单位代码 10006
学　　号 36230135
分 类 号 TP301
密　　级 不涉密

北京航空航天大學
BEIHANG UNIVERSITY

毕业设计(论文)

基于图模式的虚拟网络映射模型及算法研究

院（系）名称　高等工程学院
专业名称　计算机科学与技术
学生姓名　曹洋
指导教师　怀进鹏　Wenfei Fan

2010年6月

北京航空航天大学

本科毕业设计（论文）任务书

Ⅰ、毕业设计（论文）题目：

基于图模式的虚拟网络映射模型及算法研究

Ⅱ、毕业设计（论文）使用的原始资料（数据）及设计技术要求：

要求：（1）有系统性和深度：对虚拟网络映射问题进行系统性的理论建模、复杂度证明、可近似性证明以及相应算法设计与证明。

（2）将理论分析应用于在线密集虚拟网络映射的实际场景中。

Ⅲ、毕业设计（论文）工作内容：

1. 综合分析虚拟网络映射的应用需求、场景和特点，建立严格的数学模型：图模式模型，根据应用特点严格定义其语法和语义操作。

2. 分析并严格证明图模式模型语义操作及其若干特殊情况的计算复杂度。

3. 规范定义模型对应的最小化问题，分析其计算复杂度并设计算法解决最小化问题，给出算法的“最小性”证明。

4. 分析模型最优解求解的计算复杂度，对相应的组合优化问题的（不）可近似性分析与证明，证明各语义操作及相应的特殊情况和变种问题的优化问题可近似性分类、计算性质，并给出算法设计框架。

5. 基于图模式模型，提出在线密集型虚拟网络映射请求处理的应用需求，并结合策略博弈相关理论及图模式模型的性质给出解决方案，进行仿真实验分析。

Ⅳ、主要参考资料：

[1] NM Chowdhury and R. Boutaba. A survey of network virtualization. Computer Networks, 2009

[2] M.Yu, Y.Yi, J.Rexford, and M.Chiang Rethinking virtual network embedding: substrate support for path splitting and migration SIGCOMM Comput. Commun. Rev. 38(2):17-29 2008

[3] N.M.M.K.Chowdhury, M.R.Rahman, and R.Boutaba, Virtual network embedding with coordinated node and link mapping. INFOCOM 2009

[4] Vijay V.Vazirani Approximation Algorithms Springer 2001

[5] Michael R.Garey, David S.Johnson Computers and Intractability: A Guide to the Theory of NP-Completeness Bell Laboratories 1979

[6]Ausiello, G. Complexity and approximation: Combinatorial optimization problems and their approximability properties Springer Verlag, 1999

[7] J. Diaz, J. Petit, and M. Serna. A survey of graph layout problem. ACM Computing Surveys(CSUR), 34(3):313—356, 2002

[8] C.H. Papadimitriou. Computational Complexity, John Wiley and Sons Ltd., 2003

高等工程 学院（系） 计算机科学与技术 专业类 36230135 班

学生 曹洋

毕业设计（论文）时间： 2010 年 3 月 1 日至 2010 年 6 月 23 日

答辩时间： 2010 年 6 月 23 日

成　　绩：

指导教师： 怀进鹏

兼职教师或答疑教师（并指出所负责部分）：

系（教研室） 主任（签字）：

本人声明

我声明，本论文及其研究工作是由本人在导师指导下独立完成的，在完成论文时所利用的一切资料均已在参考文献中列出。

作者： 曹洋

签字：

时间： 2010 年 6 月

基于图模式的虚拟网络映射模型及算法研究

学　　生：　曹洋

指导教师：　怀进鹏

摘　　要

网络虚拟化在 Internet 架构、云计算基础设施、网络化软件服务等领域起着越来越重要的作用。本文分析网络虚拟化中的基本问题——虚拟网络映射问题的特点，给出了严格的数学模型：图模式模型。

针对网络虚拟化的两种层次，本文在图模式模型中定义 matching 和 embedding 两种语义操作，分析并严格证明了这两种语义操作均是 NP 难的，其若干特殊情况也是 NP 难的。

针对图模式模型及应用需求，本文还提出了模式最小化问题，设计了一种基于最大-最小化思想的“伪动规”算法能在立方时间内精确求解最小化问题，并分析了若干最小化问题的变种，给出复杂度证明。

针对图模式模型最优解问题，给出了其相应判定问题复杂度证明（是 NP 完全问题）、相应组合优化问题的不可近似性证明和优化问题分类（证明了基本优化问题是 NPO 完全问题——不可近似性，若干特例是 APX 完全问题），最后给出了算法设计框架。

