Computer Networks 59 (2014) 227–243

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Efficient Two Dimensional-IP routing: An incremental deployment design

Mingwei Xu [a], Shu Yang [a,*], Dan Wang [b], Jianping Wu [a]

[a] Department of Computer Science and Technology, Tsinghua University, and Tsinghua National Laboratory for Information Science and Technology, Beijing, China
[b] Department of Computing, The Hong Kong Polytechnic University, Hong Kong

ARTICLE INFO

Article history:
Received 21 May 2013
Received in revised form 12 October 2013
Accepted 4 November 2013
Available online 9 November 2013

Keywords:
Incremental deployment
High dimensional routing
Routing deployment

ABSTRACT

High dimensional routing has attracted more attentions to satisfy the increasing demands for more flexible services in the Internet. These routing schemes make routing decisions not only based on the destination address, but also on the source address, flow label, etc. With these schemes, networks can provide more than best-effort services.

There are several research issues towards high dimensional routing. Clearly routers face additional CPU and memory burden in looking up and maintaining the additional information. While some overheads are unavoidable, we need to minimize such burden incurred. A more important problem is the deployment. It is widely known that making changes to the network layer is notoriously difficult. The proposed scheme should have least impact on the current Internet protocols and infrastructure. A node-by-node incremental deployment scheme is highly preferred. Obviously, without full deployment, the resulting paths for traffic diversion may deviate from the pre-defined ones. The incremental deployment scheme should minimize such deviation.

In this paper, we illustrate the problem by using a real example from China Education and Research Network 2 (CERNET2). Then we formulate it as finding a deployment sequence where the traffic flows should follow the pre-defined paths given (1) the number of nodes to be deployed and (2) the extra burden each router can spare. We transform our problem to boolean clauses and develop efficient solutions following the Maximum Satisfiability (MAX-SAT) problem. We present several related algorithms for different practical scenarios.

We evaluate our algorithms using comprehensive simulations with BRITE generated topologies and real world topologies. We conduct a case study on CERNET2 configurations. Compared to an ad hoc deployment and an arbitrary TwoD-IP forwarding, our algorithms compute a deployment sequence that achieves close to optimal performance after deploying a few nodes. The CPU and memory requirement are small for packet forwarding.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Traditional networks make forwarding decisions independently at each node based on the destination address. The destination-based networks perform well for providing universal reachability services. However, with more users and enormous applications, simple best-effort services can not satisfy increasing user demands.

For example, multi-homing is prevalent and **P**rovider-**A**ggregatable (PA) address is recommended currently [1]. However, with destination-based routing, it is difficult to implement exit-routing that delivers the packets to the corresponding provider. Thus, communications may be interrupted due to ingress filtering deployed at the

* Corresponding author. Tel.: +86 (0)10 62785822; fax: +86 (0)10 6260364.
E-mail address: yangshu@csnet1.cs.tsinghua.edu.cn (S. Yang).

1389-1286/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.bjp.2013.11.004

upstream providers [2]. Other examples focus on security and performance, such as role-based access control in a multi-tenant data center, load balancing and policy routing.

High dimensional routing makes routing decisions not only based on the destination address, but also on the source address, Differentiated Services Code Point (DSCP) value and flow label. It adds more information into routing, and solves the fundamental problem of destination-based routing, i.e., lack of routing semantics.

Recent years have seen many high dimensional routing schemes. For example, policy-based routing (PBR) [3] uses access control list to implement policies set by network operators; multi-topology routing (MTR) [4] sets up multiple forwarding table to configure service differentiation. Openflow [5] uses wide flow table to enable flexible software-defined network. Class-based routing [6] proposes an Interior Gateway Protocol (IGP) based high dimensional routing protocol to facilitate exit-routing.

With an overall considerations on security, load balancing, policy routing, China Education and Research Network 2 (CERNET2), the world's largest pure-IPv6 backbone network (including 59 Giga-PoPs), is now deploying such a routing scheme called Two Dimensional-IP (TwoD-IP) routing[1] [8]. More specifically, the forwarding decisions of intermediate routers will be based not only on the destination address, but also the source address. Like other high dimensional routing schemes, the driving force of TwoD-IP is to let the networks have the ability to divert traffic flows (identified by their source and destination IPs and we call them the VIP flows) to pre-defined paths (we call them the VIP paths). CERNET2 has two international exchange centers connecting to the Internet, Beijing (CNGI-6IX) and Shanghai (CNGI-SHIX). For example, we find in operation that CNGI-6IX is very congested with an average throughput of 1.18 Gbps in February 2011; and CNGI-SHIX is much more spared with a maximal throughput of 8.3 Mbps at the same time.

During deploying, CERNET2 faces many common challenges of high dimensional routing schemes. Here, we use TwoD-IP as an example due to its simplicity and practical significance. Other high dimensional routing schemes can easily fit into this framework.

Clearly, each router faces additional processor burden in looking up the source IP and memory burden in maintaining additional source IP tables. A more important problem is the deployment of the TwoD-IP routing scheme, which requires upgrade of the CERNET2. It is widely known that making changes to the network layer is notoriously difficult. The proposed scheme should have least impact on the current Internet protocol stack and infrastructure. A node-by-node incremental deployment scheme is highly preferred.

Clearly, if only partial nodes are deployed, a VIP flow may not strictly follow its VIP path. Especially at the beginning of deployment, the ISPs only want to deploy a few routers for economic considerations. If some unimportant routers are deployed, the per-flow incremental routing can even become an impediment to a network designer.

For further incentives, the network designer needs to carefully choose the first batch of deployed routers, i.e., minimize the deviation between the pre-defined paths and real paths. In this paper, we define a deviation that is practically meaningful. We formulate a problem where we need to derive a deployment sequence and minimize the deviation given (1) the number of nodes to be deployed and (2) the spared processor capacity of each router that can be used for the source IP management; in other words, there is a constraint on the extra burden for each router.

We show that the problem is NP-complete by reducing it to a dense-k subgraph problem. We then transform our problem to boolean clauses and develop algorithms following the principle of the branch-and-bound algorithm for Maximum Satisfiability (MAX-SAT). We study several closely related problems for different practical scenarios. We develop efficient algorithms for these problems in a step by step manner. We first develop an algorithm to compute incremental deployment sequence without taking the router capacity into consideration. We then develop algorithms for the general capacity constrained problem. All these algorithms can be easily extended to other high dimensional routing schemes, e.g., with DSCP values and flow labels into consideration.

We conduct comprehensive simulations using BRITE generated topologies. We also evaluate our algorithms using the topology and 51,642 prefixes of China Education and Research Network (CERNET), a medium scale IPv4 network with 110 routers and 238 links. The results show that our algorithm can achieve close to optimal performance by deploying a few carefully selected nodes. Each router only needs to look up and maintain a small number of bits of the source IP addresses.

We carry out a case study with CERNET2 configuration and our primary concerned VIP traffic flows. We suggest the deployment sequence of TwoD-IP routing. By deploying 5 routers, we can successfully divert our concerned VIP traffic flows from the congested CNGI-6IX to CNGI-SHIX. Only 22 bits out of 128 bits in the source IP (CERNET2 is an IPv6 network) need to be looked up and maintained. We study the current burden of the CERNET2 routers, and show that this is a moderate overhead.

The rest of the paper proceeds as follows. We start by introducing the background and problem formulation in Section 2. Section 3 is devoted to the problem without capacity constraint. In Section 4, we take capacity constraint into account. We systematically evaluate our methods in Section 6. In Section 7, we study a real case from CERNET2. At last, we summarize the related work in Section 8, and conclude our paper in Section 9.

2. Background and the optimal incremental deployment problem

2.1. Background on TwoD-IP routing

We first give the background of the TwoD-IP routing in our context. Routers across the network will cooperate to

[1] It is also called source/destination routing by Cisco [7].

divert VIP traffic flows to pre-defined paths. The VIP traffic flows (such as specified traffic for policy routing, large-volumn traffic for load balancing, and sensitive traffic for security) are either selected by network operators or computed by in-network controllers.

In each router, there is a destination-IP forwarding table. Additionally, there is a source-IP forwarding table for each entry of a destination prefix.[2] There are multiple ways to implement this in a router. We show an example in Fig. 1 using linearly stored table, e.g., TCAM. Nevertheless, the table can be also structured as two dimensional tries [9], which builds a trie on the destination prefixes and hang each source tries on the leaves of the destination trie. And each trie can be either uni-bit trie or multi-bit trie, depending on the stride of each access in memory.

When a packet arrives at a router, the router maps the destination IP to a source-IP table. This is in contrast to the conventional IP routing, where the mapping leads to the next hop directly. Then the router reads the source IP of the packet and using the source-IP table, the next hop of this packet is obtained. Longest match first rule is used for both destination and source IP tables.

In the example of Fig. 1, a packet with a destination prefix of 00111 and source prefix of 11,000 will match 0011* in the destination-IP forwarding table, which will lead to a source-IP forwarding table. Then the source prefix matches 1100* in the source-IP forwarding table to get the next hop *a*.

We emphasize that this is only an illustration of the framework of TwoD-IP routing. The source IP table can be stored and retrieved in other ways or even omitted. For example, in Fig. 1, 0010* directly maps to next hop *b*. The updates of source-IP table can be manually configurable for registered VIP flows. Our current CERNET2 requirement of diverting a few traffic flows will end up in this way. In the future, when more source IP functionalities are needed and VIP traffic flows are more dynamic, an OSPF-like protocol can be developed. All these are important engineering details or require further research but they are out of the scope of this paper.

The overheads of TwoD-IP routing is the storage and the process of the forwarding operations on the source IP. These overheads can be abstracted as the number of bits that needs to be checked for the source IP. For software architecture routers, looking up less number of bits means less searches in the data structure (e.g., tries) and less storage space. For hardware architecture routers, since the width of each entry of the TCAM is configurable nowadays [10], less number of bits also means less storage space and power consumption [11]. In this paper, we focus on lookup performance because it is the key indicator in current over-provisioned ISPs [12].

2.2. The optimal incremental deployment problem

Let $G = (V, E)$ be a network, where V is the set of nodes, and E is the set of links. In this network we have multiple

[2] Some high dimensional routing schemes [7] use one source-IP forwarding table and multiple destination-IP forwarding tables, however, they are symmetric with our structure.

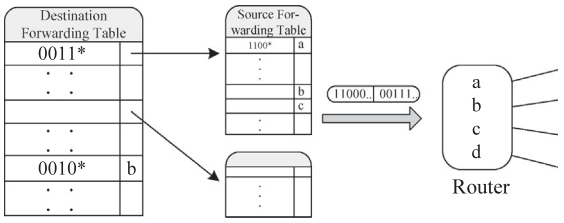


Fig. 1. Example of TwoD-IP routing.

VIP traffic flows. Let $\mathcal{T}$ denote the set of traffic flows, and $t \in \mathcal{T}$ be a VIP traffic flow. Let the source and destination prefixes of t be $P_s(t)$, $P_d(t)$, t can be represented as $\langle P_s(t), P_d(t)\rangle$. In this paper, we often omit the destination prefix, i.e., only use the source prefix to identify a flow if there is no ambiguity. For each t, the user expects it to travel on a pre-defined VIP path (which usually is not the shortest path). We use $VIP(t) = \{v_t^0, v_t^1, \ldots, v_t^j, \ldots\}$ to denote the VIP path for t.

For a destination IP prefix P_d (we omit the traffic t if there is no ambiguity), $P_d = (p_0^d p_1^d \ldots p_m^d)$. Here $p_i^d \in \{0, 1\}$ and $0 \leqslant m \leqslant 31$ for IPv4 and $0 \leqslant m \leqslant 127$ for IPv6. On a node, we call it *forwarding* an operation $\mathcal{I}^d(P_d)$ to map P_d to a set of next hops $\mathcal{I}^d(P_d) = \{a_0, a_1, \ldots, a_j, \ldots\}$, each of which can lead the packets to the destination, loop free (satisfying [13] to avoid routing loops). In conventional routing, the result of a forwarding $\mathcal{I}^d(P_d)$ is a single next hop on the shortest path.

