemma IV.7, there exists a rainbow point (b_i, t_i), where $t_i \in I$.
3) $\text{lev}(u_i) > \text{lev}(v_i)$. Then, u_i is a left node, whose leftmost address is $b_i + 1$. Note that u_i cannot be a FIB leaf as then it would belong to G_i. Furthermore, v_{i+1} is the leftmost leaf of the subtree rooted at u_i, i.e., $\text{lev}(v_{i+1}) > \text{lev}(u_i) > \text{lev}(v_i)$. This implies that v_{i+1} has to be a right node as otherwise it would belong to G_i. Furthermore, v_{i+1} has an ancestor (node u_i) in the FIB. Let $u_{i+1} = \text{lca}_{\text{F}}(v_{i+1})$ (it can be either u_i or some of its descendants). As v_{i+1} is a right node and is the leftmost leaf of u_{i+1}, node u_{i+1} determines the color of $a_{i+1} - 1$. Hence, by Lemma IV.7, there exists a rainbow point $(a_{i+1} - 1, t_i)$, where $t_i \in I$.

∎

B. Bounding the Update Cost (Proof of Lemma IV.13)

We bound the update cost over an input sequence using amortized analysis. For any node u, we define its potential at time t as

$$F_u(t) = \begin{cases} 5\alpha + 2 \cdot \min\{H_u(t), \alpha\} & \text{if } u \text{ is a FIB rule} \\ 6\alpha & \text{if } u \text{ is a FIB hidden rule} \\ 0 & \text{otherwise} \end{cases}$$

Let the total potential at time t be defined as $\Phi(t) = \sum_u F_u(t)$, where we sum over all (active) nodes from the FIB trie.

In this section, we abuse the notation, and use U-COST(HIMS) to denote the *amortized* cost of its updates,

defined as the actual cost plus the change in the potential. We show how to bound this amount in all possible cases.

Lemma IV.9. *For any time interval $I = (t_0, t_1)$ with no updates to the* FIB*, it holds that* U-COST$_I$(HIMS) $\leq 2 \cdot$ M-COST$_I$(HIMS).

Proof: There is no actual update cost. The increase of the total potential is $\Delta\Phi = \Phi(t_1) - \Phi(t_0) \leq 2 \cdot (t_1 - t_0) \cdot k$, where k is the number of FIB rules kept within I. Finally, M-COST$_I$(HIMS) $= (t_1 - t_0) \cdot k$, and thus the lemma follows. ∎

Now, we analyze the amortized update cost at any time t when the FIB is updated by HIMS. Such update is caused either by some counters reaching α or by a color change in the U-FIB that resets some counters. For the analysis, we assume that these events occur separately. Namely, we split time t into three stages:

1) In the first stage, HIMS behaves as if there was no U-FIB color update, and processes only the changes caused by some H counters that reached α.
2) In the second stage, HIMS processes the changes induced by some C counters that reached α.
3) In the third stage, HIMS processes a (single) event of a color change in the U-FIB.

We bound U-COST(HIMS) in all stages separately (see the three lemmas below). Note that it is possible that some stages are missing, and furthermore, if more than one stage is present, we possibly overestimate the cost of HIMS (for example, we may charge it for inserting a rule because of a change in C counters and then for its removal because of the U-FIB color change, while in reality HIMS would do nothing with this rule).

Lemma IV.10. *First stage: Assume that some H counters reach value α. Then,* U-COST(HIMS) ≤ 0.

Proof: If a counter H_u of a node u reaches α then if this node was a rule it becomes a hidden rule. (If it was not a rule, its status remains unchanged.) HIMS removes u from the FIB paying α and the change in the potential is $6\alpha - (5\alpha + 2 \cdot \min\{H_u, \alpha\}) = -\alpha$, i.e., the amortized update cost associated with u is zero. The lemma follows by summing the amortized cost over all affected nodes u. ∎

Lemma IV.11. *Second stage: Assume that some C counters reach value α. Then,* U-COST(HIMS) ≤ 0.

Proof: By the definition of HIMS, growing C counters can only create a semi-rule that is an ancestor of a previously existing semi-rules. Thus, if a new semi-rule u is created, then $\ell \geq 2$ semi-rules (lying in the subtree rooted at u) are removed. These actions costs at most $(\ell + 1) \cdot \alpha$ (the cost might be lower than $(\ell + 1) \cdot \alpha$, because there is no actual cost involved in creating and removing hidden rules). On the other hand, as potentials for semi-rules are always between 5α and 7α, the change in the total potential is at most $7\alpha - \ell \cdot 5\alpha$. Therefore, the amortized cost is at most $(\ell + 1 + 7 - 5\ell) \cdot \alpha = (8 - 4\ell) \cdot \alpha \leq 0$. By applying this reasoning to all newly created semi-rules, the proof follows. ∎

Lemma IV.12. *Third stage: If a node u changes its color in the* U-FIB*, then* U-COST(HIMS) $= O(w \cdot \alpha)$.

