

题 目 面向动态业务需求的

VNF 转发图扩展算法

## 专 业 信息安全

学 号 171210202

学 生 柴绪清

指 导 教 师 李宁

答 辩 日 期 2021 年 5 月 25 日

# 摘 要

传统网络架构在提供网络服务时，需要部署相应的专用物理设备（防火墙、加密设备等）。网络功能与物理设备是紧耦合的，每个网络功能都由特定的硬件平台实现，灵活性和扩展性较差，网络功能的升级和维护都受制于设备提供商。网络功能虚拟化 (Network Function Virtualization,NFV) 技术试图通过重构现有网络架构来解决上述问题。网络功能虚拟化将网络功能的软件实现与物理硬件解耦。虚拟网络功能(Virtualized Network Functions,VNFs) 可以与其他VNFs 以及物理网络功能链接，实现网络业务。租户在物理基础架构中可以部署各种服务和网络功能，而服务提供商具有动态的VNF-FG 建立、更新和扩展的需求。因此， 基础设施提供商不仅需要删除、替换、插入和添加 VNF 实例，而且还需要扩展已经实例化的服务功能链，而又不干扰先前部署的服务功能链。VNF-FG 的扩展必须在不中断最初部署的服务实例的情况下实现，必须无缝衔接到应用程序和服务中去。

本文首先提出了一种基于特征分解的启发式算法，用于求解虚拟网络功能转发图扩展问题。该算法在提供接近最优解的扩展方案的同时, 降低了算法复杂度。之后通过线性规划求解器求解虚拟网络功能转发图扩展问题，该算法作为启发式算法最终结果比较的参考。最后对两种算法的计算结果在扩展成功率和目标函数两个维度进行了比较。

关键词：网络功能虚拟化；虚拟网络功能转发图；特征分解；启发式算法

# Abstract

Traditional network architectures need to deploy dedicated physical equipment (ﬁre- walls, encryption equipment, etc.) when providing network services. The network func- tion and the physical device are tightly coupled. Each network function is implemented by a speciﬁc hardware platform, which has poor ﬂexibility and scalability. The upgrade and maintenance of network functions are all restricted by the equipment provider. The network function virtualization technology attempts to solve the above-mentioned prob- lems by reconstructing the existing network architecture. Network function virtualization decouples the software implementation of network functions from physical hardware. The virtual network function can be linked with other VNFs and physical network functions to realize network services. Tenants can deploy various services and network functions in the physical infrastructure, while service providers have dynamic VNF-FG establishment, update, and expansion requirements. Therefore, infrastructure providers not only need to delete, replace, insert, and add VNF instances, but also need to extend the service function chain that has been instantiated without interfering with the previously deployed service function chain. The expansion of VNF-FG must be realized without interrupting the initially deployed service instance, and must be seamlessly connected to applications and services.

This paper ﬁrst proposes a heuristic algorithm based on eigendecomposition to solve

the problem of virtualized network function forwarding graph expansion. This algorithm provides an expansion scheme close to the optimal solution while reducing the complexity of the algorithm. After that, the linear programming solver is used to solve the virtual network function forwarding graph expansion problem, and this algorithm is used as a reference for comparing the ﬁnal results of heuristic algorithms. Finally, the calculation results of the two algorithms are compared in the expansion success rate and objective function.

**Keywords:** NFV, VNF-FG, Eigendecomposition,Heuristic algorithm

# 目 录

[摘 要](#_bookmark0) [I](#_bookmark0)

[Abstract](#_bookmark1) [II](#_bookmark1)

[第 1 章 绪论](#_bookmark2) [1](#_bookmark2)

* 1. [课题背景](#_bookmark3) [1](#_bookmark3)
  2. [研究目的和意义](#_bookmark4) [2](#_bookmark4)
  3. [国内外研究现状](#_bookmark5) [2](#_bookmark5)
     1. [国内研究现状](#_bookmark6) [2](#_bookmark6)
     2. [国外研究现状](#_bookmark7) [3](#_bookmark7)
  4. [本文的主要研究内容](#_bookmark8) [3](#_bookmark8)

[第 2 章 虚拟网络功能转发图相关概念](#_bookmark9) [5](#_bookmark9)

* 1. [网络功能虚拟化相关概念](#_bookmark10) [5](#_bookmark10)
     1. [软件定义网络](#_bookmark11) [5](#_bookmark11)
     2. [网络功能虚拟化](#_bookmark13) [6](#_bookmark13)
  2. [虚拟网络功能转发图相关概念](#_bookmark14) [6](#_bookmark14)
     1. [服务功能链](#_bookmark15) [6](#_bookmark15)
     2. [虚拟网络功能转发图](#_bookmark16) [7](#_bookmark16)
  3. [虚拟网络功能转发图放置和扩展问题概述](#_bookmark17) [7](#_bookmark17)
  4. [本章小结](#_bookmark19) [8](#_bookmark19)

[第 3 章 VNF-FG 扩展问题的整数线性规划模型建模](#_bookmark20) [9](#_bookmark20)

* 1. [VNF-FG 扩展问题分析](#_bookmark21) [9](#_bookmark21)
     1. [物理基础设施图](#_bookmark22) [9](#_bookmark22)
     2. [虚拟网络功能转发图](#_bookmark23) [9](#_bookmark23)
     3. [虚拟网络功能转发扩展图](#_bookmark26) [10](#_bookmark26)
  2. [整数线性规划模型](#_bookmark28) [10](#_bookmark28)
     1. [相关符号变量](#_bookmark29) [10](#_bookmark29)
     2. [约束条件](#_bookmark32) [12](#_bookmark32)
     3. [目标函数](#_bookmark33) [14](#_bookmark33)
  3. [本章小结](#_bookmark35) [14](#_bookmark35)

[第 4 章 基于邻接矩阵特征分解的启发式算法研究](#_bookmark36) [15](#_bookmark36)

* 1. [Umeyama 特征分解算法与存在不足](#_bookmark37) [15](#_bookmark37)
     1. [Umeyama 特征分解算法](#_bookmark38) [15](#_bookmark38)
     2. [现有特征分解算法存在的问题](#_bookmark39) [17](#_bookmark39)
  2. [自定义特征分解算法设计](#_bookmark40) [17](#_bookmark40)
     1. [物理图和请求图的邻接矩阵设计](#_bookmark41) [17](#_bookmark41)
     2. [邻接矩阵之间距离度量的设计](#_bookmark42) [18](#_bookmark42)
     3. [特征分解计算和最优映射函数设计](#_bookmark43) [18](#_bookmark43)
  3. [基于特征分解的 VNF-FG 放置算法](#_bookmark44) [19](#_bookmark44)
     1. [VNF-FG 放置问题概述](#_bookmark45) [19](#_bookmark45)
     2. [物理图邻接矩阵变换](#_bookmark46) [19](#_bookmark46)
     3. [VNF-FG 放置算法的详细设计](#_bookmark48) [20](#_bookmark48)
     4. [VNF-FG 匹配算法的详细设计](#_bookmark51) [21](#_bookmark51)
  4. [基于特征分解的 VNF-FG 扩展算法](#_bookmark53) [23](#_bookmark53)
     1. [物理图邻接矩阵的设计](#_bookmark54) [23](#_bookmark54)
     2. [VNF-FG 扩展算法的设计](#_bookmark55) [23](#_bookmark55)
  5. [算法复杂度分析](#_bookmark56) [24](#_bookmark56)
     1. [算法时间复杂度分析](#_bookmark57) [24](#_bookmark57)
     2. [算法空间复杂度分析](#_bookmark58) [24](#_bookmark58)
  6. [本章小结](#_bookmark59) [24](#_bookmark59)

[第 5 章 仿真实验的详细设计与实现](#_bookmark60) [26](#_bookmark60)

* 1. [仿真实验整体概述](#_bookmark61) [26](#_bookmark61)
  2. [仿真环境](#_bookmark62) [26](#_bookmark62)
     1. [网络拓扑生成工具](#_bookmark63) [26](#_bookmark63)
     2. [VNF-FG 请求的时间分布设计](#_bookmark64) [28](#_bookmark64)
  3. [评价指标](#_bookmark68) [29](#_bookmark68)
  4. [仿真结果](#_bookmark69) [29](#_bookmark69)
     1. [扩展成功概率的对比与分析](#_bookmark70) [29](#_bookmark70)
     2. [目标函数的对比与分析](#_bookmark71) [30](#_bookmark71)
  5. [本章小结](#_bookmark72) [30](#_bookmark72)

[结 论](#_bookmark74) [32](#_bookmark74)

[参考文献](#_bookmark75) [33](#_bookmark75)

[哈尔滨工业大学本科毕业设计（论文）原创性声明](#_bookmark98) [35](#_bookmark98)

[致 谢](#_bookmark99) [36](#_bookmark99)

# 第 1 章 绪论

## 课题背景

当今社会互联网的快速发展给日常生活带来了很多便利，人们越来越多地使用互联网开展工作、学习、会议等活动。同时互联网行业的飞速发展也对网络的架构和性能有了更多的要求。未来的通信网络要求更好的灵活性、扩展性和易用性。日益增长的带宽需求使得在部署网络时，需要增加更多的物理专用设备。传统通信网络规模的不断膨胀，使得网络的运维成本和难度不断增加。长期的“打补丁”方式的互联网修补过程导致了互联网的僵化，采用修补方式很难从根本上解决上述问题，因此需要一种新的网络体系来解决现有网络中存在的问题。2008 年，美国斯坦福大学提出了 OpenFlow 技术[[1](#_bookmark76)]，并以此技术为基础，逐步拓展为软件定义网络（SDN）体系结构，将网络控制功能从网络转发功能中解耦出来。相比于传统网络，SDN 具有更好的灵活性、开放性、高效性等优势。

传统的网络结构在提供网络服务时，需要部署相应的专用物理设备（防火墙、加密设备等）。网络功能与物理设备是紧耦合的，每个网络功能都由特定的硬件平台实现，灵活性和扩展性较差，网络功能的升级和维护都受制于设备提供商。网络功能虚拟化（Network Function Virtualization,NFV）技术试图通过重构现有网络架构来解决上述问题。NFV 技术将网络功能和物理设备实现了解耦。网络功能虚拟化的主要思想是将网络功能的软件实现部分从原来的专有物理设备实现转变为在通用物理设备上的虚拟平台[[2](#_bookmark77)] 实现。运行在通用标准的服务器、存储设备、交换机等硬件设备上的虚拟化平台使用软件的方式实现传统网络专用硬件设备的功能[[3](#_bookmark78)]。虚拟网络功能可以根据租户的需求进行动态的实例化，而不需要增减具体的物理设备。借助NFV，服务提供商可以在标准硬件

（而非专用硬件）上运行网络功能。另外，由于网络功能已虚拟化，因此可以在单个服务器上运行多个功能。这就意味着所需的物理硬件得以减少，故而可以进行资源整合，以降低物理空间占用、功耗和总体成本。NFV 使提供商可以灵活地在不同服务器上运行VNF，或在需求发生变化时根据需要移动 VNF。这种灵活性可以加快服务提供商交付服务和应用的速度。

## 研究目的和意义

服务功能链[[4](#_bookmark79)]（Service Function Chain,SFC）是一组有序的虚拟网络功能集合，用户流量按照指定的策略一次通过多个虚拟网络功能。服务功能链可以实现用户流量按固定的顺序依次被虚拟网络功能处理。虚拟网络功能（Virtual Net- work Function，VNF）是网络功能在通用物理设备上的软件实现。服务功能链的概念可以用于传统的网络架构中，在传统的网络场景中，用户流量通常需要经过若干个网络设备，比如防火墙、入侵检测系统、负载均衡器等[[5](#_bookmark80)]。在网络功能虚拟化架构下，可以将描述虚拟网络功能连接关系的网络拓扑结构称为虚拟网络功能转发图（VNF-FG）。VNF-FG 和SFC 的关系可以简单理解为，VNF-FG用于描述多个 SFC 合并之后的网络拓扑结构。

在规划、部署SFC 应用的过程中，仍然有着很多重要的问题等待解决，包括服务功能链的映射会受限于端系统所处的物理位置以及服务提供商面临着来自租户的动态和可变的服务需求。因此不仅需要将设计的 VNF-FG 中的VNF 映射到物理资源，而且还需要在不中断最初部署和活动的服务实例的情况下实现已经部署的VNF-FG 的扩展。租户在物理和虚拟基础架构中可以部署各种服务和网络功能，而服务提供商具有动态的 SFC 建立、更新和扩展要求。因此， 基础设施提供商不仅需要删除、替换、插入和添加 VNF 实例，而且还需要扩展已经实例化的服务功能链，而又不干扰先前部署的服务功能链。在这种背景下，基础设施提供商（或更一般的 NFV 体系结构设计者和参与者）面临着来自租户的动态网络基础设施需求。服务提供商不仅需要控制、分类和引导其专用切片中的用户和应用程序流量，而且还希望通过添加新的转发路径、新的服务功能链、新的服务和虚拟化网络功能来扩展其已获取和可操作的网络切片[[6](#_bookmark81)]。VNF-FG 的扩展必须在不中断最初部署的服务实例的情况下实现，必须无缝衔接到应用程序和服务中去。

## 国内外研究现状

### 国内研究现状

2016 年，Mechtri 等人提出基于二分图构造的贪婪算法[[7](#_bookmark82)]，分两步解决了VNF-FG 放置和链接问题：首先将VNF 映射到物理主机上，然后控制主机之间的VNF 流量。该算法作为次优解决方案，往往无法得到最优解。2017 年，Rankothge等人提出了一种基于遗传算法的 VNFs 资源分配算法[[8](#_bookmark83)]，用于 VNFs 的初始布

局和 VNFs 的扩展。对于在文献中提出的现有策略的扩展，该算法从当前状态开始，搜索需要扩展的 VNF 的资源进行重新分配。因此，某些 VNF 会更改其初始位置，并且该算法尝试最小化服务器和链接配置更改的次数（以最大程度地减少对现有流量的干扰）。2020 年，Houidi 等人首先将 VNF 扩展问题建模为整数线性规划模型，并提出基于Steiner Tree 搜索算法[[6](#_bookmark81)] 求解该问题，最后对复杂性和性能进行了比较。

还有一些作者解决了 VNF 的扩展和迁移问题，以确保动态 VNF 放置可以适应不断增长的需求和流量负载。2016 年，Ayoubi 等人提出了一种用于弹性服务的可用性感知的嵌入与配置框架[[9](#_bookmark84)]。该框架包括两个主要模块，JENA 子系统用于执行虚拟网络（VN）嵌入，主要是基于托管虚拟服务器的物理服务器的可用性来提供足够的可用性保证。第二个模块ARES 是一个重新配置组件，可在扩展托管服务时调整其嵌入（通过迁移VN 或添加备份节点）。2017 年，Houidi 提出了整数线性规划（ILP）和贪婪启发式方法[[10](#_bookmark85)] 来解决网络功能的迁移和扩展。在扩展和迁移 VNF 实例时，可以对成本和性能进行折衷考虑。2018 年，Sevil 等人提出了两种算法来适应服务需求的变化：混合整数线性规划（MILP）和自定义的构造启发式算法[[11](#_bookmark86)]。

