Beyond the Stars: Revisiting Virtual Cluster Embeddings

Matthias Rost
Technische Universität Berlin

August 18, 2017, Aalborg University

Joint work with Carlo Fuerst, Stefan Schmid
Published in ACM SIGCOMM CCR, July 2015
Providing Services inside Data Centers

- Example fat tree data center topology [1]
- 2.5k switches and 27k hosts for a medium sized data center
Providing Services inside Data Centers

- Virtualization of servers allows to quickly spawn Virtual Machines (VMs) for tenants inside the data center
Providing Services inside Data Centers

- Virtualization of servers allows to quickly spawn Virtual Machines (VMs) for tenants inside the data center
- Hundreds or thousands of Virtual Machines may be requested
Providing Services inside Data Centers

- Virtualization of servers allows to quickly spawn Virtual Machines (VMs) for tenants inside the data center
- Hundreds or thousands of Virtual Machines may be requested
- Working together, communication between VMs is of paramount importance
Providing Services inside Data Centers

Problem: Performance crucially depends on bandwidth

- Network transfers: 33% of the execution time (Facebook [4])
- Data centers exhibit oversubscription factors of up to 1:240 [6]
- Customer’s performance varies dramatically depending on network load
Providing Services inside Data Centers

Problem: Performance crucially depends on bandwidth

Solution: resource isolation / Quality-of-Service
Providing Services inside Data Centers

Algorithmic Task: Graph Embedding

- find embedding, i.e. a joint mapping of VMs to servers and VM interconnections to paths
- not exceeding the data center’s resource capacities and of minimal cost
Service Abstractions: The VC Abstraction
Early 2000s: Virtual Network Embedding Problem

- Requests are specified as graphs
  - Nodes represent VMs
  - Edges represent inter-VM links

- Pro: Concise specification
- Contra: Do customers know their requirements? Generally: Challenging NP-hard problem!
Service Abstractions

Early 2000s: Virtual Network Embedding Problem

- Requests are specified as graphs
  - Nodes represent VMs
  - Edges represent inter-VM links

Pro
- *Concise specification*
Service Abstractions

Early 2000s: Virtual Network Embedding Problem

- Requests are specified as graphs
  - Nodes represent VMs
  - Edges represent inter-VM links

Pro

- *Concise specification*

Contra

- Do customers know their requirements?
- Generally: Challenging NP-hard problem!
The Right Level of Abstraction

2011: Virtual Cluster (VC) Abstraction [3]

- Allows only for ‘star’-shaped graphs
- VMs are connected to logical switch
The Right Level of Abstraction

2011: Virtual Cluster (VC) Abstraction [3]

- Allows only for ‘star’-shaped graphs
- VMs are connected to logical switch
The Right Level of Abstraction

2011: Virtual Cluster (VC) Abstraction [3]

- Allows only for ‘star’-shaped graphs
- VMs are connected to logical switch
- Requests are specified by three parameters:
  - \( N \in \mathbb{N} \) number of virtual machines
  - \( C \in \mathbb{N} \) size of virtual machines
  - \( B \in \mathbb{N} \) bandwidth to logical switch
The Right Level of Abstraction

2011: Virtual Cluster (VC) Abstraction [3]

- Allows only for ‘star’-shaped graphs
- VMs are connected to logical switch
- Requests are specified by three parameters:
  \( N \in \mathbb{N} \) number of virtual machines
  \( C \in \mathbb{N} \) size of virtual machines
  \( B \in \mathbb{N} \) bandwidth to logical switch

Pro

- Simple specification!
- Well-performing heuristics for data-center topologies [3, 8]

Contra

- The VM size and the amount of bandwidth are dictated by the maximum \( \rightarrow \) wasteful

Matthias Rost  (TU Berlin)  Revisiting Virtual Cluster Embeddings  Aalborg, August 2017 5
On Traffic Matrices

Graph Abstraction

- Allows for any traffic matrix $M$, where the bandwidth for edge $\{i, j\}$ is less than $B_{\{i,j\}}$.

VC Abstraction

- Allows for any traffic matrix $M$, where for any VM the sum of outgoing and incoming traffic is less than $B$. 

Outlook

Previous works . . .
- only considered (fat) trees
- only considered heuristics

Ballani et al.: ‘Oktopus’ [3]
“allocating virtual cluster requests on graphs with bandwidth-constrained edges is NP-hard”

Xie et al.: ‘Proteus’ [8]
“[Our algorithm] picks the first fitting lowest-level subtree out of all such lowest-level subtrees.”
Outlook

Previous works …
- only considered (fat) trees
- only considered heuristics

Ballani et al.: ‘Oktopus’ [3]
“allocating virtual cluster requests on graphs with bandwidth-constrained edges is NP-hard”

Xie et al.: ‘Proteus’ [8]
“[Our algorithm] picks the first fitting lowest-level subtree out of all such lowest-level subtrees.”

Main Questions
Is the VC embedding problem really NP-hard to solve?
Formal Definition of the VC Embedding Problem
### VC Embedding Problem Definition

**VC request:** $\mathcal{N}, \mathcal{B}, \mathcal{C}$

- $\text{VC} = (V_{\text{VC}}, E_{\text{VC}})$,
- $V_{\text{VC}} = \{1, 2, \ldots, \mathcal{N}, \text{center}\}$
- $E_{\text{VC}} = \{\{i, \text{center}\} | 1 \leq i \leq \mathcal{N}\}$

**Physical Network (Substrate)**

- $S = (V_S, E_S, \text{cap}, \text{cost})$,
- $\text{cap} : V_S \cup E_S \to \mathbb{N}$
- $\text{cost} : V_S \cup E_S \to \mathbb{R}_{\geq 0}$

**Task:** Find a mapping of...

- VMs onto substrate nodes $\text{map}_V : V_{\text{VC}} \rightarrow V_S$, and
- VC edges onto paths in the substrate $\text{map}_E : E_{\text{VC}} \rightarrow \mathcal{P}(E_S)$
The classical VC Embedding Problem

Definition: VC Embedding Problem

VC Embedding Problem Definition

VC request: \( N, B, C \)
- \( VC = (V_{VC}, E_{VC}) \),
- \( V_{VC} = \{1, 2, \ldots, N, \text{center}\} \)
- \( E_{VC} = \{\{i, \text{center}\} | 1 \leq i \leq N\} \)

