Virtual Network Embedding Approximations: Leveraging Randomized Rounding

IFIP Networking 2018

Matthias Rost
Technische Universität Berlin, Internet Network Architectures

Stefan Schmid
Universität Wien, Communication Technologies
Introduction: Virtual Network Embeddings

'Classic' Cloud Computing
- Only number and 'size' of virtual machines is given
- No guarantee on network performance

Goal: Virtual Networks (since ≈ 2006)
- Additionally: communication requirements given
- Network performance will be guaranteed
**Introduction: Virtual Network Embeddings**

**‘Classic’ Cloud Computing**
- Only number and ‘size’ of virtual machines is given
- No guarantee on network performance

**Goal: Virtual Networks (since ≈ 2006)**
- Additionally: communication requirements given
- Network performance will be guaranteed

---

**Embedding of Virtual Networks**
- Map virtual nodes to substrate nodes
- Map virtual edges to paths in the substrate
- Respecting mapping restrictions
- Respecting capacities
Introduction: Virtual Network Embeddings

Embedding of Virtual Networks

- Map virtual nodes to substrate nodes
- Map virtual edges to paths in the substrate
- Respecting mapping restrictions
- Respecting capacities

Virtual Network Embedding Problem (VNEP) ≈ 2006

Online: Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).
Offline: Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).
Introduction: Virtual Network Embeddings

Virtual Network Embedding Problem (VNEP) \(\approx 2006\)

**Online:** Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).

**Offline:** Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

Importance of the Virtual Network Embedding Problem

- Studied extensively over the last decade (> 100 publications)
- ‘Parent’ to Virtual Cluster Embeddings (\(\approx 2011\)) and Service Chain Embeddings (\(\approx 2013\))
Introduction: Virtual Network Embeddings

Virtual Network Embedding Problem (VNEP) ≈ 2006

Online: Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).
Offline: Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

Importance of the Virtual Network Embedding Problem

- Studied extensively over the last decade (> 100 publications)
- ‘Parent’ to Virtual Cluster Embeddings (≈ 2011) and Service Chain Embeddings (≈ 2013)

Virtual Cluster

Service Chain

VM₁, VM₂
VM₃, VM₄, VM₅

LB₁, LB₂
FW
NAT
Cache
Customer
Internet

cactus graphs: cycles intersect in at most one node
**Introduction: Virtual Network Embeddings**

**Virtual Network Embedding Problem (VNEP) ≈ 2006**

**Online:** Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).

**Offline:** Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

### Algorithmic Approaches to the VNEP

<table>
<thead>
<tr>
<th>Heuristics</th>
<th>Approximation Algorithms</th>
<th>Exact Algorithms</th>
</tr>
</thead>
</table>
| - no quality guarantee  
- polynomial-time  
- respects all constraints  
- very intensively studied | - quality guarantee  
- polynomial-time  
- cannot respect all constraints  
- not studied for general request graphs | - near-optimal solutions  
- exponential-time  
- respects all constraints  
- intensively studied |
Introduction: Virtual Network Embeddings

Virtual Network Embedding Problem (VNEP) \approx 2006

- **Online**: Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).
- **Offline**: Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

<table>
<thead>
<tr>
<th>Algorithmic Approaches to the VNEP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Heuristics</strong></td>
</tr>
<tr>
<td>- no quality guarantee</td>
</tr>
<tr>
<td>- polynomial-time</td>
</tr>
<tr>
<td>- respects all constraints</td>
</tr>
<tr>
<td>- very intensively studied</td>
</tr>
<tr>
<td><strong>Approximation Algorithms</strong></td>
</tr>
<tr>
<td>- <strong>quality guarantee</strong></td>
</tr>
<tr>
<td>- polynomial-time</td>
</tr>
<tr>
<td>- cannot respect all constraints(^1)</td>
</tr>
<tr>
<td>- not studied for general request graphs</td>
</tr>
<tr>
<td><strong>Exact Algorithms</strong></td>
</tr>
<tr>
<td>- near-optimal solutions</td>
</tr>
<tr>
<td>- exponential-time</td>
</tr>
<tr>
<td>- respects all constraints</td>
</tr>
<tr>
<td>- intensively studied</td>
</tr>
</tbody>
</table>

\(^1\) Not studied for general request graphs
**Introduction: Virtual Network Embeddings**

### Virtual Network Embedding Problem (VNEP) \( \approx 2006 \)

**Online:** Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).

**Offline:** Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

### Algorithmic Approaches to the VNEP

<table>
<thead>
<tr>
<th>Heuristics</th>
<th>Approximation Algorithms</th>
<th>Exact Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>no quality guarantee</td>
<td>quality guarantee</td>
<td>near-optimal solutions</td>
</tr>
<tr>
<td>polynomial-time</td>
<td>polynomial-time</td>
<td>exponential-time</td>
</tr>
<tr>
<td>respects all constraints</td>
<td>cannot respect all constraints(^1)</td>
<td>respects all constraints</td>
</tr>
<tr>
<td>very intensively studied</td>
<td>not studied for general request graphs</td>
<td>intensively studied</td>
</tr>
</tbody>
</table>

\(^1\) Not studied for general request graphs.
## Introduction: Virtual Network Embeddings

