# Overlapping decompositions of Virtual Network Embedding

Alexis Schneider
Orange Research & LIPN
Châtillon, France
alexis.schneider@orange.com

Amal Benhamiche
Orange Research
Châtillon, France
amal.benhamiche@orange.com

Pierre Fouilhoux
Université Sorbonne Paris Nord,
CNRS, LIPN
Villetaneuse, France
pierre.fouilhoux@lipn.fr

Lucas Létocart
Université Sorbonne Paris Nord,
CNRS, LIPN
Villetaneuse, France
lucas.letocart@lipn.fr

Nancy Perrot
Orange Research
Châtillon, France
nancy.perrot@orange.com

## Abstract

We study decomposition methods for the Virtual Network Embedding (VNE) problem. The Flow Formulation (FF) of the problem suffers from a weak linear relaxation. We introduce the Virtual Overlapping Partitioning Formulation (VOPF), which is based on a partition of the virtual network into several subgraphs that can overlap on some nodes. We propose several decomposition strategies based on classic graph topologies. A column generation is implemented to compute the linear relaxation of the formulation. Experiments show that automatic and star decompositions are much stronger than previous approaches from the literature, and that overlapping nodes can improve the lower bounds for a surprisingly low computational overhead. This study provides key insights on how to decompose the VNE problem.

## Keywords

Virtual Network Embedding, Decomposition Methods, Column Generation, Graph Partitioning

## 1 Introduction

Network virtualization is an emerging paradigm that enables Telecommunication Operators to share their infrastructure between multiple virtual networks [3, 9]. It is at the heart of network slicing in 5G networks, where each virtual network corresponds to a service and is tailored to meet its requirements, such as speed, security, or coverage [1]. Network slicing is essential for deploying new, promising services with specific characteristics, such as remote surgeries or autonomous vehicles.

The core combinatorial structure of network virtualization is known as the Virtual Network Embedding (VNE) problem. The VNE problem can be defined as follows. We consider simple, connected, undirected graphs. Let us denote the virtual network as $\mathcal{G}_r = (V_r, E_r)$ with $n_r$ nodes; and the substrate network as $\mathcal{G}_s = (V_s, E_s)$ with $n_s$ nodes. Allocating the substrate resources to the virtual demands is to find a *mapping* (also called an embedding) as follows:

*Definition 1.1.* A mapping $m = (m_V, m_E)$ of $\mathcal{G}_r$ on $\mathcal{G}_s$ is a pair of functions where:

- $m_V : V_r \to V_s$ is called the *node placement*;
- $m_E : E_r \to P_s$, is called the *edge routing*, where $P_s$ is the set of loop-free paths of $\mathcal{G}_s$ and for each $\overline{e} = (\overline{u}, \overline{v}) \in E_r$, $m_E(\overline{e})$ is a path of $\mathcal{G}_s$ whose endpoints are $m_V(\overline{u})$ and $m_V(\overline{v})$.

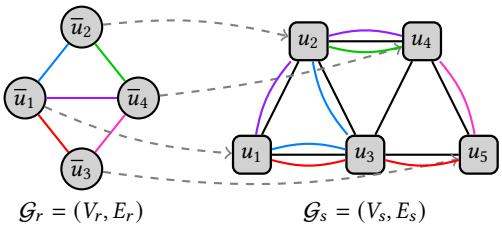


**Figure 1: Example of a mapping of a virtual network (on the left) on a substrate network (on the right).**

In this work, we consider a *one-to-one* node placement: each virtual node has to be placed on a different substrate node, which is a standard assumption in VNE literature. However, our main results can be extended ro the *many-to-one* case. A virtual node $\overline{u} \in V_r$ (resp. virtual edge $\overline{e} \in E_r$) has an integer demand $d_{\overline{u}}$ (resp. $d_{\overline{e}}$). A substrate node $u \in V_s$ (resp. a substrate edge $e \in E_s$) has an integer capacity $c_u$ (resp. $c_e$). A mapping is said to be *feasible* when the capacity constraints are satisfied, i.e., the sum of the demands of virtual nodes (resp., edges) being placed on a substrate node (resp., being routed on a substrate edge) is less than or equal to the capacity of this node (resp., edge).

Figure 1 illustrates the embedding of a virtual graph on the left over a substrate graph on the right. The dotted arrows show the placement of virtual nodes. A virtual edge of a given color is routed using a substrate path of the same color. In this example, $m_V(\overline{u}_1) = u_1$, $m_V(\overline{u}_3) = u_5$ and $m_E(\overline{u}_1, \overline{u}_3) = \{(u_1, u_3), (u_3, u_5)\}$.

Using a substrate node unit (resp. edge) induces a usage cost $w_u$ (resp. $w_e$). The cost of a mapping $m$, denoted $W_m$, corresponds to the sum of the placement and routing costs: $W_m = \sum_{\overline{u} \in V_r} d_{\overline{u}} w_{m_V(\overline{u})} + \sum_{\overline{e} \in E_r} \sum_{e \in m_E(\overline{e})} d_{\overline{e}} w_e$. The VNE problem can be formulated as follows:

*Definition 1.2.* Given a virtual network $\mathcal{G}_r = (V_r, E_r)$ with demands $d$, a substrate network $\mathcal{G}_s = (V_s, E_s)$ with capacities $c$ and costs $w$, the Virtual Network Embedding Problem (VNE) is to find the minimum cost embedding of $\mathcal{G}_r$ on $\mathcal{G}_s$

The VNE problem is known to be NP-complete [2, 25], although some polynomial cases have been exhibited [4, 24]. Heuristics and metaheuristics have been heavily investigated for the problem, see the survey in [13]. Regarding mathematical programming approaches, the compact Flow Formulation is proposed in [9, 18], but it suffers from very weak linear relaxations, as shown

*INOC '26, Liège (Belgium)*



10.48786/inoc.2026.16

in [20]. A stronger *Path Formulation*, where the virtual network is decomposed into edges, is proposed in [20, 26], and its linear relaxation is solved with Column Generation. However, we observed that this formulation remains weak. We introduced in [5] a stronger formulation based on a partition of the virtual network. In this work, we introduce an even stronger formulation, based on an overlapping decomposition of the virtual network, providing a flexible framework that can lead to very strong linear relaxations.