### 国外研究现状

2015 年，Chen 等人使用模拟退火算法[[12](#_bookmark87)]，同时只考虑一种类型的VNF 来简化整个问题，从而更快地找到了解决方案。2016 年，Li 等人为NFV 建立了一个实时资源供应系统[[13](#_bookmark88)]，称为 NFV-RT。该系统将时序分析与多种技术集成在一起，例如服务链合并和带舍入的整数线性规划。该系统通过复制完整的 VNF 链实例、重新组合链和迁移流量来实现扩展。2017 年，Liu 等人在动态情况下尝试优化新 SFC 的部署和调整现有业务链[[14](#_bookmark89)]，来满足不断变化的用户需求。

## 1.4 本文的主要研究内容

本文正文部分由 5 章节构成，其中各章节主要内容如下：

第一章是绪论，主要阐述了虚拟网络转发图扩展问题在目前研究领域的课题背景和课题研究的意义。

第二章主要介绍了虚拟网络功能转发图和论文中涉及到的有关技术和概

念。

第三章对虚拟网络功能转发图扩展问题建立了整数线性规划模型，同时提

出了扩展算法必须满足的约束条件，用于找到合适的候选物理节点。为了通过

优化求解器计算得出扩展问题的最优解，提出与完整性约束条件相关联的目标函数。

第四章首先介绍了 Umeyama 的特征分解算法，分析了现有的特征分解算法存在的问题。之后在 Umeyama 的特征分解算法的基础上进行了改进，通过对VNF-FG 请求图的邻接矩阵的重新设计以及提出启发式 VNF-FG 扩展算法解决了原有算法的不足。最后对VNF-FG 扩展算法进行了算法时间复杂度和空间复杂度的分析。

第五章详细设计了仿真实验，以扩展成功率和目标函数为评价指标对比了基于特征分解的VNF-FG 扩展算法的次优解和ILP 优化求解器计算出的最优解。

# 第 2 章 虚拟网络功能转发图相关概念

## 网络功能虚拟化相关概念

### 软件定义网络

软件定义网络（SDN）是一种新型网络架构。传统网络的转发行为是受各种网络协议控制的，而 SDN 要求集中式控制、转发和控制分离、可以直接通过应用程序来控制转发行为。SDN 的特点是将网络控制功能和网络转发功能解耦，并且使得网络控制功能可以直接被编程[[15](#_bookmark90)]。它利用 OpenFlow 协议将路由器的控制平面从数据平面中分离，改以软件方式实现，从而使得将分散在各个网络设备上的控制平面进行集中化管理成为可能，该架构可使网络管理员在不更动硬件设备的前提下，以中央控制方式用程序重新规划网络，为控制网络流量提供了新方案，也为核心网络和应用创新提供了良好平台。

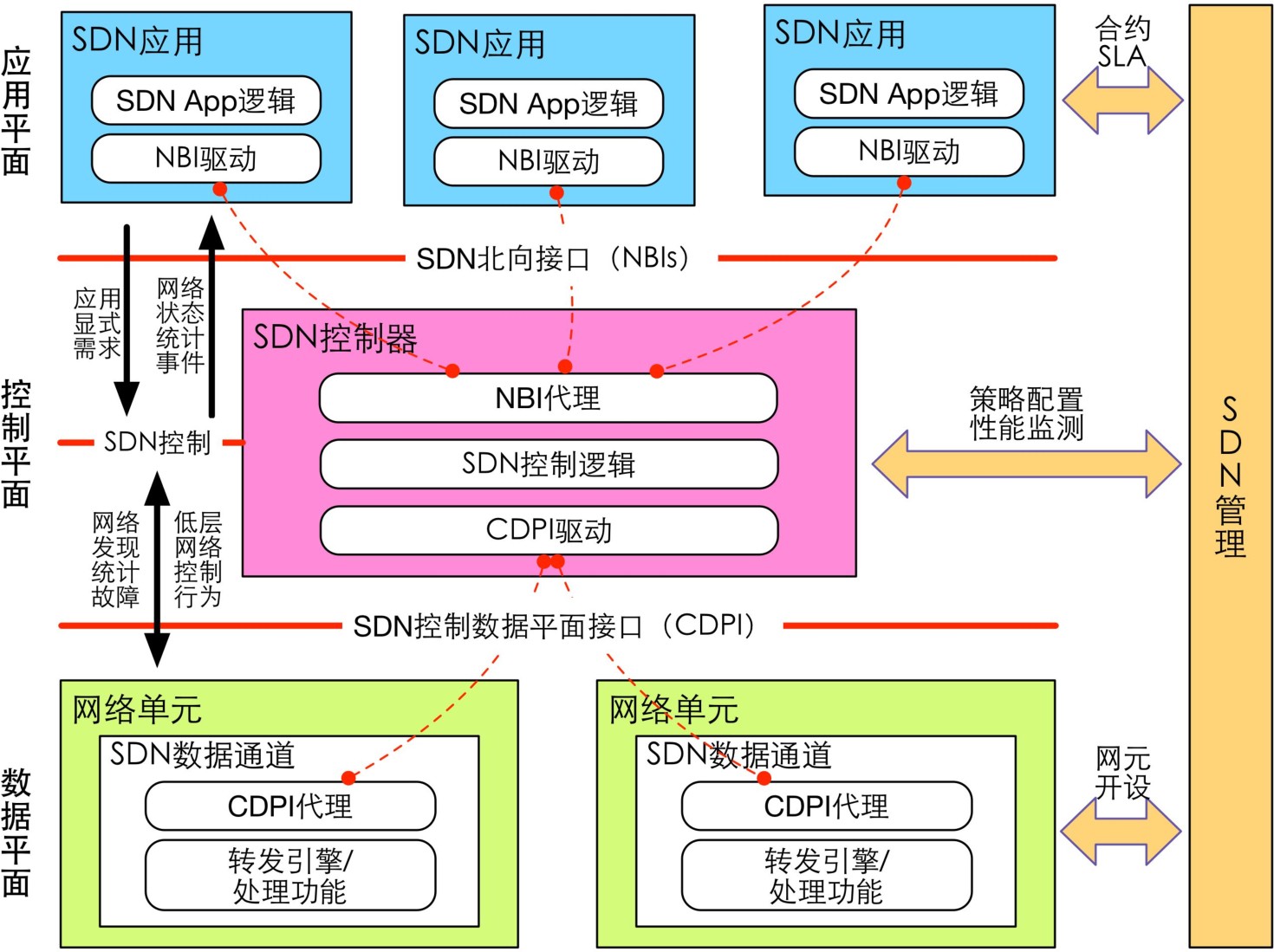
根据 2013 年标准化组织开放网络基金会（ONF）关于软件定义网络的白皮书中提出了软件定义网络的三层体系框架：数据平面、控制平面和应用平面。数据平面和控制平面之间通过 SDN 控制数据平面接口进行通信，SDN 控制数据平面接口主要采用OpenFlow 协议。控制平面与应用平面之间由SDN 北向接口进行通信, 北向接口目前尚未标准化。图 [2-1](#_bookmark12)为软件定义网络架构图。

图 2-1 软件定义网络架构图

### 网络功能虚拟化

在NVF 构架外的OSS（操作支持系统）是用于例如电信运营商（电信公司） 建立和操作服务所需的系统 (设备、软件、机制等）的总称。BSS (业务服务系统) 是用于例如电信运营商（电信公司）用于收取使用费、计费和客户维护所需的信息系统 (设备、软件、机制等) 的总称。

网络功能虚拟化设施[[16](#_bookmark91)]（Network Function Virtualization Infrastructure,NFVI） 是在虚拟化层虚拟化诸如用于计算、存储或网络功能的物理机器 (服务器) 的硬件资源的VNF 虚拟化基础架构，虚拟化层是用于虚拟计算、虚拟存储或虚拟网络的虚拟化硬件资源的管理程序或容器。

网络功能虚拟化管理和编排（Management And Network Orchestration,MANO） 部分[[17](#_bookmark92)] 主要向NFV 平台提供协调控制所有VNF 所需要的功能和操作，使所有

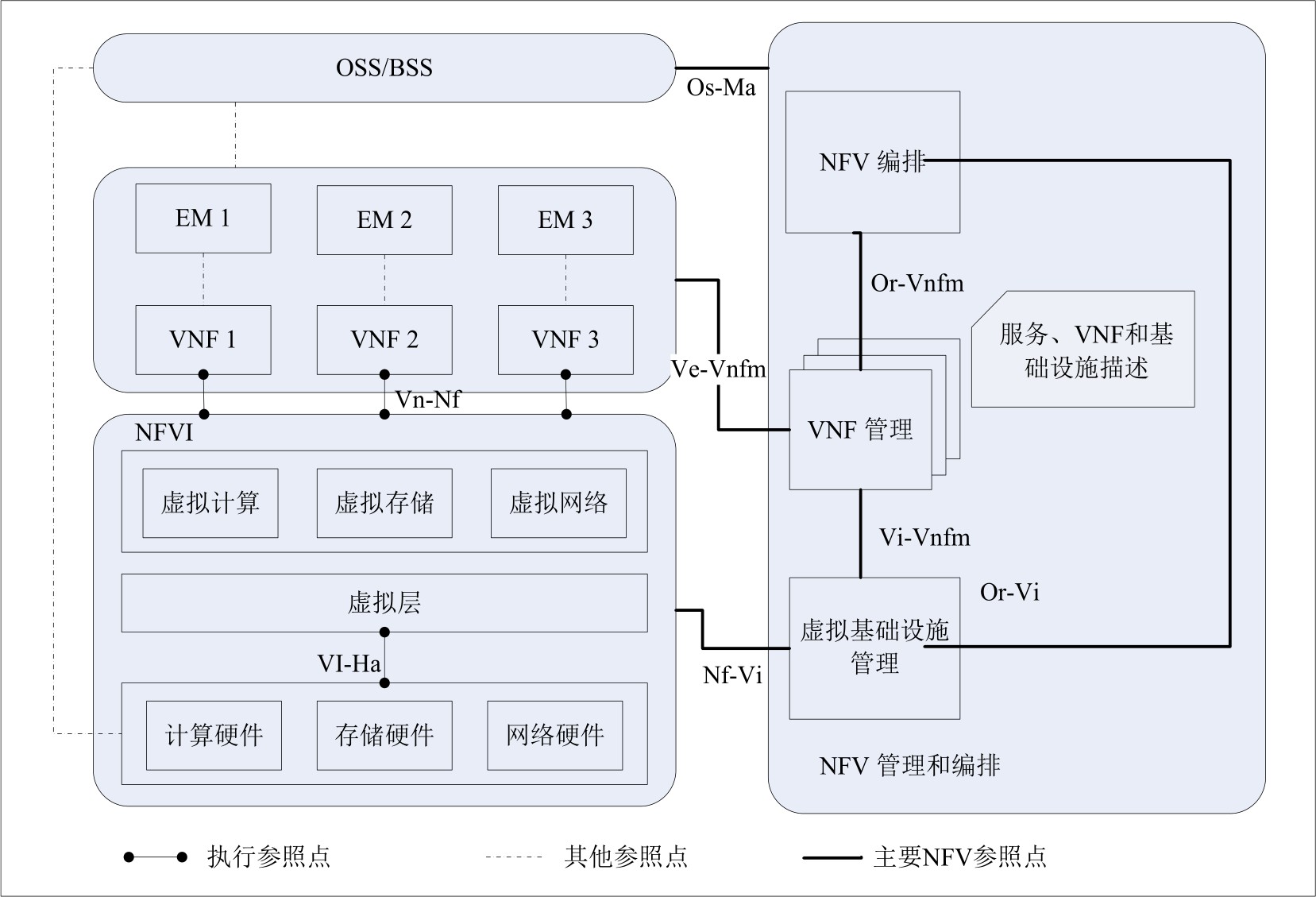
VNF 能够有序运行，MANO 主要包含虚拟化设施管理器、虚拟化网络功能管理器（VNF 管理器）和虚拟化网络功能调度器（VNF 调度器）三个部分[[18](#_bookmark93)]。

图 2-2 网络功能虚拟化技术架构图

## 虚拟网络功能转发图相关概念

### 服务功能链

服务功能链是一个包含服务功能的有序集合，对网络中的 IP 数据包、数据流按照事先规定好的分类规则和策略进行一连串的功能处理，服务功能链不局限于某种特定的网络应用场景。随着虚拟化技术的发展与逐渐成熟，目前的网

络具有按需分配资源、动态加载网络功能等新特点。SDN 技术使数据平面与转发平面分离，使得服务功能链技术更便捷地实现业务流量按既定顺序经过多个不同的业务功能节点，从而完成网络服务。

### 虚拟网络功能转发图

VNF-FG 由多个服务功能链构成，两者都表征了VNF 及其连接关系，对于服务功能链的描述同样适用于 VNF-FG。在网络场景中，数据流通常需要经过若干个网络服务设备，比如防火墙、负载均衡器、IDS 等，最后才到达目的端， 这就是所谓的SFC。SFC 的概念可以用于传统的硬件网络，但目前更多的是用于NFV/SDN 环境下，这里的网络服务通常由软件来实现，也就是 VNF。在 NFV 框架之下，我们将描述 VNF 连接关系的拓扑结构称为 VNF-FG，VNF-FG 提供了VNF 之间的逻辑连接关系。可见，VNF-FG 可以用于描述多个SFC 合并之后的 VNF 连接情况。IETF SFC 工作组使用的是 SFC 这一术语，ETSI NFV ISG 工作组则使用 VNF-FG 来描述这个概念。实际上，SFC 与 VNF-FG 的意思几乎是一样的，在 NFV/SDN 场景下两者可以互换。在本文中，为了方便问题的阐述， 对于单个服务功能链，我们使用 SFC 这一术语进行描述；对于包含多个服务功能链的拓扑图，我们使用VNF-FG 来描述。图 [2-3](#_bookmark18)(a) 和(b) 分别给出了本文语义下的 SFC 和 VNF-FG 的示意图。

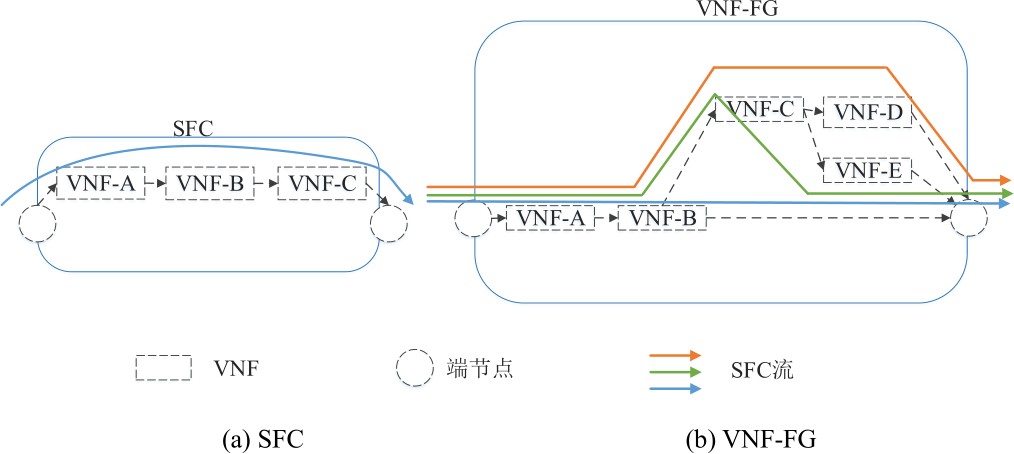


图 2-3 SFC 与 VNF-FG 示意图

## 虚拟网络功能转发图放置和扩展问题概述

VNF-FG 扩展问题是指扩展已经部署的VNF-FGs 的一般性问题，需要实现各种类型扩展，包括向现有的 VNF-FG 中插入新的 VNF 的请求、向现有的服务功能链中添加 VNFs 来扩展网络转发路径等。因此，VNF-FG 扩展不仅需要