### Virtual Network Embedding Problem (VNEP) ≈ 2006

**Online:** Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).

**Offline:** Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

### Algorithmic Approaches to the VNEP

<table>
<thead>
<tr>
<th>Heuristics</th>
<th>Approximation Algorithms</th>
<th>Exact Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>no quality guarantee</td>
<td>quality guarantee</td>
<td>near-optimal solutions</td>
</tr>
<tr>
<td>polynomial-time</td>
<td>polynomial-time</td>
<td>exponential-time</td>
</tr>
<tr>
<td>respects all constraints</td>
<td>cannot respect all constraints</td>
<td>respects all constraints</td>
</tr>
<tr>
<td>very intensively studied</td>
<td>not studied</td>
<td>intensively studied</td>
</tr>
</tbody>
</table>

---

1 Matthais Rost and Stefan Schmid. “Charting the Complexity Landscape of Virtual Network Embeddings”. In: *Proc. IFIP Networking*. 2018
Introduction: Virtual Network Embeddings

Virtual Network Embedding Problem (VNEP) ≈ 2006

Online: Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).
Offline: Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

Algorithmic Approaches to the VNEP

- **Heuristics**
  - no quality guarantee
  - polynomial-time
  - respects all constraints
  - very intensively studied

- **Approximation Algorithms**
  - quality guarantee
  - polynomial-time
  - cannot respect all constraints
  - not studied for general request graphs

- **Exact Algorithms**
  - near-optimal solutions
  - exponential-time
  - respects all constraints
  - intensively studied

---


Matthias Rost (TU Berlin)  Virtual Network Embedding Approximations: Leveraging Randomized Rounding  IFIP Networking 2018
Contributions

- **Heuristics**
  - no quality guarantee
  - polynomial-time
  - respects all constraints
  - very intensively studied

- **Approximation Algorithms**
  - quality guarantee
  - polynomial-time
  - cannot respect all constraints
  - not studied for general request graphs

- **Exact Algorithms**
  - near-optimal solutions
  - exponential-time
  - respects all constraints
  - intensively studied

**Contributions of our paper**

1. First approximation algorithm for the offline VNEP for maximizing the profit.
2. Derived heuristics and studied performance in extensive computational study.

---

1. Matthias Rost and Stefan Schmid. “Charting the Complexity Landscape of Virtual Network Embeddings”. In: *Proc. IFIP Networking*. 2018
Formal Problem Statement & Integer Program
Formal Problem Statement & Integer Program

Substrate Network
- Capacitated graph
  \( G_S = (V_S, E_S) \)

For each request \( r \in \mathcal{R} \) ...
- Capacitated graph
  \( G_r = (V_r, E_r) \)
  - Mapping restrictions
  - Profit \( p_r > 0 \)
  - Valid mappings \( \mathcal{M}_r \)
For each request $r \in \mathcal{R}$...

- Capacitated graph $G_r = (V_r, E_r)$
- Mapping restrictions
  - Profit $p_r > 0$
  - Valid mappings $\mathcal{M}_r$
Formal Problem Statement & Integer Program

**Substrate Network**
- Capacitated graph $G_S = (V_S, E_S)$

**For each request $r \in \mathcal{R}$**
- Capacitated graph $G_r = (V_r, E_r)$
- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$

---

Request 1: $G_1$
- 100$

Request 2: $G_2$
- 50$
Formal Problem Statement & Integer Program

Substrate Network

- Capacitated graph $G_S = (V_S, E_S)$

For each request $r \in \mathcal{R}$...

- Capacitated graph $G_r = (V_r, E_r)$
- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$

Valid mappings: single virtual element mappings do not violate resource or mapping restrictions.
Formal Problem Statement & Integer Program

**Substrate Network**

- Capacitated graph \( G_S = (V_S, E_S) \)

**For each request \( r \in \mathcal{R} \) ...**

- Capacitated graph \( G_r = (V_r, E_r) \)
- Mapping restrictions
- Profit \( p_r > 0 \)
- Valid mappings \( \mathcal{M}_r \)

**Valid mappings:** single virtual element mappings do not violate resource or mapping restrictions.

Matthias Rost (TU Berlin) Virtual Network Embedding Approximations: Leveraging Randomized Rounding IFIP Networking 2018 18
For each request $r \in \mathcal{R}$ ...  

- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$

Valid mappings: **single virtual element mappings** do not violate resource or mapping restrictions.

Valid mappings for request 1: $\mathcal{M}_1 = \{m_1^1, m_2^2, m_3^3, \ldots\}$
Formal Problem Statement & Integer Program

For each request \( r \in \mathcal{R} \ldots \)

- Mapping restrictions
- Profit \( p_r > 0 \)
- Valid mappings \( \mathcal{M}_r \)

Valid mappings: **single virtual element mappings** do not violate resource or mapping restrictions.

Valid mappings for request 1: \( \mathcal{M}_1 = \{ m^1_2, m^2_2, m^3_3, \ldots \} \)
Formal Problem Statement & Integer Program

Test 1: \( M_1 = \{ m_2^1, m_2^2, m_3^3, \ldots \} \)
Formal Problem Statement & Integer Program

For each request $r \in R$ ... 

- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$

Valid mappings: **single virtual element mappings** do not violate resource or mapping restrictions.