提出在线虚拟网络映射请求处理问题并改进了图模式模型，设计了一种策略博弈(LMSG 及其变种）实现了图模式模型中相应的最优化求解算法框架，证明了各博弈模型的 NE(Nash Equilibrium：纳什均衡)存在性判定问题的复杂度(基本模型的 NE 存在性判定问题是 NP 完全问题），分析变种**UNLMSG**博弈 NE 存在性并给出了基于随机化的 FPTAS 策略搜索近似算法，并基于此给出半在线虚拟网络映射请求处理方法的实施。最后通过仿真实验与已有相关算法进行了对比并取得很好效果。

最后，本文提出了若干可进一步研究的问题。

关键词： 虚拟网络，图模式，复杂度，不可近似性，近似算法，策略博弈

A Theoretical Approach to Virtual Network Mapping: Models,Complexity and Algorithms

Author： Yang Cao

Tutor： Jinpeng Huai

Abstract

Network virtualization is of great important in next generation Internet, Cloud infrastructure provision and SaaS, for its flexibility of network-scope resource isolation and on-demand building. This thesis tackles the virtual network mapping problem, which is a fundamental problem of network virutalization, from a theoretical view, including modeling, complexity analyzing, inapproximability proving and algorithms.

A graph pattern model for the virtual network mapping problem is proposed along with a strict definition of syntax and semantics of the model. There are two semantics, matching and embedding, of our model, according to the two levels of network virtualization. We give a formal proof of the complexity of the two semantics,matching and embedding. Both of them are NP-hard and even some of their special cases such as both of the pattern and base graph are DAGs.

The minimization problem of the graph pattern model is also formalized in this thesis, with a "pseudo-dynamic programming" algorithm designed based on a "maximum-minimum" approach to solve this problem in Cube-Time.

We also study the properties of the optimization problems of finding the optimal solution for matching and embedding semantics in a theoretical way, with the some complexity results of the corresponding decision problems and some non-approximability results of the optimization problems and their variants. In fact, the base optimization problems are NPO-

complete for both of matching and embedding. Even the special cases and some simplified variants are in APX-complete class. Although the optimization problems are extremely complexity and non-approximable, we provide two algorithmic framework for them in perspectives of constructive and enumerative heuristic methods.

A new pratical problem of online virtual network mapping requests processing in DCNs (data center networks)-like environment concerning about the long-term cost-reduction for InPs(infrastructure providers) is proposed. Based on the algorithmic framework of NPO-complete optimization problems in graph pattern model, a strategy game theory model, LMSG (Local Move Search Game) of which the decision problem of Nash Equilibrium existence is proved to be NP-complete in our work, and UNLMSG which is a simplified variant of LMSG with a fast convergence of Nash Equilibrium are proposed. With them, a semi-online approach is given to solve the new raised practical problem. The semi-online approach is proved well by several detailed simulation experiments.

In the end, we also present some future research problems of the work in this thesis.

Keywords: Virtual Network, Graph Pattern, Complexity, Non-Approximability, Approximation Algorithms, Strategy Game Theory.

目　录

1 绪论

本章将主要给出本论文的背景、目标及内容。首先，给出论文研究背景——网络虚拟化；然后，引出网络虚拟化中的关键问题——虚拟网络映射问题；第三，给出与虚拟网络映射问题相关的国内外研究成果，包括相关项目和已有研究进展；第四，给出当前研究的不足之处，并结合不足之处陈述本论文研究目标及研究内容；最后，给出论文规划。

1.1 选题背景及意义

1.1.1 网络虚拟化的需求

虚拟化技术近几年来得到了快速的发展，受到研究界和产业界的广泛重视。利用虚拟化技术，能在单台物理机上运行多台相互隔离的虚拟机，从底层（物理层次）对物理机资源向上层应用提供独立、隔离的视图，进而实现物理基础设施资源的共享。然而高速发展的应用需求对虚拟化技术又提出了新的要求--网络虚拟化。通过网络虚拟化，传统的虚拟化技术将由边缘支撑、辅助技术走进信息领域的核心构架中来。下面将从网络化软件应用、新一代互联网、云计算、绿色 IT 四个方面阐述信息领域对网络虚拟化的需求。

(1)网络化软件

互联网上运行着大量的分布式、网络化软件应用和服务，如网络试验床、电子商务平台、P2P 文件共享平台、高性能计算、Web2.0 数据密集型应用等。不同的网络应用对底层的运行支撑环境提出了不同的需求，如支撑环境所运行的系统版本、网络拓扑结构、网络服务质量、服务支撑成本、计算存储能力等。这种应用需求的灵活性和运行平台的异构性，对底层网络资源提出了各种不同的需求。过去大多采用软件中间件等方式在物理网络资源之上为各种网络化软件构建特定的服务层叠网，对上层网络应用之间提供资源隔离，下层实现资源共享，最终实现在互联网上运行多样的网络化软件应用。但是由于网络化软件应用的复杂性，软件并不遵循严格的层