满足服务功能链扩展的需求，还需要实现对已经部署的转发图的网络级扩展。VNF-FG 扩展可以由新的请求或 VNF-FG 生命周期管理触发，VNF-FG 生命周期管理可以决定扩展初始图以适应网络和流量负载的变化，例如添加负载平衡器、生成新的 VNFs 以吸收不断增加的负载等等。

在规划、部署SFC 应用的过程中，仍然有着很多重要的问题等待解决，包括服务功能链的映射会受限于端系统所处的物理位置以及服务提供商面临着来自租户的动态和可变的服务需求。因此不仅需要将设计的 VNF-FG 中的VNF 映射到物理资源，而且还需要在不中断最初部署和活动的服务实例的情况下实现已经部署的VNF-FG 的扩展。租户在物理和虚拟基础架构中可以部署各种服务和网络功能，而服务提供商具有动态的 SFC 建立、更新和扩展要求。因此，基础设施提供商不仅需要删除、替换、插入和添加 VNF 实例，而且还需要扩展已经实例化的服务功能链，而又不干扰先前部署的服务功能链。在这种背景下，基础设施提供商（或更一般的 NFV 体系结构设计者和参与者）面临着来自租户的动态网络基础设施需求。服务提供商不仅需要控制、分类和引导其专用切片中的用户和应用程序流量，而且还希望通过添加新的转发路径、新的服务功能链、新的服务和虚拟化网络功能来扩展其已获取和可操作的网络切片。VNF-FG 的扩展必须在不中断最初部署的服务实例的情况下实现，必须无缝衔接到应用程序和服务中去。

## 本章小结

本文主要介绍了网络功能虚拟化和虚拟网络功能转发图涉及的相关技术和概念。软件定义网络主要实现了网络控制功能和网络转发功能解耦，网络功能虚拟化主要实现了物理设备与网络功能解耦，通过软件虚拟化技术在商用硬件(即工业标准服务器、存储和交换机) 上实现了网络功能。服务功能链是指一个数据流通常需要通过多个网络服务设备，例如 IDS/IPS，防火墙，LB 等，最终才能到达目的端。虚拟网络功能转发图描述了多个 SFC 合并之后的 VNF 连接情况。虚拟网络功能转发图扩展问题需要在不中断最初部署和活动的服务实例的情况下实现已经部署的 VNF-FG 的扩展。

# 第 3 章 VNF-FG 扩展问题的整数线性规划模型建模

## VNF-FG 扩展问题分析

### 物理基础设施图

为对 VNF 转发图扩展问题进行建模，可以使用网络拓扑图来表示先前部署的 VNF 转发图、VNF 扩展请求图和 NFV 物理基础设施图。因为已经部署的VNF-FG 与新的 VNF-FG 之间存在互连性，所以在放置新的 VNF-FG 时还需要满足若干约束条件。

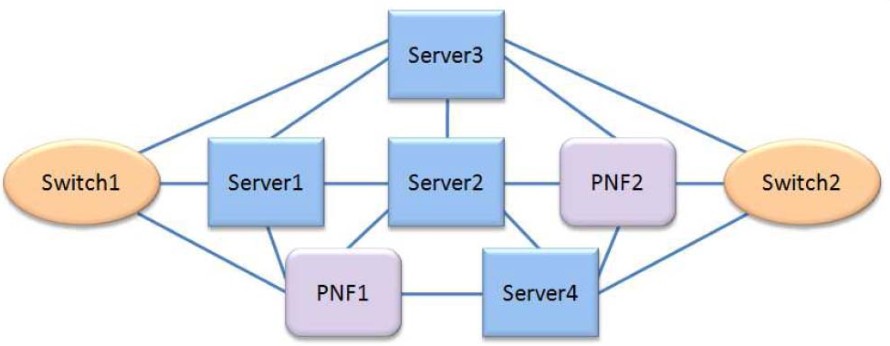
物理基础设施图被建模为一个无向加权图 *𝐺 𝑝* = (*𝑁𝑝, 𝐸 𝑝*)，*𝐸𝑝* 是物理链路的集合，*𝑁𝑝* 是物理节点的集合。对于每个 *𝑁 𝑝* 集合中的物理节点k 都具有两个特征。第一个特征是CPU 处理能力，第二个特征是虚拟网络功能（或者物理网络功能 PNF）类型。对于 *𝐸 𝑝* 集合中的每个物理链路 e 都具有带宽需求的特征。以图 [3-1](#_bookmark24)的NFV 物理基础设施图为例，该物理基础设施图包括两个物理网络功能设备、两个交换机（一个入口交换机和一个出口交换机，作为VNF-FG 中网络转发路径的输入和输出网关）以及一些互相连接的服务器。

图 3-1 NFV 物理基础设施图

### 虚拟网络功能转发图

类似地，客户的 VNF 请求转发图可以建模为 *𝐺 𝑣* = (*𝑁𝑣 , 𝐸𝑣* )，其中 *𝑁**𝑣* 是请求图中虚拟节点集合，*𝐸𝑣* 是虚拟链路集合。对于每个虚拟节点都具有 CPU 处理能力和网络功能类型两个特征。对于每个虚拟链路都具有带宽需求的特征。图 [3-](#_bookmark25)

[2](#_bookmark25)中描述了两个网络转发路径（NFP），转发路径是流量必须经过的有序 VNF 序列。根据 SFC IETF 工作组 [20] 的规定，每一个 VNF 节点都必须关联相应的网

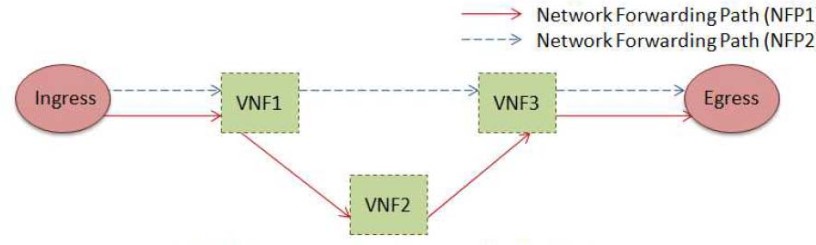
络功能类型（防火墙、NAT、负载均衡器、SSL 等）。VNF 节点只能放置在具有相同类型的服务器或 PNF 中，VNF 转发路径的首节点和尾结点必须放置在交换机上。

图 3-2 虚拟网络功能转发图

为了更好地表示出 VNF 转发图的网络拓扑，可以从 VNF 转发图中推导出网络连接拓扑图（NCT）。网络连接拓扑图表示为 *𝑁𝐶𝑇𝑣* = (*𝑁𝑣 , 𝐸𝑣* )，是一个与

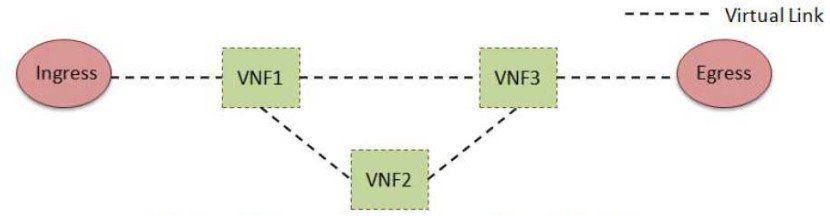
*𝐺 𝑣* 具有完全相同的节点和边集的无向加权图。惟一的区别是，网络连接拓扑图中的虚拟链路的带宽需求，是在 *𝐺𝑉* 中通过该链路的所有VNF 转发路径的带宽之和。图[3-3](#_bookmark27)为图[3-2](#_bookmark25)中的 VNF 请求转发图所对应的网络连接拓扑图。

图 3-3 网络连接拓扑图

### 虚拟网络功能转发扩展图

图 [3-4](#_bookmark30) (a) 描述了之前部署的 VNF-FG。图 [3-4](#_bookmark30) (b) 描述了接受新的 VNF-FG

扩展请求图，新的VNF-FG 扩展请求图相比于之前部署的 VNF-FG 增加了两个

VNF 节点（VNF 4 节点和VNF 5 节点），同时可以看到这两个 VNF 节点与之前部署的VNF-FG 存在着互连性。图 [3-4](#_bookmark30) (c) 描述了VNF 4 节点和VNF 5 节点分别放置在服务器 3 和服务器 4 中的扩展解决方案，可以看出新请求中的节点和链路都被成功放置。

## 整数线性规划模型

### 相关符号变量

为了更好地解决 VNF-FG 扩展问题，我们将 VNF-FG 扩展问题建模为整数

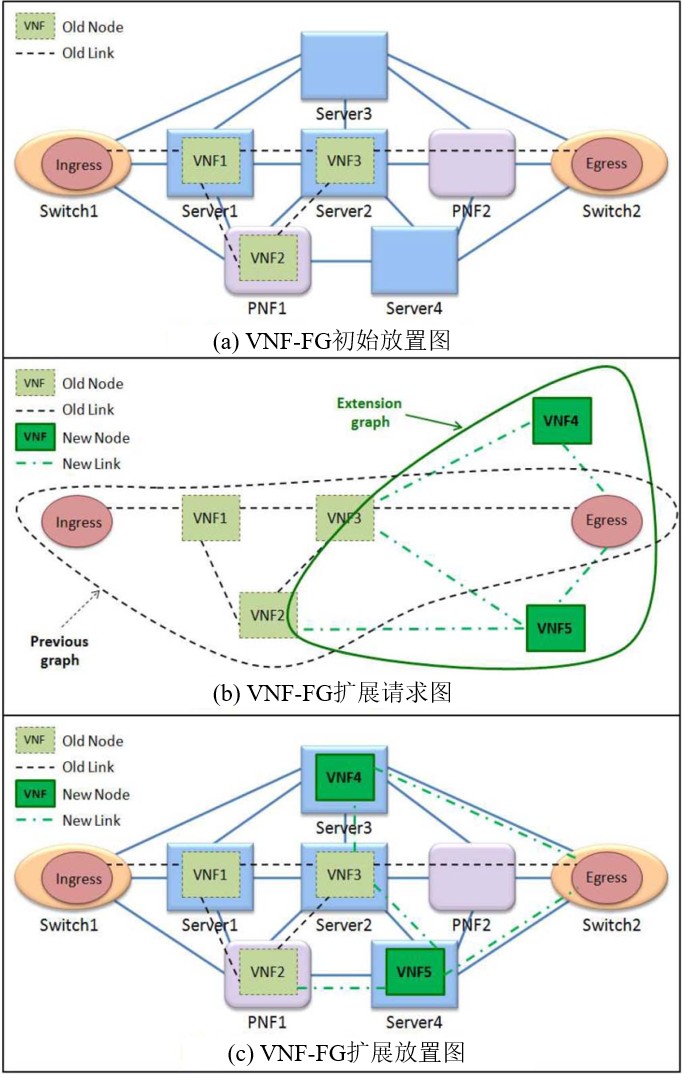


图 3-4 虚拟网络功能转发扩展图

线性规划模型。整数线性规划模型必须包括一些扩展算法必须满足的约束条件， 用于找到合适的候选物理节点。为了通过优化求解器计算得出扩展问题的最优解，需要提出与完整性约束条件相关联的目标函数。表 [3-1](#_bookmark31)是对约束条件和目标函数中涉及变量的说明。

表 3-1 约束条件涉及变量说明表

变量符号 变量描述

*𝐶𝑃𝑈𝑘* 物理节点 k 剩余计算容量

*𝑚𝑖, 𝑚 𝑗* VNF 节点 i 和 j 的初始放置位置

*𝑃𝑘𝑖 ,𝑘 𝑗* 物理节点 *𝑘𝑖* 和 *𝑘 𝑗* 之间的物理路径

C 物理节点集合

P 物理路径集合

*𝑚𝑖𝑛𝐶𝑃𝑈* C 中的最小 CPU 容量

*𝐵𝑊𝑒* 物理链路 e 的剩余链路带宽

*𝛿𝑒𝑝* 表示物理链路 *𝑒* ∈ *𝐸 𝑝* 是否属于路径 *𝑃𝑘𝑖 ,𝑘 𝑗* ∈ P: *𝛿𝑒𝑝* = 1 ⇔ *𝑒* ∈ P

*𝑁 𝑛𝑒𝑤 , 𝐸 𝑛𝑒𝑤*

扩展请求中的 VNF 节点集合，以及它们之间的链路的集合

*𝑣 𝑣*

*𝑁𝑜𝑙𝑑 , 𝐸 𝑜𝑙𝑑*

旧 VNF-FG 请求中的 VNF 节点集合，以及它们之间的链路的集合

*𝑣 𝑣*

*𝑐 𝑝𝑢𝑖* 虚拟节点 i 所需要的计算容量

*𝑏𝑤𝑒* 虚拟节点 i 和 j 之间的链路带宽

*𝑖 𝑗*

*𝑥𝑖𝑘* 此变量为二元变量，表示 VNF 节点 i 是否放置到物理节点 k 上

*𝑦𝑒𝑖 𝑗 , 𝑃𝑘𝑖 ,𝑘 𝑗* 此变量为二元变量，表示虚拟链路 *𝑒𝑖 𝑗* 是否会放置到物理路径 *𝑃𝑘𝑖 ,𝑘 𝑗*

### 约束条件

旧节点映射约束：该约束用来维护旧 VNF 节点的初始映射位置。

*𝑥𝑖𝑚𝑖*

= 1*,* ∀*𝑖* ∈ *𝑁*old

(3-1)

旧链路映射约束：该约束用来维护旧虚拟链路的初始映射位置。

*𝑣*

*𝑦𝑒𝑖 𝑗*

*, 𝑃𝑚𝑖 ,𝑚 𝑗*

= 1*,* ∀*𝑒𝑖 𝑗* ∈ *𝐸*old

(3-2)

新节点映射约束：确保每一个新的 VNF 节点都能映射到一个物理候选节点，同时 VNF 组件不能拆分。

*𝑣*

. *𝑥𝑖𝑘* = 1*,* ∀*𝑖* ∈ *𝑁*new

*𝑣*

(3-3)

*𝑘* ∈C

*𝑥𝑖𝑘*

= 1*,* if the VNF i is hosted on the substrate node k;

0*,* otherwise*.*





(3-4)

CPU 可用容量约束：确保 VNF 节点所放置物理节点的 CPU 可用容量大于等于该 VNF 节点的 CPU 容量。

.*𝑖* ∈ *𝑁*

new

*𝑣𝑒𝑤*

*𝑐 𝑝𝑢𝑖* × *𝑥𝑖𝑘* ≤ *𝐶𝑃𝑈𝑘 ,* ∀*𝑘* ∈ C (3-5)

新链路映射约束：如果虚拟链路的两端节点都是新的 VNF 节点，那么需要设置一个约束确保该链路能够被放置。

*𝑣*

*𝑣*

. . *𝑦𝑒*

*𝑖 𝑗*

*𝑘𝑖 ,𝑘 𝑗*

*𝑘𝑖* ∈C *𝑘 𝑗* ∈C

*, 𝑃*

= 1*,* ∀*𝑒𝑖 𝑗* ∈ *𝐸*new *,* ∀*𝑖, 𝑗* ∈ *𝑁*new

(3-6)

*𝑦𝑒𝑖 𝑗 , 𝑃𝑘𝑖 ,𝑘 𝑗*

 1*,* if the virtual link *𝑒𝑖 𝑗* is mapped to physical path *𝑃𝑘𝑖 ,𝑘 𝑗*

 0*,* otherwise.