Valid mappings for request 2: $\mathcal{M}_2 = \{m^1_2, m^2_2, m^3_3, \ldots\}$
Virtual Network Embedding Problem as Integer Program

- Is $k$-th mapping of request $r$ chosen?
- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $M_r$

\[ f_r^k \in \{0, 1\} \quad \forall r \in \mathcal{R}, m_r^k \in M_r \quad (1) \]

\[ \sum_{m_r^k \in M_r} f_r^k \leq 1 \quad \forall r \in \mathcal{R} \quad (2) \]

\[ \sum_{r \in \mathcal{R}} \sum_{m_r^k \in M_r} A(m_r^k, x) \cdot f_r^k \leq c_S(x) \quad \forall x \in R_S \quad (3) \]

\[ \max \sum_{r \in \mathcal{R}} \sum_{m_r^k \in M_r} p_r f_r^k \quad (4) \]
Substrate Network

For each request $r \in \mathcal{R}$ ...

- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$

Virtual Network Embedding Problem as Integer Program

- Is $k$-th mapping of request $r$ chosen?
- Select at most one mapping:

\[
\begin{align*}
  f_r^k &\in \{0, 1\} & \forall r \in \mathcal{R}, m_r^k &\in \mathcal{M}_r \quad (1) \\
  \sum_{m_r^k \in \mathcal{M}_r} f_r^k &\leq 1 & \forall r \in \mathcal{R} \quad (2) \\
  \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} A(m_r^k, x) \cdot f_r^k &\leq c_S(x) & \forall x \in \mathcal{R}_S \quad (3) \\
  \max \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} p_r f_r^k &\quad (4)
\end{align*}
\]
Virtual Network Embedding Problem as Integer Program

- Is $k$-th mapping of request $r$ chosen?
- Select at most one mapping:
- Enforce capacity for each resource $x$:

\[
\begin{align*}
 f_r^k &\in \{0, 1\} & \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r & (1) \\
 \sum_{m_r^k \in \mathcal{M}_r} f_r^k &\leq 1 & \forall r \in \mathcal{R} & (2) \\
 \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} A(m_r^k, x) \cdot f_r^k &\leq c_S(x) & \forall x \in R_S & (3) \\
 \max \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} p_r f_r^k & & & (4)
\end{align*}
\]
Formal Problem Statement & Integer Program

Substrate Network

For each request \( r \in \mathcal{R} \) ...

- Mapping restrictions
- Profit \( p_r > 0 \)
- Valid mappings \( \mathcal{M}_r \)

Virtual Network Embedding Problem as Integer Program

- Is \( k \)-th mapping of request \( r \) chosen?
- Select at most one mapping:
- Enforce capacity for each resource \( x \):
- Maximize the profit:

\[
\begin{align*}
& f_r^k \in \{0, 1\} \quad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \quad (1) \\
& \sum_{m_r^k \in \mathcal{M}_r} f_r^k \leq 1 \quad \forall r \in \mathcal{R} \quad (2) \\
& \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} A(m_r^k, x) \cdot f_r^k \leq c_S(x) \quad \forall x \in R_S \quad (3) \\
& \max \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} p_r f_r^k \quad (4)
\end{align*}
\]
Formal Problem Statement & Integer Program

Virtual Network Embedding Problem as Integer Program

- Is $k$-th mapping of request $r$ chosen?
  \[ f_r^k \in \{0, 1\} \quad \forall r \in R, \ m_r^k \in M_r \] (1)

- Select at most one mapping:
  \[ \sum_{m_r^k \in M_r} f_r^k \leq 1 \quad \forall r \in R \] (2)

- Enforce capacity for each resource $x$:
  \[ \sum_{r \in R} \sum_{m_r^k \in M_r} A(m_r^k, x) \cdot f_r^k \leq c_S(x) \quad \forall x \in R_S \] (3)

- Maximize the profit:
  \[ \max \sum_{r \in R} \sum_{m_r^k \in M_r} p_r f_r^k \] (4)

Example Solution to Integer Program: Profit 100$

Variables of request 1

- $f_1^1 = 1$
- $f_2^1 = 0$
- $f_3^1 = 0$

Variables of request 2

- $f_1^2 = 0$
- $f_2^2 = 0$
- $f_3^2 = 0$

...
### Virtual Network Embedding Problem as Integer Program

- Is $k$-th mapping of request $r$ chosen?  
  
  \[
  f_r^k \in \{0, 1\} \quad \forall r \in R, m_r^k \in M_r
  \]  

- Select at most one mapping:  
  \[
  \sum_{m_r^k \in M_r} f_r^k \leq 1 \quad \forall r \in R
  \]  

- Enforce capacity for each resource $x$:  
  \[
  \sum_{r \in R} \sum_{m_r^k \in M_r} A(m_r^k, x) \cdot f_r^k \leq c_S(x) \quad \forall x \in R_S
  \]  

- Maximize the profit:  
  \[
  \max \sum_{r \in R} \sum_{m_r^k \in M_r} p_r f_r^k
  \]

---

#### Example Solution to Integer Program: Profit 100$

<table>
<thead>
<tr>
<th>Variables of request 1</th>
<th>Variables of request 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1^1 = 1$</td>
<td>$f_2^1 = 0$</td>
</tr>
<tr>
<td>$f_2^2 = 0$</td>
<td>$f_2^2 = 0$</td>
</tr>
<tr>
<td>$f_3^3 = 0$</td>
<td>$f_3^3 = 0$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Matthias Rost (TU Berlin)  
Virtual Network Embedding Approximations: Leveraging Randomized Rounding  
IFIP Networking 2018  
28
Approximation Framework: Randomized Rounding

\footnote{P Raghavan and C D Thompson. “Provably Good Routing in Graphs: Regular Arrays”. In: Proc. 17th ACM STOC. 1985, pp. 79–87.}
Approximation Framework: Randomized Rounding

**Assumption (for now):**
Sets of valid mappings are of polynomial size and given. 
⇒ LP Formulation can be solved in polynomial-time.