For a source IP prefix, $P_s = (p_0^s p_1^s \ldots p_m^s)$. Here $p_i^s \in \{0, 1\}$ and $0 \leqslant m \leqslant 31$ for IPv4 and $0 \leqslant m \leqslant 127$ for IPv6. We call it *Two Dimensional-IP (TwoD-IP) forwarding* as an operation $\mathcal{I}^s(P_s)$ to map P_s to a next hop $a_t \in \{a_0, a_1, \ldots, a_j, \ldots\}$. *TwoD-IP routing* is that for each packet, the routers perform a forwarding operation and a TwoD-IP forwarding operation to find a next hop on the pre-defined VIP path (note that for packets of non-VIP traffic, a forwarding operation already results in a next hop). Let the path of a packet forwarded by TwoD-IP routing be $\mathcal{L}(P_s) = \{v^0, v^1, \ldots, v^j, \ldots\}$; here we only use the source prefix P_s to denote a packet for simplicity.

If we *deploy* node v, this node is TwoD-IP forwarding capable, i.e., this node can perform operation $\mathcal{I}^s(P_s)$. A *deployment* $\mathcal{G}$ is a set of nodes $V\prime \subseteq V$ that are deployed to be TwoD-IP forwarding capable. If we do not deploy node v, v will use conventional shortest path routing. In a deployment process, let κ be the number of nodes we want to deploy.

Due to the CPU and storage limitations, each router may have a *TwoD-IP forwarding capacity* (or *capacity* in short). As explained, we model this as the maximum number of bits in P_s that the router can process. Let C_v denote the capacity of router v. In other words, the router may lookup a subset of bits of the prefix. Let $\mathcal{F}(P_s) = (p_{i_0} p_{i_1} \ldots), p_{i_j} \in \{p_0, p_1, \ldots, p_m\}$ be a function that selects a subset of bits from a prefix; we call $\mathcal{F}(P_s)$ a sub-prefix. For each sub-prefix $\mathcal{F}(P_s)$, we can also perform TwoD-IP forwarding $\mathcal{I}^s(\mathcal{F}(P_s))$ to map sub-prefix $\mathcal{F}(P_s)$ to a next hop $a_{sub} \in \{a_0, a_1, \ldots, a_j, \ldots\}$. We show the notation list in Table 1.

Table 1
Notation list.

Notation	Definition	Notation	Definition
V	Set of nodes	$\mathcal{I}^d$	Forwarding operation
t	Traffic flows	$\mathcal{I}^s$	TwoD-IP forwarding
$\mathcal{T}$	Set of traffic flows	$\mathcal{G}$	A deployment
$VIP(t)$	VIP path for t	$\mathcal{L}(\mathcal{G},\mathcal{F},P_s)$	TwoD-IP path
P_s	Source IP prefix	κ	Deployed number
P_d	Destination IP prefix	$\mathcal{F}(P_s)$	A sub-prefix

Since a router may conduct TwoD-IP forwarding based on a sub-prefix $\mathcal{F}(P_s)$ and a deployment $\mathcal{G}$ may include a subset of nodes, the resulting path for a VIP traffic flow t may deviate from its $VIP(t)$. Note that given $\mathcal{G},\mathcal{F}$, for a VIP traffic t (identified by its P_s), its path is determined. Let $\mathcal{L}(\mathcal{G},\mathcal{F},P_s)$ denote such path. Clearly, we want such deviation to be small. To quantify such deviation, In our formulation, we used hamming distance to represent the deviation between paths. While hamming distance is a well-known indicator of the differences between vectors and was widely used by previous works [14,15], it is practically meaningful in some scenarios. For example, when the VIP path is to pass through a set of middle-boxes [16], hamming distance can be used to measure the number of non-passed middle-boxes. In our case study in Section 7, we show that hamming distance can be used to represent the effect of load balancing. We admit that there exists other meaningful metrics based on which we can also formulate a similar problem. We focus on hamming distance in this paper due to its simplicity. We will consider other metrics and their comparison in our future work. Note that we do not exclude other meaningful metrics and they should be discussed in our future work.

We define $D(\mathcal{L}_i,\mathcal{L}_j)=\max\{|\mathcal{L}_i|,|\mathcal{L}_j|\}-|\mathcal{L}_i\cap\mathcal{L}_j|$ the *distance* between two paths $\mathcal{L}_i$ and $\mathcal{L}_j$. Since shortest path is used if a node is not deployed, we always have $|\mathcal{L}(\mathcal{G},\mathcal{F},P_s)|\leqslant|VIP(t)|$. Thus,

Observation 1. $D(VIP(t),\mathcal{L}(\mathcal{G},\mathcal{F},P_s))=|VIP(t)|-|VIP(t)\cap\mathcal{L}(\mathcal{G},\mathcal{F},P_s)|$.

Our objective is that given the number of routers that we want to deploy, find a deployment that satisfies the routers capacities, and minimizes the total distance of the paths of the VIP flows and their pre-defined VIP paths.

Problem 1 (*Constrained Deployment*). Given κ, find a deployment $\mathcal{G}^o$ where $|\mathcal{G}^o|=\kappa$ so that $\forall v,|\mathcal{F}(P_s)|\leqslant C_v$ and $\sum_t D(VIP(t),\mathcal{L}(\mathcal{G}^o,\mathcal{F},P_s))$ is minimized.

In practice, according to different volume of the VIP traffic flows, we may need to assign different weights, i.e., w_t for flow t. In our problem formulation, we can add this weight to the distance and modify $\sum_t D(VIP(t),\mathcal{L}(\mathcal{G}^o,\mathcal{F},P_s))$ to $\sum_t w_t D(VIP(t),\mathcal{L}(\mathcal{G}^o,\mathcal{F},P_s))$ to show different importance of t in the aggregated distance. In this paper, we will focus our study on the unweighted problem. Our analysis and solutions will not change in the weighted version. We will briefly mention the weighted case in our case study.

Problem 1 has many variations; a simpler version is that, if routers across the network have enough capacity, we have

Problem 2 (*Unconstrained Deployment*). Given κ, find a deployment $\mathcal{G}^o$ where $|\mathcal{G}^o|=\kappa$ so that $\sum_t D(VIP(t),\mathcal{L}(\mathcal{G}^o,P_s))$ is minimized.

Note that we use $\mathcal{L}(\mathcal{G}^o,P_s)$ to denote $\mathcal{L}(\mathcal{G}^o,\mathcal{F},P_s)$ without capacity constraint.

As said, incremental deployment is highly favored in practice. Let $\mathcal{G}_i=V_i$ and $\mathcal{G}_j=V_j$ be two deployments. We call $\mathcal{G}_j$ *incremental* to $\mathcal{G}_i$ if $V_i\subseteq V_j$. An *incremental deployment* is a series of deployment $\mathcal{G}_0,\mathcal{G}_1,\ldots,\mathcal{G}_j,\ldots$, such that $\mathcal{G}_j$ is incremental to $\mathcal{G}_i$ if $i<j$. Thus,

Problem 3. Given $\kappa_0,\kappa_1,\ldots,\kappa_j,\ldots$, find an incremental deployment, $\mathcal{G}_0^o,\mathcal{G}_1^o,\ldots,\mathcal{G}_j^o,\ldots$, such that $|\mathcal{G}_i^o|=\kappa_i$, and $\mathcal{G}_i^o$ is a constrained (unconstrained) deployment. We call the deployment series **incremental constrained (unconstrained) deployment**.

We illustrate our definitions with an example topology in Fig. 2. Here we assume a is the source and e is the destination. For a traffic flow the shortest path from a to e is $\{a,c,e\}$. Assume there are one VIP flow t_0 and one non-VIP flow. Assume the pre-defined VIP path is $VIP(t_0)=\{a,b,c,d,e\}$. As discussed, the VIP flow is identified by its source IP prefix and destination IP prefix. Assume the source and destination prefixes of t_0 are $P_s(t_0)=(00^*)$ and $P_d(t_0)=(11^*)$; the source and destination prefixes of non-VIP flow is (10^*) and (11^*). The forwarding table on node c for conventional routing is shown in Table 2.

Assume the TwoD-IP forwarding capacities of node a,b,c,d be $C_a=0,C_b=0,C_c=1,C_d=0$. Assume node c check the 0_{th} bit; thus for node $c,\mathcal{F}(P_s(t_0))=(0)$. To achieve TwoD-IP routing, node c should forward the packets of t_0 to d, and the packets of non-VIP flow to e, that is, $\mathcal{I}^s(\mathcal{F}(P_s(t_0)))=d$. We show the forwarding table on node c for TwoD-IP routing in Table 2.

Assume we deploy one node, i.e., $\kappa=1$ and we select c, i.e., deployment $\mathcal{G}=\{c\}$. Then $\mathcal{L}(\mathcal{G},\mathcal{F},P_s)=\{a,c,d,e\}$, and $D(VIP(t_0),\mathcal{L}(\mathcal{G},\mathcal{F},P_s))=5-4=1$. The constrained deployment is $\mathcal{G}^o=\{c\}$.

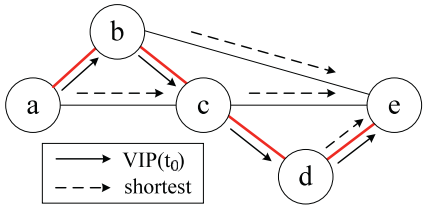


Fig. 2. Network topology and VIP paths

Table 2
Forwarding table for the routers c in Fig. 2.

Destination				$\mathcal{I}^d(\cdot)$
(a) Conventional				
$P_d = (11^*)$				$\{e\}$
Destination	$\mathcal{I}^d(\cdot)$	Source	$\mathcal{F}(\cdot)$	$\mathcal{I}^s(\mathcal{F}(\cdot))$
(b) TwoD-IP				
$P_d = (11^*)$	$\{d, e\}$	$P_s(t_0)$	(0)	d
		non-VIP	(1)	e

3. Incremental unconstrained deployment

In this section, we focus on incremental unconstrained deployment. In practice, the routers today is usually over-provisioned [17], making it possible to support TwoD-IP routing directly. Understanding of incremental unconstrained deployment also serves as a first step for the constrained version; which we will address in Section 4.

We will first develop an algorithm for unconstrained deployment given the number of nodes to be deployed (Problem 2). We show it can be naturally extended to solve the incremental unconstrained deployment (Problem 3).

Theorem 1. *Finding an unconstrained deployment (Problem 2) is NP-Complete.*

Proof. It is easy to see that the decision problem of validating a given unconstrained deployment is solvable in polynomial time. Therefore, finding an unconstrained deployment is in NP class. To show this problem is NP-hard, we reduce the dense-k subgraph problem to it; the former is known to be NP-complete [18]. The dense-k subgraph problem is, given a graph (V, E) and a positive integer k, find a subset $\mathcal{G}_k \subseteq V$ where $|\mathcal{G}_k| = k$, so that the number of induced edges is minimized, i.e., $((u, v) \in E) \cap (\mathcal{G}_k \times \mathcal{G}_k)$. We expand (V, E) to a network (V', E'), where $V\prime = \bigcup_{(u,v)\in E}\{u, v, p, q\}$, $E' = \bigcup_{(u,v)\in E}\{(u, v), (u, p), (v, p), (v, q), (q, p)\}$. And we add a VIP traffic flow $t = \{u, v, p, q\}$ for each $(u, v) \in E$. Fig. 3 shows the transformation. We next show that the unconstrained deployment $\mathcal{G}^o$ where $|\mathcal{G}^o| = \kappa = k$ in $(V\prime, E\prime)$, is equal to the solution $\mathcal{G}_k$ to dense-k problem. If $u, v \in \mathcal{G}^o$, then $\mathcal{L}(\mathcal{G}^o, P_s) = \{u, v, p, q\}$; if $u \in \mathcal{G}^o, v \notin \mathcal{G}^o$, then $\mathcal{L}(\mathcal{G}^o, P_s) = \{u, v, q\}$; else $\mathcal{L}(\mathcal{G}^o, P_s) = \{u, p, q\}$ (shortest path from u to q). Thus only if $u, v \in \mathcal{G}^o, D(VIP(t), \mathcal{L}(\mathcal{G}^o, P_s)) = 0$, else $D(VIP(t), \mathcal{L}(\mathcal{G}^o, P_s)) = 4 - 3 = 1$. Thus, $\sum_t D(VIP(t), \mathcal{L}(\mathcal{G}^o, P_s)) = |E| - |((u, v) \in E) \cap (\mathcal{G}^o \times \mathcal{G}^o)|$. So it is equal to maximize $|((u, v) \in E) \cap (\mathcal{G}^o \times \mathcal{G}^o)|$. So, $\mathcal{G}_k = \mathcal{G}^o$. □

Though the unconstrained deployment problem is NP-complete, if κ, the number of nodes to be deployed, is a constant, the problem is polynomial-time solvable even we perform exhaustive search. However, if κ is large, a straightforward exhaustive search is computationally unacceptable. Therefore, when κ is large, we develop a heuristic where we divide κ into small κ' and find the deployment for each individual κ'. This also naturally leads to an algorithm for the incremental unconstrained deployment problem.