=

(3-7)

如果虚拟链路的源节点是旧节点、目的节点是新节点，那么需要满足以下

约束。

. *𝑦𝑒*

*𝑖 𝑗*

*𝑚𝑖 ,𝑘 𝑗*

*𝑘 𝑗* ∈C

*, 𝑃*

= 1*,* ∀*𝑒𝑖 𝑗* ∈ *𝐸*new *,* ∀*𝑖* ∈ *𝑁*old *,* ∀ *𝑗* ∈ *𝑁*new

(3-8)

如果虚拟链路的源节点是新节点、目的节点是旧节点，那么需要满足以下

*𝑣*

*𝑣*

*𝑣*

约束。

. *𝑦𝑒*

*𝑖 𝑗*

*𝑘𝑖 ,𝑚 𝑗*

*𝑘𝑖* ∈C

*, 𝑃*

= 1*,* ∀*𝑒𝑖 𝑗* ∈ *𝐸*new *,* ∀*𝑖* ∈ *𝑁*new *,* ∀ *𝑗* ∈ *𝑁*old

(3-9)

带宽约束：物理路径上的每个物理链路的剩余可用带宽不能小于被放置在

*𝑣*

*𝑣*

*𝑣*

该路径上的所有虚拟链路的带宽需求的总和。

*𝑒𝑖 𝑗*.∈*𝐸*new

*𝑣*

*𝑏𝑤𝑒𝑖 𝑗* × *𝑦𝑒𝑖 𝑗 , 𝑃𝑘𝑖 ,𝑘 𝑗* × *𝛿𝑒𝑝* ≤ *𝐵𝑊𝑒* (3-10)

节点和带宽约束：当 *𝑉 𝑁 𝐹𝑖* 映射到物理候选节点 *𝑘𝑖* 时，对于每个以 *𝑉 𝑁 𝐹𝑖*

开始的虚拟链路 *𝑒𝑖 𝑗* 都必须找到一个物理路径 *𝑃𝑘𝑖 ,𝑘 𝑗* 来放置该虚拟链路.

∀*𝑒* ∈ *𝑃𝑘𝑖 ,𝑘 𝑗 ,* ∀*𝑃𝑘𝑖 ,𝑘 𝑗* ∈ P (3-11)

节点和带宽约束：当 *𝑉 𝑁 𝐹𝑗* 映射到物理候选节点 *𝑘 𝑗* 时，对于每个以 *𝑉 𝑁 𝐹𝑗*

开始的虚拟链路 *𝑒𝑖 𝑗* 都必须找到一个物理路径 *𝑃𝑘𝑖 ,𝑘 𝑗* 来放置该虚拟链路。

. *𝑦𝑒*

*𝑖 𝑗*

*𝑘𝑖 ,𝑘 𝑗*

*𝑘 𝑗* ∈C

*, 𝑃*

= *𝑥𝑖𝑘 ,* ∀*𝑒𝑖 𝑗* ∈ *𝐸*new *,* ∀*𝑘𝑖* ∈ C (3-12)

节点分离约束：该约束要求所有 VNF 节点必须分开放置在不同的物理节

*𝑖*

*𝑣*

点上，原因包括安全问题、租户需求等等。

. *𝑦𝑒*

*, 𝑃*

= *𝑥 𝑗𝑘 ,* ∀*𝑒𝑖 𝑗* ∈ *𝐸*new *,* ∀*𝑘 𝑗* ∈ C (3-13)

### 目标函数

*𝑘𝑖* ∈C

*𝑖 𝑗*

*𝑘𝑖 ,𝑘 𝑗 𝑗 𝑣*

目标函数：目标函数不仅需要考虑到运营商的利益，同时还要尽可能得满足租户的需要。在本文中，我们不失一般性地假设运营商的目标是实现物理基础设施的负载均衡。为了实现物理基础设施的负载均衡，目标函数应该倾向于选择剩余容量最多的物理节点和物理链路。实现负载均衡的一种简单而有效的办法，就是优先选择那些目前负载最小的物理节点和链路，从而渐进地、分布式地加载物理基础设施。*𝑚𝑖𝑛𝐶𝑃𝑈* 是物理候选节点中CPU 计算容量的最小值，该最小值作为归一化因子，用于消除公式 [3-14](#_bookmark34)中的不同数据之间的量纲。为了最

大化目标函数，应该尽可能选择那些拥有更多剩余计算能力的物理节点。

Z = . .

min

*𝑣*

*𝑐 𝑝𝑢𝑖*

× .. *𝐶𝑃𝑈𝑘* × *𝑥*

*𝑖𝑘* ΣΣ

*𝑖* ∈ *𝑁 𝑛𝑒𝑤*

\*𝑘𝑖* ∈C *𝑘 𝑗* ∈C

*𝑘* ∈C

*𝐶 𝑃𝑈*

(3-14)

+ ll .

\ *𝑣*

*𝑏𝑤𝑒𝑖 𝑗* × ll .

. *𝑦𝑒𝑖 𝑗 , 𝑃𝑘𝑖 ,𝑘 𝑗* \l\l

ll

*𝑒𝑖 𝑗* ∈*𝐸 𝑛𝑒𝑤*

## 本章小结

本章主要对VNF-FG 扩展问题进行了分析，同时建立了整数线性规划模型。第一节使用网络拓扑图直观地表示已部署 VNF-FG 转发图、VNF-FG 扩展请求图、物理基础设施图。作为无向加权图，具有节点集合和链路集合。同时每个节点都有 CPU 计算容量和网络功能类型两种属性，每个链路都有带宽需求的属性。第二节主要从节点、链路、VNF-FG 扩展图与已放置VNF-FG 之间的节点之间的互连性三个方面提出了 13 个约束条件，VNF-FG 扩展算法在在扩展过程中必须满足这些约束条件。在第三节中，我们不失一般性假设运营商的目标是实现物理基础设施的负载均衡。为了实现物理基础设施的负载均衡，目标函数应该倾向于选择剩余容量最多的物理节点和物理链路。实现负载均衡的一种简单而有效的办法，就是优先选择那些目前复杂最小的物理节点和链路.

# 第 4 章 基于邻接矩阵特征分解的启发式算法研究

## Umeyama 特征分解算法与存在不足

### Umeyama 特征分解算法

已部署VNF-FG 的扩展问题需要在不改变已部署VNF-FG 的情况下，将新VNF-FG 请求中的 VNF 节点和虚拟链路放置到物理基础设施图中，新部署的

VNF 节点同时还要涉及到与旧 VNF 节点的互连性问题。这是一种带有若干具体约束条件的组合优化问题。该问题直观上可以理解为在物理基础设施图中放置一个覆盖旧图的新图的问题。特征分解算法依赖于邻接矩阵，而邻接矩阵反映了 VNF-FG 的网络拓扑，因此特征分解算法是解决 VNF-FG 扩展问题的比较合适的候选方法[[19](#_bookmark94)]。为了解决 VNF 的放置问题，可以使用 Umeyama 的特征分解方法来实现加权图之间的最优匹配，然后基于匈牙利算法来提取映射结果。最优加权图匹配问题 (weighted graph matching problem,WGMP) 是在两个加

权图之间寻找最优匹配的问题，加权图的每个边都有一定的权值。加权图既可 以是无向的，也可以是有向的。加权图可以表示为 *𝐺* = (*𝑉, 𝑤*)，其中 *𝑉* 是加权图的节点集合，*𝑤* 是加权函数, 对于任意两个不同的节点 *𝑣𝑖* 和 *𝑣 𝑗* ，*𝑤* (*𝑣𝑖, 𝑣 𝑗* ) 都可以被映射为一个非负数。

*𝑤* (*𝑣𝑖, 𝑣 𝑗* ) *>*= 0*, 𝑣𝑖* ∈ *𝑉 , 𝑣 𝑗* ∈ *𝑉 , 𝑣𝑖* ≠ *𝑣 𝑗* (4-1)

假设 |*𝑉* | = *𝑛*，节点集合 *𝑉* 可以被表示为 {*𝑣*1*, 𝑣*2*, . . . , 𝑣𝑛*}。如果对于任意两个不同的节点 *𝑣𝑖* 和 *𝑣 𝑗* ，都有 *𝑤* (*𝑣𝑖, 𝑣 𝑗* ) = *𝑤* (*𝑣 𝑗, 𝑣𝑖*)，那么该加权图是无向加权图。否则，该加权图为有向加权图。加权图 *𝐺* = (*𝑉 , 𝑤*) 的邻接矩阵 *𝐴𝐺* 是一个 *𝑛* × *𝑛* 的矩阵。当加权图 *𝐺* 是无向加权图时，*𝐴𝐺* 是对称矩阵。该矩阵可以表示为以下公式：

*𝐴𝐺* = .*𝑎𝑖 𝑗* .  *𝑎𝑖 𝑗* = *𝑤* .*𝑣𝑖, 𝑣 𝑗* Σ *𝑖* ≠ *𝑗*

 *𝑎𝑖𝑖* = 0

(4-2)

假设 *𝐺* = (*𝑉*1*, 𝑤*1)*, 𝐻* = (*𝑉*2*, 𝑤*2) 是两个具有 n 个节点的加权图。最优加权图匹配问题可以转换成找到一个 *𝑉*1 和 *𝑉*2 之间的映射 *𝜙*，该映射 *𝜙* 可以最小化两个加权图 *𝐺* 和 *𝐻* 之间的距离度量。在本文中，该距离度量可以被表示为以下

公式：

*𝐽* (Φ) =

*𝑛*

*𝑖*=1

.

*𝑛*

*𝑗*=1

.

.*𝑤*1

.*𝑣𝑖, 𝑣 𝑗* Σ

− *𝑤*2

.Φ (*𝑣𝑖*) *,* Φ

.*𝑣 𝑗*

Σ Σ Σ2

(4-3)

因为 *𝐺* = (*𝑉*1*, 𝑤*1) 和 *𝐻* = (*𝑉*2*, 𝑤*2) 是两个加权图，可以将这两个加权图的邻接矩阵分别表示为 *𝐴𝐺* 和 *𝐴𝐻* 。*𝐽* (*𝜙*) 可以使用置换矩阵 P 表示成如下公式：

min *𝐽* (*𝑃*) = *𝑃 𝐴𝐺 𝑃𝑇* − *𝐴𝐻* (4-4)

2

其中，∥ · ∥ 用来表示欧几里得范数, 置换矩阵P 用来表示节点之间的映射关系 *𝜙*，欧几里得范数可以表示为以下公式:

..

.2

∥ *𝐴*∥ =

*𝑛*

*𝑖*=1

Σ*𝑛* .*𝑎*

*𝑖 𝑗*

. Σ1/2

(4-5)

因此，最优加权图匹配问题可以转换成找到一个最小化 *𝐽* (*𝑃*) 的置换矩阵

*𝑗*=1

P。如果两个加权图的节点之间存在一个映射 *𝜙*，使得对于 *𝐺* 中任意两个节点

*𝑣𝑖* 和 *𝑣 𝑗* 存在边，当且仅当 *𝐻* 中的对应的两个节点 *𝜙*(*𝑣 𝑗* ) 和 *𝜙*(*𝑣 𝑗* ) 存在边，则说明这两个加权图之间是同构的。相应地，如果两个加权图是同构的，则存在一 个映射 *𝜙* 使得 *𝐽* (*𝜙*) = 0。同时可以表示为：

*𝑃 𝐴𝐺 𝑃𝑇* = *𝐴𝐻* (4-6)

一般而言，加权图匹配问题是纯粹的组合优化问题，所以计算加权图匹配问题的最优解是一个 NP 难问题。因此，需要降低算法复杂度的前提下，找到加权图匹配问题的近似最优解。

文献 [20] 中提出了一种可以用于计算加权图匹配问题的近似最优解的特

征向量分解算法[[20](#_bookmark95)]。如果两个加权图是接近同构的，则可以将式 (4) 进行等价转化，把加权图匹配问题转化为以 *𝑼*2*𝑼*T 为效率矩阵的指派问题，其中 *𝑈*1 和 *𝑈*2

1

分别是 *𝐴*1 和 *𝐴*2 的特征向量的绝对值矩阵，最后通过匈牙利算法对 *𝑈*2 ∗

行处理可以得到最优映射函数，表示为:

T

*𝑈*1 进

max tr .*𝑷*T*𝑼*2*𝑼*TΣ (4-7)

1

其中，H(x) 表示匈牙利算法:

*𝑷* = *𝐻* .*𝑼*2*𝑼*TΣ (4-8)

1

### 现有特征分解算法存在的问题

利用特征向量分解算法进行加权图匹配需要一定的前提条件，Umeyama 特征分解算法无法直接用于解决VNF-FG 扩展问题。因此，需要对Umeyama 特征分解算法进行扩展，用于实现VNF-FG 扩展问题上的节点和链路的映射。通过对比文献[20] 中的最优加权图匹配问题和VNF-FG 扩展问题，我们可以得出以下几个结论：

* + - 1. 文献 [20] 中的最优加权图匹配问题中的两个加权图的顶点数量是相等的，而且特征分解算法只能用来进行链路之间的映射。然而VNF-FG 扩展问题中物理基础设施图和VNF-FG 扩展请求图的节点数量是不一定相等的，不仅需要进行链路之间的映射，同时还需要有节点之间的映射。我们需要改进特征分解算法，使得该算法可以用来匹配不同节点的加权图，同时支持节点之间的映射。
      2. 最优加权图匹配问题中的两个加权图中的链路的映射是一对一的映射， 然后VNF-FG 扩展问题中的VNF-FG 扩展请求图中多个虚拟链路可以放置不同的物理链路中，也可以放置在同一条物理链路中。我们需要改进特征分解算法， 使得该算法可以对两个加权图进行一对多的映射。
      3. 最优加权图匹配问题的最优解是使得所定义的距离度量最小化，如果VNF-FG 扩展问题也以距离度量最小化作为优化目标，可能会出现物理节点的计算容量小于该物理节点所放置的 VNF 节点的计算容量的问题。我们需要改进特征分解算法，使得该算法在匹配过程中需要满足若干约束条件，更好地符合 VNF-FG 扩展问题的要求。
      4. Umeyama 特征分解算法在两个加权图近似同构的情况下，计算结果可以更加接近最优解。然后 VNF-FG 扩展问题并没有要求加权图之间是同构的。我们需要改进特征分解算法，通过调整物理基础设施图的邻接矩阵使得两个加权图更加接近同构。