### Virtual Network Embedding Problem as Linear Program

- **Is** $k$-th mapping of request $r$ chosen? 
  $$ f_r^k \in [0, 1] \quad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r $$ (5)

- **Select at most one mapping:**
  $$ \sum_{m_r^k \in \mathcal{M}_r} f_r^k \leq 1 \quad \forall r \in \mathcal{R} $$ (6)

- **Enforce capacity for each resource $x$:**
  $$ \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} A(m_r^k, x) \cdot f_r^k \leq c_S(x) \quad \forall x \in \mathcal{R}_S $$ (7)

- **Maximize the profit:**
  $$ \max \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} p_r f_r^k $$ (8)
Approximation Framework: Randomized Rounding

Virtual Network Embedding Problem as Linear Program

- Is $k$-th mapping of request $r$ chosen?

$$f_r^k \in [0, 1] \quad \forall r \in R, m_r^k \in M_r \quad (5)$$

Example Solution to Linear Program: Profit 133$

Variables of request 1

- $f_1^1 = 0.5$
- $f_2^1 = 0.3$
- $f_3^1 = 0.2$

... Variables of request 2

- $f_1^2 = 0.5$
- $f_2^2 = 0.16$
- $f_3^2 = 0$
Approximation Framework: Randomized Rounding

Virtual Network Embedding Problem as Linear Program

- Is $k$-th mapping of request $r$ chosen?
- ...

$f_r^k \in [0, 1]$ \quad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \quad (5)

Example Solution to Linear Program: Profit 133$

Variables of request 1

- $f^1_1 = 0.5$
- $f^2_1 = 0.3$
- $f^3_1 = 0.2$
- ...

Variables of request 2

- $f^1_2 = 0.5$
- $f^2_2 = 0.16$
- $f^3_2 = 0$
- ...

LP solution is convex combination valid mappings!

Let $\mathcal{D}_r = \{ (f_r^k, m_r^k) | f_r^k > 0, m_r^k \in \mathcal{M}_r \}$ denote these optimal convex combinations for request $r$. 

Matthias Rost (TU Berlin)  Virtual Network Embedding Approximations: Leveraging Randomized Rounding  IFIP Networking 2018 32
Approximation Framework: Randomized Rounding

Example Solution to Linear Program: Profit 133$

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{D_r\}_{r \in \mathcal{R}}$

foreach $r \in \mathcal{R}$ do
  choose $m_r^k$ with probability $f_r^k$
end
return solution
Approximation Framework: Randomized Rounding

Example Solution to **Linear Program**: Profit $\text{133}$

Variables of request 1

| $f_1^1$ = 0.5 | $f_1^2$ = 0.3 | $f_1^3$ = 0.2 |

Variables of request 2

| $f_2^1$ = 0.5 | $f_2^2$ = 0.16 | $f_2^3$ = 0 |

Idea: Treat weights as probabilities!

**Algorithm**: RoundingProcedure

**Input**: Optimal convex combinations $\{D_r\}_{r \in \mathcal{R}}$

**foreach** $r \in \mathcal{R}$ do
  | choose $m^k_r$ with probability $f^k_r$

**end**

**return** solution

**Rounding Outcomes**

<table>
<thead>
<tr>
<th>Iter.</th>
<th>Req. 1</th>
<th>Req. 2</th>
<th>Profit</th>
<th>max Load</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Approximation Framework: Randomized Rounding

Example Solution to Linear Program: Profit 133$

Variables of request 1

$f_1^1 = 0.5$
$f_1^2 = 0.3$
$f_1^3 = 0.2$

Variables of request 2

$f_2^1 = 0.5$
$f_2^2 = 0.16$
$f_2^3 = 0$

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input: Optimal convex combinations \( \{D_r\}_{r \in \mathcal{R}} \)

foreach \( r \in \mathcal{R} \) do
  choose \( m_r^k \) with probability \( f_r^k \)
end

return solution

Rounding Outcomes

<table>
<thead>
<tr>
<th>Iter.</th>
<th>Req. 1</th>
<th>Req. 2</th>
<th>Profit</th>
<th>max Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( m_1 )</td>
<td>( m_2 )</td>
<td>150$</td>
<td>200%</td>
</tr>
</tbody>
</table>
Approximation Framework: Randomized Rounding

Example Solution to Linear Program: Profit 133$

Variables of request 1

\[ f_1^1 = 0.5 \]
\[ f_2^2 = 0.3 \]
\[ f_1^3 = 0.2 \]

Variables of request 2

\[ f_2^1 = 0.5 \]
\[ f_2^2 = 0.16 \]
\[ f_2^3 = 0 \]

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations \( \{ D_r \}_{r \in \mathcal{R}} \)

foreach \( r \in \mathcal{R} \) do
  choose \( m_r^k \) with probability \( f_r^k \)
end

return solution

Rounding Outcomes

<table>
<thead>
<tr>
<th>Iter.</th>
<th>Req. 1</th>
<th>Req. 2</th>
<th>Profit</th>
<th>max Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( m_1^1 )</td>
<td>( m_2^2 )</td>
<td>150$</td>
<td>200%</td>
</tr>
<tr>
<td>2</td>
<td>( m_3^3 )</td>
<td>( \emptyset )</td>
<td>100$</td>
<td>100%</td>
</tr>
</tbody>
</table>
Approximation Framework: Randomized Rounding

Example Solution to Linear Program: Profit 133$.

