Efficient Virtual Network Optimization across Multiple Domains without Revealing Private Information

Toru Mano*, Takeru Inoue*, Dai Ikarashi†, Koki Hamada†, Kimihiro Mizutani*, Osamu Akashi*,
*NTT Network Innovation Labs., Kanagawa 239–0847, Japan
†NTT Secure Platform Labs., Tokyo 180–0012, Japan

Email: mano.toru@lab.ntt.co.jp, takeru.inoue@ieee.org

Abstract—Building optimal virtual networks across multiple domains is an essential technology to offer flexible network services. However, existing research is founded on an unrealistic assumption; providers will share their private information including resource costs. Providers, as is well known, never actually do that to remain competitive. Technically, secure multi-party computation, which is a computational technique based on the cryptography, can be used to secure optimization, but it is too time-consuming. This paper presents a novel method to optimize virtual networks built over multiple domains, with great efficiency but without revealing any private information. Our method employs secure multi-party computation but only for masking sensitive values; it can optimize virtual networks under limited information without any time-consuming technique. It is solidly based on the theory of optimality, and is assured of finding reasonably optimal solutions. Experiments show that our method is fast and optimal in practice even concealing private information; it finds nearly optimal solutions in just a few minutes for large virtual networks with tens of nodes. This is the first work that can be implemented in practice for building optimal virtual networks across multiple domains.

Index Terms—network virtualization, network optimization, secure multi-party computation

I. INTRODUCTION

Network virtualization [1]–[3] allows logically separated virtual networks (VNs) to coexist on a common physical infrastructure, and it provides network operators with flexibility, diversity, security, and manageability. Originally studied to resolve the so-called Internet ossification problem [4], it has become essential for offering QoS service and infrastructure as a service [5]. In network virtualization, the role of traditional ISPs is divided into those of service providers (SPs) and infrastructure providers (InPs); a SP makes a request to an InP for building a VN, while the InP embeds the VN onto its physical network. VN embedding is an optimization problem, in which the InP attempts to find a minimum cost mapping between the desired VN and its physical network with respect to constraints of physical resources (nodes and links) [3]. Although this problem is known to be \textsc{np}-hard [6], many heuristics have been developed to locate better mappings [6]–[9] since InPs compete with mapping cost to win the contract.

A VN can be built across multiple InPs, and doing so will be imperative in the future of network virtualization; VNs across multiple InPs, or inter-InP VNs, give SPs a lot of benefits, such as end-to-end service delivery for several regions [10], business continuity under severe accidents, and diversity of resource functionality and performance. SPs want to get these optimal VNs at the lowest price, but this optimization is hard for SPs and InPs since no entity knows the entirety of the physical networks unlike the single InP case. Past research work on this problem [5], [10]–[13] assumed that InPs would share their private information like resource costs or prices with SPs and other InPs, but InPs have traditionally been reluctant to disclose them [14]; e.g., the disclosure allows their competitors to estimate their future bidding prices (InPs do not want to share their private information with SPs as well as other InPs though SPs are not their competitors, because SPs can often be InPs in another bid). Despite much research effort on optimizing inter-InP VNs [5], [10]–[13], their optimization methods can never be implemented in practice as long as they rely on the assumption of private information revelation.

Secure multi-party computation (MPC) [15], which enables participants to jointly compute a function while keeping the inputs private, can be used to solve this problem. However, it could take months to embed a VN to several InPs [16], since every primitive operation in MPC is executed in a distributed manner with attendant communication delays. Since candidate physical resources must be reserved during the computation, InPs have to pay very high opportunity costs [10]. In the past, MPC has been applied to various problems, including polynomial optimization problems [17], [18] and simple decision problems [19]–[21], but it has never been successfully applied to tough optimization problems like VN embedding.

In this paper, we propose a novel optimization method that minimizes inter-InP VN price without sharing InPs’ private information, while using minimal MPC operations. We show our basic idea in Fig. 1; each InP identifies the pieces of the requested VN that can be embedded to its physical network, and the SP uses just the price order calculated by an MPC sort algorithm to select the optimal set of pieces that covers the whole VN. The SP cannot estimate the price of any piece from the order (e.g., given that the SP might be an InP in a future bid, it cannot know the price used to win the bid).