## 2 Formulations

Although the edges are undirected, we consider that they are ordered sets (i.e., an arbitrary orientation is given), to formulate VNE as a flow problem. The path routing a virtual edge can contain a substrate edge $(u, v) \in E_s$ in either direction $(u, v)$ or $(v, u)$, since the network is undirected. Thus we introduce $\mathcal{G}'_s = (V_s, E'_s)$ the bi-directed version of the substrate network, where $E'_s = \bigcup_{(u,v)\in E_s} \{(u, v), (v, u)\}$. This bi-orientation technique is standard [23]. Additionally, for a substrate node $u \in V_s$, let $\delta^+(u)$ (resp. $\delta^-(u)$) be the outgoing (resp. ingoing) edges of node $u$ in $E'_s$.

### 2.1 The Flow Formulation

A Flow Formulation was introduced in the directed case in [9, 18], and adapted to the undirected version in [6]. Two sets of binary variables are considered. The placement variable $x_{\overline{u}u}$ takes the value 1 if and only if (iff) virtual node $\overline{u} \in V_r$ is placed on substrate node $u \in V_s$. The flow variable $y_{\overline{e}(u,v)}$ (resp. $y_{\overline{e}(v,u)}$) takes value 1 iff the path routing virtual edge $\overline{e} \in E_r$ uses substrate edge $(u, v) = e \in E_s$, from $u$ to $v$ (resp. from $v$ to $u$). Additionally, an easy pre-treatment is to set to 0 the variable $x_{\overline{u}u}$ (resp. $y_{\overline{e}e}$) if $d_{\overline{u}} > c_u$ (resp. $d_{\overline{e}e} > c_e$), for $\overline{u} \in V_r, u \in V_s$ (resp. $\overline{e} \in E_r, e \in E'_s$). In what follows, we thus consider that $d_{\overline{u}} \leq c_u$, for $\overline{u} \in V_r,\ u \in V_s$; and $d_{\overline{e}} \leq c_e$, for any $\overline{e} \in E_r,\ e \in E_s$. The Flow Formulation $(FF)$ is:

$$\min \quad \sum_{u\in V_s} \sum_{\overline{u}\in V_r} d_{\overline{u}} w_u x_{\overline{u}u} + \sum_{e\in E'_s} \sum_{\overline{e}\in E_r} d_{\overline{e}} w_e y_{\overline{e}e} \tag{1a}$$

$$\text{s.t.} \quad \sum_{u\in V_s} x_{\overline{u}u} = 1 \qquad \forall \overline{u} \in V_r \tag{1b}$$

$$x_{\overline{u}u} - x_{\overline{v}u} = \sum_{e\in\delta^+(u)} y_{\overline{e}e} - \sum_{e\in\delta^-(u)} y_{\overline{e}e} \qquad \forall (\overline{u}, \overline{v}) = \overline{e} \in E_r,\ \forall u \in V_s \tag{1c}$$

$$\sum_{\overline{u}\in V_r} x_{\overline{u}u} \leq 1 \qquad \forall u \in V_s \tag{1d}$$

$$\sum_{\overline{e}\in E_r} d_{\overline{e}} (y_{\overline{e}(u,v)} + y_{\overline{e}(v,u)}) \leq c_e \qquad \forall (u, v) \in E_s \tag{1e}$$

$$x_{\overline{u}u} \leq \sum_{e\in\delta^+(u)} y_{(\overline{u},\overline{v})u} \qquad \forall (\overline{u}, \overline{v}) \in E_r,\ \forall u \in V_s \tag{1f}$$

$$x_{\overline{u}u} \in \{0, 1\} \qquad \forall \overline{u} \in V_r, \forall u \in V_s,$$

$$y_{\overline{e}e} \in \{0, 1\} \qquad \forall \overline{e} \in E_r, \forall e \in E'_s$$

The objective function (1a) expresses an embedding cost taking into account placement and routing costs. The constraints (1b) ensure valid virtual node placement. The constraints (1c) are the flow conservation constraints and guarantee that each virtual edge is associated with a routing path in the substrate graph whose end nodes host its endpoints. Constraints (1d) ensure one-to-one node placement. Constraints (1e) ensure that edge capacities are respected. Constraints (1f) are the flow departure valid inequalities introduced in [6], which translates that the routing path of a virtual edge must use an outgoing substrate edge of the placement of the source of the virtual edge. Although these inequalities improve the linear relaxation of $(FF)$, the formulation remains numerically weak.

### 2.2 Virtual Overlapping Partition Formulation

To obtain better lower bounds, the Path Formulation [20, 26] was proposed, but our preliminary experiments indicate that it yields marginal improvements over $(FF)$. Our previous work [5] introduced a stronger decomposition, the Virtual Partition Formulation (VPF), based on a strict partition of the virtual network into virtual subgraphs. However, considering a strict partition is rigid and often limits decomposition quality. To overcome these limitations and enhance formulation flexibility, we propose the use of *overlapping partitions* (or covers), in which virtual nodes can appear in several virtual subgraphs, as follows.