## 自定义特征分解算法设计

### 物理图和请求图的邻接矩阵设计

为了用Umeyama 特征分解算法解决VNF-FG 扩展问题，我们需要将该问题转换为加权图匹配问题。VNF-FG 扩展问题可以理解为需要寻找物理基础设施图SG 的节点集合和VNF-FG 请求扩展图NCT 的节点集合之间的一个映射 *𝜙*，在满足若干约束条件的前提下，使得SG 和NCT 之间的距离度量最小。我们需要

对Umeyama 特征分解算法进行一些改进，用于物理基础设施图和 VNF-FG 请求扩展图之间的匹配。首先定义物理基础设施图 SG 和 VNF-FG 请求扩展图 NCT 的邻接矩阵为 *𝐴𝑆𝐺* 和 *𝐴𝑁 𝐶𝑇* ，邻接矩阵可以用以下公式计算得出：

*𝐴𝑆𝐺* =  *𝑎𝑖, 𝑗* = *𝐵𝑊𝑒𝑖 𝑗 , 𝑖* ≠ *𝑗*

 *𝑎𝑖,𝑖* = *𝐶𝑃𝑈��*

 *𝑎𝑖, 𝑗* = *𝑏𝑤𝑒𝑖 𝑗 , 𝑖* ≠ *𝑗*

*𝐴𝑁 𝐶𝑇 𝑐* =



 *𝑎𝑖,𝑖* = cpu*𝑖*)

(4-9)

(4-10)

由于加权图匹配问题处理的是相同大小的图，因此我们可以在VNF-FG 请求扩展图中添加孤立顶点，使图的大小相等。通过添加只包含 0 的行和列来将邻接矩阵 *𝐴𝑁 𝐶𝑇 𝑐* 的大小增加到邻接矩阵 *𝐴𝑆𝐺* 的大小。

### 邻接矩阵之间距离度量的设计

距离度量：定义 *𝐽* (*𝜙*) 为邻接矩阵 *𝐴𝑆𝐺* 和 *𝐴𝑁 𝐶𝑇 𝑐* 之间的距离度量，*𝐽* (*𝜙*) 可以由以下公式得出：

.

.

.

Σ

*𝐽* (*𝜙*) =

*𝑛*

*𝑖*=1

*𝑛*

*𝑗*=1

*𝑠*

*𝑖, 𝑗*

*𝑎*

2

− *𝑎 𝜙* (*𝑖*) *, 𝜙* ( *𝑗*)

*𝑐*

(4-11)

将 SG 和 NCT 的邻接矩阵表示为 *𝐴𝑆𝐺* 和 *𝐴𝑁 𝐶𝑇 𝑐* ，距离度量 *𝐽* (*𝜙*) 可以用一个置换矩阵 P 重新表示，该置换矩阵 P 用来表示映射函数 *𝜙*，距离度量可以被转换为以下公式：

*𝐽* (*𝑃*) = *𝑃 𝐴*

*𝑆𝐺*

*𝑃𝑇* − *𝐴*

*𝑁 𝐶𝑇*

*𝑐* 2 (4-12)

因此，*𝐴𝑆𝐺* 和 *𝐴𝑁 𝐶𝑇 𝑐* 之间的最优匹配问题可以被简化为寻找使J(P) 最小值的置换矩阵P 的问题。接下来，需要通过特征分解算法找到使J(P) 最小值的置换矩阵 P。

### 特征分解计算和最优映射函数设计

由于置换矩阵P 是正交的，将J 的空间扩展到使J(Q) 最小的正交矩阵Q 的空间，可以解决最小化J(P) 的置换矩阵的问题。因此，需要对 *𝐴𝑆𝐺* 和 *𝐴𝑁 𝐶𝑇 𝑐* 进行特征分解来寻找解决方案。设 *𝑈𝑆𝐺* 和 *𝑈𝑁 𝐶𝑇 𝑐* 分别为 *𝐴𝑆𝐺* 和 *𝐴𝑁 𝐶𝑇 𝑐* 的特征向量矩阵。

置换矩阵 P 可以确定为最大化 tr .*𝑃𝑇 𝑈*¯ *𝑁 𝐶𝑇 𝑐 𝑈*¯ *𝑇* Σ 的置换矩阵，该置换矩阵

*𝑆𝐺*

表示分配 (二部最大加权匹配) 问题的一种情况，以下公式表示:

tr .*𝑃𝑇 𝑈*¯ *𝑁 𝐶𝑇 𝑐 𝑈*¯ *𝑇* Σ ≤ *𝑛*  (4-13)

*𝑆𝐺*

*𝑇*

因此，P 是通过特征分解算法得到的，该算法使矩阵 *𝑀* = *𝑈 𝑁 𝐶𝑇 𝑐* × *𝑈𝑆𝐺* 的

总权值最大化。解决最大匹配问题的优化算法有很多(例如匈牙利算法)。但是， 本文提出了一种启发式算法，在满足整数线性规划模型中的节点和链路的若干约束(例如布局约束、资源容量约束、VNF-FG 路径之间的依赖性) 的情况下，找到最大的权值匹配。

## 基于特征分解的 VNF-FG 放置算法

### VNF-FG 放置问题概述

### 物理图邻接矩阵变换

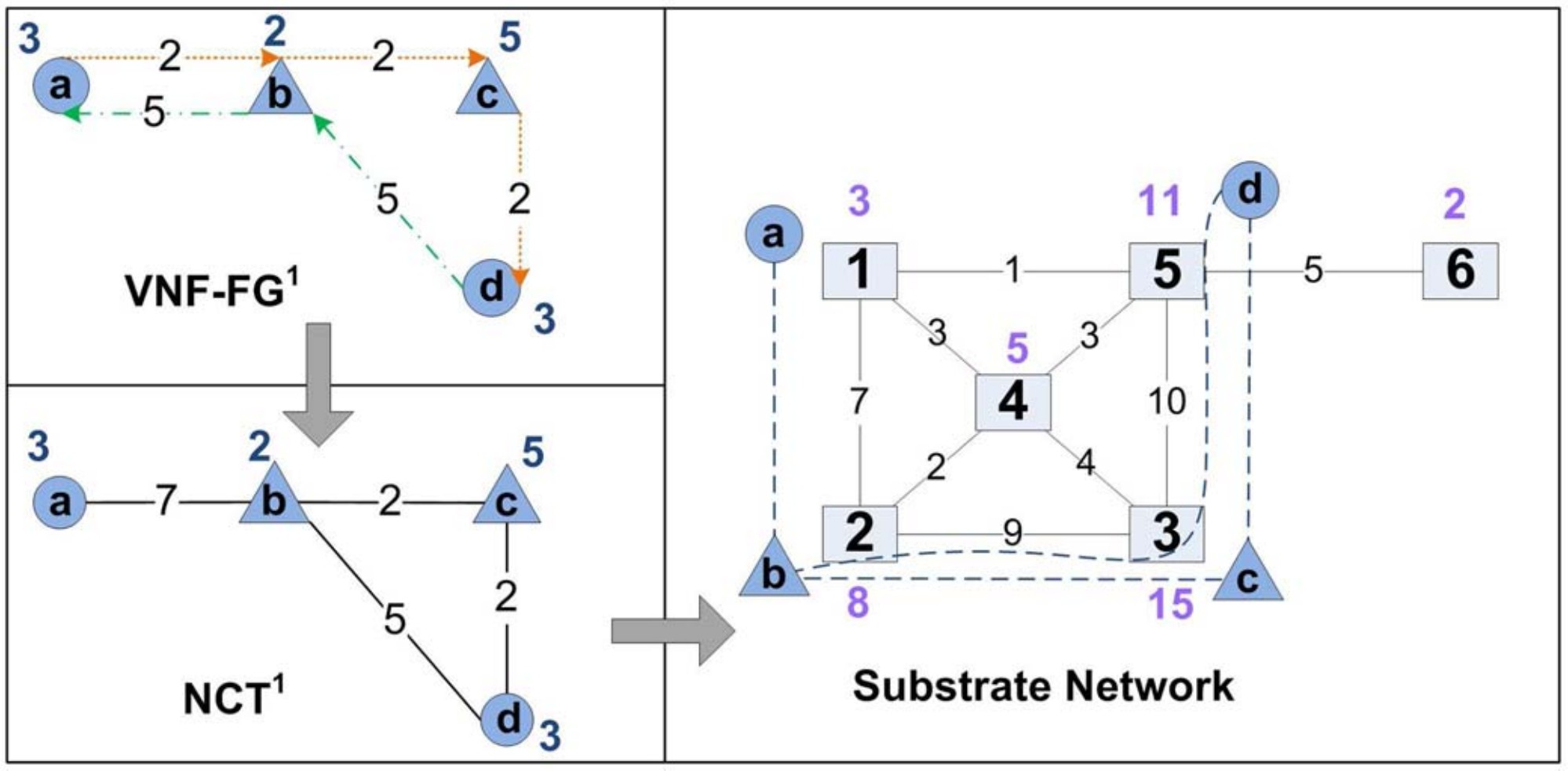
为了解释提出的算法，图 [4-1](#_bookmark47)给出了一个VNF 放置和扩展的例子，其中一个VNF-FG 被转换成一个网络连接拓扑图NCT，随后放置在物理基础设施图 SG 上。节点权重代表CPU 容量，链路权重代表带宽容量。物理链路可以承载多个虚拟链路，只要它们有足够的可用容量。每个物理节点和链路在分配虚拟资源给主机时，都有一个资源阈值，累计相加的结果不能超过该阈值。

图 4-1 VNF 放置和链路过程图

首先计算任意两个不直接相连的物理节点之间的所有路径上的最小链路带宽的最大值，将计算结果更新为基底图邻接矩阵 *𝐴𝑆𝐺* 中两个不直接相连的物理节点之间的链路带宽。

由于特征分解处理的是相同大小的图，所以需要在请求图中添加虚拟孤立顶点，以使图的大小相等。通过变换大小为 m 的邻接矩阵 *𝐴𝑁 𝐶𝑇* ，添加 n-m 行

和 n-m 列的全 0 向量，使得邻接矩阵 *𝐴𝑁 𝐶𝑇* 的大小为 n。

如图 [4-2](#_bookmark49)所示，以下矩阵分别对应基底图邻接矩阵 *𝐴𝑆𝐺* 和请求邻接矩阵

*𝐴𝑁 𝐶𝑇* 。

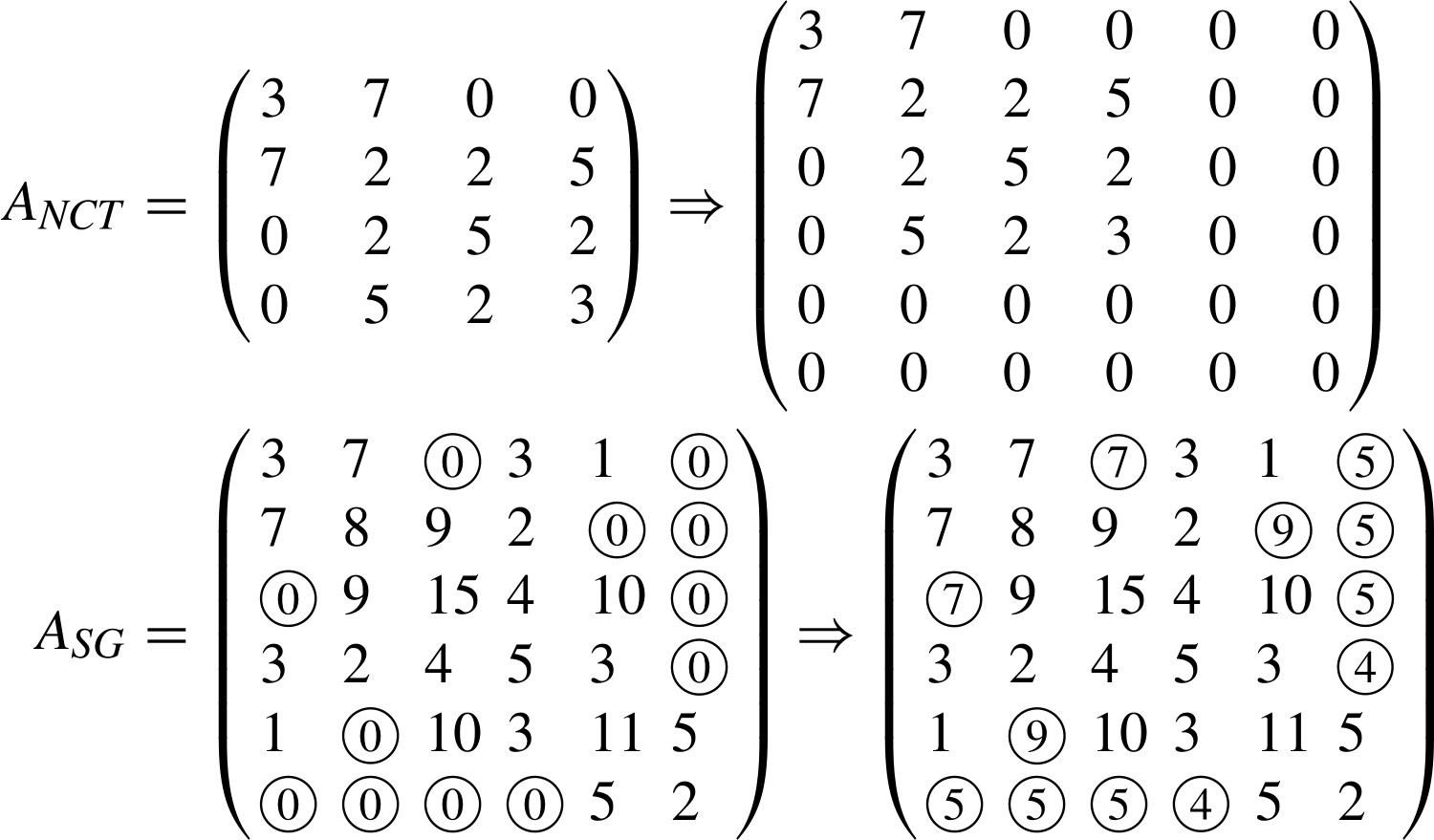


图 4-2 邻接矩阵图

### VNF-FG 放置算法的详细设计

根据算法 1, 可以计算出扩展后的 *𝐴𝑁 𝐶𝑇* 矩阵和基底图邻接矩阵 *𝐴𝑆𝐺* 的特征向量，特征向量绝对值的乘积 M 如图 [4-3](#_bookmark50)所示。