Proof: By the stick definition, u is a stick leaf. Its color change affects two group of nodes.

First, pick any node v, such that $\text{lca}_{\text{U}}(v) = u$. Nodes v with this property are active nodes belonging to sticks that are "immediately below" u, i.e., there are no sticks between u and them. The H counter of any such node v is reset to zero. If v was a hidden rule, then it becomes reinserted to the FIB (as a FIB rule). Otherwise, v remains unchanged. In the former case, the actual cost associated with v is α, while the potential change is $5\alpha + 2 \cdot \min\{H_v, \alpha\} - 6\alpha = -\alpha$. Thus, in total, the amortized cost is zero.

Second, the change of color of u also causes all the C and H counters on the path from u to the root of the stick containing u (inclusively) to be reset to zero. By the definition of HIMS, only the nodes on this path and their children are affected, i.e., only those nodes may be inserted, deleted and have their potential changed. As there are $O(w)$ such nodes and their change in the potential is at most 7α, the total amortized cost is $O(w \cdot \alpha)$. ∎

Now we may combine the lemmas above to bound the amortized cost of HIMS updates in the general case.

Lemma IV.13. *For any input sequence with m color changes,* U-COST(HIMS) $= O($M-COST(HIMS)$) + O(w) \cdot m \cdot \alpha + O(\alpha) \cdot$ SIZE(U-FIB).

Proof of Lemma IV.13: Let m be the number of color changes in the input. If we sum the guarantees of Lemma IV.9 to Lemma IV.12, we obtain that the total amortized cost is bounded by $O($M-COST(HIMS)$) + O(w \cdot \alpha \cdot m)$. Finally, we observe that the initial potential is $5\alpha \cdot$ SIZE$_0$(HIMS) $= 5\alpha \cdot$ SIZE(U-FIB), which contributes a constant $O(\alpha) \cdot$ SIZE(U-FIB) to the total cost. ∎

C. Lower Bound

We can show that Theorem IV.4 is the best we can hope for, at least for the natural class a of so-called *stick-based* algorithms: informally speaking, these algorithms do not create dependent prefixes within a single stick. We will derive an $\Omega(w)$ lower bound for such (online or offline!) algorithms.

More formally, we consider the U-FIB without superfluous nodes (recall that without loss of generality, we may assume that an algorithm removes them at the very beginning). Then, we call an algorithm *stick-based* if (i) it

never keeps an inactive node (a node outside of a stick) in the FIB, and (ii) for any two active nodes from a single stick that are in a ancestor-descendant relation, it keeps at most one of them in the FIB. Clearly, HIMS fulfills these properties and is hence an instance of a stick-based algorithm.

Theorem IV.14. *The competitive ratio of any stick-based algorithm* ALG *(even an offline one) is* $\Omega(w)$.

Proof: It is sufficient to consider a U-FIB containing a single stick S with $w+1$ leaves, corresponding to adjacent address ranges of lengths $2^{w-1}, 2^{w-2}, 2^{w-3}, \ldots, 2^2, 2^1, 2^0, 2^0$, i.e. each length except for the last one occurs exactly once. The root of S coincides with the root of the whole trie. The coloring of S is constant: the first w leaves are always black and the last one is red. In this case, ALG has to use at least $w+1$ entries in the FIB to represent such a U-FIB. On the other hand, OPT could just keep two entries in the FIB: the last red entry from U-FIB of length 2^0 and the black root (of length 2^w). Note that in the long run, the initial update cost of OPT becomes negligible and thus $\text{COST}(\text{ALG}) \geq \text{M-COST}(\text{ALG}) = \Omega(w) \cdot \text{OPT}$. ■

V. IMPLEMENTING HIMS

In this section, we show that HIMS can be implemented efficiently in the route processor. Our construction consists of two stages. First, we show how to maintain a data structure that at all times keeps the set of all semi-rules. (Recall that a semi-rule is a rule which satisfies the second and the third property given in the algorithm definition.) Additionally, for each semi-rule u, we store its color (i.e., the color of leaves of $L(u)$.) Second, we show how to augment this data structure, so that at any time we know which of the semi-rules are rules and should be kept in the FIB.