Physical Network (Substrate)
- \( S = (V_S, E_S, \text{cap}, \text{cost}) \),
- \( \text{cap} : V_S \cup E_S \rightarrow \mathbb{N} \)
- \( \text{cost} : V_S \cup E_S \rightarrow \mathbb{R}_{\geq 0} \)

Task: Find a mapping of ... 
- VMs onto substrate nodes \( \text{map}_V : V_{VC} \rightarrow V_S \), and
- VC edges onto paths in the substrate \( \text{map}_E : E_{VC} \rightarrow \mathcal{P}(E_S) \), such that

1. \( \text{map}_E(\{u, v\}) \) connects \( \text{map}_V(u) \) and \( \text{map}_V(v) \) for \( \{u, v\} \in E_{VC} \)
**VC Embedding Problem Definition**

<table>
<thead>
<tr>
<th>VC request: $\mathcal{N}, \mathcal{B}, \mathcal{C}$</th>
<th>Physical Network (Substrate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• $VC = (V_{VC}, E_{VC})$,</td>
<td>• $S = (V_S, E_S, \text{cap}, \text{cost})$,</td>
</tr>
<tr>
<td>• $V_{VC} = {1, 2, \ldots, \mathcal{N}, \text{center}}$</td>
<td>• $\text{cap} : V_S \cup E_S \to \mathbb{N}$</td>
</tr>
<tr>
<td>• $E_{VC} = {{i, \text{center}}</td>
<td>1 \leq i \leq \mathcal{N}}$</td>
</tr>
</tbody>
</table>

**Task: Find a mapping of ...**

- VMs onto substrate nodes $\text{map}_V : V_{VC} \to V_S$, and
- VC edges onto paths in the substrate $\text{map}_E : E_{VC} \to \mathcal{P}(E_S)$, such that

1. $\text{map}_E(\{i, j\})$ connects $\text{map}_V(i)$ and $\text{map}_V(j)$ for $\{i, j\} \in E_{VC}$

2. $\sum_{v' \in V_{VC} \setminus \{\text{center}\}} \mathcal{C} \leq \text{cap}(v)$ and $\sum_{e' \in E_{VC}} \mathcal{B} \leq \text{cap}(e)$ for $v \in V_S$, $e \in E_S$
VC Embedding Problem Definition

Task: Find a mapping of...

- VMs onto substrate nodes \( \text{map}_V : V_{VC} \rightarrow V_S \), and
- VC edges onto paths in the substrate \( \text{map}_E : E_{VC} \rightarrow \mathcal{P}(E_S) \), such that

1. \( \text{map}_E(\{u, v\}) \) connects \( \text{map}_V(u) \) and \( \text{map}_V(v) \) for \( \{u, v\} \in E_{VC} \)

2. \( \sum_{v' \in V_{VC} \setminus \{\text{center}\}} C \leq \text{cap}(v) \) and \( \sum_{e' \in E_{VC}} B \leq \text{cap}(e) \) for \( v \in V_S, e \in E_S \)

3. minimizing the cost \( C \cdot \sum_{v \in V_{VC} \setminus \{\text{center}\}} \text{cost}(\text{map}_V(v)) + B \cdot \sum_{e' \in E_{VC}} \text{cost}(e) \).
VC-ACE Algorithm
Key Insights

Lemma

We can assume $B = C = 1$.

Proof idea.

If $B \neq 1$, $C \neq 1$, we transform the substrate by scaling capacities and costs:

- $\text{cap}_{S'}(u) = \lfloor \text{cap}(u)/C \rfloor$ for $u \in V_S$
- $\text{cap}_{S'}(e) = \lfloor \text{cap}(e)/B \rfloor$ for $e \in E_S$
- $\text{cost}_{S'}(u) = \text{cost}(u) \cdot C$ for $u \in V_S$
- $\text{cost}_{S'}(e) = \text{cost}(e) \cdot B$ for $e \in E_S$
Key Insights

Lemma

We can solve the edge embedding problem if all nodes are placed.

Proof.

1. Construct extended graph with additional node $s^+$ and (parallel) edges: $\{(s^+, \text{map}_V(i))| i \in \{1, \ldots, N\}\}$ of capacity 1 and cost 0
2. Compute a minimum cost flow of value $N$ from $s^+$ to $\text{map}_V(\text{center})$.
3. Perform a path-decomposition to obtain mapping for edges.
Key Insights

Lemma

We can solve the embedding problem if the logical switch is placed.

Proof.

1. Construct extended graph with additional edges \( \{(s^+, u) | u \in V_S\} \), \( \text{cap}(s^+, u) = \text{cap}(u) \) and \( \text{cost}(s^+, u) = \text{cost}(u) \) for \( u \in V_S \).
2. Compute a minimum cost flow of value \( \mathcal{N} \) from \( s^+ \) to map_\(V\)(center).
3. Perform path-decomposition to obtain mapping for nodes and edges.
Key Insights

**Lemma**

*We can solve the embedding problem if the logical switch is placed.*

**Proof.**

1. Construct extended graph with additional edges \( \{(s^+, u) | u \in V_S\} \), cap\((s^+, u) = \text{cap}(u)\) and cost\((s^+, u) = \text{cost}(u)\) for \( u \in V_S \).
2. Compute a minimum cost flow of value \( \mathcal{N} \) from \( s^+ \) to map\(_V\)(center).
3. Perform path-decomposition to obtain mapping for *nodes and edges*.