**Algorithm 1** 特征分解算法

1: 计算任意两个不直接相连的物理节点之间的所有物理路径上的最小链路带宽的最大值

2: 根据计算结果更新基底图邻接矩阵 *𝐴𝑆𝐺*

3: 计算 *𝐴𝑆𝐺* 的特征向量矩阵 *𝑈𝑆𝐺*

4: 根据 VNF-FG 得到网络连接拓扑图 NCT

5: 扩展 *𝐴𝑁 𝐶𝑇* 矩阵使得 *𝐴𝑆𝐺* 和 *𝐴𝑁 𝐶𝑇* 的矩阵大小相等

6: 计算 *𝐴𝑁 𝐶𝑇* 的特征向量矩阵 *𝑈𝑁 𝐶𝑇*

*𝑇*

7: *𝑀* ← *𝑈 𝑁 𝐶𝑇 𝑐* × *𝑈𝑆𝐺*

8: 根据算法 2 计算出置换矩阵 P

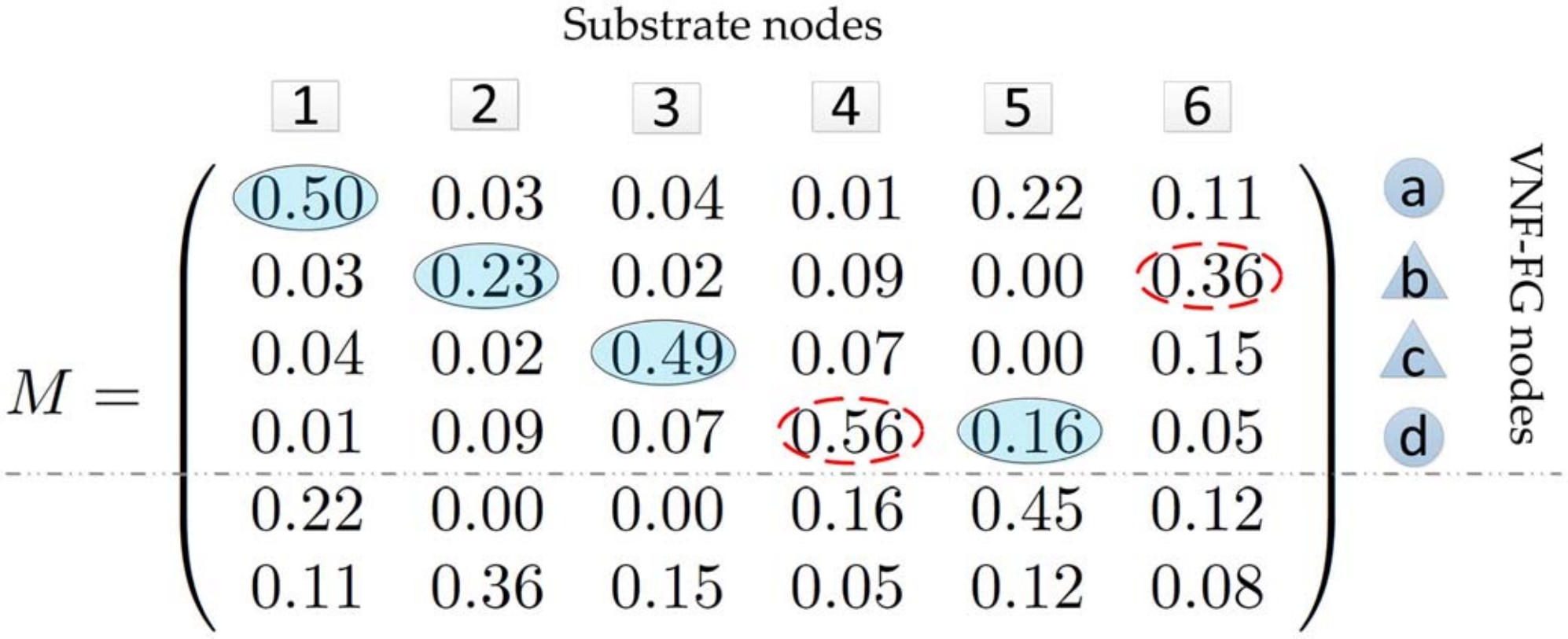


图 4-3 特征向量乘积计算结果

### VNF-FG 匹配算法的详细设计

下一步通过 VNF-FG 匹配算法来计算出置换矩阵 P。对于 VNF-FG 初始请求的每个 VNF，矩阵 M 中更高的元素值反映了最可能的最佳匹配。这些最可能的匹配需要由算法 2 进行校验，以验证节点、链路的所有约束条件都得到满足。

在置换矩阵 P 中，成功放置的节点所对应的元素的值会被设置为 1。根据算法 2 计算出的置换矩阵 P 如图 [4-4](#_bookmark52)所示。

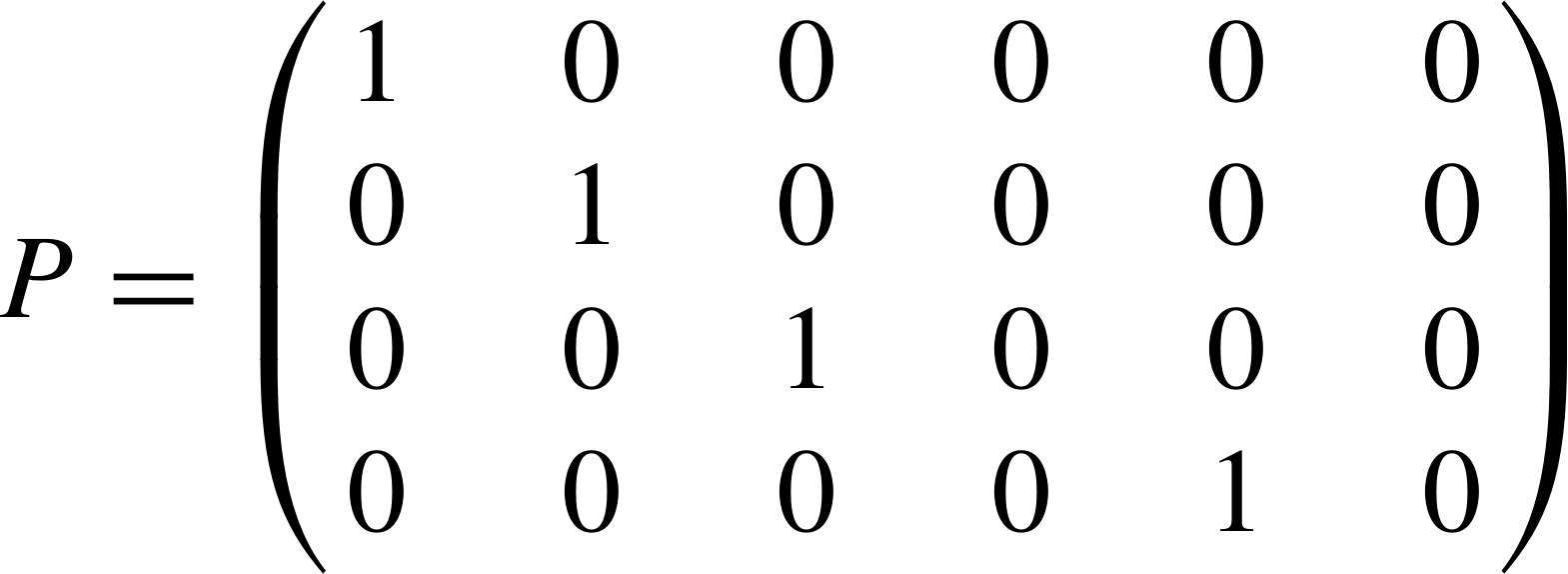


图 4-4 置换矩阵 P

例如，托管请求的 VNF 节点b 的最可能的候选节点是节点 6(值为 0.36)。然而，节点 6 并没有被选中，因为不能符合 VNF 节点 a 与 b 之间链路带宽大于 7 的要求。节点 1 和节点 6 之间的最大可用带宽仅为 5。同样，对于放置 VNF 节点

d 的候选节点 4，尽管在矩阵 m 中的值更高，但是不能满足链路带宽约束。映射VNF 节点b 和VNF 节点d 之间的链路，只有候选节点 5 满足所有要求(节点

2→ 节点 3→ 节点 5)。

算法 2 中的步骤 1 到步骤 3，从特征分解算法的角度来匹配最合适的候选节点。M 的每一行中的最大的元素值，代表了最佳的节点匹配。步骤 4 到 8 用来验证候选节点是否满足整数线性规划模型中设计的所有约束条件。如果候选节点不能满足所有约束条件，算法返回到步骤 3，继续验证该元素在矩阵m 中的所在行的下一个最大元素值所在节点是否符合所有约条件。步骤 19 到 28 对应于由步骤 5 调用的函数，该函数用来验证“所选节点及其相邻节点的所有链路”是否都满足约束条件。实际上，算法 1 中基于特征分解的方法计算出最适合放置VNF 节点的物理节点，元素值越大就意味着更合适的节点匹配，但最终的匹配结果由算法 2 计算得出。

特征分解算法是一种在尽可能降低算法复杂度的情况下用于解决VNF-FG 扩展问题的算法。在算法 2 的步骤 1 中，由于对VNF 节点的放置具有一定的特定顺序，导致算法可能计算出局部最优解。在算法 2 中并没有进行回溯，这是因为作为一种启发式算法，在矩阵 M 中每一行中的最高的元素值就代表着最

**Algorithm 2** VFG-FG 匹配算法

**Input:** *𝑁𝐶𝑇 𝑐* = (*𝑁 𝑐, 𝐸 𝑐*) ; *𝑆𝐺* = (*𝑁 𝑠, 𝐸 𝑠*) ; *𝑀* = *𝑈*¯ *𝑁 𝐶𝑇 𝑐* × *𝑈*¯ *𝑇*

*𝑆𝐺*

**Output:** 置换矩阵 P;VNF-FG 最终的放置和扩展结果

1: **for** each *𝑛𝑐* ∈ *𝑁 𝑐* **do**

*𝑖*

2: *𝑅𝑜𝑤* ← 矩阵 M 的第 i 行

3: *𝑘* ← 计算 Row 中最大值的下标

4: **if** *𝑐 𝑝𝑢*(*𝑛𝑠* ) − *𝑐 𝑝𝑢*(*𝑛𝑐*) ≥ 0 and (其他的节点约束都符合) **then**

*𝑘*

*𝑖*

5: **if** CheckLinks (*𝑛𝑐, 𝑛𝑠* )= True **then**

*𝑖 𝑘*

6: (*𝑛𝑐*) ← *𝑛𝑠*

*𝑖*

*𝑘*

7: *𝑃*(*𝑖, 𝑘*) ← 1

8: 更新邻接矩阵 *𝐴𝑆𝐺*

9: **else**

10: *𝑅𝑜𝑤* [*𝑘*] ← *𝑛𝑢𝑙𝑙*

11: Goto Step 3

12: **end if**

13: **else**

14: *𝑅𝑜𝑤* [*𝑘*] ← *𝑛𝑢𝑙𝑙*

15: Goto Step 3

16: **end if**

17: **end for**

18: **return** P

19:

20: **function** CHECKLINKS(*𝑛𝑐, 𝑛𝑠* )

*𝑖 𝑘*

21: **for** each *𝑙* ∈ *𝑛𝑒𝑖𝑔ℎ𝑏𝑜𝑢𝑟 𝑠*(*𝑛𝑐*)

*𝑖*

22: **if** 节点 l 已经被放置 **then**

23: **if** [*𝑏𝑤* (*𝑒𝑠* (*𝑛𝑠 ,* (*𝑙*))) − *𝑏𝑤* (*𝑒𝑐* (*𝑛𝑐, 𝑙*))] *<* 0 and (符合节点约束) **then**

*𝑘 𝑖*

24: **return** False

25: **end if**

26: **end if**

27: **end for**

28: **end function**

合适的候选节点，可以更快地计算出符合条件的候选节点。对于 VNF 节点的顺序进行重新排序和在算法 2 中进行回溯会使得算法复杂度更高特征分解算法在更低的算法复杂度和更好的放置结果之间进行了均衡。

## 基于特征分解的 VNF-FG 扩展算法

### 物理图邻接矩阵的设计

算法 1 和算法 2 主要解决了 VNF-FG 的初始放置问题，特征分解算法同样可以用来解决VNF-FG 扩展问题，同时保证新的 VNF-FG 扩展请求图与已部署的 VNF-FG 的节点之间的互连性。同样，初始部署的 VNF-FG 也是由特征分解算法计算得到的。在计算初始部署的VNF-FG 的放置方案时，使用的是初始的物理基础设施图的邻接矩阵和初始的VNF-FG 的邻接矩阵。在计算VNF-FG 扩展问题时，需要使用新的VNF-FG 请求扩展图的邻接矩阵，同时需要对物理基础设施图的邻接矩阵进行更新。为了实现扩展，需要在原有算法的基础上，对基底图的邻接矩阵 *𝐴𝑆𝐺* 进行一些更新:

* + - 1. 首先需要删除旧图中所利用 CPU 资源和链路带宽资源，同时固定先前部署的节点在矩阵 M 中的位置。将旧图中已被放置的物理节点的 CPU 剩余容量设置为 0，避免新的 VNF 节点放置到已经放置过的物理节点中, 同时保证了物理基础设施的负载均衡。
      2. 新的 VNF-FG 扩展请求矩阵包括扩展请求的新节点和连接到旧图的节点，这些旧的节点特别标记来实现受限放置。扩展请求节点不受限制。

### VNF-FG 扩展算法的设计

特征分解算法需要计算出新的VNF-FG 扩展请求的邻接矩阵与物理基础设施图的邻接矩阵之间的最优映射，即置换矩阵 P。新的 VNF-FG 扩展请求的邻接矩阵的设计需要考虑到VNF-FG 扩展请求与已部署VNF-FG 的节点之间的互连性。这个矩阵需要包含 VNF-FG 扩展请求的新节点和旧图 (已部署 VNF-FG) 中与VNF-FG 请求扩展图存在边的节点，这些节点需要被特别标记以实现放置在之前被放置的物理节点上。将VNF-FG 中与旧图存在边的新节点，在扩展放置时需要保证放置的物理节点存在与相连的旧图节点所放置的物理节点存在边， 并且该边的链路带宽需要大于两个 VNF 节点之间的虚拟链路的带宽需求。其他的VNF-FG 扩展节点是不受约束的，并且按照整数线性规划模型中所定义的约束条件进行放置。

## 算法复杂度分析

### 算法时间复杂度分析

特征分解算法的复杂性取决于算法 1 和算法 2 的复杂性。NCT 和 SG 的邻接矩阵的特征分解的复杂度直接取决于它们的大小n，并且这个复杂度已知为

*𝑂* (*𝑛*3)。算法1 的步骤8 调用了算法2，算法2 的算法复杂度应该为*𝑂* (*𝑚*∗*𝑛*∗(*𝑛*−1))。在算法 3 的步骤 1, 涉及到m 次循环。在步骤 11 和步骤 15 中，在最坏情况下可能会涉及到 n 次循环。在步骤 20 中，需要对候选节点的所有邻接节点进行遍历，所以最坏算法复杂度为 *𝑂* (*𝑛* − 1)。综上所述，对于我们的解决方案的最坏算法时间复杂度为 *𝑂* (*𝑛*3)。

