Constructing Optimal Non-overlap Routing Tables

Tong Yang, Ting Zhang, Shenjiang Zhang and Bin Liu*
Dept. of Computer Science and Technology, Tsinghua University, Beijing China

Abstract—**The size of routing tables has been growing rapidly, while the link transmission speed of Internet backbone has increased up to 100Gbps commercially and towards 400Gbps Ethernet for laboratory experiments. In order to alleviate the pressure from both the huge large routing table and very high interface speed, ISPs are trying to find ways to compress the table while striving to design a more powerful lookup engine. To address this issue, we propose an algorithm, named Optimal Non-overlap Routing Table Constructor (ONRTC), to compute an equivalent routing table with a minimal number of prefixes under the constraint that all the prefixes are not overlapped. Experimental evaluations show that, for large backbone routing tables, the ONRTC algorithm requires only about 71% of the original number of prefixes. We release ONRTC's source code in [10].**

I. INTRODUCTION

Internet has maintained a rapid growth for years, which brings two main issues: a) due to a roughly 15% [1] increase per year of routing table size, ISPs struggle to suppress the table growth, so as to further postpone the requirement for upgrading the infrastructure; b) to handle hundreds of gigabit-per-second traffic, the backbone routers must be able to forward hundreds of millions of packets per second, thus bringing huge pressure to routing lookup.

Current solutions to address these issues can be divided into two categories: the first category only strives to solve the routing table growth problem, while the second only focus on the routing lookup. ORTC [2] and 4-level [3] algorithm etc., belong to the first category, while the [4-8] belong to the second.

An ideal goal is to achieve both the good routing table compression and the fast packet lookup. To achieve this, a big obstacle lies in the prefix overlap, due to the introduction of CIDR. Prefix overlap refers to some prefixes are a part of others. This brings many difficulties when operating routing lookup, especially when TCAM-based solution is adopted. A detailed analyze is given in the following.

1) Layout in TCAM: Ternary Content Addressable Memories (TCAMs) [6] are fully associative memories that allow a "*don't care*" state to be stored in each memory cell in addition to *0*s and *1*s. One TCAM access can finish one routing lookup operation. Due to the overlap, prefixes must be ordered by the prefixes length when being stored in TCAM chips. Meanwhile, a priority encoder is indispensible.

2) Handling Update: when updates occur, due to the prefix overlap, many prefixes might need to move, which is called domino effect in this paper. Although some algorithms manage to resist this kind of domino effect, but redundancy will be introduced.

3) Power Consumption: the main shortcoming of TCAM is its high power consumption. Partitioning the routing table and fitting them into TCAM's buckets is an effective solution to save power. Because only one partition of TCAM works for each packet lookup, thus the power consumption can be significantly reduced. There are several partition algorithms, such as ID-bit partition [6, 7] and sub-tree partition [8]. ID-bit partition cannot split the table evenly. Sub-tree partition works better, but it will introduce redundancy.

If the prefix overlap is eliminated, these issues of TCAM-based scheme can be well addressed: 1) prefixes can be stored in TCAM arbitrarily; 2) the priority encoder in TCAM is no longer needed. This not only reduces hardware cost, but also decreases the TCAM lookup latency; 3) domino effect will be avoided; 4) TCAM partition can be strictly even without redundancy.

Therefore, overlap elimination is very important. There are several approaches to reduce or eliminate overlap, such as [4] and [5]. In [4], the routing table is divided into two parts: the overlapping part and the non-overlapping part. This approach can only reduce the number of overlapped prefixes, but cannot totally avoid the overlap. As far as we know, only the leaf-pushing algorithm proposed in [5] can totally eliminate the prefix overlap. Unfortunately, leaf-pushing algorithm causes routing table expanding too much to be tolerant.

We propose ONRTC algorithm, which constructs optimal non-overlap routing tables. Specifically, we achieve the following major contributions:

- We propose ONRTC algorithm as well as its incremental update algorithm.
- Extensive experiments on twelve real backbone routing tables and an 18-month long real update data are conducted to evaluate the performance of ONRTC.