1We confirmed this view by private interviews with network operators.
InPs are not informed of the price order, and so they cannot estimate the prices of the other InPs’ pieces at all. Since MPC is used just for sorting the pieces, our method can finish in a short time.

In order to resolve this optimization problem given only the price order of VN pieces, we develop a new theory that defines the optimality of SPs and InPs; this theory gives a solid guideline for designing algorithms to find optimal solutions for SPs and InPs.

- For SPs, we introduce a theorem to minimize the worst-case VN price based on just the price order. We then utilize a sophisticated search algorithm with a cost function derived from the theory, which efficiently finds the optimal set of pieces. (Section III-A)
- For InPs, we present another theorem that states that it is sufficient for InPs to enumerate just the Pareto set of pieces, since other pieces play no role in the SP’s optimization. Since the resulting algorithm enumerates many fewer pieces, the SP’s search space is greatly reduced without losing the opportunity of any InP for winning the bid. (Section III-B)

We also design protocols to securely execute this distributed optimization process using MPC; e.g., InPs’ private information is kept confidential, and InPs’ cheating acts to illegally win a bid are deterred. (Section IV)

Thorough numerical experiments show that our method finds VNs that are twice as optimal than those without price order; surprisingly, the VNs found by our method are only 20% worse than those found with full price value disclosure. Our method requires just a couple of minutes to find optimal VNs even those with 40 nodes, due to the limited use of MPC. (Section V)

Our method is designed to be practical enough to follow the complex policies of SPs and InPs; no existing work concerns itself with both SP and InP policies, as discussed in related work in Section VI. Our method allows InPs to filter out embeddable VN pieces, if these pieces violate their policies. SPs can examine VN pieces while considering InP reputation.

This is the first work acceptable for InPs, since our method does not force them to share their private information. SPs are encouraged to use our method, since it can find near optimal VNs in reasonable computation time. We believe that our method will play a key role in building VNs across multiple InPs in the future Internet.

II. Virtual Network Optimization with Multiple Providers

This section defines the VN optimization problem. A physical network is represented as an undirected graph consisting of physical nodes and links. Each physical node is associated with available capacity (CPU and/or memory) and type (functionalities and/or locations), and belongs to a single InP. Each link has available capacity (bandwidth), and belongs to an InP, or to two InPs if it is an inter-InP link. Nodes and links are both referred to as resources.

A virtual network is also an undirected graph, \( G = (V, E) \). Each virtual node and virtual link, \( v \in V \) and \( e \in E \), specifies the corresponding capacity requirement as, \( r(v) \) and \( r(e) \), respectively. A virtual node can be embedded in a physical node that has enough capacity with matched type. A virtual link is mapped to a physical path; all physical links on the path must have enough capacity, while a part of the path can be shared with other paths (Fig. 1(4) includes examples of a virtual link mapped to a physical path). Virtual links across InPs are configured with inter-InP links based on mutual agreement between the InPs [10]. Cost of virtual resources is determined based on required capacity and the physical resources to which the virtual resources are embedded. VN price, \( f(G) \), is determined by InPs based on the resource cost; we assume the price function is monotonic, i.e., \( f(G') \leq f(G) \) if \( G' \subseteq G \).

Embedding a VN in a single InP is an optimization problem in which the InP finds the minimum cost embedding satisfying the constraints of resource capacity and node type. SPs are not involved in the optimization process.
VN optimization across multiple InPs involves a SP and InPs. The SP attempts to find the optimal VN partition that minimizes the price of the whole VN, while each InP embeds a given piece of the VN in its physical network. This is quite a tough problem, since SP’s optimization methods cannot rely on InPs’ private information, such as resource costs and prices, available resource capacities, and node types. The price of each piece is determined by the corresponding InP, and InPs try to lease larger (more profitable) VN pieces to the SP. We assume that the SP doesn’t have both roll of SP and InP simultaneously. We also assume that each VN request is processed independently and separately, and so we consider a single request at a time in this paper.

III. THEOREMS AND ALGORITHMS

This section proposes two algorithms with related theorems. Section III-A discusses an algorithm in which a SP finds an optimal set of VN pieces as shown in Fig. 1(3), while Section III-B describes another algorithm for InPs to enumerate minimum necessary pieces as in Fig. 1(1).