We first define *passing-through property* of each node. Intuitively, v has passing-through property if v is on both the path of the VIP traffic flow and its pre-defined VIP path.

Definition 1. Node v has passing-through property for traffic t under deployment $\mathcal{G}$, if $v \in VIP(t) \cap \mathcal{L}(\mathcal{G}, P_s(t))$.

Clearly, the more nodes have passing-through property, the smaller the total distance. Let $\mathcal{C}(v, P_s(t))$ denote the passing-through property of v for traffic t. We develop Algorithm Passing-Through() to evaluate $\mathcal{C}(v, P_s(t))$ of all nodes. Intuitively, evaluation of passing-through properties of all nodes needs to check all possible deployments $\mathcal{G}$, which is exponential. We develop an Algorithm Passing-Through() which does not need to perform an exhaust search in the solution space. We will use Algorithm Passing-Through() as a subroutine to solve the unconstrained deployment problem.

Let θ_v be an indicator variable for node v, if v gets deployed, $\theta_v = 1$, else $\theta_v = 0$. Our idea is that we do not need to make an assignment to θ at the beginning (which will reflect to a specific deployment). We use this abstract θ and transfer the problem to a generalized MAX-SAT problem. Our solution thus does not need to specify the deployment.

We first use an example in Fig. 2 to explain our idea. For example, node c has (or does not have) passing-through property if and only if $\{(\theta_a \wedge \theta_b) \vee \neg\theta_a\}$ is equal to one (or zero). We see that node c has passing-through property in two conditions: (1) a is not deployed or (2) both a and b are deployed. In condition (1), since a is not deployed, the traffic will follow conventional shortest path routing. In condition (2), since a is deployed, a will perform TwoD-IP forwarding and the traffic will flow the VIP path to node b. Since b is also deployed and will perform TwoD-IP forwarding, the traffic will be forwarded to c. Correspondingly, $\{(\theta_a \wedge \theta_b) \vee \neg\theta_a\}$ is equal to 1 if (1) $\theta_a = 0$ or (2) $\theta_a = 1$ and $\theta_b = 1$. We thus provide a mapping between clause satisfaction and our problem. Similarly, we can see that the passing-through property of node a, b, d, e can be transformed to clauses '1', $\{\theta_a\}$, $\{(\theta_a \wedge \theta_b \wedge \theta_c) \vee (\neg\theta_a \wedge \theta_c)\}$, $\{(\theta_a \wedge \theta_b \wedge \neg\theta_c) \vee (\neg\theta_a \wedge \neg\theta_c) \vee (\theta_a \wedge \theta_b \wedge \theta_c) \vee (\neg\theta_a \wedge \theta_c) \vee (\theta_a \wedge \neg\theta_b)\}$ respectively.

We define $s\_child(v, t)$ as the first successor node that is on both $VIP(t)$ and the shortest path. For example, in Fig. 2, $s\_child(a, t_0) = c$, indicating that c is the first successor node on $VIP(t_0) = \{a, b, c, d, e\}$ and also on the shortest path from a to e. Algorithm Passing-Through() computes the passing-through properties of each node as follows.

Algorithm 1. Passing-Through($VIP(t)$)

```
Output : C(v, P_s(t)), ∀v ∈ VIP(t)
1  begin
2    C(v_t^0, P_s(t)) ←'1'// v_t^0 is the source node
3    s_child(v_t^0, t) ← compute_s_child(v_t^0, t)
4    for i = 1 to |VIP(t)| − 1 do
5      s_child(v_t^i, t) ← compute_s_child(v_t^i, t)
6      if v_t^i = s_child(v_t^{i−1}, t) then C(v_t^i, P_s(t)) ← C(v_t^{i−1}, P_s(t))
7      else C(v_t^i, P_s(t)) ← C(v_t^{i−1}, P_s(t)) ∧ (θ_{v_t^{i−1}})
8      for j = 0 to i − 2 do
9        if v_t^i = s_child(v_t^j, t) then
           C(v_t^i, P_s(t)) ← C(v_t^i, P_s(t)) ∨ (C(v_t^j, P_s(t)) ∧ (¬θ_{v_t^j}))
10 end
```

The input of Algorithm Passing-Through() is a VIP path $VIP(t)$, and the output is the passing-through properties of each node on $VIP(t)$. Basically, Algorithm

Passing-Through() follows a dynamic programming structure. We show an example of Algorithm Passing-Through() using Fig. 2 as the input. We show the last round execution of Algorithm Passing-Through() to compute $\mathcal{C}(e, P_s(t_0))$. As shown in Fig. 2, $s\_child(d, t_0) = e$ and d is the predecessor node of e along $VIP(t_0)$. And node b, c satisfy $s\_child(b, t_0) = e$, $s\_child(c, t_0) = e$. So $\mathcal{C}(e, P_s(t_0)) = \mathcal{C}(d, P_s(t_0)) \vee (\mathcal{C}(b, P_s(t_0)) \wedge \neg\theta_b) \vee (\mathcal{C}(c, P_s(t_0)) \wedge \neg\theta_c)$.

Theorem 2. *The complexity of Algorithm Passing-Through() is $O(|VIP(t)|^2)$, which is bounded by $O(|V|^2)$.*

Proof. The complexity of *compute_s_child*(v, t) is bounded by $O(|VIP(t)|)$. Thus, the theorem gets proved. □

We now solve the unconstrained deployment problem. Recall the generalized MAX-SAT problem as: given a set U of variables θ_i, a collection of clauses, where each clause is a disjunction of conjunction of literals (e.g., $(\theta_i \wedge \theta_j) \vee \neg\theta_j$), find a truth assignment such that the number of satisfied clauses is maximized. We develop our UOpt-Deploy() following the branch-and-bound algorithm for MAX-SAT [19]. We improve the branch-and-bound algorithm by exploring the search tree in a depth-first order. At each node, the algorithm compares the number of clauses violated (unsatisfied with certainty) by the best assignment (called upper bound *ub*), with the number of clauses violated by the current assignment plus an underestimation. The underestimation is the number of clauses that become violated if we extend the current partial assignment into a complete assignment. If the current assignment plus the underestimation is greater than the best assignment, the subtree of the node is pruned, else the algorithm searches one level deeper into the tree. If a clause is certain to be satisfied, it will be removed from the union of clauses (Γ). Initially, *ub* is computed by a local random search procedure call *GSAT*() [20]. When each step the algorithm searches deeper, a variable in Γ is selected following J–W rule [21], which gives precedence to variable in shorter clauses.

The inputs of Algorithm UOpt-Deploy() are ub, Γ and κ. The outputs of the Algorithm UOpt-Deploy() are the unconstrained deployment $\mathcal{G}^o$ and the minimum distance.

Algorithm 2. UOpt-Deploy(ub, Γ, κ)

```
   Initialize : Γ ← ⋃_{v∈VIP(t),θ_v≠1} C(v, P_s(t)), ub ← GSAT(Γ)
   Output     : < G^o, ub >
1  begin
2      if Γ = ∅ or Γ only contains violated clauses then
           // return the number of violated clauses
3          return < ∅, violated_num(Γ) >
4      if lower_bound(Γ) > ub then return < ∅, ∞ >
5      u ← select_variable(Γ)// following J-W rule
6      if Σ_{v∈V} θ_v < κ then // set θ_v to be 1 in Γ
7          < G, ub_1 > ← UOpt-Deploy(ub, Γ|θ_u, κ)
8          if ub_1 ≤ ub then ub ← ub_1
9      < G', ub_2 > ← UOpt-Deploy(ub, Γ|¬θ_u, κ) // set θ_v to be 0
10     if ub_2 ≤ ub then ub ← ub_2, return < G', ub >
11     else return < G ∪ {u}, ub >
12 end
```

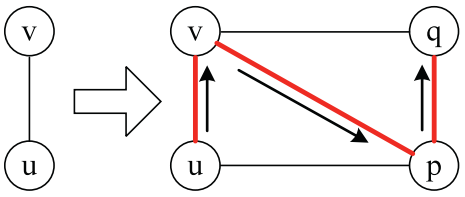


Fig. 3. Transformation from a graph.

Theorem 3. *The complexity of Algorithm UOpt-Deploy() is $O(\binom{|V|}{\kappa} \times |\mathcal{T}| \times \max_t |VIP(t)|^2)$, bounded by $O(|V|^{\kappa+2} \times |\mathcal{T}|)$; The network is loop-free after deploying $\mathcal{G}^0$.*

Proof. The algorithm has to check at most $\binom{|V|}{\kappa}$ cases. Each case has at most $|\mathcal{T}| \times \max_t |VIP(t)|$ clauses, each clause has at most $\max_t |VIP(t)|$ variables. Because $|VIP(t)| < |V|$, so the complexity is bounded by $O(|V|^{\kappa+2} \times |\mathcal{T}|)$.

Without loss of generality, we prove that VIP flow t is loop-free. As we mentioned, we follow the rules in [13] when selecting the next hops. Suppose we use the "one hop down" rule (other rules can be proved in a similar way), i.e., the deployed routers only forward packets to neighbors that are closer to the destination. Assume there is a loop for t, there must be at least two nodes of $VIP(t)$ on the loop path.[3] Let them be v_t^i and v_t^j, then v_t^i is closer than v_t^j and v_t^j is closer than v_t^i, which contains contradictions. □

This complexity is exponential. However, we can see that if κ is constant, the complexity becomes polynomial and we can perform exhaustive search. But when κ is large, the computing time increases fast with κ. To reduce computing time, we develop a heuristic, which computes UOpt-Deploy(ub, Γ, κ) by running UOpt-Deploy(ub, Γ, 1) for κ times. The complexity is then reduced to $O(|V|^3 \times |\mathcal{T}| \times \kappa)$.

Note that this heuristic can naturally be used to solve our incremental unconstrained deployment problem. We call it Inc-UDeploy() for future reference.

Next, we reduce the search space of our problem given a key observation as follows.

Observation 2. $D(VIP(t), \mathcal{L}(\mathcal{G}, \mathcal{F}, P_s)) = 0$ if and only if for $0 \leqslant i \leqslant |VIP(t) - 2|, \theta_{v_t^i} = 1$ when $s\_child(v_t^i) \neq v_t^{i+1}$.

Proof. The correctness proof is in Appendix A.1. □

Let $K = \{v_t^i | s\_child(v_t^i) \neq v_t^{i+1},\ \forall t,\ 0 \leqslant i \leqslant |VIP(t) - 2|\}$, Observation 2 shows that we only need to deploy nodes in K to guarantee that the paths that VIP flows are identical with pre-defined VIP paths. We call the nodes in K *key* nodes.

[3] We do not consider the routing oscillation and instability [22].