The remaining parts of this paper are organized as follows. Section II surveys the related work. Section III presents ONRTC algorithm and Section IV illustrates its fast incremental update algorithm. The mechanism of routing lookup based on TCAM is described in Section V. Section VI conducts extensive experiments over a large real data set, and finally we conclude our paper in Section VII.

II. RELATED WORK

As mentioned above, two problems are involved in this paper: 1) routing table scalability; 2) fast routing lookup.

Some solutions focused on the routing table scalability problem by using Forwarding Information Base (FIB) compression. Draves et al. proposed the most famous ORTC algorithm [2] to construct optimal IP routing tables. In [3], Xin Zhao et al. presented 4-level algorithm at the cost of changing the forwarding behavior of routers.

*Corresponding author: liub@tsinghua.edu.cn.
Others: yangtongemail@gmail.com, zhangting_825@yahoo.cn, zhangshenjiang @gmail.com. Supported by NSFC (61073171, 60873250), Tsinghua University Initiative Scientific Research Program, the Specialized Research Fund for the Doctoral Program of Higher Education of China (20100002110051).

With respect to routing lookup, TCAM has been widely used, and the typical papers are CoolCAMs [6], Kai's TCAM-parallel lookup algorithm [7] and Dong's TCAM-cache algorithm [8]. But these solutions do not eliminate overlap, thus face a lot of challenges during the TCAM lookup and update, such as redundancy introduction, domino effect, etc.

Therefore, if overlap is eliminated, the performance of TCAM-based solution will be significantly improved. In [4], the routing table is divided into two parts: the overlapping part and the non-overlapping part. This approach can only reduce overlap. The controlled prefix expansion algorithm [5] proposed a disjoint tree by using leaf-pushing. However, its massive new born prefixes make it impractical to the current routers and have to control the prefix expansion, which doesn't eliminate overlap totally. Therefore, we proposed ONRTC to construct optimal non-overlap routing tables.

III. ONRTC ALGORITHM

A. *Terms and Definitions*

The following terms will be used in this paper, so their definitions are given in Table I.

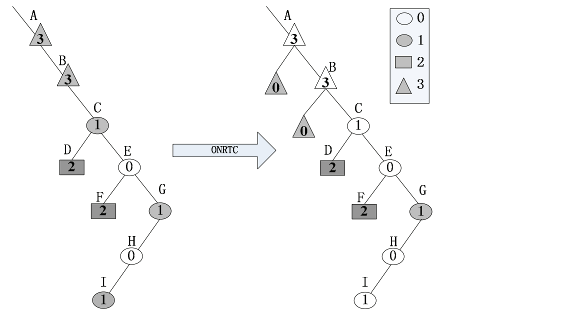
TABLE I. TERMS AND DEFINITIONS

Terms	Definitions
FIB size	the number of FIB entries
Oldport	the next hop of an entry in FIB before compressed
Newport	the next hop of an entry in FIB after compressed
Insertport	the next hop of the update message in the operation of insertion and changing
Default-oldport	the next hop of the nearest and non-empty ancestor node before compressed
Oldport/Newport	the next hop of a prefix in FIB before and after compression

As shown in Figure 1, each node has two next hops: Oldport/Newport. Newport is represented by the shape, Oldport is represented by the number in the node (such as 0 in ▲0), and the hollow node (such as ①) means that its Newport is 0. For convenience, three next hops are introduced: solid ellipse, solid rectangle, and solid triangle, representing the Newport of 1, 2, and 3, respectively. For example, the shape of ▲3 is triangle, so its Newport is 3; ⓪ is represented by 0/0, indicating its Oldport is 0, and Newport is 0.

B. ONRTC Algorithm

1) *An Example*



a) The original trie b) the compress trie compressed by ONRTC

Figure 1. An example of ONRTC.

In order to make a clear picture of ONRTC algorithm, an example is given in Figure 1. Figure 1(a) is the original trie, and Figure 1(b) is the compressed trie. In this paper, compression means the number of solid node (FIB size) decreases, because usually only solid nodes needs to be stored in fast memory. This example shows FIB size of the compressed trie by ONRTC is 5, while the original is 7.