Variables of request 1:
- $f_1^1 = 0.5$
- $f_1^2 = 0.3$
- $f_1^3 = 0.2$

Variables of request 2:
- $f_2^1 = 0.5$
- $f_2^2 = 0.16$
- $f_2^3 = 0$

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{D_r\}_{r \in R}$
foreach $r \in R$ do
    choose $m_r^k$ with probability $f_r^k$
end
return solution

Rounding Outcomes

<table>
<thead>
<tr>
<th>Iter.</th>
<th>Req. 1</th>
<th>Req. 2</th>
<th>Profit</th>
<th>max Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$m_1^1$</td>
<td>$m_2^1$</td>
<td>150$</td>
<td>200%</td>
</tr>
<tr>
<td>2</td>
<td>$m_1^3$</td>
<td>$\emptyset$</td>
<td>100$</td>
<td>100%</td>
</tr>
<tr>
<td>3</td>
<td>$m_1^1$</td>
<td>$m_2^1$</td>
<td>150$</td>
<td>200%</td>
</tr>
</tbody>
</table>
Approximation Framework: Randomized Rounding

Example Solution to Linear Program: Profit 133$

Variables of request 1

\[ f_1 = 0.5 \]
\[ f_2 = 0.3 \]
\[ f_3 = 0.2 \]

Variables of request 2

\[ f_2^1 = 0.5 \]
\[ f_2^2 = 0.16 \]
\[ f_2^3 = 0 \]

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations \( \{D_r\}_{r \in R} \)

foreach \( r \in R \) do

| choose \( m_r^k \) with probability \( f_r^k \)

end

return solution

Rounding Outcomes

<table>
<thead>
<tr>
<th>Iter.</th>
<th>Req. 1</th>
<th>Req. 2</th>
<th>Profit</th>
<th>max Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( m_1^1 )</td>
<td>( m_2^2 )</td>
<td>150$</td>
<td>200%</td>
</tr>
<tr>
<td>2</td>
<td>( m_1^3 )</td>
<td>( \emptyset )</td>
<td>100$</td>
<td>100%</td>
</tr>
<tr>
<td>3</td>
<td>( m_1^1 )</td>
<td>( m_1^2 )</td>
<td>150$</td>
<td>200%</td>
</tr>
<tr>
<td>4</td>
<td>( m_1^2 )</td>
<td>( m_2^1 )</td>
<td>150$</td>
<td>200%</td>
</tr>
</tbody>
</table>

Matthias Rost (TU Berlin) Virtual Network Embedding Approximations: Leveraging Randomized Rounding IFIP Networking 2018 38
Approximation Framework: Randomized Rounding

Example Solution to Linear Program: Profit 133$

Variables of request 1
- $f_1^1 = 0.5$
- $f_2^1 = 0.3$
- $f_3^1 = 0.2$

Variables of request 2
- $f_2^2 = 0.5$
- $f_1^2 = 0.16$
- $f_3^2 = 0$

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations \( \{D_r\}_{r \in \mathcal{R}} \)

foreach \( r \in \mathcal{R} \) do
  choose \( m_r^k \) with probability \( f_r^k \)
end

return solution

Rounding Outcomes

<table>
<thead>
<tr>
<th>Iter.</th>
<th>Req. 1</th>
<th>Req. 2</th>
<th>Profit</th>
<th>max Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( m_1 )</td>
<td>( m_2 )</td>
<td>150$</td>
<td>200%</td>
</tr>
<tr>
<td>2</td>
<td>( m_3 )</td>
<td>( \emptyset )</td>
<td>100$</td>
<td>100%</td>
</tr>
<tr>
<td>3</td>
<td>( m_1 )</td>
<td>( m_1 )</td>
<td>150$</td>
<td>200%</td>
</tr>
<tr>
<td>4</td>
<td>( m_2 )</td>
<td>( m_2 )</td>
<td>150$</td>
<td>200%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Approximation Algorithm for VNEP & Derived Heuristics
Randomized Rounding Approximation

**Algorithm: VNEP Approximation**

// perform preprocessing
compute optimal LP solution
compute \( \{D_r\}_{r \in \mathcal{R}} \) from LP solution
do
| solution \( \leftarrow \) RoundingProcedure(\( \{D_r\}_{r \in \mathcal{R}} \))
while (solution not \((\alpha, \beta, \gamma)\)-approximate and rounding tries not exceeded)

**Algorithm: RoundingProcedure**

Input : Optimal convex combinations \( \{D_r\}_{r \in \mathcal{R}} \)
foreach \( r \in \mathcal{R} \) do
| choose \( m_r^k \) with probability \( f_r^k \)
end
return solution
Approximation Algorithm for VNEP

Randomized Rounding Approximation

Algorithm: VNEP Approximation

// perform preprocessing
compute optimal LP solution
compute \( \{D_r\}_{r \in \mathcal{R}} \) from LP solution
do
| solution \( \leftarrow \) RoundingProcedure(\( \{D_r\}_{r \in \mathcal{R}} \))
while \( \text{solution not } (\alpha, \beta, \gamma)\)-approximate
and rounding tries not exceeded

Main Theorem: First Approximation for the Virtual Network Embedding Problem

The Algorithm returns \((\alpha, \beta, \gamma)\)-approximate solutions for the VNEP\(^a\) of at least an \( \alpha \) fraction of the optimal profit, and allocations on nodes and edges within factors of \( \beta \) and \( \gamma \) of the original capacities, respectively, with high probability.