A. Selecting Optimal Virtual Network by Service Provider

1) Definitions: We define a VN piece as a subgraph of VN, \( P \subseteq G \), which is embedded in a single InP. These pieces are enumerated by InPs, and they are used to cover the VN specified by the SP. Links that connect to another InP have resources at enumeration, since they are not commonly found in a given piece of the VN in its physical network. This is quite a tough problem, since SP’s optimization methods cannot rely on InPs’ physical networks, they just try to cover the whole VN without considering available capacities of physical resources. Capacity requirement of inter-InP virtual links will be examined after the optimization, while pieces (virtual resources within an InP) are mapped to physical resources at enumeration.

We begin with constraints. A SP selects zero or one piece from each InP:

\[
|\{ P \in X^i \cap X^j : P \text{ is a piece} \}| \leq 1 \ (\forall i = 1, \ldots, n),
\]

where \( n \) is the number of InPs. This is because InPs do not guarantee to embed multiple pieces. Any number of bonds can be selected if necessary. We refer to this constraint as \( g_1(X^i) \).

In order to connect a pair of open-ended virtual links, there must exist a sequence of bonds between the InPs; e.g., an open-ended virtual link, \( \{ u, v \} = e \), must be connected with a sequence of bonds between the InP of virtual node \( u \) and that of \( v \),

\[
\forall \{ u, v \} = e \in P \in X^u : e \text{ is open-ended},
\]

\[
\exists b_{InP(u,i)}, \ldots, b_{InP(v)} \in X^v,
\]

where virtual node \( v \) is mapped to InP\((v)\). This constraint is referred as \( g_2(X^i) \).

Given subset \( X^u \), we can determine the VN price as follows,

\[
f(X^u) = \sum_{i=1}^{n} f(X^i) + \sum_{e \in E^0(X^u)} \sum_{b \in B(e)} f_0(b)r(e),
\]

where \( E^0(X^i) \) is a set of inter-InP virtual links, and \( B(e) \) is a set of bonds on inter-InP virtual link \( e \). The first term means the price of pieces, and the second term is that of bonds.

We define the SP’s optimization problem as follows,

\[
\min_{X^u \subseteq X} f(X^u)
\]

s.t.

\[
g_1(X^u) \quad \text{and} \quad g_2(X^u),
\]

\[
\forall v \in V, \exists P \in X^u : v \in P,
\]

\[
\forall e \in E, \exists P \in X^u : e \in P.
\]
where constraints (3) and (4) ensure that the whole VN is covered.

The SP has to solve this problem without price values $f(X)$’s, i.e. using just the given $X$ sorted in descending order by extended price, $f(X)$. We denote the sorted list of $X$ by $[X_1, \cdots, X_m]$, where $m$ is the size of $X$ ($m=5$ in Fig. 1), and $X = \bigcup_{i=1}^m X_i$. Although it is quite hard to find an optimal solution without price values, we have the following theorem on the optimality yielded by the price order.

**Theorem 1.** Given a subset of pieces and bonds, $X^* \subseteq X$, the price of a VN built with them is bounded by,

\[
    f(X^*) \leq (1 + w_c)n\bar{f}_{\text{max}}(X^*) = O(\bar{f}_{\text{max}}(X^*)),
\]

where $\bar{f}_{\text{max}}(X^*)$ is the maximum extended price of elements contained in $X^*$, i.e., $\bar{f}_{\text{max}}(X^*) = \max\{f(X_k) : X_k \in X^*\}$.

**Proof:** The VN price of subset $X^*$ is bounded as follows,

\[
    f(X^*) \leq n\bar{f}_{\text{max}}(X^*) + \sum_{e \in E^b(X^*)} \sum_{b \in \mathcal{B}(e)} w_e\bar{f}_{\text{max}}(X^*) \sum_{e' \in E} r(e') r(e) \\
    \leq (1 + w_c)n\bar{f}_{\text{max}}(X^*),
\]

where we use the price monotonicity and $\bar{f}(b) = f_0(b)\sum_{e \in E} r(e)/w_e \leq \bar{f}_{\text{max}}(X^*)$.