![Diagram](image.png)
The ‘classical’ VC Embedding Problem

VC-ACE Algorithm

Algorithm 1: The VC-ACE Algorithm

Input: Substrate $S = (V_S, E_S)$, request $(N, B, C)$
Output: Optimal VC mapping $\text{map}_V, \text{map}_E$ if feasible

$(\hat{f}, \hat{v}) \leftarrow (\text{null}, \text{null})$

for $v \in V_S$ do
    $V_S' = V_S \cup \{s^+\}$ and $E_S' = E_S \cup \{(s^+, u) | u \in V_S\}$
    $\text{cap}_{S'}(e) = \begin{cases} 
    \lfloor \text{cap}(e)/B \rfloor, & \text{if } e \in E_S \\
    \lfloor \text{cap}(u)/C \rfloor, & \text{if } e = (s^+, u) \in E_S 
    \end{cases}$
    $\text{cost}_{S'}(e) = \begin{cases} 
    \text{cost}(e) \cdot B, & \text{if } e \in E_S \\
    \text{cost}(u) \cdot C, & \text{if } e = (s^+, u) \in E_S 
    \end{cases}$
    $f \leftarrow \text{MinCostFlow}(s^+, v, N, V_S', E_S', \text{cap}_{S'}, \text{cost}_{S'})$
    if $f$ is feasible and $\text{cost}(f) < \text{cost}(\hat{f})$ then
        $(\hat{f}, \hat{v}) \leftarrow (f, v)$
    if $\hat{f} = \text{null}$ then
        return null
return DecomposeFlowIntoMapping($\hat{f}, \hat{v}$)

Idea

Simply iterate over possible locations for the center.
VC-ACE Algorithm

**Algorithm 2: The VC-ACE Algorithm**

**Input:** Substrate $S = (V_S, E_S)$, request $(\mathcal{N}, B, C)$

**Output:** Optimal VC mapping $map_V, map_E$ if feasible

$(\hat{f}, \hat{v}) \leftarrow (null, null)$

for $v \in V_S$ do

$V_{S'} = V_S \cup \{s^+\}$ and $E_{S'} = E_S \cup \{(s^+, u) | u \in V_S\}$

$\text{cap}_{S'}(e) = \begin{cases} \lfloor \text{cap}(e) / B \rfloor, & \text{if } e \in E_S \\ \lfloor \text{cap}(u) / C \rfloor, & \text{if } e = (s^+, u) \in E_S \end{cases}$

$\text{cost}_{S'}(e) = \begin{cases} \text{cost}(e) \cdot B, & \text{if } e \in E_S \\ \text{cost}(u) \cdot C, & \text{if } e = (s^+, u) \in E_S \end{cases}$

$f \leftarrow \text{MinCostFlow}(s^+, v, \mathcal{N}, V_{S'}, E_{S'}, \text{cap}_{S'}, \text{cost}_{S'})$

if $f$ is feasible and $\text{cost}(f) < \text{cost}(\hat{f})$ then

$(\hat{f}, \hat{v}) \leftarrow (f, v)$

if $\hat{f} = \text{null}$ then

return null

return DecomposeFlowIntoMapping($\hat{f}, \hat{v}$)

---

**Idea**

Simply iterate over possible locations for the center.

**Theorem**

*Correctness follows from the lemma on the previous slide.*
The 'classical' VC Embedding Problem

VC-ACE Algorithm

Theorem

The runtime is $O\left(N(n^2 \log n + n \cdot m)\right)$ with $n = |V_S|$ and $m = |E_S|$, when using the successive-shortest path for the flow computation.

Corollary.

The VC Embedding Problem can be solved optimally in polynomial time.

Algorithm 3: The VC-ACE Algorithm

Input: Substrate $S = (V_S, E_S)$, request $(N, B, C)$
Output: Optimal VC mapping $map_V, map_E$ if feasible

$(\hat{f}, \hat{v}) \leftarrow (null, null)$

for $v \in V_S$ do

- $V'_S = V_S \cup \{s^{+}\}$ and $E'_S = E_S \cup \{(s^{+}, u) | u \in V_S\}$
- $\text{cap}_{S'}(e) = \begin{cases} \lfloor \text{cap}(e) / B \rfloor, & \text{if } e \in E_S \\ \lfloor \text{cap}(u) / C \rfloor, & \text{if } e = (s^{+}, u) \in E_S \end{cases}$
- $\text{cost}_{S'}(e) = \begin{cases} \text{cost}(e) \cdot B, & \text{if } e \in E_S \\ \text{cost}(u) \cdot C, & \text{if } e = (s^{+}, u) \in E_S \end{cases}$

$f \leftarrow \text{MinCostFlow}(s^{+}, v, N, V'_S, E'_S, \text{cap}_{S'}, \text{cost}_{S'})$

if $f$ is feasible and $\text{cost}(f) < \text{cost}(\hat{f})$ then

$(\hat{f}, \hat{v}) \leftarrow (f, v)$

if $\hat{f} = \text{null}$ then

return null

return $\text{DecomposeFlowIntoMapping}(\hat{f}, \hat{v})$
We can compute optimal solutions in polynomial-time.

Can we do even better?
We can compute optimal solutions in polynomial-time.

Can we do even better?

Introducing the Hose-Based Virtual Cluster
VC Abstraction

- Allows for any traffic matrix $M$, where for any VM the sum of outgoing and incoming traffic is less than $B$
motivation

starting from scratch

vc abstraction

allows for any traffic matrix \( M \), where for any VM the sum of outgoing and incoming traffic is less than \( B \)

Question:
What is the purpose of the switch?
VC Abstraction

Question:
What is the purpose of the switch?

Ballani et al. ‘Oktopus’ [3]
“Oktopus’ allocation algorithms assume that the traffic between a tenant’s VMs is routed along a tree.”

Answer:
To route the traffic along a tree.

- Allows for any traffic matrix $M$, where for any VM the sum of outgoing and incoming traffic is less than $B$
Starting from Scratch

**VC Abstraction**

1. Allows for any traffic matrix $M$, where for any VM the sum of outgoing and incoming traffic is less than $B$

**Question:** Can we do without the switch?
Hose-Based Virtual Cluster Embeddings

Starting from Scratch

Motivation

VC Abstraction

Question:
Can we do without the switch?

Ballani et al. ‘Oktopus’ [3]

“Alternatively, the NM [Network Manager] can control datacenter routing to actively build routes between tenant VMs [..]”

“We defer a detailed study of the relative merits of these approaches to future work.”

Answer:
Yes!

Allows for any traffic matrix $M$, where for any VM the sum of outgoing and incoming traffic is less than $B$. 
Starting from Scratch

VC Abstraction

- Allows for any traffic matrix $M$, where for any VM the sum of outgoing and incoming traffic is less than $B$

Hose-Based VC Abstraction

- Allows for any traffic matrix $M$, where for any VM the sum of outgoing and incoming traffic is less than $B$
Motivating Example

There exists no solution in the classic VC embedding model.

\( N = 6, \ B = C = 1 \)

\( \text{cap} = 1 \)
\( \text{cap} = 2 \)

ring substrate
There exists no solution \ldots
\ldots\text{in the classic VC embedding model.}
Motivating Example

There exists a solution in the hose-based VC model!