ONRTC algorithm derives from Election and Representative (EAR) algorithm, which is proposed in [10]. Although EAR algorithm produces the minimal Routing Tables, it doesn't eliminate overlap. Under the constraint of Non-overlap, EAR is evolved into ONRTC. Both EAR and ONRTC follow a process which is similar to the election process of the democratic society. Each node owns a next hop, while each candidate has a vote. Actually, any candidate's next hop can be selected as representative. However, all the nodes which own the same next hop with the representative can be deleted. Therefore, in order to achieve optimal compression, the most popular next hop should be chosen, in other words, should be elected as representative. This is the rationale of EAR and ONRTC.

2) *Election and Representative*

Both ONRTC and EAR consist of two basic operations, named "election" and "representative". "Representative" operation is executed after a successful "election" operation immediately. Those nodes (candidate nodes) participating in elections must meet the following requirements: they are 1) solid or hollow; 2) siblings (if a node has no sibling node, a sibling node must be created with the next hop of 0/Default-oldport); 3) elected representatives (If not, the point must be traced to a leaf node in the sub-tree rooted at the unelected node, then recursive election should be done step by step).

Election: two or more nodes elect their common ancestor node, under the constraint that no solid node appears in the path from the candidate nodes to the common ancestor node. The most popular node will be elected as representative, and then the common ancestor's next hop will be replaced by the representative's next hop. If the most popular node is not only one, election fails. At this moment, the common ancestor's next hop will be set to zero, and then the common ancestor will participate in the next round of election.

Representative: after a successful election, the common ancestor node will exercise the right of representative immediately: the Newport of its voters which own the same next hop with representative is set to 0.

3) *Atomic Equivalent Models of ONRTC Algorithm*

TABLE II. NODE'SATTRIBUTES

single-node attributes (Category 1)				**two-node attributes (Category 2)**	
the first attribute		***the second attribute***			
solid	hollow	has got brother	no brother	own the same hop	own different hops

As shown in Table II, in order to cover all possible situations, candidate nodes' attributes are classified into two categories. According to these attributes, all atomic election models can be enumerated.

Category 1: single-node election. In this case, an electing node has no brother. If the node is solid, model 1 emerges. If the node is hollow, model 2 emerges.

Model 1: as shown in Figure 2(a), node A has no brother. According to the requirements of election, at least two nodes are needed, so node B and C are created with 0/Default-oldport. Then node D is set to Oldport/0.

Model 2: as shown in Figure 2(b), the election and representative process is similar to model 1.

Category 2: according to the two-node attributes, four models emerge.

Model 3: as shown in Figure 2(c), node A is solid and B is hollow. Node A and the sub-tree rooted at node B participate in the election. In this case, just set C to Oldport/0.

Model 4: as shown in Figure 2(d), both A and B are hollow. The two sub-trees rooted at node A and B participate in the election. In this case, just set C to Oldport/0.

Model 5: as shown in Figure 2(e), both A and B are solid, and own the same Newport, so the common Newport is elected as the Newport of node C.

Model 6: as shown in Figure 2(f), both A and B are solid, but own different Newport. In this case, just set C's Newport to zero.

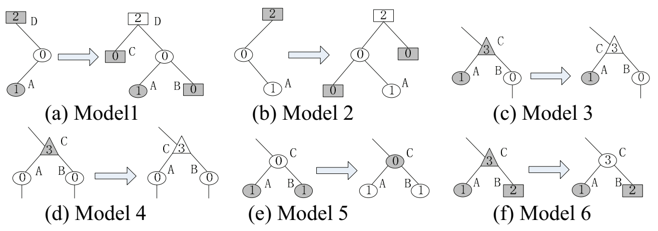


Figure 2. ONRTC atomic equivalent models.

Any trie can be compressed fast according to these six models, which covers all election situations. Although the models seem to be complicated, the whole algorithm requires only one post-order traversal, which is of high efficiency.