4. Increment constrained deployment

4.1. Feasible bit-lookup set

We now take the router capacity constraint into account. We first need to guarantee that all packets are forwarded to the right next hops. Let $\mathcal{B}_v(d)$ be a subset of the bits of source IP address for router v and destination d.

Definition 2. A bit set $\mathcal{B}_v(d)$ is **feasible** if packets of VIP traffic t towards d will be forwarded to next hop a_t on $VIP(t)$, and other packets will be forwarded on the shortest path.

We want $\mathcal{B}_v(d)$ to be as small as possible. For example, in TCAM-based solutions, less bits can reduce the width of each TCAM entry, as the width of TCAM chips can be configured in current routers [23]. Thus it can reduce the storage space and power consumption of TCAM-based solutions. In trie-based solutions, less bits directly lead to less memory accesses and less memory space, thus making the lookup speeds of routers faster.

We illustrate our definition with an example. In Fig. 4, there are three VIP flows towards destination d, t_0, t_1, and t_2. On node v, the next hops of the VIP flows are $a_{t_0} = a, a_{t_1} = a, a_{t_2} = c$. There is a non-VIP flow, whose next hop is b. The source prefixes of each VIP and non-VIP are shown in Fig. 4. In this example, $\{1,2\}, \{0,1,2\}, \{1,2,3\}, \{0,1,2,3\}$ are all feasible bit-lookup sets. Take $\{1,2\}$ as an example. When a packet towards destination d arrives, the 1_{st} and 2_{nd} bits will be checked in the source IP: if the 1_{st} bit is '1', the packet will be forwarded to a; if the 1_{st} is '0' and the 2_{nd} is '1', the packet will be forwarded to c; else the packet will be forwarded to b.

Intuitively, for traffic t_i, t_j (or non-VIP) where the next hops for t_i, t_j (or non-VIP) are different, the bits that need to be checked for t_i and t_j (or non-VIP) should have at least one bit in difference.[4]

4.2. Minimum bit-lookup set

We first develop an algorithm to search for the minimum bit-lookup set for each router. This will become a subroutine for the overall algorithm. We show that the problem is NP-complete and we will solve it with a greedy based algorithm from the minimum cover set [25].

Before deriving the complexity, we define $\mathcal{E}(p,q)$ be the set of the positions of the different bits for two prefixes p and q. For example, for $p = \langle 0011^* \rangle, q = \langle 00001 \rangle$, the 2_{nd} and 3_{rd} bits are different. Thus $\mathcal{E}(p,q) = \{2,3\}$ (Note that the 4th bits of p, q are $*, 1$, and $\mathcal{E}(p,q)$ does not consider this bit).

Let $\mathcal{N}$ be the set of non-VIP prefixes, and a_n be the next hop for non-VIPs (next hop on the shortest path) for node v and destination d.

Theorem 4. *Finding minimum bit-lookup set is NP-complete.*

[4] Note that two prefixes can partially overlap, making it ambiguous to find the bit difference. In such case, a preprocess can be conducted to remove such overlapping [24]. Consequently, in the rest of the paper, we assume that no two prefixes overlap.

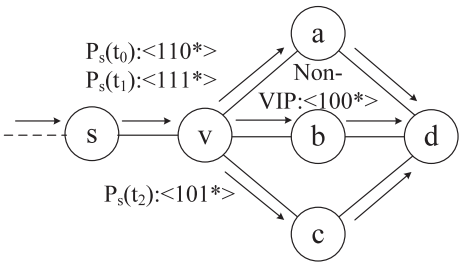


Fig. 4. Feasible bit-lookup set.

Proof. We prove that finding the minimum bit-lookup set is equivalent to the minimum hitting set problem. The minimum hitting set problem is equivalent to minimum cover set problem, which is known to be NP-complete [26]. The minimum hitting set problem is, given a collection E of subsets of a finite set F, find a solution $B \subseteq F$, such that $B \cap e \neq \emptyset, \forall e \in E$ and $|B|$ is minimized. Let F be the set of all bits in source address. Let E includes $\mathcal{E}(P_s(t_i), P_s(t_j))$, $\forall i,j$, if $a_{t_i} \neq a_{t_j}$, and $\mathcal{E}(P_s(t_i), p)$, $\forall i$, $\forall p \in \mathcal{N}$, if $a_{t_i} \neq a_n$. It's easy to see that the solution of minimum bit-lookup set is equivalent to the solution of minimum hitting set. □

We next present a greedy based algorithm to solve the minimum bit-lookup set problem. At each stage, the algorithm chooses the bit contained in most sets and then exclude the sets that are already covered by the selected bit.

Algorithm 3. Min-BitLookup(v, d)

```
   Output: B_v(d)
1  begin
2      P ← {t | P_d(t) = d, C(v, P_s(t)) = 1}
       // flows passing by v to d
3      S ← {E(P_s(t_i), P_s(t_j)) | a_i ≠ a_j, ∀t_i, t_j ∈ P}
       ∪{E(P_s(t_i), p) | a_i ≠ a_n, ∀t_i ∈ P, ∀p ∈ N}
4      while S ≠ ∅ do
5          s ← greedy_select_bit(S), B_v(d) ← B_v(d) + {s}
6          S ← S − {S ∈ S | s ∈ S}
7  end
```

Theorem 5. *The complexity of Min-BitLookup() is* $O(|\mathcal{T}|^2)$.

Proof. $|\mathcal{N}|$ is a constant, so $|\mathcal{S}|$ is bounded by $O(|\mathcal{T}|^2)$. Because $|\mathcal{B}_v(d)| \leqslant 32$ (or 128), thus theorem is proved. □

The algorithm achieves an approximation ratio of $H(m)$, where m is the maximum number of sets that contain the same bit, and $H(n)$ is the n_{th} harmonic number. The greedy algorithm is proved to be the best-possible polynomial time approximation for set cover problem [27].

4.3. Incremental constrained deployment

The constrained deployment problem can be transformed into a partial generalized MAX-SAT problem [28], which has some clauses marked as soft (relaxable) clauses and the others as hard (non-relaxable) clauses. The objective is to find a value assignment that satisfies all hard

Table 3
Algorithm table.

Algorithm	Problem definition	Problem
UOpt-Deploy()	Unconstrained deployment	Problem 2
GOpt-Deploy()	Constrained deployment	Problem 1
Inc-UDeploy()	Incremental unconstrained deployment	Problem 3
Inc-GDeploy()	Incremental constrained deployment	Problem 3
Adp-Forward()	Adaptive forwarding	Problem 4

clauses and maximum soft clauses. Transforming to our situation, the soft clauses are the passing through properties on each node (i.e., $\mathcal{C}(v, P_s(t))$) and the hard clauses represent the solution space where the capacity constraint can be satisfied. For example, $(\neg\theta_a) \wedge (\neg\theta_b)$ means a and b should not be deployed at the same time.

However, we cannot know the hard clauses in advance. We learn the hard clauses during the computation process. Iteratively, we use UOpt-Deploy() to compute a deployment $\mathcal{G}$, and use Min-BitLookup() to validate if $\mathcal{G}$ satisfies the capacity constraints on all nodes. If the capacity constraints are satisfied, we output the final solution. Else we learn a hard clause as $\neg v_i(\text{Min} - \text{BitLookup}(v_i, d) - C_{v_i} > \text{Min} - \text{BitLookup}(v_j, d) - C_{v_j}, \forall j \neq i)$, which excludes the most congested node.

Algorithm 4. GOpt-Deploy(κ)

```
Output: G^o
1 begin
2   satisfied ← False, ϒ ← ∅ // ϒ: the set of hard clauses
3   while satisfied = False do
        // UOpt-Deploy'() is a modified version satisfying ϒ
4       satisfied ← True, < G, ub > ← UOpt-Deploy'(∞, ∅, κ)
5       if ∃v, d, Min-BitLookup(v, d) − C_v > 0 then
6           ϒ ← ϒ ∪ learn_hardclause(), satisfied ← False
7   if satisfied = TRUE then return G
8 end
```

Theorem 6. *The complexity of Algorithm GOpt-Deploy() is* $O((|V| - \kappa) \times (|V|^3 \times |\mathcal{T}| \times \kappa + |V| \times |\mathcal{T}|^3))$, *which is bounded by* $O(|V|^2 \times |\mathcal{T}| \times (|V|^2 \times \kappa + |\mathcal{T}|^2))$.

Proof. The loop runs for at most $|V| - \kappa$ times. □

We can derive a heuristic for the incremental constrained deployment problem by natural extension of GOpt-Deploy(), using the same method in Section 3. We call it Inc-GDeploy() for future references. For clarity, we summarize our algorithms developed in this paper in Table 3.

5. Discussion and improvement

The deployment sequence derived from Algorithm Inc-UDeploy() is fixed. In practice, the deployment may need to be more flexible. Even the deployment sequence is settled, the network or VIP paths may change. As such, we need a more adaptive deployment scheme.

To address this problem, our idea is to make the deployed nodes adaptive in the forwarding operation, i.e., $\mathcal{I}^s(\mathcal{F}(P_s))$. More specifically, a deployed node may "turn-off" the TwoD-IP forwarding operation to a subset of VIP flows. Let $\lambda(v, t)$ denote this turn-off action, that is, if $\lambda(v, t) = 1$, v forwards t on $VIP(t)$, else v forwards t on the shortest path.

Let $\mathcal{J}(\mathcal{G}) = \{\lambda(v, t) | v \in \mathcal{G}, v \in VIP(t)\}$, we call $\mathcal{J}(\mathcal{G})$ an *adaptive forwarding* for $\mathcal{G}$. Our objective is that given $\mathcal{G}$, find an adaptive forwarding that minimizes the total distance between the paths of the VIP traffic flows and their pre-defined VIP paths. Formally,

Problem 4 (*Adaptive Forwarding*). Given a deployment $\mathcal{G}$, find an adaptive forwarding $\mathcal{J}(\mathcal{G})$ that minimizes the total distance $\sum_t D(VIP(t), \mathcal{L}(\mathcal{G}, P_s))$.

To solve this problem, we develop a dynamic programming based algorithm Adp-Forward() which is optimal.

Algorithm 5. Adp-Forward($\mathcal{G}$)

```
   Output : J(G)
1  begin
2    f(v_t^{|VIP(t)|−1}) = 1
3    for t ∈ T do
4      for i = |VIP(t)| − 2 to 0 do
5        if v_t^i ∈ G_i then f(v_t^i) ← 1 + f(s_child(v_t^i, t))
6        else
7          if f(v_t^{i+1}) > f(s_child(v_t^i, t)) then
             f(v_t^i) ← 1 + f(v_t^{i+1}), λ(v_t^i, t) ← 1
8          else f(v_t^i) ← 1 + f(s_child(v_t^i)), λ(v_t^i, t) ← 0
9  end
```

Theorem 7. *The complexity of Adp-Forward() is* $O(\max_i |VIP(t)|^2 \times |\mathcal{T}|)$, *which is bounded by* $O(|V|^2 \times |\mathcal{T}|)$.

Proof. For each VIP path, Adp-Forward() computes from the last node to the first node along $VIP(t)$, and the complexity for computing *s_child*() is bounded by $O(|VIP(t)|)$. □

With adaptive forwarding, the performance of incremental deployment can be improved, i.e, reduce the total distance between the paths of VIP traffic flows and the pre-defined VIP paths, we can then derive a property of the deployment sequence found by Adp-Forward().

Lemma 1. *Given an incremental deployment series* $\mathcal{G}_0, \mathcal{G}_1, \ldots$, *then* $\mathcal{J}_0(\mathcal{G}_0), \mathcal{J}_1(\mathcal{G}_1), \ldots$ *computed by Adp-Forward() satisfies that, if* $i \leqslant j$, *the total distance within* $\mathcal{J}_i(\mathcal{G}_i)$ *is greater than or equal to total distance within* $\mathcal{J}_j(\mathcal{G}_j)$.