**Lemma 1.** Given just the price order, the most rational strategy for minimizing the worst case price is to simply minimize the maximum price rank of the selected subset; i.e.,

\[
    \min_{X^* \subseteq X} \max\{k : X_k \in X^*\},
\]

under the same constraints, (1), (2), (3), and (4), where $\max\{k : X_k \in X^*\}$ is the maximum price rank of subset $X^*$.

This is because bound (5) is determined just by the maximum extended price, which is associated with the maximum rank.

---

**Algorithm 1:** Select optimal VN by SP

**Input:** $[X_1, \cdots, X_m]$ // sorted pieces/bonds list

**Output:** $X^* \subseteq X$ // optimal subset

1. $T \leftarrow \{\emptyset\}$
2. for $k \leftarrow 1$ to $m$ do
   3. if $X_k$ is a piece then
      4. $T \leftarrow T \cup \{X^* \cup \{X_k\} : X^* \in T, g_1(X^* \cup \{X_k\})\}$
         // constraint of (1)
   else
      6. $T \leftarrow \{X^* \cup \{X_k\} : X^* \in T\}$
   7. foreach $X^* \in T$ do
      8. if $X^*$ covers $G$ then // (2), (3), and (4)
         9. return $X^*$
   10. ($T$ is reduced as a heuristic)

---

3) Algorithms: We design an algorithm to find the optimal subset of pieces and bonds, $X^* \subseteq X$, based on Lemma 1. Algorithm 1 shows an exhaustive version of it (ignore line 10 for a while). The SP examines all possible subsets stored in $T$, while adding a piece or bond of maximum rank $k$ to each subset in $T$. The constraint of single piece per InP, (1), is checked at line 4. If a subset satisfies the constraints of (2), (3), and (4) at line 8, an optimal subset is successfully found. This algorithm obviously finds the optimal subset minimizing the maximum rank, since it is exhaustive. However, the number of subsets examined, $|T|$, grows exponentially with the number of pieces and bonds, $m$, and so becomes computationally intractable.

Our solution is a heuristic search algorithm of polynomial computational complexity that introduces a cost function based on Lemma 1. At the end of each iteration (line 10 of Algorithm 1), some “better” subsets are chosen and others are removed from $T$, which allows us to easily control the computation complexity. The subsets are prioritized based on the closing rank, which is the maximum rank of a given subset when the subset grows to cover the whole VN. It is formally defined as follows, for subset $X^* \in T$,

\[
    \min\{\max\{k : X_k \in X'\} : X' \supseteq X^*, X' \text{ covers } G\}.
\]

We choose subsets of smaller closing rank, since we are minimizing the maximum rank of the subset when it covers the VN. Here we employ the coupon collector’s problem [22] to estimate the closing rank. This is because this well-known problem gives us a tool to count coupons (subsets) with overlaps among them. The closing rank of subset $X^*$ is roughly estimated as follows, for iteration $i$,

\[
    k(X^*, i) = i\frac{H_{|V| + |E|}}{H_{|V| + |E|} - H_{\{|v \in V^e \cup e \in E^b : e \in V^e \cap e \in E^b \}|}},
\]

where $H_n$ is the $n$-th harmonic number.

Restricting the maximum size of $T$ up to $T$, i.e., $|T| \leq T$, the time complexity of heuristic search is $O(mT(n|V| + |E| + \log T))$, while the space complexity is $O(T)$.

In this algorithm, SPs can take account of InP reputation; e.g., the piece size of ill-reputed InPs can be limited, or some penalty is added to their closing rank.

---

B. Enumerating Virtual Network Pieces by Infrastructure Providers

InPs enumerate their pieces and bonds in the beginning of our method, as shown in Fig. 1(1). The number of bonds cannot be so large, since it is bounded by the number of inter-InP links. On the other hand, since any subgraph of VN can be a piece, the number of pieces increases exponentially with VN size. For SPs, relatively few pieces should be enumerated, since the time complexity of the optimization algorithm increases with the number of pieces. InPs, however, want to enumerate as many pieces as possible in order to enhance the chance of winning a bid. We propose an algorithm to select the minimal necessary pieces without losing any InP’s chance based on Pareto optimality.
Algorithm 2: Enumerate pieces and bonds by i-th InP