4) Mathmatical Proof of the Equivalent Models

To ensure the correctness of the six equivalent models, we have finished mathematical proof. Due to space limitation, the details are given in [10]. Because ONRTC only traverses the trie once, its time complexity is *O(n)* (n is the nodes number of the trie).

IV. Fast Incremental Update Algorithm

A. Updating Metrics While Applying FIB Compression

When updates occur, incremental update algorithm should run in partial range as fast as possible under the constraint of all the prefixes are not overlapped. Then how to evaluate the performance of incremental update algorithm? Two metrics: Time to Fresh (TTF) and re-compression interval, are defined.

TTF refers to the average computing time of updating a message. It indicates a router's sensitivity to the changes of the network. The smaller the TTF is, the more sensitive the router will be. If no compression algorithm is adopted, TTF is minimal, and is regarded as ground-truth.

During the process of re-compression, the router can't conduct routing lookup based on the newest FIB. The interval between the two adjacent events of re-compressing the whole routing table is called "re-compression interval". Our objective is to make re-compression time shorter and re-compression interval longer.

For incremental update, one important issue is worth being mentioned here. In order to achieve fast update, we need to confine the update scope, and visit as few nodes as possible. Unlike ORTC and 4-level algorithm, the update scope of ONRTC is not easy to confine. The constraint of non-overlap makes the scope can hardly be the sub-tree rooted at the update node. If the update node has an ancestor node whose Oldport is not NULL, then the scope should be the sub-tree rooted at the nearest ancestor. Otherwise, just update the sub-tree rooted at the update node. Updating a sub-tree is equivalent to compressing a sub-tree. Therefore, the faster the compression algorithm runs, the faster the update algorithm runs.

B. Update Algorithm

There are two kinds of update messages: announcement and withdrawal, which can be further divided into "insertion", "changing" and "deletion" operation. ONRTC's incremental algorithm is divided into three steps:

1) Lookup the Prefix in the Trie

When an update message arrives, update algorithm firstly locates the prefix in the trie. Sometimes the prefix doesn't exist. In this case, if the type of updating message is "announcement", update algorithm must create a path to the update node; and if its type is "withdrawal", it means to delete a node which doesn't exist, so algorithm ends. In this step, the NOT NULL ancestor node should be recorded if it exists.

2) Refresh the Update Node

After located in the trie, the update node should be refreshed according to the update operation.

3) Update the Subtree

The process of updating a sub-tree is compressing the sub-tree with ONRTC. This step spends much more time than the first two. Therefore, the faster the compression algorithm runs, the faster the update algorithm runs.

V. Lookup Based on TCAM

Because of the so many advantages which ONRTC brings, our TCAM scheme is divided into the following three steps:

1) Step 1: Even Partition. After compression, suppose the number of prefixes is M, and TCAM is divided into N buckets. Our scheme traverses the trie recursively from the root node in inorder or preorder, and groups the prefixes with the number of M/N. In this way, all prefixes are divided into N parts evenly.

2) Step II: after even partition, just put the prefixes in every bucket arbitrarily.

3) Step III: Set N-1 Registers.

When looking up a prefix, in order to locate the corresponding bucket, every bucket needs two registers which store the boundary points. Therefore, N buckets means N-1 registers.

Through the above three steps, the power consumption is only 1/N of the original TCAM. When updates occur, as mentioned above, domino effect never happens. The update process is divided into two steps:

1) Step1: Update the Trie

This update process is illustrated in detail above, and it is simple and fast.

2) Step II: Update the TCAM

The update of TCAM is very simple. During an insertion operation, a prefix will be compared with the boundary points and be copied to any empty place of the corresponding bucket. With respect to changing and deletion operation, just locate the corresponding bucket, and update the corresponding prefix.

VI. Evaluation of ONRTC Algorithm

A. Experimental Settings

1) Data Set

The routing tables are taken from www.ripe.net [9] at RIPE NCC, which collects default free routing updates from peers. In order to evaluate the performance of ONRTC algorithm in an objective and complete way, the RIB packets at 8:00 on August 8 in 2011 from 12 routers are selected (There are 16 routers' tables available in www.ripe.net, but other four routers don't update to present). In addition, the routing tables of rrc01 over recent 12 months are also selected to evaluate ONRTC algorithm. Table III shows these routers' locations.