Proof. Suppose there exist $\mathcal{J}'_j$, such that if $v \notin \mathcal{G}_i$ and $v \in \mathcal{G}_j$, let $\lambda(v, t) = 0$, $\forall t \in \mathcal{T}$, else let $\lambda(v, t)$ be the same with $\lambda(v, t)$ in $\mathcal{J}_i$. Then the total distance within $\mathcal{J}'_j(\mathcal{G}_j)$ is the same with the distance within $\mathcal{J}_i(\mathcal{G}_i)$. According to the problem definition in Problem 4, distance within $\mathcal{J}_j(\mathcal{G}_j)$ is smaller than distance within $\mathcal{J}'_j(\mathcal{G}_j)$. Thus we can conclude that the distance within $\mathcal{J}_j(\mathcal{G}_j)$ is always smaller than or equal to the distance within $\mathcal{J}_i(\mathcal{G}_i)$, Thus the lemma is proved. □

Another improvement is to further reduce the minimum bit-lookup set. One key is to differentiate VIP traffic flows and non-VIP traffic flows. There are many options to achieve this and one is to use a bit marked in the VIP packets.

6. Performance evaluation

6.1. Simulation setup

We evaluate the performance of our TwoD-IP routing scheme using both BRITE [29] generated topologies and CERNET. A case study on CERNET2 is in the next section.

6.1.1. BRITE topology

We generate topologies with nodes from 50 to 400. To set up a VIP flow, we first randomly select a pair of source-destination nodes, and then randomly select its VIP path which (1) is not the shortest path and (2) satisfies the rules in [13] to prevent routing loops. Each VIP flow is associated with multiple source and destination prefixes. We set the number of source prefixes (SP) associated with each VIP flow to be 50, and the number of destination prefixes (DP) with each VIP flow to be 600. The number of VIP flows is between 10 and 100. For each node, we set the capacity constraint to be 0 to 32. For simplicity, we set the capacity of each router to be the same over the network, to show that our algorithm can reduce the number of bits needed to be checked. However, the problem formulation does not change if the capacity is a variable for each router. The default values and other parameters of our evaluation are in Table 4.

6.1.2. Real topology

We obtain the real topology of CERNET, which is a medium-scale IPv4 network with 110 routers and 238 links. We collected the prefixes from the RIB table of the router in Beijing. There are 51,642 entries and 18 next AS hops. We need to find the next router hop for each prefix at each router. Similar to [30,31] we select 18 egress routers and compute the next hop for each prefix using these egress routers as the destinations by Dijkstra. We also obtain four real topologies (AS 1239, AS 1221, AS 1755, AS 3257, AS 3967, AS 6461) from Rocketfuel [32]. The details of real topologies are in Table 5.

We compare our algorithm with random deployment (RD). We admit that random deployment is artificial, and service providers may attempt other schemes. We thus compare with human-like deployment (HD), that only selects key nodes to deploy (see Observation 2). Note however, HD is based on an observation within the contribution of our study. Besides, we also set full deployment (FD) as a benchmark for comparison. Our evaluation metric is the averaged normalized distance (or in short *a-distance* thereafter) between the paths computed by our algorithms and the pre-defined VIP paths. Formally, a-distance equals to $\frac{\sum_t D(\mathcal{L}(\mathcal{G},\mathcal{F},P_s),VIP(t))}{\sum_t |VIP(t)|}$. The smaller the a-distance, the better. If a-distance equals to zero, the performance is identical to the FD, i.e., the computed paths are identical to the pre-defined VIP paths. The results shown in this section are averaged by ten random and independent experiments.

Table 4
Parameter table.

VIP flows	Capacity	SP	DP	No. nodes
30	16	50	600	150
Links/new node	Mode	Model	Placement	α/β
3	Router only	Waxman	Random	0.15/0.2

Table 5
Parameter table of real topologies.

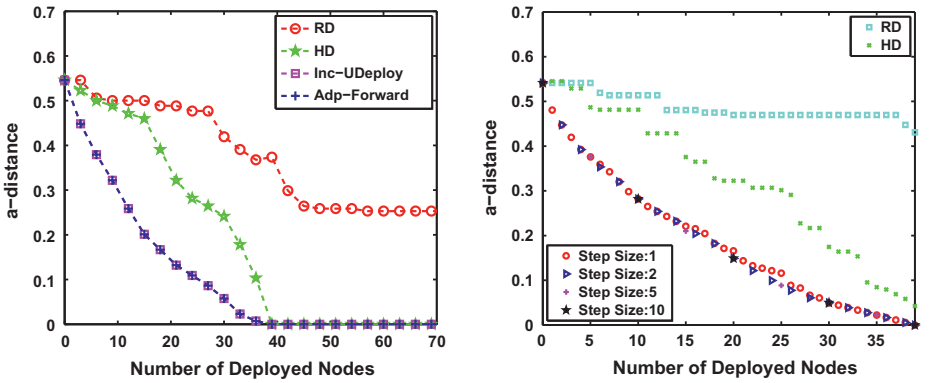
	No. routers	No. edges	Avg. links/new router
AS1239	315	972	3.09
AS1221	104	151	1.45
AS1755	87	161	1.85
AS3257	161	328	2.04
AS3967	79	147	1.86
AS6461	128	372	2.90

6.2. Simulation results

6.2.1. Incremental unconstrained deployment

Fig. 5(a) shows a typical incremental deployment process and compares different deployment algorithms. There are 150 nodes and 30 VIP flows. We deploy three nodes in every incremental step. In Fig. 5(a), we see that even we do not deploy any router, the a-distance is still around 50%. This is because the shortest path can travel through a few VIP nodes already. When we deploy more nodes, the a-distance decreases. However, Algorithm Inc-UDeploy() performs better than RD and HD. For example, after we deploy 33 nodes, the nodes chosen by Algorithm Inc-UDeploy() match the nodes on the VIP paths very well, and the a-distance is only 2.02%. On the contrary, if we randomly choose 36 nodes to deploy, the a-distance is 39.08%, if randomly choose 36 key nodes to deploy, the a-distance is 17.82%. When we look into the details of the simulation trace, we see that there are only 4 nodes that do not match the VIP nodes using Algorithm Inc-UDeploy(), while there are 68 VIP nodes not covered using RD, 31 VIP nodes not covered by HD. We emphasize again that the essence of incremental deployment is to demonstrate the benefits of TwoD-IP routing when deploying as few nodes as possible. Clearly, our algorithm achieves this.

In Fig. 5(a), we see that for the paths computed by RD, the a-distance to the VIP paths can even increase after more nodes are deployed. For example, when 36 nodes are deployed, the a-distance is 36.78%. After we deploy another 3 nodes, the a-distance increases to 37.36%. On the contrary, we see in Fig. 5, this never happens to Inc-UDeploy(). We also evaluate adaptive forwarding Adp-Forward() in Fig. 5(a). We see that its impact is quite small. Both HD and Inc-UDeploy() will achieve optimal performance after deploying all key nodes, however, the a-distance of Inc-UDeploy() decreases much faster than HD.



(a) Comparison of different deployment algorithms

(b) Impact of step size

Fig. 5. a-distance as a function of number of deployed nodes. (a) Comparison of different deployment algorithms. (b) Impact of step size.

Fig. 5(b) shows the impact of the incremental step size. We compare the performance for four step sizes, 1, 2, 5, 10 (i.e., 1, 2, 5, 10 nodes are deployed in each incremental step). We see that there is not much difference between different step sizes. Therefore, our heuristic algorithm, using small step size to reduce the computational complexity, is satisfactory.

In Fig. 6, we study the computation time of each algorithm. Because RD and HD both consume negligible computation time, we only study Inc-UDeploy(). We can see that the computation time increases with the number of deployed nodes, because we have to run UOpt-Deploy() for more times with more deployed nodes. The computation time increases exponentially with the step size, for example, it only takes 1 second to compute 40 deployed nodes when the step size is 1, while it takes more than 5 minutes when the step size is 10. This is because the complexity of UOpt-Deploy() increases exponentially with the number of deployed nodes in each step, according to Theorem 3.

In Fig. 7, we study the impact of the network size. We see that for Inc-UDeploy() and HD, the a-distance decreases when the network size increases. For RD, the a-distance increases when the network size increases. This is because when the network is larger, the number of nodes that belong to VIP paths increases slower than the total number of nodes. Since Inc-UDeploy(), and HD always choose from this set of nodes, the performance improves. On the contrary, RD selects randomly from all nodes, making its performance decrease. In Fig. 7, we also compare different deploy algorithms within two different deployment ratios, 10%, 20% (i.e., 10%, and 20% of all nodes are deployed). Clearly, the more nodes deployed, the shorter the a-distance. And by deploying 10% more nodes within RD, the improvement (i.e., the gap between 10% and 20% deployment ratio) is smaller than Inc-UDeploy() and HD. Although the a-distance both Inc-UDeploy() and HD decrease with network size, the a-distance of Inc-UDeploy() drops more quickly to be zero, especially when deployment ratio is low. The result indicates that better performance can be achieved by deployed carefully selected nodes based on our algorithm, especially at the initial stage of incremental deployment.

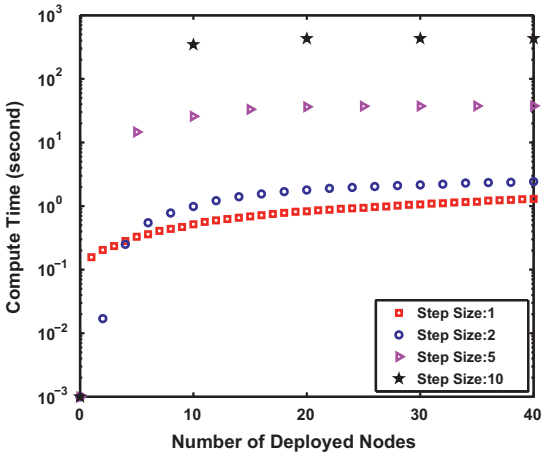


Fig. 6. Computation time of Inc-UDeploy().

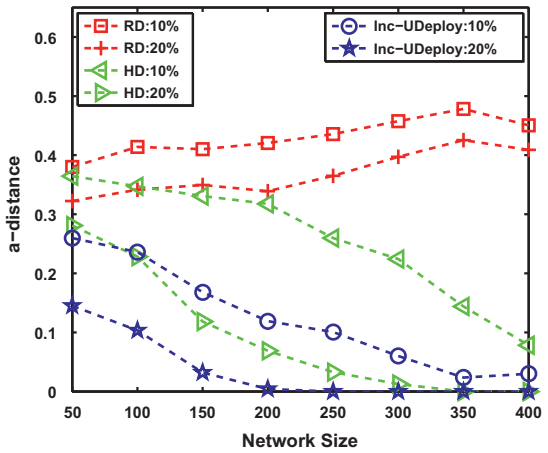


Fig. 7. a-distance as a function of network size.

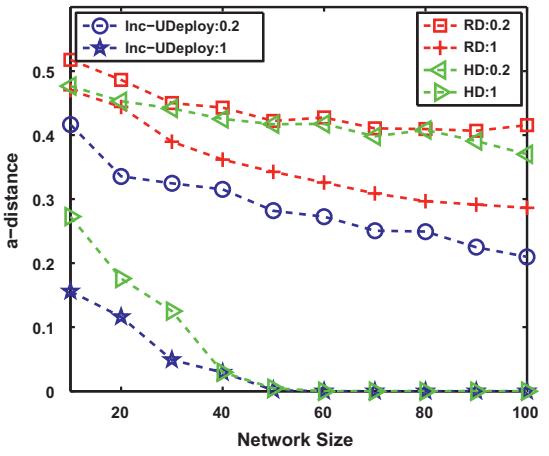


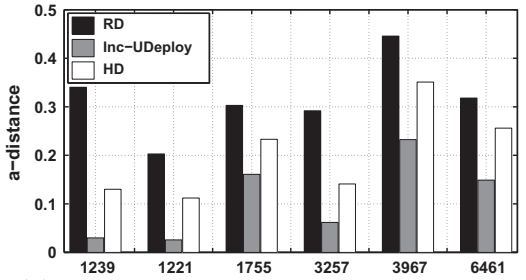
Fig. 8. a-distance as a function of number of VIP flows.