Embedding on the same substrate
Motivating Example

There exists a solution ...  
... in the hose-based VC model!
Motivating Example

Why allocations of 2 are sufficient:

- Consider edge $e$ between VMs 6 and 5.
- The edge is used by routes $R(e) = \{(1, 5), (2, 5), (3, 6), (4, 6), (5, 6)\}$.
- Any valid traffic matrix $M$ will respect:
  - $M_{1,5} + M_{2,5} \leq 1$
  - $M_{3,6} + M_{4,6} + M_{5,6} \leq 1$
- Hence $\sum_{(i,j) \in R(e)} M_{i,j} \leq 2$ holds.
Motivating Example II

\[ N = 6, \ B = C = 1 \]

\[ \text{Solution costs...} \]
\[ \text{...can be arbitrarily higher under the classic star-interpretation!} \]
Hose-Based Virtual Cluster Embedding Problem
Hose-Based VC Embedding Problem (HVCEP)

**Definition (Clique Graph)**

- $V_C = \{1, \ldots, N\}$, $E_C = \{(i, j) | i, j \in V_C, i < j\}$

**Task: Find a mapping of ...**

- VC nodes onto substrate nodes $\text{map}_V : V_C \rightarrow V_S$, and
- VC routes onto paths in the substrate $\text{map}_E : E_C \rightarrow \mathcal{P}(E_S)$, such that

1. \textit{route} $(i, j) \in E_C$ connects $\text{map}_V(i)$ and $\text{map}_V(j)$,
2. the mapping of VMs must not violate node capacities (cf. slide 12),
Hose-Based VC Embedding Problem (HVCEP)

Definition (Clique Graph)

- \( V_C = \{1, \ldots, N\} \), \( E_C = \{(i, j)|i, j \in V_C, i < j\} \)

Task: Find a mapping of...

- VC nodes onto substrate nodes map_V : \( V_C \rightarrow V_S \), and
- VC routes onto paths in the substrate map_E : \( E_C \rightarrow \mathcal{P}(E_S) \), and
- integral bandwidth reservations \( l_{u,v} \leq \text{cap}(u, v) \) for \( \{u, v\} \in E_S \), s.t.
  1. route \((i, j) \in E_C\) connects map_V(i) and map_V(j),
  2. the mapping of VMs must not violate node capacities (cf. slide 12),
  3. for all valid traffic matrices \( M \) – i.e. \( \sum_{(i,j) \in E_C} M_{i,j} + M_{j,i} \leq B \) holds – the bandwidth reservation is not exceeded on any edge \( \{u, v\} \in E_S: \sum_{\{i,j\} \in E_C: \{u,v\} \in \text{map}_E(\{i,j\})} M_{ij} \leq l_{u,v} \).
Hose-Based VC Embedding Problem (HVCEP)

**Definition (Clique Graph)**

- \( V_C = \{1, \ldots, N\} \), \( E_C = \{(i, j) | i, j \in V_C, i < j\} \)

**Task: Find a mapping of ...**

- VC nodes onto substrate nodes \( \text{map}_V : V_C \rightarrow V_S \), and
- VC routes onto paths in the substrate \( \text{map}_E : E_C \rightarrow \mathcal{P}(E_S) \), and
- integral bandwidth reservations \( l_{u,v} \leq \text{cap}(u,v) \) for \( \{u,v\} \in E_S \), such that
  - route \( (i,j) \in E_C \) connects \( \text{map}_V(i) \) and \( \text{map}_V(j) \),
  - the mapping of VMs must not violate node capacities (cf. slide 12),
  - for all valid traffic matrices \( M \) – i.e. \( \sum_{(j,i) \in E_C} M_{ji} + M_{ij} \leq B \) holds – the bandwidth reservation is not exceeded on any edge \( \{u,v\} \in E_S \):
    \[
    \sum_{\{i,j\} \in E_C: \{u,v\} \in \text{map}_E(\{i,j\})} M_{ij} \leq l_{u,v},
    \]
  - minimizing \( C \cdot \sum_{i \in V_C} \text{cost}(\text{map}_V(i)) + B \cdot \sum_{e \in E_S} l_{u,v} \cdot \text{cost}(e) \).
Computational Complexity of HVC Embeddings
Computational Complexity of Finding HVC Embeddings

Theorem (via the Virtual Private Network Problem [7])
Finding a feasible solution for the HVCEP is NP-hard. This still holds if the VMs are already mapped.

Theorem (via the Virtual Private Network Conjecture [5])
Algorithm VC-ACE solves the HVCEP when capacities are sufficiently large.
Computing (Fractional) HVC Embeddings
A Mixed-Integer Programming Formulation for the HVCEP

Mixed-Integer Program 1: HVC-OSPE

\[
\begin{align*}
\text{min} & \quad \sum_{i \in V_C, u \in V_S} \text{cost}_u \cdot x^i_u + \sum_{\{u,v\} \in E_S} \text{cost}_{u,v} \cdot l_{uv} \\
& \quad \sum_{u \in V_S} x^i_u = 1 \quad \forall i \in V_C. \\
& \quad \sum_{u \in V_S} \sigma_u \cdot (x^i_u - x^{i+1}_u) \leq 0 \quad \forall i \in V_C \setminus \{N\}. \\
& \quad \sum_{i \in V_C} C \cdot x^i_u \leq \text{cap}_u \quad \forall u \in V_S. \\
& \quad l_{uv} \leq \text{cap}_{uv} \quad \forall \{u, v\} \in E_S. \\
& \quad \sum_{(u, v) \in \delta_+^u} y^i_{uv} - \sum_{(v, u) \in \delta^-_u} y^i_{vu} = x^i_u - x^i_j \quad \forall (i,j) \in E_C, \forall u \in V_S. \\
& \quad \sum_{i \in V_C} B \cdot \omega^i_{uv} \leq l_{uv} \quad \forall \{u, v\} \in E_S. \\
& \quad y^i_{uv} + y^j_{vu} \leq \omega^i_{uv} + \omega^j_{uv} \quad \forall (i,j) \in E_C, \forall \{u, v\} \in E_S.
\end{align*}
\]

Variables

- \(x^i_u\): mapping of VM \(i\) onto node \(u\)
- \(y^i_{uv}\): mapping of link \((i,j)\) onto (directed) substrate edge \((u,v)\)
- \(l_{uv}\): load on substrate edge \(\{u,v\}\)
- \(\omega^i_{uv}\): ‘dual variable’ for allocation of communications of VM \(i\) on edge \(\{u,v\}\)