An overlapping partition (or cover) $\sigma$ of $\mathcal{G}_r$ into $k_r$ subgraphs is a set of connected virtual subgraphs, such that nodes are in at least one subgraph, and edges are in at most one (overlapping edges are not allowed). We denote a virtual subgraph as $\mathcal{H} = (V_r^{\mathcal{H}}, E_r^{\mathcal{H}})$. Some edges are not included in any subgraph: these are the *virtual cut edges*, denoted $E_r^0$. The set of overlapping nodes of an *overlapping partition* is denoted $O_r$; and, for each virtual node $\overline{u}$, we denote the set of subgraphs that contains $\overline{u}$ by $\sigma_{\overline{u}}$. Each subgraph of that set is also assigned an arbitrary index $\mathcal{H}_{\overline{u}}^i$, with $i \in \{1, \ldots, |\sigma_{\overline{u}}|\}$. For notational purpose, we consider that $\mathcal{H}_{\overline{u}}^{|\sigma_{\overline{u}}|+1} = \mathcal{H}_{\overline{u}}^1$

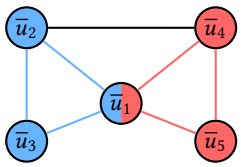


**Figure 2: Overlapping partition of a virtual network**

Figure 2 gives an example of an overlapping partition of a small virtual network. Here, the two virtual subgraphs are ( $\{\overline{u}_1, \overline{u}_2, \overline{u}_3\}, \{(\overline{u}_1, \overline{u}_2), (\overline{u}_2, \overline{u}_3), (\overline{u}_3, \overline{u}_1)\}$ ) (in blue) and ( $\{\overline{u}_1, \overline{u}_4, \overline{u}_5\}, \{(\overline{u}_1, \overline{u}_4), (\overline{u}_4, \overline{u}_5), (\overline{u}_5, \overline{u}_1)\}$) (in red); $O_r = \{\overline{u}_1\}$; and $E_r^0 = \{(\overline{u}_2, \overline{u}_4)\}$.

For a virtual subgraph $\mathcal{H} \in \mathsf{H}$, the set of mappings is denoted $\mathcal{M}_{\mathcal{H}}$. For a mapping $m \in \mathcal{M}_{\mathcal{H}}$, $\mathrm{x}_{\overline{u}u}^m$ is equal to one iff $\overline{u}$ is placed on $u$ in $m$, $\mathrm{y}_{\overline{e}e}^m$ is equal to one iff the path routing $\overline{e}$ uses substrate edge $e \in E'_s$ in $m$. the cost of a submapping $m \in \mathcal{M}_{\mathcal{H}}$ is then:

$$w_m = \sum_{\overline{u}\in V_r^{\mathcal{H}}} \sum_{u\in V_s} d_{\overline{u}} \frac{\mathrm{x}_{\overline{u}u}^m}{|H_{\overline{u}}|} w_u + \sum_{\overline{e}\in E_r^{\mathcal{H}}} \sum_{e\in E'_s} d_{\overline{e}} \mathrm{y}_{\overline{e}e}^m w_e \tag{2}$$

We now introduce the new formulation. Two sets of binary variables are considered. The *submapping* variables, denoted $\lambda_m^{\mathcal{H}}$, takes the value 1 iff the mapping $m \in \mathcal{M}_{\mathcal{H}}$ is selected for subgraph $\mathcal{H} \in \sigma$. The flow variables $y_{\overline{e}e}$ take value 1 iff the path routing virtual cut edge $\overline{e} \in E_r^0$ uses substrate edge $e \in E'_s$. The Virtual Overlapping Partition Formulation, for a given overlapping partition $\sigma$, is denoted $(VOPF_\sigma)$ and is the following.

$$\min \quad \sum_{\mathcal{H}\in\sigma}\sum_{m\in\mathcal{M}_{\mathcal{H}}} w_m\lambda_m^{\mathcal{H}} + \sum_{\overline{e}\in E_r^0}\sum_{e\in E_s'} d_{\overline{e}} w_e y_{\overline{e}e} \tag{3a}$$

$$\text{s.t.} \quad \sum_{m\in\mathcal{M}_{\mathcal{H}}} \lambda_m^{\mathcal{H}} \geq 1 \qquad \forall \mathcal{H}\in\sigma \tag{3b}$$

$$\sum_{m\in\mathcal{M}_{\mathcal{H}_{\overline{o}}^i}} x_{\overline{o}u}^m \lambda_m^{\mathcal{H}_{\overline{o}}^i} = \sum_{m\in\mathcal{M}_{\mathcal{H}_{\overline{o}}^{i+1}}} x_{\overline{o}u}^m \lambda_m^{\mathcal{H}_{\overline{o}}^{i+1}} \qquad \forall i\in\{1,\dots,|\sigma_{\overline{o}}|\}, \; \forall u\in V_s, \forall \overline{o}\in O_r \tag{3c}$$