In Fig. 8, we study the impact of the number of VIP traffic flows. In practice, when the number of VIP traffic flows increases, we will also deploy more nodes to be TwoD-IP capable. Thus, we set an increasing ratio x between the number of VIP flows and the nodes deployed, i.e., every x additional nodes will be deployed when there is one additional VIP traffic flow. We compare different deploy algorithms within two different ratios 1 and 0.2. We see that the a-distance decreases when the number of VIP flows increases for all algorithms. This shows that deploying more nodes has higher positive impact. RD performs the worst, even the increasing ratio of RD is 1, its performance is still worse than Inc-UDeploy() with increasing ratio of 0.2. When the ratio is high, the a-distance of HD is quite small. However, when the ratio is low, the a-distance of HD is almost as worse as RD.

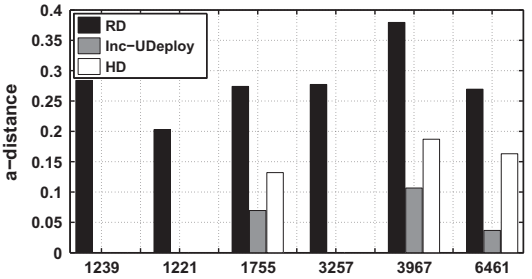
In Fig. 9, we show the performance of different algorithms when the deployment ratio is 0.1 or 0.2. We can see that Inc-UDeploy() performs much better than HD and RD, especially in large scale networks, such as AS 1239. HD performs better than RD, because HD selects deployed nodes in key nodes. In Fig. 9(b), when the number of deployed nodes exceeds the number of key nodes, the a-distance of Inc-UDeploy and HD both achieve 0. This further validates our results.

6.2.2. Adaptive forwarding

Fig. 10 shows the impact of adaptive forwarding on random deployment processes (RD and HD) that do not follow the optimal incremental deployment. We deploy three nodes in every incremental step. We see that adaptive forwarding improves both RD and HD. For example, if we randomly choose 90 nodes to deploy, the a-distance is 34.71%, however, the a-distance decreases to be 27.65% if we use adaptive forwarding; if we randomly choose 12 key nodes to deploy, the a-distance is 28.49%, and the a-distance decreases to be 22.93% if adaptive forwarding is used. In the extreme case, adaptive forwarding improves RD by 7.1% maximally, and improves HD by 5.5% maximally. Adaptive forwarding has greater impact on RD, because RD will also



(a) a-distance in different topologies when deployment ration is 0.1



(b) a-distance in different topologies when deployment ration is 0.2

Fig. 9. a-distance in different topologies. (a) A-distance in different topologies when deployment ration is 0.1. (b) A-distance in different topologies when deployment ration is 0.2.

choose non-key nodes, which can also act as relay nodes to further decrease the a-distance.

In Fig. 11, we study the benefits of adaptive forwarding when VIP flows change. After setting up 30 VIP flows and choosing 30 nodes to deploy according to Inc-UDeploy(), we generate 10–70 new VIP flows. We see that when the number of new VIP flow increases, the a-distance also increases. The more the new VIP flows, the larger the a-distance. However, with adaptive forwarding, the a-distance increases slower, i.e., the gap between Inc-UDeploy() and Adp-Forward() becomes larger when more VIP flows are

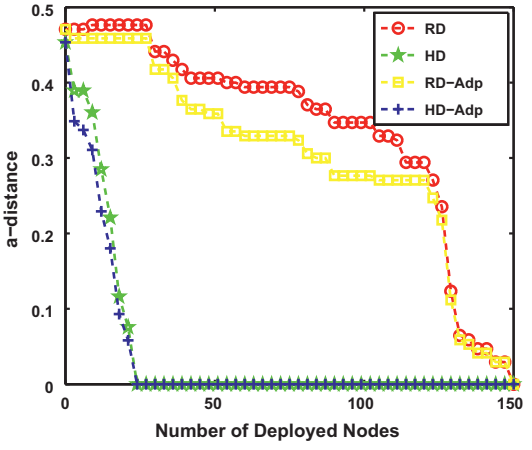


Fig. 10. A-distance as a function of number of deployed nodes.

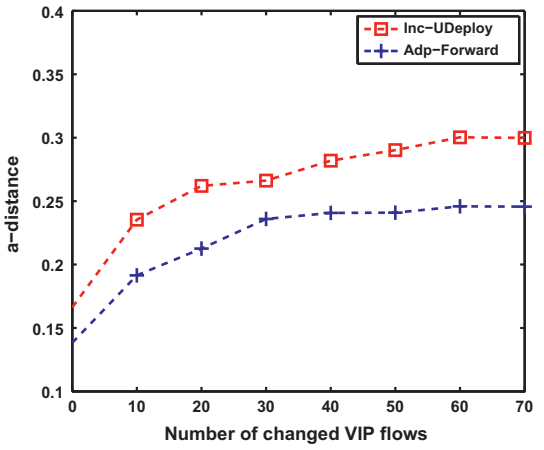


Fig. 11. A-distance as a function number of vip flows.

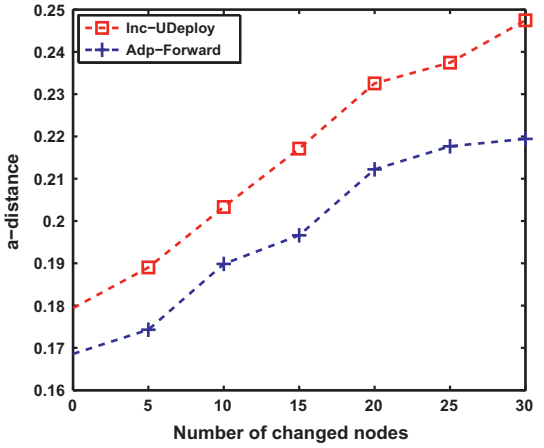


Fig. 12. A-distance as a function of number of changed nodes.

generated. This is because adaptive forwarding can manage the VIP flows that are away from their pre-defined VIP paths.

In Fig. 12, we study the benefits of adaptive forwarding when topology changes. After deploying 30 nodes according to Inc-UDeploy(), we randomly add 5-30 new nodes[5]. The a-distance will increase when more nodes are added. This is because when topology changes, the forwarding action (e.g., the next hop on the shortest path) of each node may change. Thus the a-distance will increases because the deployed nodes are computed according to the original topology. With adaptive forwarding, the a-distance increases slower. The result shows that we can achieve substantial benefits through adaptive forwarding when VIP flows or network topology change.

6.2.3. Incremental constrained deployment

Fig. 13 shows an incremental deployment process with capacity on each node be 16. We also deploy three nodes in every incremental step. In Fig. 13, we see that the a-distance decreases when we deploy more nodes, and Inc-GDeploy() always outperforms RD and HD. For example, after deploying 18 nodes, the a-distance computed by Inc-GDeploy() is 18.33%, while the a-distance computed by HD is 35.0%. After deploying 27 nodes, the a-distance computed by Inc-GDeploy() is only 10.0%, while the a-distance computed by RD is 39.40%. We see that the performance is worse than Inc-UDeploy() in Fig. 5. This is not surprising as we have more constraints in the routers. In addition, different from Fig. 5, the a-distance of the constrained deployment will not decrease to 0 even all nodes get deployed. This is because some bottleneck nodes cannot forward along the pre-defined paths.

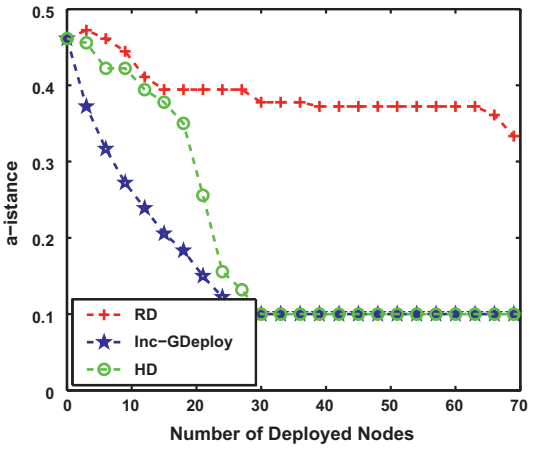


Fig. 13. A-distance as a function of number of deployed nodes.

We then study the impact of the capacity of each node in Fig. 14 within two deployment ratios, 10% and 20%. We see that a-distance decreases when capacity increases. We can also see that when the capacity is 15, the performance is the same as unconstrained deployment. This also suggests that in general TwoD-IP routing does not incur an additional 32-bit processing overhead for the routers.

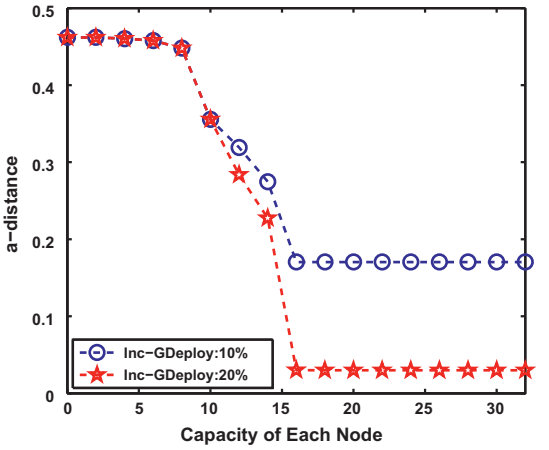
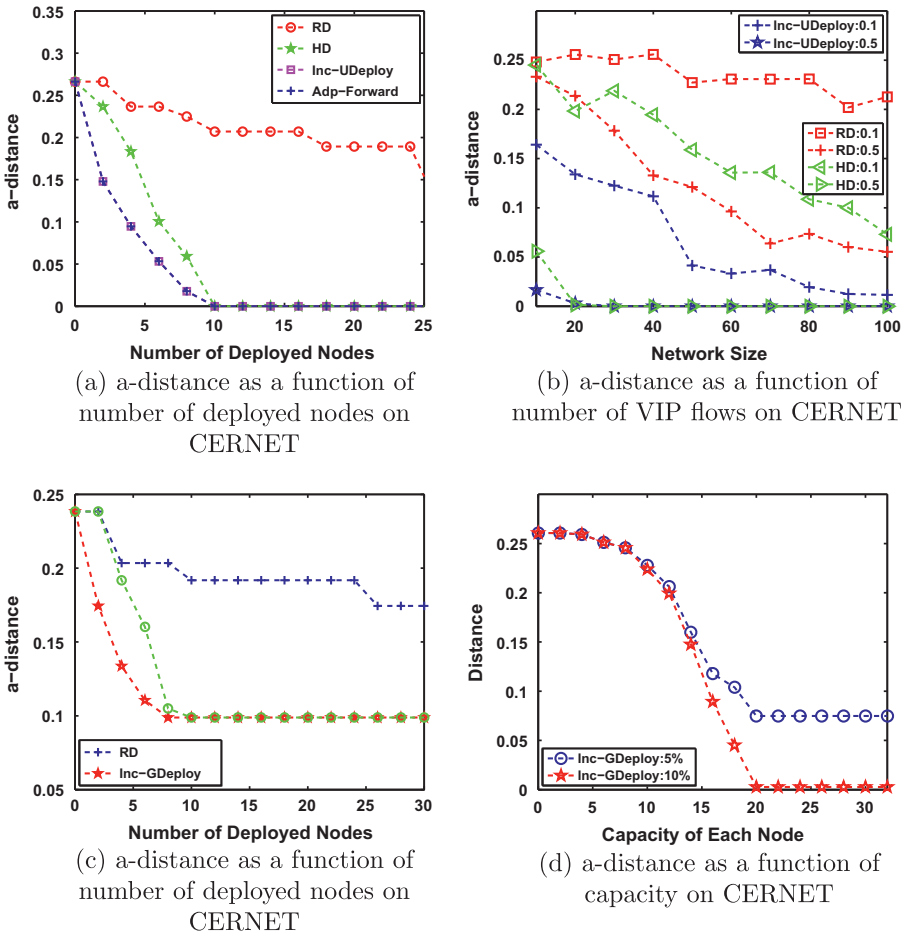


Fig. 14. A-distance as a function of capacity.