\(^a\) restricted on cactus request graphs
Approximation Algorithm for VNEP

Randomized Rounding Approximation

Algorithm: VNEP Approximation

// perform preprocessing
compute optimal LP solution
compute \(\{D_r\}_{r \in R}\) from LP solution

\[\text{do}\]
\[\quad\text{solution } \leftarrow \text{RoundingProcedure}(\{D_r\}_{r \in R})\]
\[\quad\text{while } \left(\text{solution not } (\alpha, \beta, \gamma)\)-approximate \text{ and rounding tries not exceeded}\right)\]

Definition of Parameters

\[\alpha = 1/3\] (relative achieved profit)
\[\beta = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(R^V_S) \cdot \log(|R^V_S|)})\] (max node load)
\[\gamma = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(E_S) \cdot \log(|E_S|)})\] (max edge load)
\[\varepsilon = \max_{r \in R, x \in R_S} \frac{d_{\text{max}}(r, x)}{c_{\text{S}}(x)} \leq 1\] (max demand/capacity)

\[\Delta(X) = \max_{x \in X} \sum_{r \in R} (A_{\text{max}}(r, x)/d_{\text{max}}(r, x))^2\] (sum over \(R\) of squared max (total / single) alloc)

Main Theorem: First Approximation for the Virtual Network Embedding Problem

The Algorithm returns \((\alpha, \beta, \gamma)\)-approximate solutions for the VNEP\(^a\) of at least an \(\alpha\) fraction of the optimal profit, and allocations on nodes and edges within factors of \(\beta\) and \(\gamma\) of the original capacities, respectively, with high probability.

\(^{a}\text{restricted on cactus request graphs}\)
Approximation Algorithm for VNEP

**Randomized Rounding Approximation**

**Algorithm: VNEP Approximation**

// perform preprocessing
compute optimal LP solution
compute \( \{D_r\}_{r \in R} \) from LP solution
do
| solution ← RoundingProcedure(\( \{D_r\}_{r \in R} \))
| solution not \((\alpha, \beta, \gamma)\)-approximate
while (and rounding tries not exceeded)

**Definition of Parameters**

\[
\begin{align*}
\alpha &= \frac{1}{3} \quad \text{(relative achieved profit)} \\
\beta &= (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(X) \cdot \log(|R|)}) \quad \text{(max node load)} \\
\gamma &= (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(X) \cdot \log(|E|)}) \quad \text{(max edge load)} \\
\varepsilon &= \max_{r \in R, x \in R_S} \frac{d_{\max}(r, x)}{c_S(x)} \leq 1 \quad \text{(max demand/capacity)}
\end{align*}
\]

\[
\Delta(X) = \max_{x \in X} \sum_{r \in R} \left(\frac{A_{\max}(r, x)}{d_{\max}(r, x)}\right)^2 \quad \text{(sum over } R \text{ of squared max (total / single) alloc)}
\]

**Applicability in Practice: Computing \( \beta \) and \( \gamma \) is hard**

**Option 1: Overestimating \( \beta \) and \( \gamma \)**

→ bad solution returned after few iterations

**Option 2: Underestimating \( \beta \) and \( \gamma \)**

→ no solution returned after many iterations
Approximation Algorithm for VNEP

Randomized Rounding Approximation

Algorithm: VNEP Approximation

// perform preprocessing
compute optimal LP solution
compute \{D_r\}_{r \in R} from LP solution
do
| solution ← RoundingProcedure(\{D_r\}_{r \in R})
while (solution not (\alpha, \beta, \gamma)-approximate and rounding tries not exceeded)

Definition of Parameters

\[ \alpha = \frac{1}{3} \] (relative achieved profit)
\[ \beta = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(R^V_S) \cdot \log(|R^V_S|)}) \] (max node load)
\[ \gamma = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(E^S) \cdot \log(|E^S|)}) \] (max edge load)
\[ \varepsilon = \max_{r \in R, x \in R^S} \frac{d_{\text{max}}(r, x)}{c_S(x)} \leq 1 \] (max demand/capacity)

\[ \Delta(X) = \max_{x \in X} \sum_{r \in R} (A_{\text{max}}(r, x) / d_{\text{max}}(r, x))^2 \] (sum over R of squared max (total / single) alloc)

Applicability in Practice: Computing \( \beta \) and \( \gamma \) is hard

Option 1: Overestimating \( \beta \) and \( \gamma \)
→ bad solution returned after few iterations

Option 2: Underestimating \( \beta \) and \( \gamma \)
→ no solution returned after many iterations

Option 3: Consider Heuristics
Return best solution found within \( X \) iterations.
Derived Heuristics

Randomized Rounding Approximation

**Algorithm:** VNEP Approximation

// perform preprocessing
compute optimal LP solution
compute \( \{D_r\}_{r \in R} \) from LP solution
do
| solution ← RoundingProcedure(\( \{D_r\}_{r \in R} \))
while (solution not \((\alpha, \beta, \gamma)\)-approximate and rounding tries not exceeded)
Heuristic Idea: Return best of $X$

Algorithm: Heuristic Adaptation

// perform preprocessing
compute optimal LP solution
compute $\{D_r\}_{r \in R}$ from LP solution
do
| solution ← RoundingProcedure($\{D_r\}_{r \in R}$)
while rounding tries not exceeded
return best solution