$$\sum_{\mathcal{H}\in\sigma_{\overline{u}}}\sum_{m\in\mathcal{M}_{\mathcal{H}}} \frac{x_{\overline{u}u}^m}{|\sigma_{\overline{u}}|}\lambda_m^{\mathcal{H}} - \sum_{\mathcal{H}\in\sigma_{\overline{v}}}\sum_{m\in\mathcal{M}_{\mathcal{H}}} \frac{x_{\overline{v}u}^m}{|\sigma_{\overline{v}}|}\lambda_m^{\mathcal{H}} = \sum_{e\in\delta^+(u)} y_{\overline{e}e} - \sum_{e\in\delta^-(u)} y_{\overline{e}e} \qquad \forall (\overline{u},\overline{v}) = \overline{e}\in E_r^0, \; \forall u\in V_s \tag{3d}$$

$$\sum_{\mathcal{H}\in\sigma}\sum_{m\in\mathcal{M}_{\mathcal{H}}}\sum_{\overline{u}\in V_r^{\mathcal{H}}} \frac{x_{\overline{u}u}^m}{|\sigma_{\overline{u}}|}\lambda_m^{\mathcal{H}} \leq 1 \qquad \forall u\in V_s \tag{3e}$$

$$\sum_{\mathcal{H}\in\sigma}\sum_{m\in\mathcal{M}_{\mathcal{H}}}\sum_{\overline{e}\in E_r^{\mathcal{H}}} d_{\overline{e}}(\gamma_{\overline{e}(u,v)}^m + \gamma_{\overline{e}(v,u)}^m)\lambda_m^{\mathcal{H}} + \sum_{\overline{e}\in E_r^0} d_{\overline{e}}(y_{\overline{e}(u,v)} + y_{\overline{e}(v,u)}) \leq c_e \qquad \forall (u,v) = e\in E_s \tag{3f}$$

$$\sum_{\mathcal{H}\in\sigma_{\overline{u}}}\sum_{m\in\mathcal{M}_{\mathcal{H}}} \frac{x_{\overline{u}u}^m}{|\sigma_{\overline{u}}|}\lambda_m^{\mathcal{H}} \leq \sum_{e\in\delta^+(u)} y_{(\overline{u},\overline{v})e} \qquad \forall (\overline{u},\overline{v})\in E_r^0, \; \forall u\in V_s \tag{3g}$$

$$\lambda_m^{\mathcal{H}} \in \{0,1\} \qquad \forall m\in\mathcal{M}_{\mathcal{H}}, \forall \mathcal{H}\in\sigma$$

$$y_{\overline{e}e} \in \{0,1\} \qquad \forall \overline{e}\in E_r^0, \forall e\in E_s'$$

The objective function (3a) expresses the embedding cost. Constraints (3b) are the convexity constraints, ensuring that one submapping is used for each virtual subgraph. Constraints (3c) ensure that the node placement of an overlapping node is the same across the submappings of all subgraph containing that overlapping node. Constraints (3d) are the flow conservation constraints for the virtual cut edges. Constraints (3e) ensures one-to-one node placement. Constraints (3f) ensure that edge capacities are respected. Constraints (3*g*) are the flow departure inequalities for the virtual cut edges. Compared to the formulation introduced in [5], the constraints (3c) are added, while the other constraints are adapted to take into account the presence of overlapping nodes.

Note that $(VOPF_\sigma)$ provides a linear relaxation at least as tight as $(FF)$ for any overlapping partition $\sigma$, as shown in [5], with trivial cases where it is stronger. This formulation has the particularity that the variable $\lambda_m^{\mathcal{H}}$ represents the same structure as the solution of the original problem: a feasible mapping. Such decompositions are seldom explored in the literature (see, e.g., [8, 27]). Moreover, it has a similar structure to (FF): the placement variables are replaced with the submapping variables, and the flow variables remain. Actually, if subgraphs are single nodes, $(VOPF_\sigma) = (FF)$. Another original aspect is the presence of overlapping elements in the substructure. This is known, in the context of Lagrangian Relaxation, as *Lagrangian Decomposition* [14], but has rarely been investigated through Column Generation (see, e.g., [10, 17]). It typically results in better linear relaxations, but a difficult convergence, due to equalities (3c) that are in large numbers and require many columns to be covered.

## 3 Decomposition strategies

An important advantage of our new formulation lies in its flexibility. While traditional decomposition methods typically follow a fixed problem structure (e.g., individual commodities in flow problems), our framework allows for decomposition along any substructure of the virtual network. This leads to a fundamental research question with theoretical and practical implications: which virtual substructure induces the best decomposition? In this section, we propose strategies based on classic graph topologies and a graph partition tool. For each strategy, we describe the algorithm implemented (strict and overlapping versions), and show the partitions on the Polska network, from SNDlib [21] (Figure 3).

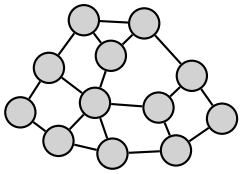

**Figure 3: The Polska network**

*Edge decomposition:* In the edge decomposition, every virtual edge $\overline{e}\in E_r$ is a virtual subgraph. Hence, virtual nodes with a degree of more than two are overlapping, as they are present in each subgraph of their adjacent edge. With this strategy, $(VOPF_\sigma)$ is equivalent to the *Path Formulation (PF)* proposed in [19, 20].

*Star decompositions:* The *strict star partition* is computed as follows. The biggest star (with adjacent edges) is iteratively removed from the residual virtual network (which is the original virtual network at the beginning) until this residual network only contains single nodes. The partition obtained is the stars removed and the single nodes remaining. The *overlapping star partition* is computed similarly, but only the center node of the selected star is removed from the residual virtual network. The strict (left) and overlapping (right) star partition obtained on the Polska network are shown in Figure 4. Each virtual subgraph has a different color. Overlapping nodes are shown with several colors. Note that the overlapping partition contains larger stars and fewer cut edges.