Lemma V.1. *It is possible to maintain the set of semi-rules (in the route processor) using a data structure of size* $O(\text{SIZE}(\text{U-FIB}))$*, so that any sequence of events to* U-FIB *can be processed in time* $O(w)$ *on average.*

Proof: For maintaining the set of all semi-rules, we need to track two types of events: (i) a U-FIB color change may force some C counters to be reset to zero and (ii) some counters may reach the value of α. (Note that H counters are irrelevant for computing semi-rules.) Any such change in the counter of a node v affects the state (i.e., being or not being a semi-rule) of at most three nodes: v itself and if v is not a stick leaf, then also its two children. Thus, for a single event, we may update our semi-rules set in constant time.

We first observe that the number of active nodes (for which we want to track the C counters) is at most twice the number of U-FIB rules. Second, instead of storing counters C_u, we just keep the timestamps of the last reset of C_u to zero. These values are sorted in non-increasing order, kept in a linked queue Q with additional references from and to the nodes of the U-FIB trie. Finally, we store a pointer q to the place in Q that splits Q into two parts: the left one keeping counters strictly smaller than α and the right one with counters that are at least α.

When a counter is reset to zero, it is moved to the front of the list with its timestamp updated to the current time. For tracking the second type of events, we set an alarm to the time when the first counter would reach α. (This is the counter immediately to the left of q.) When the alarm goes off, we shift q to the left accordingly and set the alarm for the next element. It is possible that many counters, say k, reach α simultaneously, in which case we shift q by k positions. By using a standard amortization argument, we may assign a constant cost to counter resets and zero cost to the event where a counter reaches α.

Finally, we observe that the resetting of counters can only occur when there is a color change (of a stick leaf u) in the U-FIB. Such a change affects at most $w+1$ counters of the nodes on a path from u to the root of the stick u belongs to. Thus, the total cost of maintaining the structure is at most $O(w)$ times larger than the number of color changes in the input sequence. ■

To compute the set of rules instead of semi-rules, one could try to keep H counters in a similar data structure. However, a color change of a single node may affect H counters of virtually all possible nodes. (Consider the U-FIB described in Lemma II.2, where the root frequently changes color from red to black, and back.) Below, we show that while sometimes we indeed need to make a lot of updates, their number can be asymptotically bounded by the number of updates to the FIB.

Theorem V.2. HIMS *can implemented in the route processor using a data structure of size* $O(\text{SIZE}(\text{U-FIB}))$*, so that any sequence of* m_1 *events to the* U-FIB *that entails* m_2 *updates to the* FIB *can be processed in expected time* $O(m_1 \cdot w + m_2)$.

Proof: We want to augment the data structure described in Lemma V.1. By the definition of HIMS, we know that a semi-rule is *not* a FIB rule if and only if $H_u \geq \alpha$, which is equivalent to $C(\text{lca}_\text{U}(u)) \geq \alpha$ and the color of $\text{lca}_\text{U}(u)$ being the same as the color of u. These conditions can be checked in constant time and when the semi-rule is created. When a semi-rule is deleted (and was a rule), it is also removed from the FIB. Below, we show that we may also keep track when a node which is uninterruptedly a semi-rule starts or ceases to be a FIB rule.

To this end, for any node v and color c, we keep the set of all semi-rules u of color c, such that $\text{lca}_\text{U}(u) = v$. We denote such a set by $P(v, c)$, and we denote the union of these sets over possible colors by $P(v)$. To maintain these sets, for each semi-rule u we keep a bidirectional pointer

to $\text{lca}_U(u)$. To keep the memory requirement asymptotically the same as for storing semi-rules only, we just store those sets $P(v,c)$ that are non-empty. This can be achieved by keeping a hashing table for each node v, whose keys are colors c and values are non-empty sets $P(v,c)$.

Now, whenever the counter of v is reset to zero or reaches α, we may easily enumerate those semi-rules (and only them) that need to be added to or removed from the FIB. Namely, when $C_v \geq \alpha$ and v has color b, all semi-rules from $P(v,b)$ are hidden rules and all semi-rules from $P(v) \setminus P(v,b)$ are FIB rules. When $C_v < \alpha$, all semi-rules from $P(v)$ are FIB rules. Thus, the additional time overhead for modifying the data structure is proportional to the number of FIB updates. ■

VI. Open Questions

Our work opens several interesting directions for future research. First, obviously, the optimality of our competitive ratio only holds for a restricted class of algorithms, and it will be interesting to generalize the lower bound or prove that this is not possible. Another open question regards the design of *offline algorithms*: while it is quite easy to see that under certain circumstances, e.g., when there can be at most one color change per unit time, the problem is *fixed parameter tractable* (i.e., optimal solutions can be computed in time $f(\alpha) \cdot n^{O(1)}$ where n denotes the number of prefixes and f is a function of α), it remains an open question whether a polynomial time algorithm exists.