\[
\begin{align*}
\textbf{Input:} & \quad G, r \quad \text{ // requested VN} \\
\textbf{Output:} & \quad \mathcal{X}^i \quad \text{ // embeddable pieces and bonds} \\
& \quad \mathcal{P} \leftarrow \text{embeddable pieces of } i\text{-th InP, found with } G \text{ and } r \\
& \quad B \leftarrow \text{bonds of } i\text{-th InP} \\
& \quad \textbf{foreach} \quad P \in \mathcal{P} \textbf{ do} \\
& \quad \quad \textbf{foreach} \quad P^* \in \mathcal{P} \setminus \{P\} \textbf{ do} \\
& \quad \quad \quad \textbf{if} \quad P^* \preceq P \textbf{ then} \\
& \quad \quad \quad \quad \quad \mathcal{P} \leftarrow \mathcal{P} \setminus \{P\} \quad \textbf{if } P \text{ is not Pareto} \\
& \quad \quad \textbf{return} \quad \mathcal{P} \cup B \\
\end{align*}
\]

We first define the partial order over pieces. Piece $P^*$ precedes another piece $P$, if $P^*$ is a supergraph of $P$ and $P^*$’s extended price is smaller than $P$’s; i.e.,

\[ P^* \preceq P \iff P^* \trianglerighteq P \wedge \bar{f}(P^*) \leq \bar{f}(P). \]

A Pareto set of pieces is defined as follows. Given a set of pieces, the set is a Pareto set if no piece in the set precedes any other piece in the set; i.e., a set of pieces, $\mathcal{P}$, is a Pareto set, $\mathcal{P}^*$, if,

\[ P^* \trianglerighteq P \quad \forall P \in \mathcal{P}, P^* \in \mathcal{P} \setminus \{P\}. \]

In a Pareto set, no piece dominates others in terms of coverage and extended price.

Based on this Pareto optimality, we have the following theorem and lemma (proofs are omitted due to space limits).

**Theorem 2.** The assumption is that SPs follow Lemma 1 as their most rational strategy. Given a set of pieces and bonds, $\mathcal{X}$, which contains pieces of the non-Pareto set, these pieces are never included in the optimal subset, $\mathcal{X}^*$; i.e.,

\[ \mathcal{X}^* \cap \mathcal{P}^* = \emptyset, \]

where $\mathcal{P}^*$ is the complement set of Pareto set $\mathcal{P}^*$.

**Lemma 2.** InPs can reduce the number of pieces to enumerate without losing a chance to win a bid, if the Pareto set of pieces is selected.

We design an algorithm for InPs to enumerate a set of pieces and bonds following Lemma 2, as presented in Algorithm 2. Each InP first finds embeddable pieces and bonds, lines 1–2. After selecting a Pareto set of pieces at lines 3–6, the union of the selected pieces and bonds is obtained at line 7.

As mentioned at the beginning of this section, there can be a huge number of embeddable pieces, and so it is intractable to find all of them at line 1. InPs, however, do not want to miss a chance to cover a VN by overlooking some embeddable pieces. In addition, it seems difficult to find many embeddable pieces, since embedding just a single VN is $\mathcal{NP}$-hard as is well known.

We introduce here a greedy algorithm to find embeddable pieces such that all virtual nodes are covered by at least one piece (if possible). The algorithm starts with a virtual node, and tries to embed it in a physical node with maximum available capacity. The requested VN is expanded until no more virtual resource is embedded to the physical network. This expanding process is repeated with each virtual node $v \in V$. Although pieces are not necessarily optimally embedded in this greedy manner, InPs can re-embed pieces of the SP’s optimal selection in the final stage of our method.

The time complexity of Algorithm 2 with greedy embedding is $O(d|V|^2(|E|^P + |V^p| \log |V^p|) + |\mathcal{P}|^2)$, where $d$ is the maximum degree of the VN, and $V^p$ and $E^p$ are sets of physical nodes and links in the InP. The space complexity is $O(|\mathcal{X}^i|)$.

Each InP independently enumerates a set of pieces and bonds, $\mathcal{X}^i$, and then the union of these sets, $\mathcal{X}^*$, is sorted with MPC and passed to the SP. Although the union set is not necessarily a Pareto set, the SP must not make it Pareto removing some pieces from it. This is because at most one piece is allowed to select for each InP due to constraint (1), and non-Pareto pieces could be Pareto if some pieces of other InPs were excluded by the constraint.