### 算法空间复杂度分析

首先对算法 1 进行空间复杂度分析，对于算法 1 中所涉及的邻接矩阵 *𝑂* (*𝑛*3)

和 *𝑂* (*𝑛*3)，*𝑂* (*𝑛*3) 经过添加 n-m 行和 n-m 列的全 0 向量，使得 *𝐴𝑁 𝐶𝑇* 的大小变成

*𝑂* (*𝑛*3)。同样，邻接矩阵 *𝑂* (*𝑛*3) 的大小也是 *𝑂* (*𝑛*3)，两个矩阵的空间复杂度都是

*𝑂* (*𝑛*2)。同理，邻接矩阵 *𝑂* (*𝑛*3) 和 *𝑂* (*𝑛*3) 经过特征分解得出的特征向量矩阵 *𝑂* (*𝑛*3) 和 *𝑂* (*𝑛*3) 以及特征向量矩阵的乘积 M 的空间复杂度都为 *𝑂* (*𝑛*3)。置换矩阵 P 的大小为 *𝑂* (*𝑛*3)，其算法复杂度同样为 *𝑂* (*𝑛*3)。综上所述，对于我们的解决方案的最坏算法空间复杂度为 *𝑂* (*𝑛*3)。

## 4.6 本章小结

本章详细介绍了基于特征分解的VNF-FG 扩展算法，同时分析了算法的时间复杂度和空间复杂度。第一节中主要介绍了加权图匹配问题和 Umeyama 特征分解算法。加权图匹配问题是指对于两个加权图找到一个映射函数（或置换矩阵），使得其中一个加权图经过映射后与另一个加权图的欧几里得范数最小。

Umeyama 特征分解算法是指如果两个加权图是接近同构的，则可以将把加权图匹配问题转换成以两个加权图的邻接矩阵的特征向量矩阵为效率矩阵的的指派问题，最后通过匈牙利算法得到最优映射函数。Umeyama 特征分解算法不能直接用于解决 VNF-FG 扩展问题，加权图匹配问题与VNF-FG 扩展问题不完全相同。在第二节中，我们对物理基础设施图和VNF-FG 请求扩展图的邻接矩阵进行了调整，同时定义了邻接矩阵之间的距离度量以及最优映射函数。在第三节中，我们详细设计了基于特征分解的 VNF-FG 放置算法，用于解决 VNF-FG 初

始放置问题。在第四节中，我们将物理基础设施图的邻接矩阵进行了调整，同时提出了基于特征分解的 VNF-FG 扩展算法，用于解决 VNF-FG 扩展问题。在第五节中，主要分析了算法的时间复杂度和空间复杂度。

# 第 5 章 仿真实验的详细设计与实现

## 仿真实验整体概述

本文使用专业网络拓扑图形生成工具 (如 GT-ITM 工具[[21](#_bookmark96)]) 用于生成 VNF-

FG 初始请求和 VNF-FG 扩展请求，物理基础设施网络拓扑是使用真实的网络基础设施拓扑(如Germany50 网络拓扑[[22](#_bookmark97)]) 绘制的。网络拓扑图形生成工具GT-

ITM 可以直接生成网络拓扑，然后将该网络拓扑分成VNF-FG 初始请求和扩展请求。Germany50 网络拓扑是真实的网络基础设施拓扑，可以最大程度地符合现实场景中的网络拓扑的情况。

仿真系统的环境配置为Windows 10 操作系统，Intel i5-8265U 处理器，Python 环境为 3.7.7。仿真实验中使用 Python 程序计算整数线性规划模型的最优解，该程序会对所有可用的物理节点进行遍历，找出符合约束条件的所有放置方案， 最后选择目标函数最大的放置方案作为结果返回。

## 仿真环境

### 网络拓扑生成工具

GT-ITM(即 Georgia Tech Internet work Topology Models) 拓扑生成器可用于生成平面随机图和两种类型的层次性拓扑图。该生成器首先产生 SGB 格式的拓扑文件，再将 SGB 文件转换为网络仿真器 NS2 拓扑格式的脚本，以用于网络仿真。 GT-ITM 拓扑产生器可以创建一种平面随机的拓扑和两种层次型的拓扑

（N-level 和 transit-stub）。网络拓扑结构可以用各种平面随机图作为模型。最简单的一种为随机模型：节点在平面上随机分布，任意两个节点间存在边的概率为常数 0.2。由于它不能很好的反应现实网络的拓扑结构，所以在此基础上提出了其他几种模型，节点仍然在平面上随机分布，只是对生成边的概率函数做了不同的修改。分别是Waxman 1 模型、Waxman 2 模型、Doar-Leslie 模型、指数模型、Locality 模型。

N-level 通过递归的形式生成网络拓扑图。首先用上述 6 种模型中的一种模型生成一个平面随机图，作为首层图。然后用平面随即图代替首层图中的每一个节点，并且依次替代下去。用来代替同层（非同层）节点的平面随机图要求相同（也可不同）。上一层与下一层随即平面图有多种连接方式。

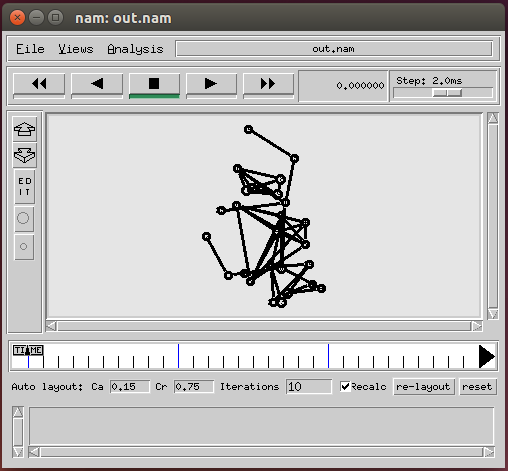


图 5-1 网络拓扑图形化结果

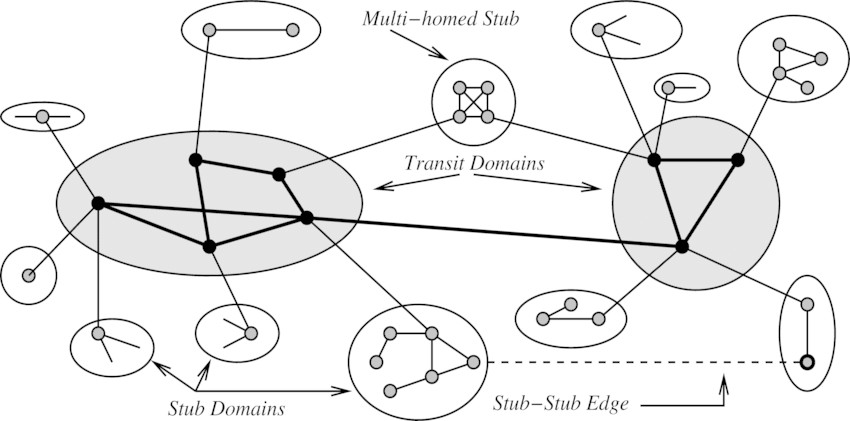
GT-ITM 是 Transit-Stub 模型的最早实现。Transit-Stub 将节点划入不同类型的域，再将这些域连接起来。首先生成一个平面随机图，图中的每一个节点代表一个 transit 域。然后用平面随机图代替这些 transit 域，表示这些 transit 域的骨干拓扑。对transit 域中的每个节点，生成一个或多个随机平面图作为stub 域， 并将其和节点连接起来。最后还可以在特定的节点对之间增加一些额外边，节点对需要满足：一个在 transit 域。一个在 stub 域，或者在不同的 stub 域。

图 5-2 Transit-Stub 网络拓扑图

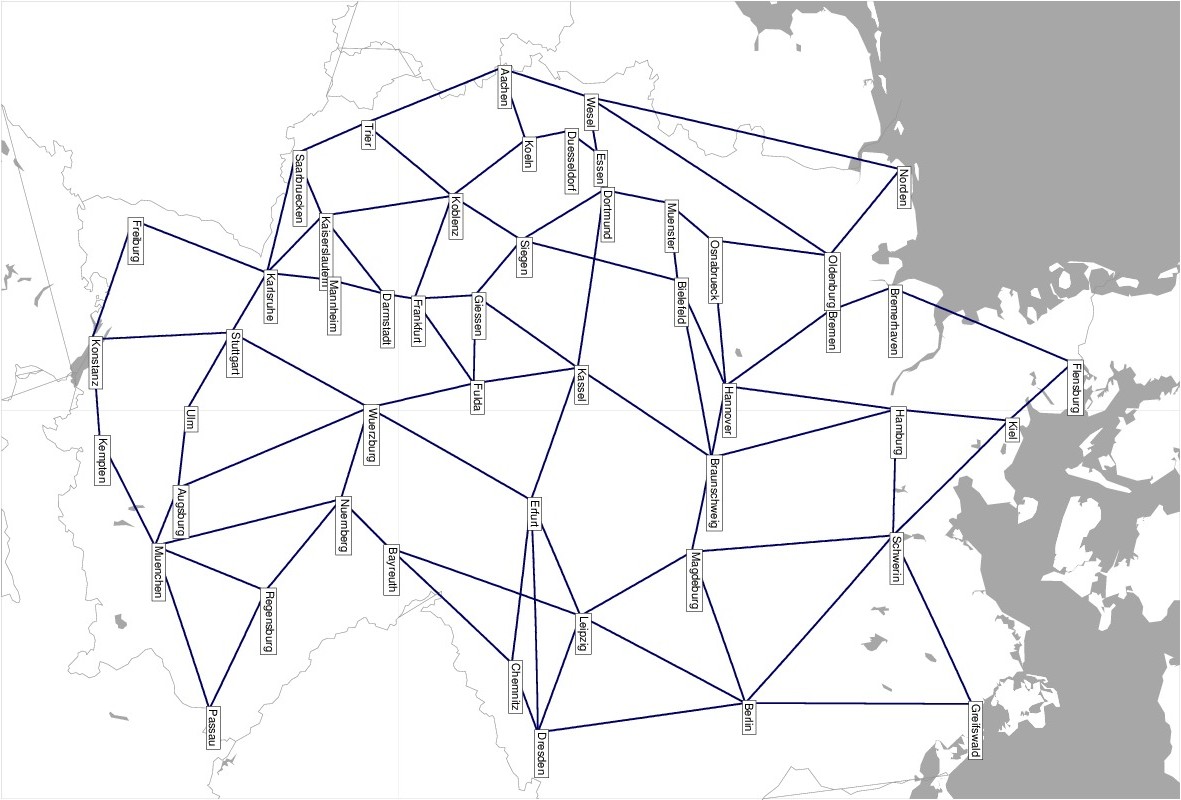
Germany50 网络拓扑对应的是德国某地区真实的公用网络拓扑(50 个节点)， 可以最大程度地符合现实场景中的网络拓扑的情况。这种拓扑结构由德国国家研究和教育网络(German National Research and Education Network，DFN) 设计。该网络有 50 台物理节点和 88 条物理链路。图 [5-3](#_bookmark65)为 Germany50 网络拓扑图。

图 5-3 Germany50 网络拓扑图

### VNF-FG 请求的时间分布设计

日常生活中，大量事件是有固定频率的。例如，某网站平均每分钟有 2 次访问、某医院平均每小时出生 3 个婴儿。这些事件的特点是可以预估这些事件的总数，但是无法知道具体的发生时间。泊松分布就是描述某段时间内，事件具体的发生概率。泊松过程是一系列离散事件的模型，事件之间的平均时间是已知的，但事件发生的确切时间是随机的。一个事件的到来与之前的事件无关

（事件之间的等待时间是无记忆的）。泊松分布的公式如 [5-1](#_bookmark66)所示。其中，P 表示概率，N 表示某种函数关系，t 表示时间，n 表示发生事件的数量。

(*𝜆𝑡*)*𝑛𝑒*−*𝜆𝑡*

*𝑃*(*𝑁* (*𝑡*) = *𝑛*) =

(5-1)

*𝑛*!

为了最大程度上模拟现实场景中出现的情况，VNF-FG 请求使用泊松过程生成，平均每 100 个时间单位 2 个请求，每个请求距离下一个请求的平均时间间隔为 50 个时间单位。请求的生命周期初始放置、扩展和离开，生命周期内部的三个状态之间同样遵循泊松分布。VNF-FG 请求模拟分布如图 [5-4](#_bookmark67)所示。从图中可以看出，不同的 VNF-FG 请求的初始到达存在前后关系，同时 VNF-FG 请

求的生命周期之间存在重合，使得不同的VNF-FG 请求之间初始放置和扩展过程中发生关联，所以需要保证初始放置和扩展过程中不能放置到其他 VNF-FG 已经放置的物理节点中。

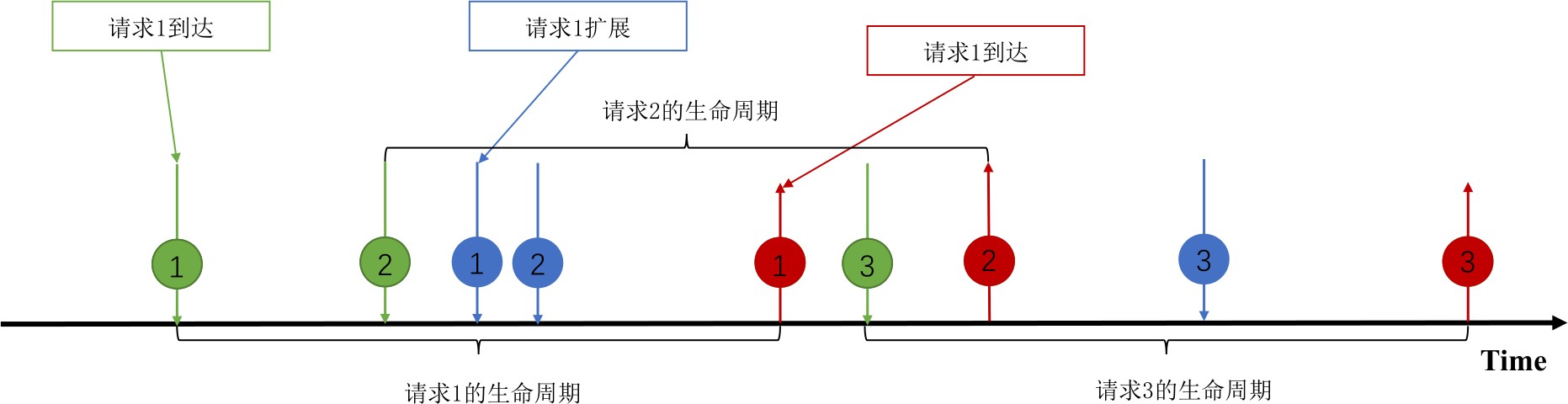


图 5-4 VNF-FG 请求模拟分布图

物理节点和链路的容量在[100,120] 的间隔内随机生成。初始 VNF-FG 请求被设置为 4 个节点。扩展的请求被设置为 3 个节点。初始 VNF-FG 设置为每个虚拟节点 10 个CPU 单元，每个虚拟链路 10 个单元。扩展VNF-FG 设置为每个虚拟节点 20 个 CPU 单元，每个虚拟链路 20 个单元。VNF-FG 节点之间的连通性设置为 0.3(30%)。