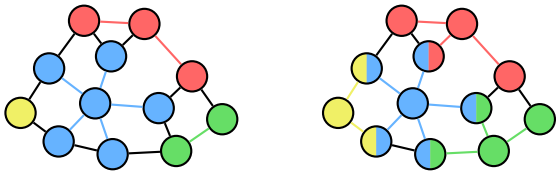

**Figure 4: Stars decomposition of Polska**

*Path decomposition:* The *strict path partition* is computed as follows. The longest shortest path between any pair of virtual nodes in the residual virtual network is iteratively removed, until only single nodes remain. The partition obtained consists of the paths removed and the single nodes remaining. In the *overlapping path partition*, only the edges of the paths constructed are removed, while the nodes can be used in new paths. The strict (left) and overlapping (right) path partitions obtained on the Polska network are shown in Figure 5. Note that the overlapping partition contains more paths, longer paths, and many overlapping nodes, because all edges are in a subgraph (as for the edge

decomposition).

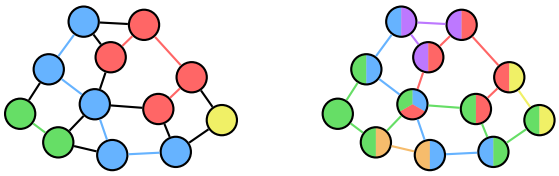

**Figure 5: Paths decompositions of Polska**

*Cycle decomposition:* The *strict cycle partition* is computed as follows. The smallest cycle from a cycle basis of the residual virtual network is removed iteratively. The partition obtained consists of the cycles removed and the single nodes remaining. In the *overlapping cycle partition*, only the edges of the cycle constructed are removed, while the nodes remain and can be used in other cycles. In early tests, we also considered using the longest cycle from the cycle basis, which could result in stronger relaxation. However, since cycles often share edges, this resulted in a very small number of cycles being generated (often only one). The strict (left) and overlapping (right) cycle partitions obtained on the Polska network are shown in Figure 6: for this network, they are the same.

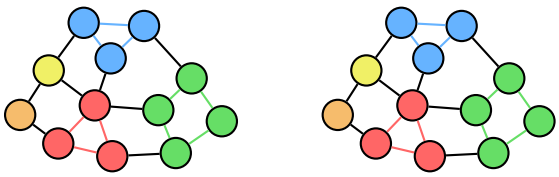

**Figure 6: Cycles decompositions of Polska**

*Automatic partition:* Finally, we consider partitions given by the automatic graph partitioner METIS [15], as we have done in [5]. METIS aims to produce a partition of the graph into a given number of connected subgraphs, such that the sizes of the subgraphs are balanced and that the number of cut edges is minimized. For the size of networks considered, it runs extremely fast.

METIS produces the *strict automatic partitions*. To compute *overlapping automatic partitions*, we implement an algorithm that adds an overlapping node to a strict partition iteratively as follows. The virtual subgraphs of the original METIS partition are removed from the residual virtual network. Then, the virtual node that is the most connected to another virtual subgraph (at least two neighbors) on the residual virtual network is added to that subgraph. Edges that connect that node to its new subgraphs are added to the subgraph and removed from the residual virtual network. The algorithm runs until $k_o$ overlapping nodes are added, or until no such nodes are found. The strict (left) and overlapping (right) automatic partitions obtained on the Polska network, $k_r = 2$ and $k_o = 3$, are shown in Figure 6. The algorithm adds only one overlapping virtual node to the original partition, which allows the removal of two cut edges.

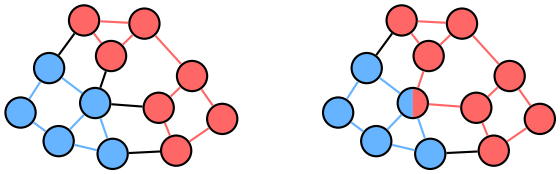

**Figure 7: Automatic decompositions of Polska**

## 4 Column Generation

Column generation is an algorithm that solves linear programs with an exponential number of variables [11], by adding iteratively the variables, or *columns*. The Restricted Master Problem (*RMP*) refers to the linear relaxation of $(VOPF_\sigma)$ with a subset of columns. A *Pricer Problem* generates the column with the minimal reduced costs. In our case, for a given virtual subgraph $\mathcal{H} \in \sigma$, the pricing problem $PP_\mathcal{H}$ is actually a VNE problem of $\mathcal{H}$ on $\mathcal{G}_s$. It can be solved using the Flow Formulation, with the following objective:

$$
\begin{aligned}
v(PP_\mathcal{H}) = \min -\theta_\mathcal{H} &+ \sum_{\overline{u} \in V_r^\mathcal{H}} \sum_{u \in V_s} \frac{(w_u - \gamma_u)}{|\sigma_{\overline{u}}|} x_{\overline{u}u} + \sum_{\overline{e} \in E_r^\mathcal{H}} \sum_{e \in E_s} (w_e - \delta_e)\, y_{\overline{e}e} \\
&+ \sum_{\substack{\overline{o} \in O_r, i \in \{1,\dots,|\sigma_{\overline{o}}|\} \\ \mathcal{H} = \mathcal{H}^i_{\overline{o}}}} \sum_{u \in V_s} (-\alpha_{\overline{o},u,i} + \alpha_{\overline{o},u,i+1}) x_{\overline{o}u} \\
&+ \sum_{\substack{(\overline{u},\overline{v}) = \overline{e} \in E_r^0 \\ \overline{v} \in V_r^\mathcal{H}}} \sum_{u \in V_s} \frac{\beta_{\overline{e}u}}{|\sigma_{\overline{v}}|} x_{\overline{v}u} - \sum_{\substack{(\overline{u},\overline{v}) = \overline{e} \in E_r^0 \\ \overline{u} \in V_r^\mathcal{H}}} \sum_{u \in V_s} \frac{\beta_{\overline{e}u}}{|\sigma_{\overline{u}}|} x_{\overline{u}u} \\
&- \sum_{u \in V_s} \sum_{\substack{(\overline{u},\overline{v}) = \overline{e} \in E_r^0, \\ \overline{u} \in V_r^\mathcal{H}}} \frac{\epsilon_{\overline{e}u}}{|\sigma_{\overline{u}}|} x_{\overline{u}u}
\end{aligned}
\tag{4}
$$