In Algorithm 2, InPs can embed their pieces following their policies, with some modification of our algorithm. InPs are also allowed to eliminate pieces and bonds that do not follow the policies.

**IV. PROTOCOLS**

This section describes protocols between an SP and InPs; these protocols securely control the whole optimization process. Section IV-A describes a protocol to sort the VN pieces with MPC as shown in Fig. 1(2), and Section IV-B discusses another protocol to conclude the optimization process (Fig. 1(4)). We assume that the SP and InPs do not collude with each other, but each InP might cheat to win a bid illegally by itself if possible. In an MPC process, SPs and InPs follow the predefined protocols and do not try to corrupt the computation, since it has been theoretically proved that no participant change the results to what it wants [23].

**A. Sorting Protocol**

In this protocol, pieces and bonds enumerated by InPs are sorted with MPC by their extended prices, without revealing the price values. In MPC, secret value $x$ is distributed among MPC participants, each of which is allocated a share of $x$; i.e., $x$ is divided into $d$ shares which are combined together; individual shares of $x$ have no use on their own. A share of $x$ allocated to the $i$-th participant is denoted by $[[x]]_i$. In our method, MPC participants are chosen from InPs by the SP, while shares of secret values (extended prices) are provided by all InPs that join our method.

Figure 2 describes our sorting protocol. The $i$-th InP enumerates its pieces and bonds, $\mathcal{X}^i$, with Algorithm 2. These pieces and bonds are associated with IDs, $\{(id_x, X) : X \in \mathcal{X}^i\}$, where $id_x$ is the ID of piece or bond $X$, and are sent to the SP. Upon receiving these sets from all InPs, the SP randomly chooses MPC participants from
InPs. The \( j \)-th participant receives shares of extended prices, \( \{[[id_X]], [f(X)]_i, X \in \mathcal{X}^r \} \), from InPs. The participants use an MPC sort algorithm to collaboratively sort them by extended price. Since the sorted list is distributed as secret shares among participants, e.g., \( \{[id_{X_1}, \cdots, id_{X_n}]_i \} \) for \( j \)-th participant, the SP gathers these shares and reconstructs the sorted list. Finally, pieces and bonds are associated with their order, which becomes the input of Algorithm 1.

Since extended prices are always exchanged as secret shares, as shown in Fig. 2, no one can reconstruct them. In this way, InPs’ private information is protected in this protocol.

### B. Closing Protocol

This protocol embeds inter-InP virtual links to physical paths, and unveils the price of the optimal VN to the SP. We begin with inter-InP virtual links. Since optimal pieces has been selected by Algorithm 1, we are now allowed to embed inter-InP virtual links to physical paths. In order to embed them, we utilize PolyViNE [10], which is an existing protocol to embed a VN over multiple InPs. In the original PolyViNE, InPs jointly partition the requested VN while sharing their private information, but since the VN has been partitioned by the SP in our method, InPs can embed the VN without raising privacy issues.

At the end, the price of the optimal VN is unveiled to the SP. The SP gathers secret shares of optimal pieces and bonds from the MPC participants, and reconstructs their extended prices. These prices cannot be necessarily identical with those that will be charged by InPs, since they have been determined without embedding inter-InP virtual links. However, this price check is indispensable, because without it InPs could give unfairly low prices for sorting and charge much higher prices for payment.

Figure 3 shows the closing protocol. After finding the optimal VN, the SP asks InPs whether inter-InP virtual links can be embedded; the SP tells InPs only their assignment without the entire VN mapping, so as not to reveal selected pieces of other InPs. InPs then try to embed them with PolyViNE modified so as not to rely on private information. If embedding fails, the SP resumes Algorithm 1 and finds the next optimal VN. If successful, the SP collects the secret shares and reconstructs their extended prices, \( f(\mathcal{X}) \)’s.

The price of the optimal VN is unveiled to just the SP, not InPs. Prices of other pieces and bonds cannot be unveiled to anyone, since collecting the shares is not permitted.