Vanilla Rounding: $\text{RR}_{\text{MinLoad}}$

- still may exceed capacities
- return solution with least resource violations (among those: highest profit)
Derived Heuristics

**Heuristic Idea:** Return best of $X$

**Algorithm:** Heuristic Adaptation

// perform preprocessing
compute optimal LP solution
compute $\{D_r\}_{r \in R}$ from LP solution
do
| solution ← RoundingProcedure($\{D_r\}_{r \in R}$)
while rounding tries not exceeded
return best solution

**Vanilla Rounding:** $\text{RR}_{\text{MinLoad}}$
- still may exceed capacities
- return solution with least resource violations
  (among those: highest profit)

**Heuristic Rounding:** $\text{RR}_{\text{Heuristic}}$
- RoundingProcedure:
  discard chosen mappings exceeding capacities
- always yields feasible solutions
- return solution with highest profit

**Algorithm:** RoundingProcedure (Heuristic)

Input : Optimal convex combinations $\{D_r\}_{r \in R}$
foreach $r \in R$ do
| choose $m_r^k$ with probability $f_r^k$
| discard mapping if capacity violated
end
return solution
Taking a Step Back: How to compute LP Solutions?
Taking a Step Back: How to Compute LP Solutions?

**Assumption (for now):**
Sets of valid mappings are of polynomial size and given. 
⇒ LP Formulation can be solved in polynomial-time.

How to compute optimal convex combinations \( \{D_r\}_{r \in \mathcal{R}} \)?
Taking a Step Back: How to Compute LP Solutions?

How to compute optimal convex combinations $\{D_r\}_{r \in \mathcal{R}}$?

Obtaining convex combinations $\{D_r\}_{r \in \mathcal{R}}$ is challenging!

1. Presented LP has exponential size and cannot be used.
2. Classic LP formulation may yield meaningless solutions for cyclic graphs:
   - Theorem: Solution to classic LP Formulation cannot be decomposed into valid mappings.
   - Theorem: Classic LP Formulation has infinite integrality gap.
Taking a Step Back: How to Compute LP Solutions?

How to compute optimal convex combinations \( \{D_r\}_{r \in \mathcal{R}} \)?

Obtaining convex combinations \( \{D_r\}_{r \in \mathcal{R}} \) is challenging!

1. Presented LP has exponential size and cannot be used.
2. Classic LP formulation may yield **meaningless** solutions for cyclic graphs:
   - **Theorem**: Solution to classic LP Formulation **cannot be decomposed** into valid mappings.
   - **Theorem**: Classic LP Formulation has **infinite integrality gap**.

---

**Classic LP Formulation**

**Formulation 1:** Classic MCF Formulation for the VNEP

\[
\begin{align*}
\text{max} & \quad \sum_{r \in \mathcal{R}} c_r \cdot x_r, \\
\text{s.t.} & \quad \sum_{r \in \mathcal{R}} y_{ri} = c_r, \quad \forall r, i \in \mathcal{V}, \\
& \quad \sum_{r \in \mathcal{R}} y_{ri} = 0, \quad \forall r, i \in \mathcal{V}, \\
& \quad 0 \leq x_{ij} \leq x^*_{ij}, \quad \forall (i, j) \in \mathcal{E}, \\
& \quad \sum_{(i, j) \in \mathcal{E}} x_{ij} = x^* \cdot \sum_{r \in \mathcal{R}} y_{ri}, \\
& \quad \sum_{r \in \mathcal{R}} c_r \cdot x_r \leq c \cdot (x, y), \quad \forall (x, y) \in \mathcal{R}. 
\end{align*}
\]

---

**Structural Deficiency of Classic LP Formulation**

- **Request** \( G_r \)
- **Substrate** \( G_S \)
- **Classic LP Solution**
- **Decomposition Attempt**

Matthias Rost (TU Berlin)  Virtual Network Embedding Approximations: Leveraging Randomized Rounding  IFIP Networking 2018
How to compute optimal convex combinations \( \{D_r\}_{r \in R} \)?

**Novel Decomposable Linear Programming Formulation** (Details in the paper)

- **Intuition – ‘breaking cycles’**: fix any node on a cycle \( \rightarrow |V_S| \) copies of the classic Formulation.
- Formulation size increases by factor \( \mathcal{O}(|V_S|) \) and is only applicable for cactus request graphs.
Taking a Step Back: How to Compute LP Solutions?

How to compute optimal convex combinations \( \{D_r\}_{r \in \mathcal{R}} \)?

Novel Decomposable Linear Programming Formulation (Details in the paper)

- **Intuition – ‘breaking cycles’:** fix any node on a cycle \( \rightarrow |V_S| \) copies of the classic Formulation.
- **Formulation size increases by factor** \( \mathcal{O}(|V_S|) \) and is only applicable for cactus request graphs.

\[
\begin{align*}
\text{Virtual Cluster} & \quad \text{Service Chain} \\
\text{VM}_5 & \quad \text{VM}_1 \quad \text{VM}_2 \\
\text{VM}_4 & \quad \text{VM}_3
\end{align*}
\]

\[
\begin{align*}
\text{Customer} & \quad \text{Cache} \quad \text{FW} \quad \text{Internet} \\
\text{LB}_1 & \quad \text{LB}_2 \quad \text{NAT}
\end{align*}
\]

**cactus graphs:** cycles intersect in at most one node.
Taking a Step Back: How to Compute LP Solutions?

How to compute optimal convex combinations \( \{D_r\}_{r \in R} \)?