次结构，导致不同网络化软件应用在不同层次间存在着运行环境交叉、资源使用冲突等问题。例如对底层运行系统配置文件的共享，文件系统的共享，对网络资源如网卡、协议栈、IP 地址、端口等的共享，会导致不同软件之间的运行兼容性和冲突的问题。这些冲突制约了网络资源（计算、数据、通信）的利用效率。根据美国国家标准局（NIST）的调查结果，全球数据中心包含的 1180 万台服务器的 CPU 平均利用率只有 15%。

为了解决这些问题，避免共享造成的网络软件应用之间的冲突，需要将隔离的层次向底层发展，同时增大隔离的粒度，支持对网络资源的隔离。原因有以下几点：

- 第一，低层次的隔离能减少资网络基础设施资源共享造成冲突的种类。底层硬件层提供隔离时由于资源（只能是硬件资源）共享造成的冲突种类少，规律性强，而在上层软件层提供隔离则会受运行环境影响而产生多类型和无规律性的冲突。

- 第二，低层次的隔离时所产生的底层资源共享造成的冲突大多是硬件资源（CPU、内存）的冲突，已有较成熟且广泛应用的处理方法。

- 第三，粗粒度、网络化隔离能为网络化软件的网络运行环境提供隔离的网络资源，比如带宽、节点、协议栈、IP、端口等，为网络化软件应用的部署和运行提供和独立的可定制的网络资源。

于是，基于单机虚拟化和网络技术发展而来的网络虚拟化技术成为替网络化软件提供纯净、隔离运行支撑环境的手段，并能提高网络化软件应用及服务对网络资源的利用效率。网络化软件应用及服务的发展，提出了网络虚拟化的需求。

综上，网络化软件应用及服务需要网络虚拟化的支撑。

(2)Internet 结构

Internet 已有了 30 多年的历史，并已根本性的改变了人们的学习、研究、生活、生产，具有伟大的历史意义。Internet 通过支持单机之间的通信，是分布式、网络应用的基础，并能同时支持多个分布式、网络应用的运行。然而，目前 Internet 的这种特点正制约着下一代互联网的发展，原因如下：

- 第一，Internet 的结构正阻碍着自身的发展。Internet 的特点是由多个 ISP 互联提供的，并且被多个用户使用的全球分布的网络，组织复杂，变更 Internet 的底层架构会涉及到众多 ISP 厂商和网络用户（终端用户和网络应用）的协调，工作量巨大且难以实现。例如 IPv6 的部署。

- 第二，新的网络应用在 Internet 上的部署和运行往往需要新的网络协议，甚至是新的网络拓扑结构，对 Internet 提出了众多的各不相同发展需求。

- 第三，各种网络化软件应用及服务需要在 Internet 上增加各种中间件及类似的"补丁"，造成 Internet 越来越"臃肿"，网络软件运行环境和平台的可靠性越来越低，发生故障的概率也越来越大，而对于故障往往又是加上新的"软件补丁"解决，进而形成了恶性循环。

Internet 目前处于一种"一体适用"，协议栈越来越高、越来越胖，造成网络质量和可靠性的保障越来越困难。借助网络虚拟化技术，能从硬件层对网络基础设施进行划分和隔离，直接提供给 Internet 上的应用。每个网络应用只需在所分配到的隔离的基础设施上安装必须的协议栈即可。因此，网络虚拟化能解决目前 Internet"一体适用"的模式，成为"多体合一"的模式，成为下一代 Internet 的发展方向。

综上，Internet 的发展需要网络虚拟化提供基础。

(3)云计算

根据 Berkeley 的解释，云计算是指在互联网上以服务形式提供的应用系统程序，又指在数据中心用来提供这些服务的硬件和系统软件，是 SaaS 和公用计算（以现用现付的方式提供给大众的服务）的并集，人们可以成为 SaaS 的用户或供应商，也可以成为公用计算的用户和提供商。云计算模式为企业用户提供了商业模式上的革新，企业（云用户）无需事先投入，只需按照需求给云提供商付费，并由云服务商块数部署并提供相应的资源给付费用户。这里面的关键点在于云端服务的部署与提供是透明且快速的。这种快速按需提供服务的能力需要云端快速根据用户需求划分资源并进行相应配置。由于云端的数据中心的设备的计算能力和网络能力较强，为了同时为多个用户提供这种快速的服务，需要一种即用即部署、按需部署、高度共享、可伸

缩的数据中心资源管理功能，而虚拟机正具备这种功能。因此虚拟化技术目前已在云计算数据总新中得到广泛应用，如 Amazon EC2。

而随着网络应用的普及，不仅仅对数据中心中的节点资源（存储、计算）提出更高要求，对链路资源（带宽等）也提出了新的高要求，比如有研究成果显示 Amazon EC2 的网络资源使用效率并不理想。然而网络应用对网络质量的需求增长速度往往超过网络技术发展的速度。因此，为了解决云计算中应用对网络资源（节点、带宽）迅速增长的需求，不仅需要提高节点资源的利用率，也需要提高链路资源的利用率。因此，在利用虚拟化分配和使用网络计算资源时，有必要考虑单机资源分配之外的虚拟化的网络资源分配方法。