[5] The influenced VIP flows will be regenerated



(a) a-distance as a function of number of deployed nodes on CERNET

(b) a-distance as a function of number of VIP flows on CERNET

(c) a-distance as a function of number of deployed nodes on CERNET

(d) a-distance as a function of capacity on CERNET

Fig. 15. Simulation on CERNET. (a) A-distance as a function of number of deployed nodes on CERNET. (b) A-distance as a function of number of VIP flows on CERNET. (c) A-distance as a function of number of deployed nodes on CERNET. (d) A-distance as a function of capacity on CERNET.

6.2.4. Simulation results on CERNET

We further validate our results using CERNET.

Fig. 15(a) shows an incremental unconstrained deployment process by deploying two nodes at each step. We see that the a-distance is 26.7% even we do not deploy any router, this is because of the low connectivity of CERNET topology. When we deploy more nodes, the a-distance decreases and Inc-UDeploy() performs much better than RD and HD. After deploying 8 nodes, the a-distance of Inc-UDeploy() is 1.78%, the a-distance of RD is still 22.49%, and the a-distance of HD is 5.92%. Fig. 15(b) shows the impact of the number of VIP flows. Here, we set the increasing ratio to be 0.1 and 0.5, i.e., deploying 0.1 and 0.5 nodes after adding a VIP flow. In Fig. 15(b), a-distance decreases with the number of VIP flows. Inc-UDeploy() decreases faster and performs the best. HD achieves nearly optimal result when the ratio is high, but much worse when the ratio is low.

Fig. 15(c) shows an incremental constrained deployment process. When we deploy more nodes, a-distance decreases and Inc-GDeploy() outperforms both RD and HD. In Fig. 15(d), we study the impact of node capacity on Inc-GDeploy and RD within two different deployment ratios, 5% and 10%, we can see that a-distance decreases with capacity. The simulation results is similar with the results on BRITE generated topologies.

7. TwoD-IP Routing for CERNET2: a case study

We conduct a case study with the real topology and prefix information of CERNET2. Our work also serves as a reference for the future deployment of TwoD-IP routing on CERNET2.

We want to move the out-going International traffic of three universities, i.e., THU (in Beijing, with 38 prefixes), HUST (in Wuhan, with 18 prefixes) and SCUT (in Guangzhou, with 28 prefixes) to CNGI-SHIX (Shanghai portal). There will be three VIP paths, $VIP(t_0)$ ={Beijing, Tianjin, Jinan, Hefei, Nanjing, Shanghai}, $VIP(t_1)$ ={Wuhan, Nanjing, Shanghai} and $VIP(t_2)$ ={Guangzhou, Xiamen, Hangzhou, Shanghai}.

In the CERNET2 scenario, we apply a variation of the weighted version of our problems. Each VIP traffic flow t is assigned a weight w_t that is proportional to the traffic volume. Besides, each node $v \in VIP(t)$ is also assigned a weight $w_t(v)$, which is set according to the importance of

node v. $w_t(v)$ satisfies that $\sum_{v\in VIP(t)} w_t(v) = w_t$. Let the total distance be $\sum_i (w_t - \sum_v w_t(v) \times I_{VIP(t)\cap\mathcal{L}(\mathcal{F}(P_s))}(v))$ where $I_A(w)$ is an indicator function (if $w \in A, I_A(w) = 1$, else $I_A(w) = 0$). The solutions above can be easily modified to solve the variant version .

In our case, the traffic volume from THU is 1955 (Mbps), the traffic volume from HUST is 510 (Mbps), and the traffic volume from SCUT is 1107 (Mbps). Thus we set the weights to $w_{t_0} = 19.6, w_{t_1} = 5.1, w_{t_2} = 11.0$. Because our purpose is to let the traffic flows travel through Shanghai to CNGI-SHIX (let v_s represent the router in Shanghai), thus let $w_{t_0}(v_s) = 19.6,\ w_{t_1}(v_s) = 5.1,\ w_{t_2}(v_s) = 11.0,$ and $w_t(v) = 0,\ \forall t,\ \forall v \neq v_s$.

Fig. 16(a) shows the incremental unconstrained deployment process, where we deploy nodes one by one. We compare Inc-UDeploy() and HD algorithm (here nodes are randomly selected from the VIP paths, the candidate node set includes Beijing, Tianjin, Jinan, Hefei, Nanjing, Shanghai, Wuhan, Guangzhou, Xiamen, Hangzhou). For Inc-UDeploy(), the deployment sequence we suggest is Guangzhou, Wuhan, Jinan, Tianjin and Beijing. We see that after deploying the router of Guangzhou (occupies 30.8% of the total traffic), the a-distance falls from 100% to 69.2%. For HD to achieve similar performance, we need to deploy routers at five cities. (See Fig. 17).

Fig. 16(b) shows the size of minimum bit-lookup set on each router. Note that CERNET2 is IPv6; thus the IP addresses for lookup are 128 bits. Out of 128 bits, each router has to check at most 22 bits in the source addresses. The normal TCAM width is 36, 72, 144 bits, and CERNET2 is using 144 bits TCAM now. Thus, the width can be reduced to be 36 bits using the minimum bit-lookup set. If CERNET2 switches to using multi-bit trie (with strides of 4 bits), then the number of accesses in memory per lookup can be reduced from $\lceil \frac{128}{4} \rceil = 32$ to $\lceil \frac{22}{4} \rceil = 6$. This proves that ISPs can benefit from our algorithms no matter trie-based or TCAM-based storage structures are used.

In Table 6, we show the current maximum utilization of 5 key routers in CERNET2. The most loaded one (Guangzhou), even at its peak state, has more than 35% redundant processing capacity. As we also plan to add more linecards for a router if it is chosen to be deployed, we consider it might be reasonable to assume that the routers are capable to handle the overheads for TwoD-IP routing.

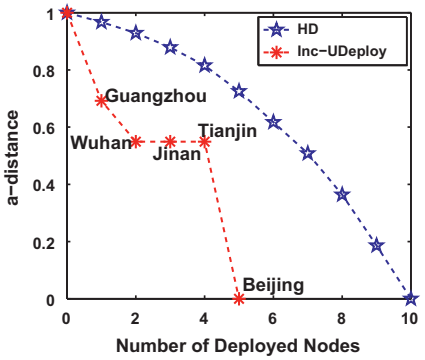
8. Related work

Recent years see more research on improving routing with richer semantics. Such as policy based routing [3], customer-specific routing [33], user-directed routing [34], class-based routing [6], and source/destination routing [7]. They all add more information into the routing system, to satisfy various demands from users.

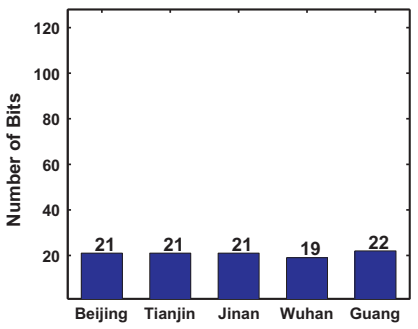
Due to security and accounting problems, CERNET2 has deployed SAVI (Source Address Validation Improvement) [35] to validate the source address of each packet at the edge points of the network. Currently, confirmed SAVI users are more than 900,000 [36]. CERNET2 then decides to integrate the source address lookup into IP routing and deploy TwoD-IP.

Higher dimensional routing introduces overheads on routers, including lookup time and storage. For fast lookup, previous studies make use of TCAM in hardware, the de facto standard in industry due to its constant lookup speeds. However, TCAM-based solutions have limited capacity, high price and high power consumption [37]. Many previous works proposed algorithmic solutions, such as trie-based solutions [9]. Unlike TCAM-based solutions, these solutions need multiple accesses in memory during each lookup. [38,39] propose different approaches to make the trie-based structure more compact, and reduce the number of accesses for faster speed. In this paper, to make our study focused, we use the number of bits to be checked as an indicator for the overheads. We want to emphasize that less bits also lead to less storage and less energy consumption with both TCAM-based or trie-based solutions.

Incremental design is advocated [40] for network layer proposals. Many incremental solutions have been



(a) a-distance as a function of number of deployed nodes on CERNET2



(b) Size of minimum bit-lookup set on routers on CERNET2

Fig. 16. Simulation on CERNET2. (a) A-distance as a function of number of deployed nodes on CERNET2. (b) Size of minimum bit-lookup set on routers on CERNET2.

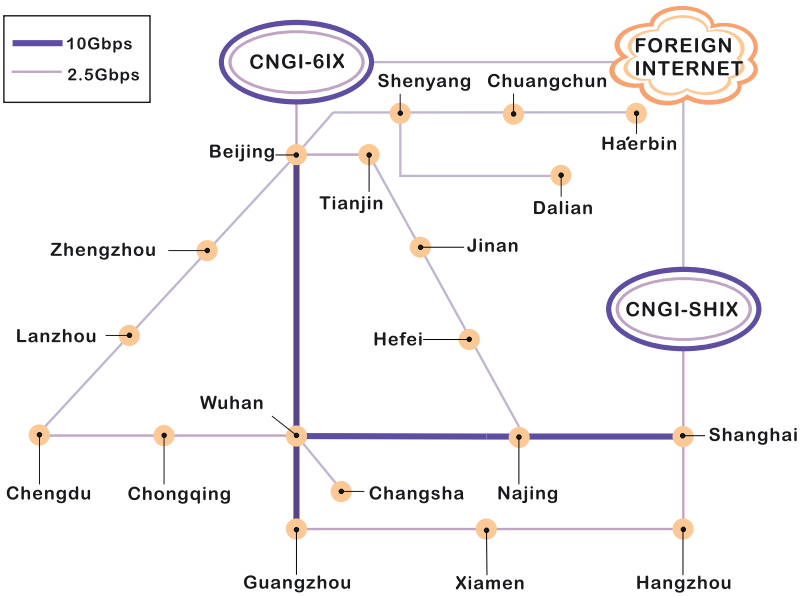


Fig. 17. CERNET2 topology.

Table 6
Maximum utilization of routers in CERNET2.

Beijing	Jinan	Tianjin	Wuhan	Guangzhou
28.57%	39.56%	46.82%	30.20%	64.28%

proposed for IPv6 routing [41], interdomain security routing [42], multicast routing [43], information centric network [44], overlay network [13], and various future Internet routing schemes [45]. In this paper, we focus on high dimensional routing, which needs a design from scratch. Beside, most previous works just sketch their incremental plans, while we want to find the route-level deployment, that lets ISPs gain the most and pay the least during partial deployment.

We use TwoD-IP as an example, because (1) it make our paper succinct while it can be easily extended to higher dimensions (e.g., including DSCP values and flow labels); (2) destination and source addresses contain the most important information in networks; and (3) CERNET2 is facing such deployment issues in the real-world.

9. Conclusion and future work

In this paper, we presented a study of TwoD-IP routing where the forwarding decisions is not only based on the destination IP addresses but also on the source IP addresses. Our focus is on incremental deployment requirement, a practical concern of CERNET2. We formulated our problem such that we need to find a deployment sequence under the constraints of router capacity. We proved the problem to be NP-complete. We then transformed our problem to boolean clauses and developed an efficient algorithm following the principles of branch-and-bound algorithm for MAX-SAT.

We evaluated our algorithms comprehensively using CERNET and other topologies. We showed that by deploying a few nodes suggested by our deployment sequence can successfully manage the traffic flows and introduces small overheads to the routers. We then presented a case study on CERNET2 and provided a fine deployment sequence.