where the dual variables $\theta, \alpha, \beta, \gamma, \delta, \epsilon$ correspond to respectively inequalities (3b), (3c) (3d), (3e), (3f), (3g), and $v$ corresponds to the optimal cost of a problem. When no column with negative reduced costs is found, the value of the linear relaxation of $(VOPF_\sigma)$ has been solved to optimality.

*Lagrangian bound.* The full convergence of the Column Generation algorithm is typically computationally expensive, but a *Lagrangian Bound* proposes a valid lower bound for the problem and can be easily computed here at any iteration of the algorithm:

$$
LGB = v(RMP) + \sum_{i \in \{1,\dots,k_r\}} v(PP_\mathcal{H}) \tag{5}
$$

*Stabilization.* We observed oscillations in the dual variable values during our tests, in particular for the dual variables $\alpha$ and $\beta$, which correspond to constraints (3c) and (3d). These duals usually have large values, leading to an important impact in the reduced costs. However, since they are equalities covered with, in the early stage of the convergence, a few columns, the dual costs are very unstable, which leads to the production of weak columns. To mitigate these oscillations, we adopt a standard stabilization technique: dual variable smoothing [22]. At each iteration, instead of using the current dual variables from the restricted master problem, we use a convex combination of the previous stabilized duals and the current ones. The parameter $\alpha \in [0, 1]$ controls the smoothing strength. At a given iteration of the column generation, the stabilized counterpart $\hat{\omega}$ of the dual variable $\omega$ is:

$$
\hat{\omega} = \alpha \hat{\omega}' + (1 - \alpha)\omega
$$

where $\hat{\omega}'$ is the stabilized variable at the previous iteration. In practice, we use $\alpha = 0.9$, which showed good results in [22] and in our early experiments.

*Pricer solver.* To solve the pricer subproblems, we use an MIP solver and the Flow Formulation for all decompositions. In some cases, the subproblems can be solved in polynomial-time: edge subgraphs with the shortest path algorithm from [20], and star subgraphs with the integer flow algorithm from [24]. For the other decompositions, which have NP-complete subproblems, as shown in [4], efficient heuristics can be designed. These pricer improvements would significantly accelerate the Column Generation algorithm of each decomposition. However, the goal of this study is to evaluate the polyhedral strength of each decomposition, rather than raw speed. Consequently, we opt for a common solver, rather than specialized prices, such that decompositions are on equal terms.

# 5 Experimental study

## 5.1 Instances

We consider topologies from SNDlib [21] and Topology Zoo [16], which correspond to real networks. The topologies are described in Table 1. The virtual networks have different sizes and densities. We use uniform demands on every virtual node and edge to limit the amount of randomness used in the generation of the instances. The substrate networks have roughly the same sizes (30-40 nodes), but different densities. All substrate nodes have a unit capacity, while the edge capacity is randomly generated. The cost is based on the node and edge connectivity and capacity, as is done in practice.

| Virtual Networks | | | | Substrate Networks | | | |
|---|---|---|---|---|---|---|---|
| **Name** | $\|V_r\|$ | $\|E_r\|$ | **Origin** | **Name** | $\|V_s\|$ | $\|E_s\|$ | **Origin** |
| polska | 12 | 18 | SNDlib | INS | 33 | 41 | TopologyZoo |
| nobel-germany | 17 | 26 | SND-Lib | Xspedius | 34 | 108 | TopologyZoo |
| HiberniaUs | 21 | 29 | TopologyZoo | india35 | 35 | 80 | SNDlib |
| Abovenet | 23 | 31 | SNDlib | BtNorthAmerica | 36 | 76 | TopologyZoo |
| France | 25 | 45 | SNDlib | Valleynet | 39 | 51 | TopologyZoo |
| Integra | 27 | 36 | TopologyZoo | giul39 | 39 | 86 | SNDlib |

**Table 1: List of virtual and substrate network instances**

Note that all instances are relatively small and can be solved exactly in less than 1 hour using a commercial solver.

## 5.2 Implementation

The Column Generation is implemented in Julia [7], using the modeling tool JuMP [12]. The Restricted Master Problem and the pricers are solved with the commercial solver IBM ILOG CPLEX 22.1. The algorithm stops when the gap between the Lagrangian bound $LGB$ and $v(RMP)$ reaches 1% or less, or a 3600 seconds time limit. Computational experiments are conducted on a machine equipped with an Intel(®) Xeon® E5-2650 v3 CPU running at 2.30 GHz.