### V. Experiments

We first evaluate our optimization algorithm described in Section III-A, and then examine the output of our enumeration algorithm in Section III-B. Finally, the computation time of our whole method is measured. We did not compare our method with any existing work, since, to the best of our knowledge, no practical method exists that protects private information.

Parameters used in the experiments are chosen following [5], [10] (Table I). Physical and virtual networks are generated by GT-ITM [24]. The unit price of physical resources are randomly determined based on mean and variance, which are defined for each InP. The piece price is the sum of embedded physical resources.

The experiments were conducted on a machine with Xeon E3 3.50GHz \( \times 4 \), emulating long round-trip time of 300 msec (assuming SPs and InPs are distributed on different continents). Our method is implemented in C++ and Haskell with a fast MPC sort algorithm [25].
We evaluate two variants of our optimization algorithm of Section III-A: “price search” and “coverage search”. The price search uses price values to exhaustively search for the optimal subset; this gives a lower bound of VN price, though it takes quite a long time. The coverage search relies on just piece coverage, i.e., no price-related information; this is similar to Algorithm 1, but it sorts pieces and bonds by the coverage (assuming bonds precede pieces) and reduces \( T \) by the coverage of \( \mathcal{X} \). Figure 4 shows optimal VN prices normalized by those of price search. Our algorithm is, surprisingly, only 2–19 % worse than the price search on average. It outperforms the coverage search by 35–173 %. As shown in Fig. 5, our algorithm finds the same VNs as the price search once in five times, and it never outputs VNs that are more than twice as expensive as the price search results, unlike coverage search. We believe that these promising results, which are due to our search strategy based on Lemma 1 and the heuristic, are good enough for SPs to choose our method.

Figure 6 shows the number of pieces and bonds enumerated by our method; reduced by about 2/3. We solved 30 instances for each bar. The means are represented by the marks, and standard deviations are indicated by error bars.

Figure 7 shows the computation time of our method for the four stages of Fig. 1. Our method finishes in a couple of minutes for large VNs.

### VI. RELATED WORK

Several papers have discussed the inter-InP VN optimization problem. Most of them [5], [11]–[14] assume that InPs would share their private information with the SP, which partitions the requesting VNs based on the private information including resource costs. These methods are unacceptable for InPs, which have traditionally been reluctant to disclose the private information. In addition, VNs are partitioned by the SP without consideration of the InPs’ policies; this is why we allow the
InPs to enumerate embeddable pieces at first. PolyViNE [10] forces InPs to share the private information with each other. InPs use shared information including resource costs to jointly build the requested VN. Although PolyViNE is well designed to find feasible embedding between InPs, this method ignores the privacy issue. SPs’ policies are ignored since they are not involved in the embedding. To the best of our knowledge, our past work [16] is the only study that does not reveal InPs’ private information, but its computation time can be excessive, sometimes several months, due to the heavy use of MPC.

Many algorithms have been proposed for the VN optimization problem with a single InP [6]–[9]. They complement our work by finding better embedding in the closing protocol.

We are unaware of earlier techniques that apply MPC to complex optimization problems. The closest related works we know of are linear programming [26] and shortest path search between two parties [17], [18]. They are simple polynomial optimization problems, and so we had to develop a new approach for VN optimization which is NP-hard. Other MPC applications include anomaly detection [19], [20] and policy checking [21], which are also much simpler than our problem since they do check some invariants with MPC. Reference [27] applied MPC to sugar beet auction, which is a pricing mechanism with a single commodity; our problem is much harder, since it involves multiple commodities with complex constraints located in multiple providers.

VII. CONCLUSIONS

In this paper, we proposed an optimization method to build a VN across multiple InPs without revealing their private information. Although much effort has been devoted to this problem, existing efforts require the private information to be shared in the optimization, which makes them impractical to implement. We introduced a theory that can find the optimal VN given just the price order computed with MPC, and developed an efficient search algorithm based on the theory. We also discussed the Pareto optimality of VN pieces enumerated by InPs, in order to reduce the search space. Numerical experiments showed that our method is fast and optimal in practice. We have shown how to solve this quite hard optimization problem under limited information for the first time ever.

Future work includes development of faster search algorithms, game theoretic analysis of the relationship between SP and InPs, and large-scale experiments with actual providers.

REFERENCES