## 评价指标

1. VNF-FG 请求成功在物理图中扩展的数量：该评价指标主要反映两种算法可以成功扩展的 VNF-FG 数量，宏观上反映两种算法的最终表现。
2. VNF-FG 成功扩展次数与成功初始放置次数的比率：该评价指标主要体现算法的相对效率，反映在成功初始放置的基础上能够实现成功扩展的能力， 实现对两种算法更公平的比较。
3. 目标函数：为了最大化目标函数，应该尽可能选择那些拥有更多剩余计算能力的物理节点。通过对比特征分解算法的计算结果与ILP 最优解的目标函数，可以更好地验证特征分解算法实现物理基础设施的负载均衡的效果。

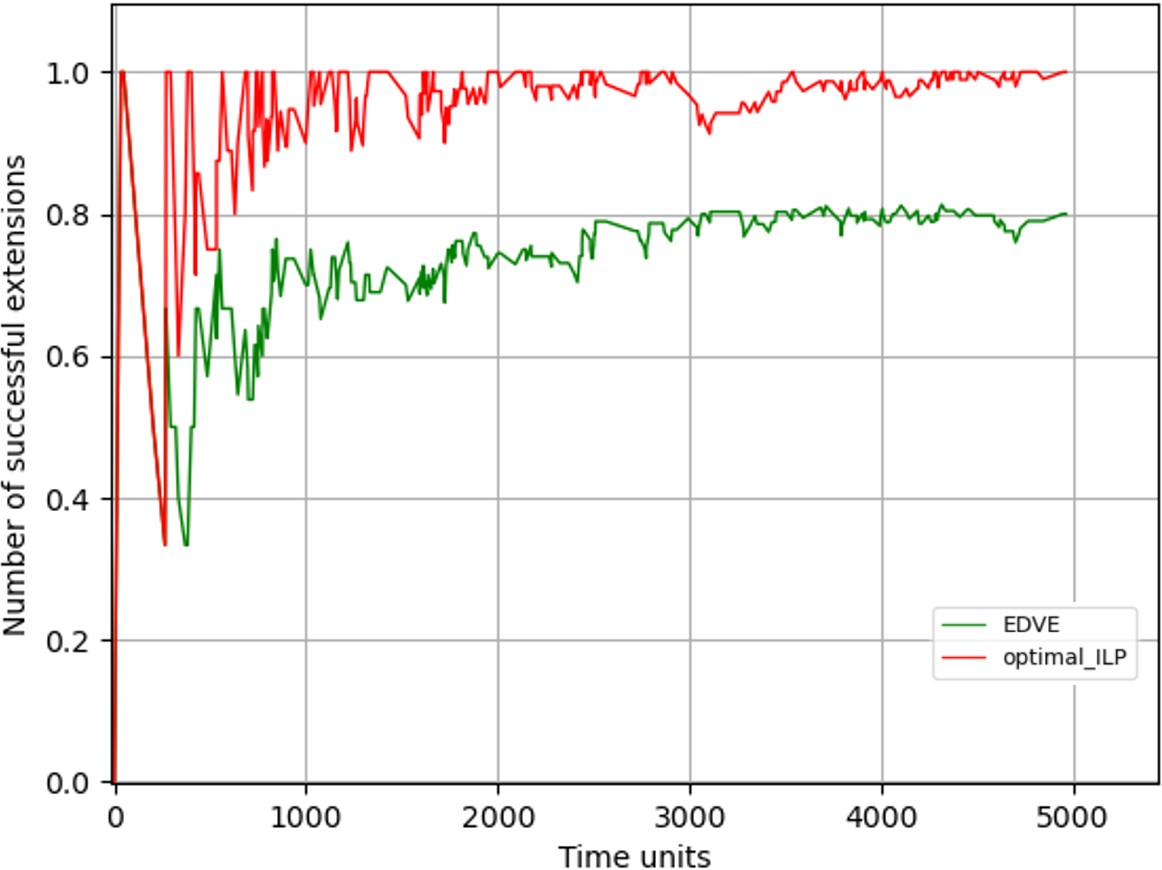
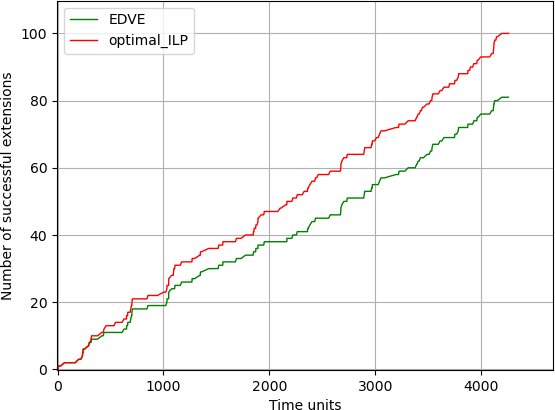
## 仿真结果

由于该拓扑的规模较小，ILP 算法在可接受的时间内产生最优解，因此可以在 Germany50 网络上对两种算法的性能进行评估。

### 扩展成功概率的对比与分析

左图描述了成功放置在物理图中的 VNF-FG 的请求数量随时间变化的曲线，x 轴为时间维度，y 轴为扩展请求中成功放置的次数。从图中可以看出，ILP

算法计算出的最优解几乎实现了对VNF-FG 的所有扩展，而特征分解算法的扩展成功次数为 82 次。右图描述了成功扩展次数与所有生成的 VNF-FG 请求次数的比率随时间变化的曲线，x 轴为时间维度，y 轴为成功扩展次数与所有生成的 VNF-FG 请求次数的比率。从图中可以看出，ILP 算法随着扩展次数的增加，扩展成功的概率稳定在 99% 附近。而特征分解算法的扩展成功的概率逐渐稳定在 80% 附近。



* + - 1. 扩展成功数量 b) 扩展成功比率

图 5-5 特征分解算法扩展结果图

### 目标函数的对比与分析

图 [5-6](#_bookmark73)描述了特征分解算法的扩展结果的目标函数随时间变化的曲线，用于表示所提出的算法的实现质量。在本文中，该函数描述了算法在给定的物流网络基础设施上实现负载均衡的能力。从图 [5-6](#_bookmark73)可以看出，基于特征分解算法的VNF-FG 扩展算法相比于最优解存在一些差距，但是在大部分时间段中两者差距不大。这同时也表明了基于特征分解算法的VNF-FG 扩展算法计算出的次优解比较接近于最优解。

## 5.5 本章小结

本章主要介绍了仿真实验的设计与实现, 同时对仿真结果进行了对比与分析。第一节主要对仿真实验做了整体概述，VNF-FG 的网络拓扑使用GT-ITM 生成，物理基础设施网络拓扑使用 Germany50 网络拓扑，同时对仿真实验的环境配置进行了说明。第二节是对仿真环境的详细说明，首先对 GT-ITM 和Germany50 的特点进行说明，然后介绍了 VNF-FG 请求到达的时间分布，VNF-FG 的初始请求与 VNF-FG 扩展请求之间符合泊松分布，VNF-FG 请求之间也符合泊松分布。初始 VNF-FG 请求被设置为 4 个节点，扩展 VNF-FG 请求被设置为 3 个节

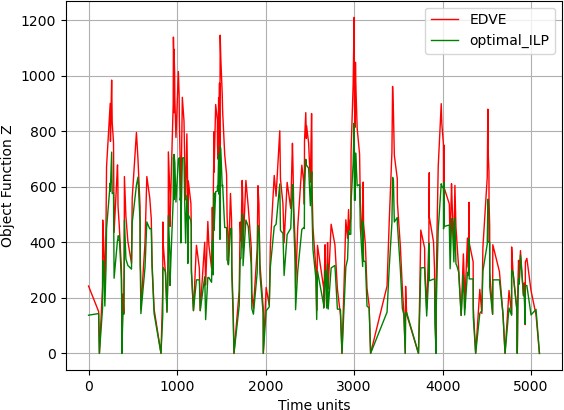


图 5-6 目标函数图

点。第三节介绍了三种评价指标，分别是VNF-FG 请求成功在物理图中扩展的数量、VNF-FG 成功扩展次数与成功初始放置次数的比率和目标函数。第四节在这三种评价指标上对两种算法进行了对比与分析。

# 结 论

在本文提出的基于特征分解的虚拟网络功能转发图扩展算法中，主要的得到的成果有：

（1）为更好地解决 VNF-FG 扩展问题，本文将 VNF-FG 扩展问题建模为整数线性规划模型。整数线性规划模型包括若干约束条件，用于找到合适的候选物理节点。同时为了通过优化求解器计算得出扩展问题的最优解，本文提出了与完整性约束条件相关联的目标函数。

（2）本文对Umeyama 特征分解算法进行改进，使得该算法可以用来匹配不同节点的加权图，同时支持节点之间的映射。同时该算法可以对两个加权图进行一对多的映射，以及在匹配过程中需要满足 ILP 模型中的约束条件。

（3）本文通过仿真实验对比了基于特征分解的VNF-FG 扩展算法计算出的次优解与优化求解器计算出的最优解，同时在三种评价指标上进行了对比与分析。

由于研究时间和资金有限，今后本研究课题依旧存在一定的改进空间：

（1）本课题没有搭建虚拟网络功能转发图部署原型系统，将本方法在真实环境中进行验证。

（2）虽然在本文的仿真实验中可以正常运行，但实际环境存在很多不稳定的复杂因素，所以仿真环境的设计应该考虑到更多的随机性故障事件。

（3）在算法复杂度方面依旧有一定优化空间，可以提出新的启发式算法降低算法时间复杂度。

# 参考文献

1. McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks[J]. ACM SIGCOMM computer communication review, 2008, 38(2): 69-74.
2. Uhlig R, Neiger G, Rodgers D, et al. Intel virtualization technology[J]. Computer, 2005, 38(5): 48-56.
3. Han B, Gopalakrishnan V, Ji L, et al. Network function virtualization: Challenges and opportunities for innovations[J]. IEEE Communications Magazine, 2015, 53(2): 90-97.
4. Li T, Zhou H, Luo H. A new method for providing network services: Service function chain[J]. Optical Switching and Networking, 2017, 26: 60-68.
5. Bari M F, Chowdhury S R, Ahmed R, et al. On orchestrating virtual network func- tions[C] // 2015 11th International Conference on Network and Service Management (CNSM). 2015: 50-56.
6. Houidi O, Soualah O, Louati W, et al. Dynamic vnf forwarding graph extension algorithms[J]. IEEE Transactions on Network and Service Management, 2020, 17(3): 1389-1402.
7. Mechtri M, Ghribi C, Zeghlache D. VNF placement and chaining in distributed cloud[C] // 2016 IEEE 9th International Conference on Cloud Computing (CLOUD). 2016: 376-383.
8. Rankothge W, Le F, Russo A, et al. Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms[J]. IEEE Transactions on Network and Service Management, 2017, 14(2): 343-356.
9. Ayoubi S, Zhang Y, Assi C. A reliable embedding framework for elastic virtualized services in the cloud[J]. IEEE Transactions on Network and Service Management, 2016, 13(3): 489-503.
10. Houidi O, Soualah O, Louati W, et al. An eﬃcient algorithm for virtual network function scaling[C] // GLOBECOM 2017-2017 IEEE Global Communications Con- ference. 2017: 1-7.
11. Dräxler S, Karl H, Mann Z Á. Jasper: Joint optimization of scaling, placement, and routing of virtual network services[J]. IEEE Transactions on Network and Service Management, 2018, 15(3): 946-960.
12. Li X, Qian C. The virtual network function placement problem[C] // 2015 IEEE Con- ference on Computer Communications Workshops (INFOCOM WKSHPS). 2015: 69-70.
13. Li Y, Phan L T X, Loo B T. Network functions virtualization with soft real-time guar- antees[C] // IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. 2016: 1-9.
14. Liu J, Lu W, Zhou F, et al. On dynamic service function chain deployment and readjustment[J]. IEEE Transactions on Network and Service Management, 2017, 14(3): 543-553.
15. Kreutz D, Ramos F M, Verissimo P E, et al. Software-deﬁned networking: A com- prehensive survey[J]. Proceedings of the IEEE, 2014, 103(1): 14-76.
16. Soares J, Gonçalves C, Parreira B, et al. Toward a telco cloud environment for service functions[J]. IEEE Communications Magazine, 2015, 53(2): 98-106.
17. Herrera J G, Botero J F. Resource allocation in NFV: A comprehensive survey[J]. IEEE Transactions on Network and Service Management, 2016, 13(3): 518-532.
18. Pujol P C. Deployment of NFV and SFC scenarios[D]. [S.l.]: Universitat Politècnica de Catalunya. Escola d’Enginyeria de Telecomunicació …, 2017.
19. Mechtri M, Ghribi C, Zeghlache D. A scalable algorithm for the placement of service function chains[J]. IEEE Transactions on Network and Service Management, 2016, 13(3): 533-546.
20. Umeyama S. An eigendecomposition approach to weighted graph matching prob- lems[J]. IEEE transactions on pattern analysis and machine intelligence, 1988, 10(5): 695-703.
21. Zegura E. How to model an internetwork[C] // Proc. IEEE INFOCOM. 2000.
22. Orlowski S, Wessäly R, Pióro M, et al. SNDlib 1.0—Survivable network design library[J]. Networks: An International Journal, 2010, 55(3): 276-286.

# 哈尔滨工业大学本科毕业设计（论文）原创性声明

本人郑重声明：在哈尔滨工业大学攻读学士学位期间，所提交的毕业设计

（论文）《面向动态业务需求的 VNF 转发图扩展算法》，是本人在导师指导下独立进行研究工作所取得的成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明，其它未注明部分不包含他人已发表或撰写过的研究成果，不存在购买、由他人代写、剽窃和伪造数据等作假行为。

本人愿为此声明承担法律责任。

作者签名： 日期： 年 月 日

# 致 谢

历经五个多月的毕业设计终于要结束了，同时我的本科四年生活也要结束了。在这里我要感谢北京邮电大学的王颖老师和实验室的师姐，从毕业设计的选题、实验以及论文的写作都给予了我很大的帮助。每一次的组会都帮助我解决了很多困惑，没有她们的耐心指导，我很难顺利地完成这次毕业设计。同时我还要感谢校内的李宁老师，在论文的写作上给了我很多指导意见。最后需要感谢我的爸爸妈妈，当我遇到困难不知所措的时候一直在鼓励着我，希望我继续坚持下去。每次当我想到自己从小就是他们的骄傲，就有勇气去挑战遇到的难题。

回首本科四年，我付出了很多努力去做自己想做的事，四年光阴没有虚度。同时因为一些原因，也让自己的本科生活留下了一些遗憾。临渊羡鱼，不如退而结网。希望自己永远保持对编程的热爱，在自己喜欢的领域深耕下去。