## 5.3 Results

The aggregated results are reported in Table 2. The first column indicates the decomposition strategy and the second the average of the Lagrangian bounds (column LGB) obtained, over all virtual networks and all substrate networks. The remaining columns show the average of the computing times (in seconds), the number of iterations of the column generation, the number of columns generated, the gaps (i.e., improvement) between LGB and the linear relaxation of the flow formulation (GapFF), and between LGB and the optimal value (GapOpt).

| Decomposition | LGB | Time | #iters | #cols | GapFF | GapOpt |
|---|---|---|---|---|---|---|
| edge | 67.3 | 150.8 | 126 | 4097 | 0.00 | 0.28 |
| star strict | 76.9 | 234.8 | 186 | 970 | 0.11 | 0.19 |
| star overlap. | 80.5 | 425.3 | 232 | 1524 | 0.15 | 0.15 |
| path strict | 71.5 | 147.6 | 157 | 908 | 0.05 | 0.24 |
| path overlap. | 77.8 | 677.2 | 434 | 3093 | 0.12 | 0.18 |
| cycle strict | 72.0 | 79.4 | 86 | 1134 | 0.06 | 0.23 |
| cycle overlap. | 74.8 | 119.4 | 93 | 1002 | 0.09 | 0.2 |
| auto. overlap. ($k_r$=3) | 82.2 | 268.4 | 99 | 297 | 0.16 | 0.14 |
| auto. overlap. ($k_r$=3, $k_o$=3) | 85.0 | 436.2 | 98 | 292 | 0.19 | 0.11 |
| auto. overlap. ($k_r$=2) | 85.7 | 1258.0 | 90 | 179 | 0.2 | 0.1 |
| auto. overlap. ($k_r$=2, $k_o$=3) | 88.0 | 1341.7 | 94 | 188 | 0.22 | 0.07 |

**Table 2: Overall results**

The automatic decompositions clearly provide the strongest lower bounds, especially with smaller $k_r$ (larger virtual subgraphs). The star decompositions are the second best, indicating that this decomposition has a lot of potential for a future Branch&Price, since the pricing problems could be solved in polynomial-time using the algorithm of [24]. Next, cycle and path decompositions provide weaker but reasonable bounds. The path decomposition, in particular, offers weaker gain in bounds compared to the number of iterations and columns generated. This shows that star virtual subgraphs induce more bound improvements than path subgraphs.

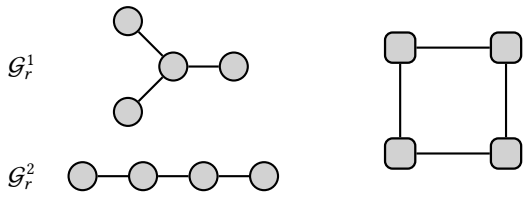


**Figure 8: Simple star and path virtual networks**

We now illustrate why this holds. Figure 8 shows 4-nodes virtual star and path (left) and a simple substrate network (right). Any mapping of the virtual star on the substrate network requires at least one 2-edge path for routing one of the virtual edges, whereas mappings of the virtual path can contain only one-edge routing paths. This is because the maximum degree of the substrate network is 2, whereas the center of the virtual star has a degree of 3. Thus, despite having the same number of edges and nodes, the mappings of the virtual star are more expensive than the path ones (as will be the submappings in the decompositions).

Finally, the edge decomposition, which is equivalent to the Path Formulation proposed in [20, 26], offers almost no improvement over the Flow Formulation.

*Column Generation convergence.* The convergence time of the Column Generation is long, in some cases longer than solving the actual problem, but it could be greatly reduced using dedicated pricers, as discussed above. The most demanding pricers are those for the automatic decompositions, where the virtual subgraphs are larger, especially for $k_r = 2$. However, the number of columns and iterations is surprisingly much lower for these decompositions. Thus, if the pricers are improved, these decompositions, which yield the best bounds, could become very competitive.

*Impact of overlapping nodes.* The use of overlapping nodes consistently leads significant improvements in the lower bounds. However, our results reveal two distinct convergence behaviors based on the degree of overlap. For the edge, star, and path overlapping decompositions, which typically have many overlapping nodes, the table shows a marked increase in the number of iterations, the total number of columns generated, and the solving times. Conversely, when there are few overlapping nodes (automatic or cycle overlapping decompositions), the bound is still improved, but the solving time increases only marginally, and the number of iterations and columns can, surprisingly, decrease. These findings indicate that using a few well-connected overlapping nodes is the most efficient strategy. In particular, our proposed algorithm for augmenting METIS partitions with a few overlapping nodes is highly effective.

*Network topologies.* The detailed results show that the star and automatic decompositions perform especially well (higher GapFF) on denser virtual networks, such as France or Nobelgermany (larger stars, denser subgraphs), compared to edge and path decompositions. The size of the network is also an important factor, as large virtual networks require more columns and more time. Yet, we observed that the gaps, for the different decompositions, depend more on the density than on the size of the virtual network. Finally, all decompositions become stronger on denser substrate networks, but the number of columns and solving times increase noticeably.

*Substrate capacities.* In the previous experiments, all substrate nodes had a capacity of one. We ran additional experiments in which some nodes had zero capacity. We observed that, for all decompositions, GapOpt remains stable, but GapFF increases as more substrate nodes have zero capacity.

## 6 Conclusion

In this paper, we introduced a new extended formulation for the Virtual Network Embedding (VNE) problem, based on overlapping partitions of the virtual network. This original decomposition approach offers a flexible framework that generalizes previous approaches. We investigated several decomposition strategies based on fundamental topologies and developed a Column Generation scheme to evaluate the resulting linear relaxations. Experiments showed that the star and the automatic decompositions are the most effective, yielding significantly tighter lower bounds than the edge decomposition proposed in the literature [20, 26], or the path decomposition, giving new polyhedral insights for VNE. We also established that adding a few overlapping nodes provides further bound gains with surprisingly little computational overhead. Future research will focus on refining the partitioning heuristics and accelerating Column Generation convergence with dedicated pricers. Ultimately, these advancements provide the foundations for a Branch&Price capable of solving large-scale VNE instances to optimality.