Matthias Rost (TU Berlin)
A Mixed-Integer Programming Formulation for the HVCEP

Mixed-Integer Program 2: HVC-OSPE

\[
\begin{align*}
\text{min} & \sum_{i \in V_C, u \in V_S} \text{cost}_u \cdot x_u^i + \sum_{\{u, v\} \in E_S} \text{cost}_{u,v} \cdot l_{uv} \quad (1) \\
& \sum_{u \in V_S} x_u^i = 1 \quad \forall i \in V_C. \quad (2) \\
& \sum_{u \in V_S} \sigma_u \cdot (x_u^i - x_u^{i+1}) \leq 0 \quad \forall i \in V_C \setminus \{N\}. \quad (3) \\
& \sum_{i \in V_C} C \cdot x_u^i \leq \text{cap}_u \quad \forall u \in V_S. \quad (4) \\
& l_{uv} \leq \text{cap}_{uv} \quad \forall \{u, v\} \in E_S. \quad (5) \\
& \sum_{(u, v) \in \delta_u^+} y_{uv}^{ij} - \sum_{(v, u) \in \delta_u^-} y_{vu}^{ij} = x_u^i - x_u^j \quad \forall (i, j) \in E_C, \forall u \in V_S. \quad (6) \\
& \sum_{i \in V_C} B \cdot \omega_{uv}^i \leq l_{uv} \quad \forall \{u, v\} \in E_S. \quad (7) \\
& y_{uv}^{ij} + y_{vu}^{ij} \leq \omega_{uv}^i + \omega_{uv}^j \quad \forall (i, j) \in E_C, \forall \{u, v\} \in E_S. \quad (8)
\end{align*}
\]

Explanation

- (2) - (4) control the VM embedding
- (5) - (8) is adapted from Altin et al. [2] for computing the optimal hose allocations on edges
# A Mixed-Integer Programming Formulation for the HVCEP

## Explanation

1. (2) - (4) control the VM embedding
2. (5) - (8) is adapted from Altin et al. [2] for computing the optimal hose allocations on edges

## Observation

There are $O(N^2 \cdot |E_S|)$ binary variables for computing paths.

## Mixed-Integer Program 3: HVC-OSPE

\[
\begin{align*}
\text{min} & \quad \sum_{i \in V_C, u \in V_S} \text{cost}_u \cdot x_u^i + \sum_{\{u, v\} \in E_S} \text{cost}_{u, v} \cdot l_{uv} \\
& \quad \sum_{u \in V_S} x_u^i = 1 \quad \forall i \in V_C. \\
& \quad \sum_{u \in V_S} \sigma_u \cdot (x_u^i - x_u^{i+1}) \leq 0 \quad \forall i \in V_C \setminus \{N\}. \\
& \quad \sum_{i \in V_C} C \cdot x_u^i \leq \text{cap}_u \quad \forall u \in V_S. \\
& \quad l_{uv} \leq \text{cap}_{uv} \quad \forall \{u, v\} \in E_S. \\
& \quad \sum_{(u, v) \in \delta_u^+} y_{uv}^i - \sum_{(v, u) \in \delta_u^-} y_{vu}^i = x_u^i - x_u^j \quad \forall (i, j) \in E_C, \forall u \in V_S. \\
& \quad \sum_{i \in V_C} B \cdot \omega_{uv}^i \leq l_{uv} \quad \forall \{u, v\} \in E_S. \\
& \quad y_{uv}^i + y_{vu}^i \leq \omega_{uv}^i + \omega_{uv}^j \quad \forall (i, j) \in E_C, \forall \{u, v\} \in E_S.
\end{align*}
\]
A Mixed-Integer Programming Formulation for the HVCEP

**Mixed-Integer Program 4: HVC-OSPE**

\[
\begin{align*}
\text{min} & \quad \sum_{i \in V_C, u \in V_S} \text{cost}_u \cdot x^i_u + \sum_{\{u, v\} \in E_S} \text{cost}_{u,v} \cdot l_{uv} \\
\sum_{u \in V_S} x^i_u &= 1 \quad \forall i \in V_C. \\
\sum_{u \in V_S} \sigma_u \cdot (x^i_u - x^{i+1}_u) &\leq 0 \quad \forall i \in V_C \setminus \{N\}. \\
\sum_{i \in V_C} C \cdot x^i_u &\leq \text{cap}_u \quad \forall u \in V_S. \\
l_{uv} &\leq \text{cap}_{uv} \quad \forall \{u, v\} \in E_S. \\
\sum_{(u,v) \in \delta^+_u} y^i_{uv} - \sum_{(v,u) \in \delta^-_u} y^j_{vu} &= x^i_u - x^j_u \quad \forall (i,j) \in E_C, \forall u \in V_S. \\
\sum_{i \in V_C} B \cdot \omega^i_{uv} &\leq l_{uv} \quad \forall \{u, v\} \in E_S. \\
y^i_{uv} + y^j_{vu} &\leq \omega^i_{uv} + \omega^j_{uv} \quad \forall (i,j) \in E_C, \forall \{u, v\} \in E_S.
\end{align*}
\]

**Observation**

There are \(O(N^2 \cdot |E_S|)\) binary variables for computing paths.

**Initial Computational Results**

Solving this formulation may take up to 1800 seconds for embedding a 10-VM VC onto a 20 node substrate.
Further Observations

- The hardness result has shown that the problem is hard even if the VMs are fixed.
- The large number of variables necessary for computing each end-to-end path between VMs renders solving even the linear relaxation – i.e. dropping integrality constraints – computationally hard.