Acknowledgments

We thank Fred Baker from Cisco/IETF and Magnús M. Halldórsson from Reykjavik Uni for their valuable feedback. Marcin Bienkowski was supported by the Polish National Science Centre grant DEC-2013/09/B/ST6/01538, Stefan Schmid by the EIT ICT project 'Mobile SDN'.

References

[1] Routeviews project, 2013. http://www.routeviews.org/.

[2] M. Bienkowski and S. Schmid. Competitive FIB aggregation for independent prefixes: Online ski rental on the trie. In *Proc. of the 20th Colloq. on Structural Information and Communication Complexity (SIROCCO)*, pages 92–103, 2013.

[3] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[4] T. Bu, L. Gao, and D. Towsley. On characterizing BGP routing table growth. *Computer Networks*, 45:45–54, 2004.

[5] L. Cittadini, W. Muhlbauer, S. Uhlig, R. Bush, P. Francois, and O. Maennel. Evolution of internet address space deaggregation: myths and reality. *IEEE Journal on Selected Areas in Communications*, 28:1238–1249, 2010.

[6] R. P. Draves, C. King, S. Venkatachary, and B. D. Zill. Constructing optimal IP routing tables. In *Proc. of the IEEE INFOCOM*, pages 88–97, 1999.

[7] A. Elmokashfi, A. Kvalbein, and C. Dovrolis. BGP churn evolution: a perspective from the core. *IEEE/ACM Trans. on Networking*, 20(2):571–584, 2012.

[8] E. Karpilovsky, M. Caesar, J. R. amnd Aman Shaikh, and J. E. van der Merwe. Practical network-wide compression of IP routing tables. *IEEE Transactions on Network and Service Management*, 9:446–458, 2012.

[9] J. Li, M. Guidero, Z. Wu, E. Purpus, and T. Ehrenkranz. BGP routing dynamics revisited. *SIGCOMM Computer Communication Review*, 37:5–16, 2007.

[10] Y. Liu, B. Zhang, and L. Wang. Fast incremental fib aggregation. In *Proc. of the IEEE INFOCOM*, pages 1213–1221, 2013.

[11] Y. Liu, X. Zhao, K. Nam, L. Wang, and B. Zhang. Incremental forwarding table aggregation. In *Proc. Global Communications Conference (GLOBECOM)*, pages 1–6, 2010.

[12] L. Luo, G. Xie, S. Uhlig, L. Mathy, K. Salamatian, and Y. Xie. Towards TCAM-based scalable virtual routers. In *Proc. of the 8th Int. Conf. on Emerging Networking Experiments and Technologies (CoNEXT)*, pages 73–84, 2012.

[13] L. Luo, G. Xie, S. Uhlig, L. Mathy, K. Salamatian, and Y. Xie. A trie merging approach with incremental updates for virtual routers. In *Proc. of the IEEE INFOCOM*, pages 1222–1230, 2013.

[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Computer Communication Review*, 38:69–74, 2008.

[15] D. Medhi and K. Ramasamy. *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann Publishers Inc., 2007.

[16] G. Rétvári, J. Tapolcai, A. Kőrösi, A. Majdán, and Z. Heszberger. Compressing IP forwarding tables: Towards entropy bounds and beyond. In *Proc. of the ACM SIGCOMM*, pages 111–122, 2013.

[17] N. Sarrar, M. Bienkowski, S. Schmid, S. Uhlig, and R. Wuttke. Exploiting locality of churn for FIB aggregation. Number TR 2012/12, TU Berlin, 2012.

[18] N. Sarrar, S. Uhlig, A. Feldmann, R. Sherwood, and X. Huang. Leveraging Zipf's law for traffic offloading. *SIGCOMM Computer Communication Review*, 42(1):16–22, 2012.

[19] S. Suri, T. Sandholm, and P. R. Warkhede. Compressing two-dimensional routing tables. *Algorithmica*, 35(4):287–300, 2003.

[20] Z. A. Uzmi, M. Nebel, A. Tariq, S. Jawad, R. Chen, A. Shaikh, J. Wang, and P. Francis. SMALTA: Practical and near-optimal FIB aggregation. In *Proc. of the Int. Conf. on Emerging Networking Experiments and Technologies (CoNEXT)*, 2011.

[21] X. Zhao, Y. Liu, L. Wang, and B. Zhang. On the aggregatability of router forwarding tables. In *Proc. of the 29th IEEE INFOCOM*, pages 848–856, 2010.