## References


[1] NGMN Alliance. 2016. Description of network slicing concept. *NGMN 5G P* 1, 1 (2016), 1–11.

[2] E. Amaldi, S. Coniglio, A. Koster, and M. Tieves. 2016. On the computational complexity of the virtual network embedding problem. *Electronic Notes in Discrete Mathematics* 52 (2016), 213–220.

[3] T. Anderson, L. Peterson, S. Shenker, and J. Turner. 2005. Overcoming the Internet impasse through virtualization. *Computer* 38, 4 (2005), 34–41.

[4] A. Benhamiche, P. Fouilhoux, L. Létocart, N. Perrot, and A. Schneider. 2025. Complexity of the Virtual Network Embedding with uniform demands. *arXiv preprint arXiv:2501.10154* (2025).

[5] A. Benhamiche, P. Fouilhoux, L. Létocart, N. Perrot, and A. Schneider. 2025. Using integer programming to embed large virtual networks. In *2025 11th International Conference on Control, Decision and Information Technologies (CoDIT)*, Vol. 1. IEEE, 2768–2773.

[6] A. Benhamiche, P. Fouilhoux, L. Létocart, N. Perrot, and A. Schneider. 2026. On the Virtual Network Embedding polytope. *arXiv preprint arXiv:2601.11419* (2026).

[7] J. Bezanson, A. Edelman, S. Karpinski, and V. Shah. 2017. Julia: A fresh approach to numerical computing. *SIAM review* 59, 1 (2017), 65–98.

[8] Alberto Caprara, Fabio Furini, Enrico Malaguti, and Emiliano Traversi. 2016. Solving the temporal knapsack problem via recursive Dantzig–Wolfe reformulation. *Inform. Process. Lett.* 116, 5 (2016), 379–386.

[9] Mosharaf Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. 2011. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on networking* 20, 1 (2011), 206–219.

[10] J.V. Clausen, R. Lusby, and S. Ropke. 2022. Consistency cuts for Dantzig-Wolfe reformulations. *Operations Research* 70, 5 (2022), 2883–2905.

[11] J. Desrosiers and M. Lübbecke. 2005. A primer in column generation. In *Column generation*. Springer, 1–32.

[12] I. Dunning, J. Huchette, and M. Lubin. 2017. JuMP: A modeling language for mathematical optimization. *SIAM review* 59, 2 (2017), 295–320.

[13] A. Fischer, J. Botero, M. Beck, H. De Meer, and X. Hesselbach. 2013. Virtual network embedding: A survey. *IEEE Communications Surveys & Tutorials* 15, 4 (2013), 1888–1906.

[14] M. Guignard and S. Kim. 1987. Lagrangean decomposition: A model yielding stronger Lagrangean bounds. *Mathematical programming* 39, 2 (1987), 215–228.

[15] G. Karypis and V. Kumar. 1997. METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. (1997).

[16] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan. 2011. The internet topology zoo. *IEEE Journal on Selected Areas in Communications* 29, 9 (2011), 1765–1775.

[17] L. Létocart, A. Nagih, and N. Touati-Moungla. 2012. Dantzig-Wolfe and Lagrangian decompositions in integer linear programming. *International Journal of Mathematics in Operational Research* 4, 3 (2012), 247–262.

[18] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha. 2013. Optimal virtual network embedding: Node-link formulation. *IEEE Transactions on Network and Service Management* 10, 4 (2013), 356–368.

[19] R. Mijumbi, J. Serrat, J. Gorricho, and R. Boutaba. 2015. A path generation approach to embedding of virtual networks. *IEEE Transactions on Network and Service Management* 12, 3 (2015), 334–348.

[20] LFS. Moura, LP. Gaspary, and LS. Buriol. 2018. A branch-and-price algorithm for the single-path virtual network embedding problem. *Networks* 71, 3 (2018), 188–208.

[21] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski. 2010. SNDlib 1.0—Survivable network design library. *Networks: An International Journal* 55, 3 (2010), 276–286.

[22] A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck. 2013. In-out separation and column generation stabilization by dual price smoothing. In *International Symposium on Experimental Algorithms*. Springer, 354–365.

[23] C. Raack, A. Koster, S. Orlowski, and R. Wessäly. 2011. On cut-based inequalities for capacitated network design polyhedra. *Networks* 57, 2 (2011), 141–156.

[24] M. Rost, C. Fuerst, and S. Schmid. 2015. Beyond the stars: Revisiting virtual cluster embeddings. *ACM SIGCOMM Computer Communication Review* 45, 3 (2015), 12–18.

[25] M. Rost and S. Schmid. 2020. On the hardness and inapproximability of virtual network embeddings. *IEEE/ACM transactions on networking* 28, 2 (2020), 791–803.

[26] Y. Wang, Q. Hu, and X. Cao. 2016. A branch-and-price framework for optimal virtual network embedding. *Computer Networks* 94 (2016), 318–326.

[27] D. Warrier, W. Wilhelm, J. Warren, and I. Hicks. 2005. A branch-and-price approach for the maximum weight independent set problem. *Networks: An International Journal* 46, 4 (2005), 198–209.