Assumptions for obtaining a ‘solvable’ formulation

- Assume that the VMs are already mapped.
- Assume that the hose-paths are *splittable*, i.e. each VMs are connected by a set of (weighted) paths.
Assumptions

- Assume that the VMs are already mapped.
- Assume that the hose-paths are *splittable*. Arbitrarily many paths.
Computing Splittable HVC Embeddings

\[ \sum_{u \neq v} l_{uv} \leq \text{cap}_{uv} \quad \forall \{u, v\} \in E_S. \tag{5} \]

\[ \sum_{(u, v) \in \delta^+_u} y^{ij}_{uv} - \sum_{(v, u) \in \delta^-_u} y^{ji}_{vu} = x^i_u - x^j_u \quad \forall (i, j) \in E_C, \forall u \in V_S. \tag{6} \]

\[ \sum_{i \in V_C} B \cdot \omega^i_{uv} \leq l_{uv} \quad \forall \{u, v\} \in E_S. \tag{7} \]

\[ y^{ij}_{uv} + y^{ji}_{vu} \leq \omega^i_{uv} + \omega^j_{uv} \quad \forall (i, j) \in E_C, \forall \{u, v\} \in E_S. \tag{8} \]

This type of constraint is equivalent to (6).
Computing Splittable HVC Embeddings

\[ \sum_{(u,v)\in \delta^+} y_{uv}^{ij} - \sum_{(v,u)\in \delta^-} y_{vu}^{ij} = x_u^i - x_v^j \quad \forall \{u, v\} \in E_s. \] 
\[ y_{uv}^{ij} + y_{vu}^{ij} \leq \omega_{uv}^i + \omega_{uv}^j \quad \forall \{i, j\} \in E_C, \forall \{u, v\} \in E_s. \] 
\[ \sum_{(u,v)\in \delta^+(W)} y_{uv}^{ij} \geq 1 \quad \forall \{i, j\} \in E_C, \forall W \subset V_S : \text{map}_V(i) \in W, \text{map}_V(j) \notin W. \]

Derivation of a new constraint

- Across a cut \( W \), the amount of flow must be greater than 1 \((6\star)\).
Computing Splittable HVC Embeddings

\[
\sum_{(u,v) \in \delta_u^+} y_{uv}^i - \sum_{(v,u) \in \delta_u^-} y_{vu}^j = x^i_u - x^j_u \quad \forall\{u, v\} \in E_S. \tag{5}
\]
\[
\sum_{i \in V_C} B \cdot \omega_{uv}^i \leq l_{uv} \quad \forall\{u, v\} \in E_S. \tag{7}
\]
\[
y_{uv}^i + y_{vu}^j \leq \omega_{uv}^i + \omega_{uv}^j \quad \forall\{i, j\} \in E_C, \quad \forall\{u, v\} \in E_S. \tag{8}
\]
\[
\sum_{(u,v) \in \delta^+(W)} y_{uv}^i \geq 1 \quad \forall\{i, j\} \in E_C. \quad \forall W \subset V_S: \quad \text{map}_V(i) \in W, \quad \text{map}_V(j) \notin W \tag{6\star}
\]

Derivation of a new constraint

- Across a cut \( W \), the amount of flow must be greater than 1 \((6\star)\).
- By summing up Constraints \((8)\) accordingly, we obtain for \((i, j) \in E_C\) and cut \(W\) that
  \[
  \sum_{(u,v) \in \delta^+(W)} (\omega_{uv}^i + \omega_{uv}^j) \geq 1 \quad \text{holds.}
  \]
Computing Splittable HVC Embeddings

\[ l_{uv} \leq \text{cap}_{uv} \quad \forall \{u, v\} \in E_S. \tag{5} \]

\[ \sum_{(u,v) \in \delta^+} y_{uv}^i - \sum_{(v,u) \in \delta^-} y_{vu}^j = x_u^i - x_u^j \quad \forall (i,j) \in E_C, \quad \forall u \in V_S. \tag{6} \]

\[ \sum_{i \in V_C} B \cdot \omega_{uv}^i \leq l_{uv} \quad \forall \{u, v\} \in E_S. \tag{7} \]

\[ y_{uv}^i + y_{vu}^j \leq \omega_{uv}^i + \omega_{uv}^j \quad \forall (i,j) \in E_C, \quad \forall \{u, v\} \in E_S. \tag{8} \]

\[ \sum_{(u,v) \in \delta^+(W)} y_{uv}^i \geq 1 \quad \forall (i,j) \in E_C. \forall W \subset V_S:\ \text{map}_V(i) \in W, \ (6\star) \]

\[ \sum_{(u,v) \in \delta^+(W)} (\omega_{uv}^i + \omega_{uv}^j) \geq 1 \quad \forall (i,j) \in E_C. \forall W \subset V_S:\ \text{map}_V(i) \in W, \ (9) \]

\[ \text{map}_V(j) \notin W \]

**Derivation of a new constraint**

- Across a cut \( W \), the amount of flow must be greater than 1 \((6\star)\).
- By summing up Constraints \((8)\) accordingly, we obtain for \( (i,j) \in E_C \) and cut \( W \) that \( \sum_{(u,v) \in \delta^+(W)} (\omega_{uv}^i + \omega_{uv}^j) \geq 1 \) holds.
Hose-Based Virtual Cluster Embeddings

Computing (Fractional) HVC Embeddings

Computing Splittable HVC Embeddings

\[
\sum_{(u,v) \in \delta_u^+} y_{uv}^{ij} - \sum_{(v,u) \in \delta_u^-} y_{vu}^{ij} = x_i^u - x_i^u \quad \forall (i,j) \in EC, \quad \forall u \in V_S. \tag{6}
\]

\[
\sum_{i \in V_C} b \cdot \omega_{uv}^i \leq l_{uv} \quad \forall \{u,v\} \in E_S. \tag{7}
\]

\[
y_{uv}^{ij} + y_{vu}^{ij} \leq \omega_{uv}^i + \omega_{uv}^j \quad \forall (i,j) \in EC, \quad \forall \{u,v\} \in E_S. \tag{8}
\]

\[
\sum_{(u,v) \in \delta^+(W)} y_{uv}^{ij} \geq 1 \quad \forall (i,j) \in EC, \forall W \subset V_S : \map_V(i) \in W, \ (6\star) \map_V(j) \notin W \tag{9}
\]

\[
\sum_{(u,v) \in \delta^+(W)} (\omega_{uv}^i + \omega_{uv}^j) \geq 1 \quad \forall (i,j) \in EC, \forall W \subset V_S : \map_V(i) \in W, \map_V(j) \notin W \tag{9}
\]

Remarks

- **Given (9), we can always (re-)construct the flow variables** \(y_{uv}^{ij}\) **afterwards by breadth-first searches.**

- **Furthermore, this property does not depend on** (6\star).
### Computing Splittable HVC Embeddings