TABLE III. Locations of Routers

ID	Location	ID	Location
rrc00	RIPE NCC, Amsterdam	rrc11	New York (NY), USA
rrc01	LINX, London	rrc12	Frankfurt, Germany
rrc03	AMS-IX, Amsterdam	rrc13	Moscow, Russia
rrc04	CIXP, Geneva	rrc14	Palo Alto, USA
rrc05	VIX, Vienna	rrc15	Sao Paulo, Brazil
rrc06	Otemachi, Japan	rrc16	Miami, USA
rrc07	Stockholm, Sweden		

With regard to the update experiments, two group of data set are selected. Firstly, in order to measure TTF, the update data from 2011.01.01/08:00 to 2011.01.02/08:00 is selected. Secondly, in order to test the re-compression interval, the update data over the recent 18 months from 2009.12 to 2011.05, which is about 40GBytes, is downloaded and parsed.

With regard to the partition experiment, the routing tables at 8:00 on August 8 in 2011 from rrc01, rrc03, and rrc04, are selected to evaluate the three partition algorithms.

2) Computer Configuration

Our experiments have been done on a windows XP sp3 machine with Pentium (R) Dual-Core CPU 5500@2.80GHz and 4G Memory.

B. Experiments on FIB Compression

The compressed FIB size and the original table size are shown in Figure 3 and Figure 4. Figure 3 shows the compression results of 12 routers, while Figure 4 shows the compression results of the recent 12 months on rrc01.

In Figure 3, the taller histogram is the original FIB size, the lower histogram is the FIB size after compression by ONRTC algorithm, and the curve is the compression time of ONRTC algorithm, and the unit is microsecond. It can be observed that all compressed routing tables are about 70% of the original tables, and the compression time is 39 milliseconds. Similar with Figure 3, Figure 4 shows the FIB size and compression time over the recent 12 months on rrc01, and the result is similar.

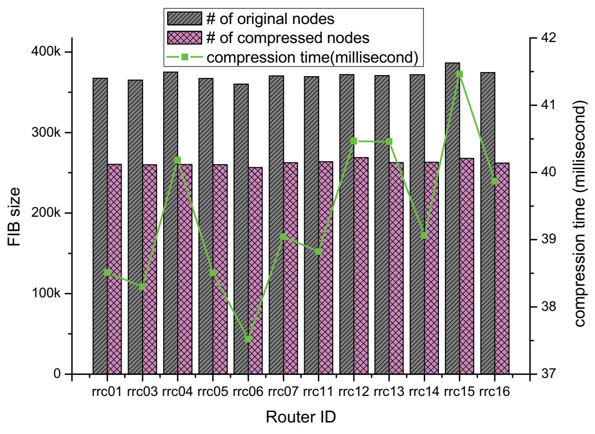


Figure 3. FIB size before and after compression and compression time on 12 routers.

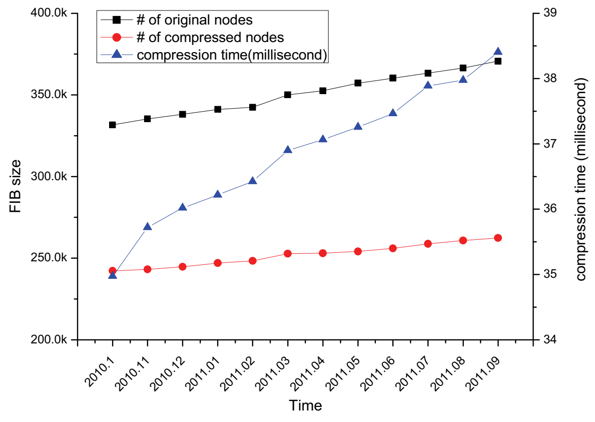


Figure 4. FIB size and compression time over 12 months on rrc01.