Novel Decomposable Linear Programming Formulation (Details in the paper)

- **Intuition – ’breaking cycles’:** fix any node on a cycle \( \rightarrow |V_S| \) copies of the classic Formulation.
- Formulation size increases by factor \( O(|V_S|) \) and **is only applicable for cactus request graphs**
- Generalization to arbitrary request graphs is possible\(^a\), but ...
  - Formulation size increases **super-polynomially** \( \rightarrow \) **fixed-parameter tractable** approximations.
  - No polynomial-time approximations can exist for arbitrary request graphs, unless \( \mathcal{P} = \mathcal{NP} \).

---

Computational Evaluation
Computational Evaluation

Substrate: GEANT Network

Requests: Synthetic Cactus Requests

Generation Parameters for 1,500 instances

Number of requests: 40, 60, 80, 100

Node-Resource Factor (NRF): 0.2, 0.4, 0.6, 0.8, 1.0

Edge-Resource Factor (ERF): 0.25, 0.5, 1.0, 2.0, 4.0

Instances per combination: 15

Computational Evaluation

Baseline Algorithm – $\text{MIP}_{\text{MCF}}$: solve classic MIP Formulation for up to 3 hours

<table>
<thead>
<tr>
<th>Acceptance Ratio</th>
<th>Avg. Node Load$^3$</th>
<th>Avg. Edge Load$^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Edge Resource Factor</td>
<td>Number of Requests</td>
<td>Number of Requests</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>4.0</td>
<td>97.0 98.1 98.2 98.0</td>
<td>100</td>
</tr>
<tr>
<td>2.0</td>
<td>84.8 85.0 85.1 85.1</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>71.2 69.2 68.2 66.1</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>72.4 62.5 56.2 55.3</td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>81.8 69.9 60.3 53.7</td>
<td></td>
</tr>
<tr>
<td>Number of Requests</td>
<td>40 60 80 100</td>
<td>40 60 80 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Computational Evaluation: Results

Vanilla Rounding Performance

- Relative profit $\approx 80 - 120\%$
- Resource augmentations mostly $< 200\%$
Computational Evaluation: Results

Vanilla Rounding Performance

- Relative profit $\approx 80 - 120\%$
- Resource augmentations mostly $< 200\%$
Relative profit ≈ 80 - 120%
Resource augmentations mostly < 200%
Relative profit ≈ 80 - 120%
Resource augmentations mostly < 200%

Relative profit ≈ 65 - 90%
min: 22.5% / mean: 73.8% / max: 101%
Computational Evaluation: Results

Vanilla Rounding Performance

- Relative profit ≈ 80 - 120%
- Resource augmentations mostly < 200%

Heuristic Rounding (w/o augmentations)

- Relative profit ≈ 65 - 90%
- min: 22.5% / mean: 73.8% / max: 101%
Computational Evaluation: Results

**Vanilla Rounding Performance**

- Relative profit ≈ 80 - 120%
- Resource augmentations mostly < 200%

**Heuristic Rounding (w/o augmentations)**

- Relative profit ≈ 65 - 90%
- min: 22.5% / mean: 73.8% / max: 101%
Conclusion
Conclusion: A First Step Towards provably Good Algorithms for the VNEP!

Contributions of our paper

1. First approximation algorithm for the offline VNEP for maximizing the profit.
2. Derived heuristics (w/o) resource augmentations achieves 73.8% on average.

Main Challenge: Computing Decomposable LP Solutions

- Classic LP Formulation
  - non-decomposable solutions
  - infinite integrality gap
- Novel LP Formulation
  - decomposable formulation for cactus request graphs
  - formulation size increases by factor $O(|V_S|)$
  - generalization to arbitrary request graphs possible

Future Work

- Other Rounding Heuristics / Column Generation for Solving the LP / Online Problem

---

Conclusion: A First Step Towards provably Good Algorithms for the VNEP!

Contributions of our paper

1. First approximation algorithm for the offline VNEP for maximizing the profit.
2. Derived heuristics (w/o) resource augmentations achieves 73.8% on average.

Main Challenge: Computing Decomposable LP Solutions

<table>
<thead>
<tr>
<th>Classic LP Formulation</th>
<th>Novel LP Formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>non-decomposable solutions</td>
<td>decomposable formulation for cactus request graphs</td>
</tr>
<tr>
<td>infinite integrality gap</td>
<td>formulation size increases by factor $O(</td>
</tr>
<tr>
<td></td>
<td>generalization to arbitrary request graphs possible$^4$</td>
</tr>
</tbody>
</table>

Future Work

Other Rounding Heuristics / Column Generation for Solving the LP / Online Problem

Conclusion: A First Step Towards *provably* Good Algorithms for the VNEP!

**Contributions of our paper**
1. First approximation algorithm for the offline VNEP for maximizing the profit.
2. Derived heuristics (w/o) resource augmentations achieves 73.8% on average.

**Main Challenge: Computing Decomposable LP Solutions**

**Classic LP Formulation**
- non-decomposable solutions
- infinite integrality gap

**Novel LP Formulation**
- decomposable formulation for cactus request graphs
- formulation size increases by factor $O(|V_S|)$
- generalization to arbitrary request graphs possible

**Future Work**

Other Rounding Heuristics / Column Generation for Solving the LP / Online Problem

---