Let

\[ l_{uv} \leq \text{cap}_{uv} \quad \forall \{u, v\} \in E_S. \]  \hspace{1cm} (5)

\[
\sum_{(u,v)\in \delta^+_u} y^i_{uv} - \sum_{(v,u)\in \delta^-_u} y^i_{vu} = x^i_u - x^i_v \quad \forall (i,j) \in E_C, \quad \forall u \in V_S. \]  \hspace{1cm} (6)

\[
\sum_{i \in V_C} B \cdot \omega^i_{uv} \leq l_{uv} \quad \forall \{u,v\} \in E_S. \]  \hspace{1cm} (7)

\[
y^i_{uv} + y^j_{vu} \leq \omega^i_{uv} + \omega^j_{uv} \quad \forall (i,j) \in E_C, \forall \{u,v\} \in E_S. \]  \hspace{1cm} (8)

\[
\sum_{(u,v)\in \delta^+(W)} y^i_{uv} \geq 1 \quad \forall (i,j) \in E_C, \forall W \subset V_S : \text{map}_V(i) \in W, \text{map}_V(j) \notin W \]  \hspace{1cm} (9)

\[
\sum_{(u,v)\in \delta^+(W)} (\omega^i_{uv} + \omega^j_{uv}) \geq 1 \quad \forall (i,j) \in E_C, \forall W \subset V_S : \text{map}_V(i) \in W, \text{map}_V(j) \notin W \]  \hspace{1cm} (6*)

### Remarks

- Therefore Constraints (6), (8), and (6*) are not needed anymore!
Algorithm 5: HMPR

\[
\begin{align*}
\text{min} & \quad \sum_{\{u,v\} \in E_S} \text{cost}_{u,v} \cdot l_{uv} \\
\text{s.t.} & \quad l_{uv} \leq \text{cap}_{uv} \quad \forall\{u, v\} \in E_S. \\
& \quad \sum_{i \in V_C} \beta \cdot \omega^i_{uv} \leq l_{uv} \quad \forall\{u, v\} \in E_S. \\
& \quad \sum_{(u,v) \in \delta^+(W)} (\omega^i_{uv} + \omega^j_{uv}) \geq 1 \quad \forall (i, j) \in E_C. \forall W \subset V_S: \map_V(i) \in W, \map_V(j) \notin W
\end{align*}
\]
Computing Splittable HVC Embeddings

Algorithm 6: HMPR

\[
\min \sum_{\{u,v\} \in E_S} \text{cost}_{u,v} \cdot l_{uv}
\]

\[
l_{uv} \leq \cap_{uv} \quad \forall\{u, v\} \in E_S.
\]

\[
\sum_{i \in V_C} B \cdot \omega_{iuv} \leq l_{uv} \quad \forall\{u, v\} \in E_S.
\]

\[
\sum_{(u,v) \in \delta^+(W)} (\omega_{iuv}^i + \omega_{juv}^j) \geq 1 \quad \forall (i, j) \in E_C. \forall W \subset V_S : \map_V(i) \in W, \map_V(j) \notin W
\]

Exponential number of constraints, ...

... which can be separated in polynomial time.
Computing Splittable HVC Embeddings

**Algorithm 7: HMPR**

\[
\text{min} \sum_{\{u,v\} \in E_S} \text{cost}_{u,v} \cdot l_{uv} \tag{10}
\]

\[
l_{uv} \leq \text{cap}_{uv} \ \forall\{u,v\} \in E_S. \tag{11}
\]

\[
\sum_{i \in V_C} B \cdot \omega^i_{uv} \leq l_{uv} \ \forall\{u,v\} \in E_S. \tag{12}
\]

\[
\sum_{(u,v) \in \delta^+(W)} (\omega^i_{uv} + \omega^j_{uv}) \geq 1 \ \forall (i,j) \in E_C. \ \forall W \subset V_S : \text{map}_V(i) \in W, \text{map}_V(j) \notin W \tag{13}
\]

---

Exponential number of constraints, ...

...which can be separated in polynomial time.

Number of variables, ...

...in the order of \(O(N \cdot |E_S|)\) instead of \(O(N^2 \cdot |E_S|)\).
Computing Splittable HVC Embeddings

We can compute fractional edge embeddings, ... 

...but how to find node locations?
Heuristic Idea

- without capacities: “VC = HVC”
- reuse VC-ACE algorithm, but allow violation of capacities w.r.t. VC model
- violating capacities induces $k$ times the cost of the original edge

Algorithm 5: The HVC-ACE Embedding Heuristic

**Input:** Substrate $S = (V_S, E_S)$, request $VC(N, B, C)$, cost factor $k \geq 1$

**Output:** Splittable HVC-Embedding $map_V, map_E$

$E_{S'} \leftarrow \emptyset$

for $e \in E_S$ do

- $E_{S'} = E_{S'} \cup \{e, e'\}$
- $cap_{S'}(e) = cap(e)$ and $cap_{S'}(e') = \infty$
- $cost_{S'}(e) = cost(e)$ and $cost_{S'}(e') = cost(e) \cdot k$

$map_V, map_E \leftarrow VC-ACE(V_S, E_{S'}, VC(N, B, C))$

if $map_V \neq \text{null}$ then

- $map_E \leftarrow HMPR(VC(N, B, C), map_V)$

if $map_E \neq \text{null}$ then

- return $map_V, map_E$

return null
Computational Evaluation

What do we get by using HVC-ACE?
Topologies

- Fat trees with 12 port switches and 432 server overall
- MDCubes consisting of 4 BCubes with 12 port switches and \( k = 1 \), such that the topology contains 576 server

Figure : Fat tree (\( n=4 \))

Figure : MDCube (\( n=2, k=1 \))
Setup

Generation of Requests

- $N$ is chosen uniformly at random from the interval $\{10, \ldots, 30\}$.
- $B$ is uniformly distributed in the interval of $\{20\%, \ldots, 100\%\}$ w.r.t. to the available capacity of an unused link.
- $C = 1$ and the capacity of servers are 2.