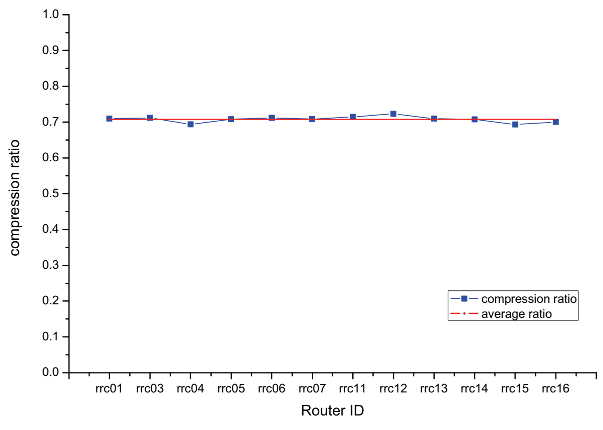


Figure 5. Compression ratio over 12 routers.

The compression ratios of 12 routers are shown in Figure 5. Compression ratio is defined as the ratio of the nodes number in compressed trie to that of the original trie. Results show that the compression ratios are between 0.7232 and 0.6934 with a mean of 0.7077.

C. *Experiments of Fast Incremental FIB Updating*

The x-axis of Figure 6 and 7 means the time when the update messages arrive. For example, 201010231945 means the time of 2010.10.10/23:19:45. The two update metrics (TTF and re-compression interval) are evaluated by the following two experiments.

1) *Experiment I: Update Experiment over a Day*

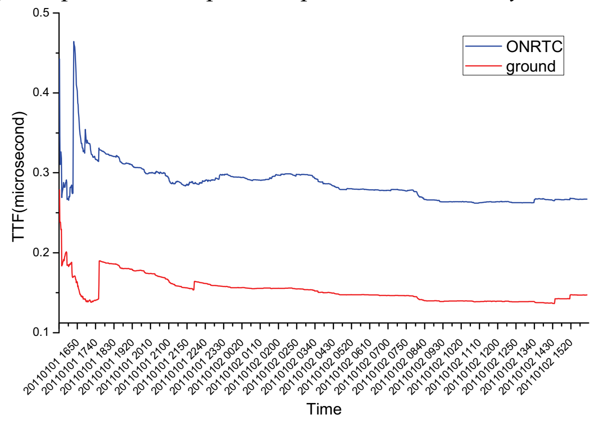


Figure 6. TTF comparison between ONRTC and groud-truth.

Figure 6 shows TTF of ONRTC and ground-truth from 2011.01.01/08:00 to 2011.01.02/08:00. It can be observed that TTF of ONRTC is a little bigger than ground-truth. TTF of ONRTC ranges from 0.2621 microseconds to 0.4642 microseconds with a mean of 0.2864 microseconds. It means that ONRTC algorithm can handle 3.49 million updates per second in average on a common computer. According to our data mining results, even in the worst case, the updates messages of the backbone routers are only 35K per second at most. Therefore, the update algorithm of ONRTC is fast enough to handle the current updates.

2) *Experiment II: Update Experiment over 18 Months*

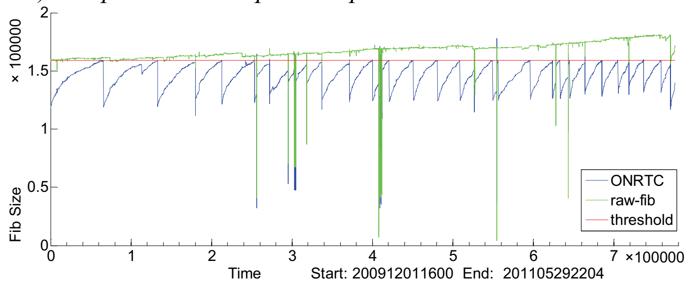


Figure 7. The growing stability of the FIB size with re-compression over a continous time span of 18 months.