In our current study, we did not take network failure into consideration. We put it into our future work as network failure recovery could cause traffic rerouting. In [46], a general architecture is proposed for joint failure recovery and traffic engineering where it could be applied to our scenario.

Based on our specific consideration, we use hamming metric to represent the deviation between paths. While there are other meaningful metrics in graph theory, using some of them (such as Jaccard index [47]) also has the property, where the base configuration of each iteration evolves from the final state of the previous iteration. In this paper, we just use hamming distance as an example, because it is simple, widely used and meaningful in real-world. Our next work includes trying different metrics for other considerations. We assertively use summation of individual path distance to express the total deviation, validation and comparison with other metrics (e.g., maximum path deviation) also belong to our future work.

Acknowledgment

The research is supported by the National Basic Research Program of China (973 Program) under Grant 2012CB315803, the National Natural Science Foundation

of China (61073166, 61133015),the National High-Tech Research and Development Program of China (863 Program) under Grants 2011AA01A101.

Appendix A

A.1. Proof for Observation 2

Proof. $D(VIP(t), \mathcal{L}(\mathcal{G}, \mathcal{F}, P_s)) = 0$ if and only if $\mathcal{C}(v, P_s(t)) = 1, \forall v \in VIP(t)$.

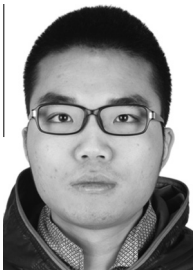
It's easy to prove that it is a sufficient condition. With precondition that $\mathcal{C}(v_t^0, P_s(t))$ is 1, suppose that for all $j < k$, $\mathcal{C}(v_t^j, P_s(t))$ are 1. Then if $s\_child(v_t^{k-1}, P_s(t)) = v_t^k$, $\mathcal{C}(v_t^k, P_s(t))$ must be 1, else according to Observation 2 we have $\theta_{v_t^{k-1}} = 1$, thus $\mathcal{C}(v_t^k, P_s(t))$ is 1. So we can conclude that all clauses $\mathcal{C}(v, P_s(t))$ are 1.

Then we prove that the condition is necessary. According to Algorithm 1, if $s\_child(v_t^j, P_s(t)) \neq v_t^{j+1}$ and $\theta_{v_t^j} \neq 1$, there must exist $k_0 < j$ such that $s\_child(v_t^{k_0}, P_s(t)) = v_t^{j+1}$ ($s\_child(v_t^{k_0}, P_s(t)) \neq v_t^{k_0+1}$) and $\theta_{v_t^{k_0}} \neq 1$. Indicating there must be $k_1 < k_0$ such that $s\_child(v_t^{k_1}, P_s(t)) \neq v_t^{k_1+1}$ and $\theta_{v_t^{k_1}} \neq 1$. Thus there is a integer sequence $k_0 > k_1 > k_2 \ldots$ until $k_l = 0 (l > 0)$, obviously, if $s\_child(v_t^0, P_s(t)) \neq v_t^1, \theta_{v_t^0}$ must equal to 1 (else $\mathcal{C}(v_t^1, P_s(t))$ will be 0). Thus the assumption is false. □

References

[1] C. De Launois, M. Bagnulo, The paths toward IPV6 multihoming, Communications Surveys Tutorials, IEEE 8 (2) (2006) 38–51.
[2] F. Baker, P. Savola, Ingress Filtering for Multihomed Networks, RFC 3704 (Best Current Practice), March 2004.
[3] Cisco, Policy-Based Routing (White Paper), 1996.
[4] Juniper, Multi-Topology Routing (White Paper), Augest 2010.
[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Openflow: enabling innovation in campus networks, SIGCOMM Computer Communication Review 38 (2) (2008) 69–74.
[6] F. Baker, Routing a traffic class, Internet Draft, draft-baker-fun-routing-class-00, January 2012.
[7] F. Baker, IPv6 Source/Destination Routing using OSPFv3, Internet Draft, draft-baker-ipv6-ospf-dst-src-routing-00, February 2013.
[8] M. Xu, S. Yang, D. Wang, J. Wu, Two dimensional-ip routing, in: Proc. IEEE ICNC'13, San Diego, USA, 2013.
[9] G. Varghese, Network Algorithmics: An Interdisciplinary Approach to Designing Fast Networked Devices, Morgan Kaufmann, Waltham, MA, 2005.
[10] F. Yu, R. Katz, T. Lakshman, Gigabit rate packet pattern-matching using tcam, in: Proc. IEEE ICNP'04, Berlin, Germany, 2004.
[11] B. Agrawal, T. Sherwood, Modeling tcam power for next generation network devices, in: Proc. IEEE ISPASS'06, Austin, Texas, 2006.
[12] R. Zhang-Shen, N. McKeown, Designing a predictable internet backbone with valiant load-balancing, in: Proc. IEEE IWQoS'05, Passau, Germany, 2005.
[13] X. Yang, D. Wetherall, Source selectable path diversity via routing deflections, in: Proc. ACM SIGCOMM'06, New York, NY, 2006.
[14] S. Ho, M. Gendreau, Path relinking for the vehicle routing problem, Journal of Heuristics 12 (2006) 55–72.
[15] L. Trevisan, When hamming meets euclid: the approximability of geometric tsp and mst, in: Proc. ACM STOC'97, El Paso, TE, 1997.
[16] V. Sekar, S. Ratnasamy, M.K. Reiter, N. Egi, G. Shi, The middlebox manifesto: enabling innovation in middlebox deployment, in: Proc. ACM HotNets'11, Cambridge, Massachusetts, 2011.
[17] K. Fall, P.B. Godfrey, G. Iannaccone, S. Ratnasamy, Routing tables: is smaller really much better? in: Proc. ACM HotNets'09, New York, NY, 2009.
[18] U. Feige, G. Kortsarz, D. Peleg, The dense k-subgraph problem, Algorithmica 29 (2001) 410–421.
[19] T. Alsinet, F. Manya, J. Planes, Improved branch and bound algorithms for max-sat, in: Proc. Theory and Applications of Satisfiability Testing (SAT'03), Portofino, Italy, 2003.
[20] B. Selman, H. Levesque, D. Mitchell, A new method for solving hard satisfiability problems, in: Proc. of the Tenth National Conference on Artificial Intelligence (AAAI'92), San Jose, CA, 1992.
[21] J. Hooker, V. Vinay, Branching rules for satisfiability, Journal of Automated Reasoning 15 (1995) 359–383.
[22] C. Labovitz, G. Malan, F. Jahanian, Internet routing instability, IEEE/ACM Transactions on Networking 6 (5) (1998) 515–528.
[23] C. Meiners, A. Liu, E. Torng, Hardware Based Packet Classification for High Speed Internet Routers, Springer, 2010.
[24] V. Srinivasan, G. Varghese, Faster ip lookups using controlled prefix expansion, in: Proc. ACM SIGMETRICS '98, Madison, Wisconsin, United States, 1998.
[25] P. Slavík, A tight analysis of the greedy algorithm for set cover, in: Proc. ACM STOC'96, Philadelphia, Pennsylvania, United States, 1996.
[26] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, New York, NY, 1990.
[27] U. Feige, A threshold of ln n for approximating set cover, Journal of the ACM 45 (1998) 634–652.
[28] J. Argelich, Max-sat formalisms with hard and soft constraints, AI Communications 24 (2011) 101–103.
[29] Brite: Boston University Representative Internet Topology Generator. <http://www.cs.bu.edu/brite>.
[30] X. Zhao, Y. Liu, L. Wang, B. Zhang, On the aggregatability of router forwarding tables, in: Proc. IEEE INFOCOM'10, San Diego, CA, 2010.
[31] Q. Li, D. Wang, M. Xu, J. Yang, On the scalability of router forwarding tables: Nexthop selectable fib aggregation, in: Proc. IEEE INFOCOM'11, Mini-Conference, Shanghai, China, 2011.
[32] N. Spring, R. Mahajan, D. Wetherall, T. Anderson, Measuring isp topologies with rocketfuel, IEEE/ACM Transactions on Networking 12 (1) (2004) 2–16.
[33] J. Fu, J. Rexford, Efficient ip-address lookup with a shared forwarding table for multiple virtual routers, in: Proc. ACM CoNEXT'08, Madrid, Spain, 2008.
[34] P. Laskowski, B. Johnson, J. Chuang, User-directed routing: from theory, towards practice, in: Proc. ACM NetEcon'08, Seattle, WA, USA, 2008.
[35] J. Wu, J. Bi, M. Bagnulo, F. Baker, C. Vogt, Source address validation improvement framework, Internet Draft, draft-ietf-savi-framework-04, March 2011.
[36] Cngi-cernet2 savi Deployment Update, March 2011. <http://www.ietf.org/proceedings/80/slides/savi-2.pdf>.
[37] C.R. Meiners, A.X. Liu, E. Torng, J. Patel, Split: optimizing space, power, and throughput for tcam-based classification, in: Proc. ACM/IEEE ANCS'11, Brooklyn, NY, 2011.
[38] H. Song, J. Turner, J. Lockwood, Shape shifting tries for faster ip route lookup, in: Proc. IEEE ICNP'05, Boston, MA, USA, 2005.
[39] M. Degermark, A. Brodnik, S. Carlsson, S. Pink, Small forwarding tables for fast routing lookups, in: Proc. ACM SIGCOMM'97, Cannes, France, 1997.
[40] D. Thaler, B. Aboba, What Makes for a Successful Protocol? in: RFC 5218 (Informational), July 2008.
[41] J. Wu, Y. Cui, C. Metz, E. Rosen, Softwire Mesh Framework, RFC 5565 (Standards Track), June 2009.
[42] J. Rexford, J. Feigenbaum, Incrementally-deployable security for interdomain routing, in: Proc. IEEE CATCH'09, Washington, DC, USA, 2009.
[43] X. He, C. Papadopoulos, P. Radoslavov, Incremental deployment strategies for router-assisted reliable multicast, IEEE/ACM Transactions on Networking 14 (4) (2006) 779–792.
[44] S. Fayazbakhsh, A. Tootoonchian, Y. Lin, A. Ghodsi, K. Ng, B. Maggs, V. Sekar, S. Shenker, Less pain, most of the gain: incrementally deployable icn, in: Proc. ACM SIGCOMM'13, Hongkong, China, 2013.
[45] R. Grandl, D. Han, S.-B. Lee, H. Lim, M. Machado, M. Mukerjee, D. Naylor, Supporting network evolution and incremental deployment with xia, SIGCOMM Computer Communication Review 42 (4) (2012) 281–282.
[46] M. Suchara, D. Xu, R. Doverspike, D. Johnson, J. Rexford, Network architecture for joint failure recovery and traffic engineering, in: Proc. ACM SIGMETRICS'11, San Jose, CA, 2011.
[47] Jaccard Index. <http://www.en.wikipedia.org/wiki/Jaccard_index>.

Mingwei Xu received the B.Sc. degree and the Ph.D. degree from Tsinghua University. He is a full professor in Department of Computer Science at Tsinghua University. His research interest includes computer network architecture, high-speed router architecture and network security. He is a member of the IEEE.



Shu Yang received the B.Sc. degree from Beijing University of Posts and Telecommunications. He is a Ph.D. candidate in Department of Computer Science at Tsinghua University. His research interest includes computer network architecture, network security and high performance router.

Dan Wang received his B. Sc from Peking University, Beijing, M. Sc from Case Western Reserve University, Cleveland, OH, and Ph. D. from Simon Fraser University, Vancouver, Canada, all in Computer Science. He is an Assistant Professor of Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. His research interest includes Sensor Networks, Internet Routing and Applications. He is a member of the IEEE.

Jianping Wu received his B.S., M.S., and Ph.D. from Tsinghua University. He is a Full professor and director of Network Research Center, Ph.D. Supervisor of Department of Computer Science, Tsinghua University. From 1994, he has been in charge of China Education and Research Network (CERNET). His research interests include next generation Internet, IPv6 deployment and technologies, Internet protocol design and engineering. He is an IEEE fellow.