Generation of Scenarios

- Requests are embedded over time using the VC-ACE algorithm.
- After system stabilization, a single data point is generated by considering the performance of both algorithms on the same substrate state and the same request.
**Metrics**

**Acceptance Ratio**
How many requests can VC-ACE embed compared to HVC-ACE?

**Footprint Change**
Assuming that both algorithms have found a solution, how much resources do we save by using HVC-ACE (compared to VC-ACE using 100%).
Results
Results on Fat Tree Topology

HVC-ACE can improve acceptance ratio dramatically.
Results on Fat Tree Topology

HVC-ACE saves $>10\%$ of resources for more than $20\%$ of the requests.
HVC-ACE can improve acceptance by around 20% on average.
Results on MDCube

HVC-ACE saves no resources.
Conclusion

Contributions

- Showed how to solve the classic VC embedding problem optimally.
- Defined formally the hose-based VC embedding problem and studied its computational complexity.
- Derived compact formulation for the splittable hose embedding.
- Validated that hose model can save a substantial amount of resources and increase the acceptance ratio.
Conclusion

Contributions

- Showed how to solve the classic VC embedding problem optimally.
- Defined formally the hose-based VC embedding problem and studied its computational complexity.
- Derived compact formulation for the splittable hose embedding.
- Validated that hose model can save a substantial amount of resources and increase the acceptance ratio.

Bottomline

- Complexity of specification can often be traded-off with the complexity of the respective embedding algorithms.
- We need to understand this trade-off better and explore the boundaries of specifications that we can efficiently embed.
References

A scalable, commodity data center network architecture.

Provisioning virtual private networks under traffic uncertainty.

Towards predictable datacenter networks.

Managing Data Transfers in Computer Clusters with Orchestra.
In *ACM SIGCOMM*, 2011.

The VPN conjecture is true.


Excursion: VPN Embeddings and the VPN Conjecture

Definition (VPN Embedding Problem (VPNEP) [7])

Given:
- Substrate network $G = (V, E)$ with edge costs $\text{cost} : E \rightarrow \mathbb{R}_0^+$
- Set of terminals $W \subseteq V$ with demands $b(i) \in \mathbb{R}^+$ for $i \in W$

Task:
- Find paths $P_{\{i,j\}}$ for all pairs $i, j \in W$, $i \neq j$, and
- bandwidth allocations $x_e \in \mathbb{R}_0^+$ on edges $e \in E$, s.t.
- $\sum_{i,j \in W : i \neq j, P_{\{i,j\}}} M_{\{i,j\}} \leq x_e$ holds for traffic matrices $M$,
- minimizing the cost $\sum_{e \in E} \text{cost}(e) \cdot x_e$. 
Excursion: VPN Embeddings and the VPN Conjecture

**Definition (VPN Embedding Problem (VPNEP) [7])**

Given:
- Substrate network $G = (V, E)$ with edge costs $\text{cost} : E \rightarrow \mathbb{R}_0^+$
- Set of terminals $W \subseteq V$ with demands $b(i) \in \mathbb{R}^+$ for $i \in W$

Task:
- Find paths $P_{\{i, j\}}$ for all pairs $i, j \in W$, $i \neq j$, and
- bandwidth allocations $x_e \in \mathbb{R}_0^+$ on edges $e \in E$, s.t.
  - $\sum_{i,j \in W : i \neq j, P_{\{i, j\}}} M_{\{i, j\}} \leq x_e$ holds for traffic matrices $M$,
  - minimizing the cost $\sum_{e \in E} \text{cost}(e) \cdot x_e$.

**Theorem**

*Finding a feasible solution for the capacitated VPNEP is NP-hard [7].*
Excursion: VPN Embeddings and the VPN Conjecture

Definition (VPN Embedding Problem (VPNEP) [7])

Given:
- Substrate network $G = (V, E)$ with edge costs $\text{cost} : E \rightarrow \mathbb{R}^+_0$
- Set of terminals $W \subseteq V$ with demands $b(i) \in \mathbb{R}^+$ for $i \in W$

Task:
- Find paths $P_{\{i,j\}}$ for all pairs $i, j \in W$, $i \neq j$, and
- bandwidth allocations $x_e \in \mathbb{R}^+_0$ on edges $e \in E$, s.t.
  - $\sum_{i,j \in W: i \neq j} M_{\{i,j\}} \leq x_e$ holds for traffic matrices $M$,
  - minimizing the cost $\sum_{e \in E} \text{cost}(e) \cdot x_e$.

Theorem

Finding a feasible solution for the capacitated VPNEP is NP-hard [7].

Theorem (By reduction from the VPNEP)

Finding a feasible solution for the HVCEP is NP-hard.
Excursion: VPN Embeddings and the VPN Conjecture

### Definition (VPN Embedding Problem (VPNEP) [7])

**Given:**
- Substrate network $G = (V, E)$ with edge costs $\text{cost} : E \rightarrow \mathbb{R}^+_0$
- Set of terminals $W \subseteq V$ with demands $b(i) \in \mathbb{R}^+$ for $i \in W$

**Task:**
- Find paths $P_{\{i,j\}}$ for all pairs $i, j \in W$, $i \neq j$, and
- bandwidth allocations $x_e \in \mathbb{R}^+_0$ on edges $e \in E$, s.t.
- $\sum_{i,j \in W: i \neq j, P_{\{i,j\}}} M_{\{i,j\}} \leq x_e$ holds for traffic matrices $M$,
- minimizing the cost $\sum_{e \in E} \text{cost}(e) \cdot x_e$.

### Theorem (VPN Conjecture)

*Tree routing and arbitrary routing solutions coincide for the VPNEP on uncapacitated graphs.* [5].
Excursion: VPN Embeddings and the VPN Conjecture

Definition (VPN Embedding Problem (VPNEP) [7])

Given:
- Substrate network $G = (V, E)$ with edge costs $\text{cost} : E \rightarrow \mathbb{R}^+$
- Set of terminals $W \subseteq V$ with demands $b(i) \in \mathbb{R}^+$ for $i \in W$

Task:
- Find paths $P_{\{i,j\}}$ for all pairs $i, j \in W$, $i \neq j$, and
- bandwidth allocations $x_e \in \mathbb{R}^+$ on edges $e \in E$, s.t.
- $\sum_{i,j \in W : i \neq j} M_{\{i,j\}} \leq x_e$ holds for traffic matrices $M$,
- minimizing the cost $\sum_{e \in E} \text{cost}(e) \cdot x_e$.

Theorem (VPN Conjecture)

Tree routing and arbitrary routing solutions coincide for the VPNEP on uncapacitated graphs. [5].

Theorem (Via the VPN Conjecture)

Algorithm VC-ACE solves the HVCEP when capacities are sufficiently large!