Suppose the FIB size is just the same as the memory size on a line card on 2009.12.01/08:00, which is considered as the threshold. In order to evaluate the re-compression interval of ONRTC update algorithm, we plot the size of the routing table from 2009.12.01/08:00 to 2011.05.01/08:00. In Figure 7, the top curve, which is called raw-fib, is the FIB size without compression. This figure shows ONRTC re-compresses 409 times in the 18 months. It means routers only need re-compress once in 1.32 days in average, which costs about 30 microseconds, and this is an easy task for a router.

D. *Experiment of TCAM Partition*

Figure 8 shows the partition results among three algorithms: Kai's partition algorithm, Dong's sub-tree partition algorithm and ours. This experiment are finished in the 12 routers, only 2 routers' results are shown here, because the results are similar and there are not enough space. It can be seen that Kai's algorithm cannon split the prefixes evenly, and Dong's algorithm split the prefixes evenly at the cost of some redundancy. In contrast, ONRTC algorithm splits the prefixes strictly evenly without redundancy, and puts much fewer prefixes in one bucket than Kai's and Dong's. If more parts are split, Kai's algorithm and Dong's algorithm will introduce more redundancy (see Figure 6 in [8]), while our algorithm still introduces no redundancy.

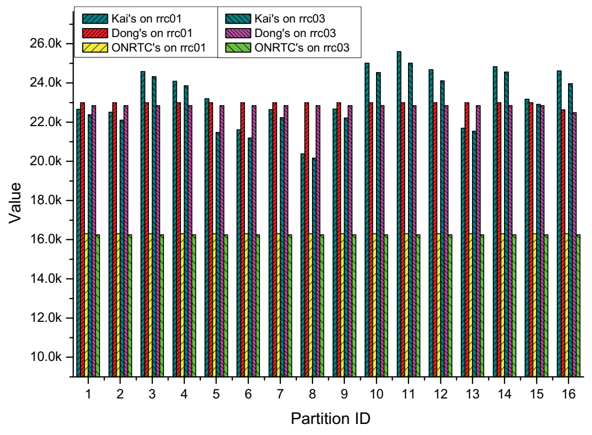


Figure 8. partition comparison among three algorithms.

VII. CONCLUSION

In this paper we have presented ONRTC algorithm to construct optimal non-overlap routing tables. Experiments on twelve real backbone routing tables show that ONRTC can achieve 71% compression ratio in average.

REFERENCES

[1] X. Meng, Z. Xu, B. Zhang, G. Huston, S. Lu, and L. Zhang, IPv4 Address Allocation and the BGP Routing Table. ACM SIGCOMM Computer Communication Review, vol. 35, pp. 71–80, January 2005.

[2] R.Draves, C.King, S.Venkatachary, and B.D.Zill. Constructing Optimal IP Routing Tables. In Proc. IEEE INFOCOM, 1999.

[3] X. Zhao, Y. Liu, L. Wang, and B. Zhang. On the Aggregatability of Router Forwarding Tables. In Proc. IEEE INFOCOM, 2010.

[4] Bin Zhang, Jiahai Yang, Jianping Wu, Qi Li, Donghong Qin. An Efficient Parallel TCAM Scheme for the Forwarding Engine of the Next-generation Router. In Proc. IFIP/IEEE IM, 2011.

[5] V. Srinivasan and G. Varghese, Fast IP lookups using controlled prefix expansion, ACM TOCS, vol. 17, pp. 1–40, Feb. 1999.

[6] F. Zane, G. Narlikar, A. Basu, CoolCAMs: Power-Efficient TCAMs for Forwarding Engines, In Proc. INFOCOM, 2003.

[7] Zheng, K., Hu, C., Lu, H., Liu, B.: A TCAM-based distributed parallel IP lookup scheme and performance analysis. IEEE/ACM Trans. Netw. 14, 863–875, 2006.

[8] Lin, D., Zhang, Y., Hu, C., Liu, B., Zhang, X., Pao, D.: Route Table Partitioning and Load Balancing for Parallel Searching with TCAMs. In Proc. IPDPS, 2007.

[9] RIPE Network Coordination Centre. http://www.ripe.net/data-tools/stats/ris/ris-raw-data.

[10] Routing Table Compression and Update Website. http://s-router.cs.tsinghua.edu.cn/~yangtong