综上，云计算基础设施中的虚拟化应用有单机虚拟化向网络虚拟化发占是必然趋势。

(4)GreenIT

全球技术研究和咨询公司 Gartner 研究表明，目前 IT 行业和 IT 企业面临的一个重大机遇是利用信息与通信技术减少企业及其供应链、产品和服务对环境的影响。IT 企业和 IT 基础架构的关键政策将侧重在数据中心和个人电脑的能耗上[1]。2005 年，电力成本占数据中心成本的 40%（￥10M），数据中心用电占世界电能的 2%。同时资源利用率和能耗不成正比，空转的 4 核服务器耗电 400W，空转的 1.5 万转磁盘耗电 300W-450W。在网络基础设施、数据中心中能耗直接关系到企业和基础设施提供商的成本，关系到绿色 IT 的发展。为此，提供更高密度的动态负载汇聚是关键。一方面减少基础设施提供商的运营成本，另一方面，减少散热、二氧化碳排放量等以达到绿色环保指标。

而在网络环境下实现负载汇聚，网络虚拟化仍是首选方案。通过网络虚拟化技术，以虚拟机网络的形式封装网络应用及相应的运行支撑环境，并支持多个虚拟机网络同时运行在一个物理网络基础设施上，结合虚拟网络中的节点迁移、动态路由等技术，实现动态负载汇聚，减少，缩小工作运行状态的基础设施规模，降低能耗和成本。

综上，绿色 IT 需要网络虚拟化的支持。

1.1.2 虚拟网络资源分配

虚拟网络是网络虚拟化中资源隔离的粒度单位。将虚拟网络利用到网络化软件应用及服务、云计算、下一代互联网的过程，即使对物理网络基础设施资源分配的过程。下面，本文将对虚拟网络相关概念，以及在应用虚拟网络对物理网络进行资源分配过程中存在的基本问题进行介绍。

（一）虚拟网络简介

(1)体系结构与模型

虚拟网络的主要作用是提供对网络基础设施资源的隔离，实现多个网络应用对基础设施的更高效率的共享。在物理网络基础设施层之上引入虚拟网络层后，资源的基本实体是虚拟网络，并且网络环境的体系结构发生了变化。网络结构从底至上依次为：物理网络基础设施层，网络服务层，网络应用层。其中网络应用层根据具体的应用需求，向网络服务层提出相应的运行支撑网络环境需求；而网络服务层则根据上层的网络应用需求，向下层物理网络基础设施提供商按需租赁网络资源，并进行相应的协议栈、运行系统、运行库等处理，以虚拟网络的形式为相应网络应用提供隔离的、直接可用的、最优的、有 QoS 保障的网络运行环境支撑；物理网络基础设施层接收网络服务层的租赁请求（虚拟网络请求），根据请求所描述的虚拟网络拓扑结构、节点链路资源请求、QoS 等，分配相应基础设施给网络服务层，分配过程对网络服务层保持透明。

依照这种体系结构，传统的 ISP 与网络用户（网络应用、网络终端用户）式的商业模型也随之有所变动。传统的 ISP 将会被基础设施提供商和网络服务商。基础设施提供商负责运营和管理底层物理网络基础设施资源，并接收并处理来自网络服务商的基础设施租赁请求。网络服务商则接收来自终端用户的有端到端服务质量要求的服务请求，通过从一个或多个基础设施提供商按需租赁基础设施资源（虚拟网络形式），并根据上层应用用户安装特定的网络协议栈、支撑系统等，为上层网络用户（网络应用、网络终端用户）提供特定的、隔离的运行时环境（网络服务）。网络用户也能从多个网络服务商所提供的虚拟网络上获取所需服务。

(2)相关概念

网络虚拟化的引入，对网络应用和服务的模式产生了新的影响，并在运算基础设施以及下一代互联网研究中扮演着重要角色。为了便于描述网络虚拟化相关问题，下面介绍虚拟网络相关的重要概念。

物理网络：一个物理网络可以有物理网络拓扑以及拓扑节点和边上的属性表示。数学描述为一个带权无向图。拓扑图中每个节点和边都有相应的属性向量，分别表示物理网络节点的 CPU、内存、磁盘容量、网卡个数等，以及链路的带宽容量、延时等。

虚拟网络：（又称虚拟机网络）一个虚拟网络由虚拟网络拓扑及拓扑节点和边上的属性表示。其拓扑图中每个节点和边分别表示网络服务商需要向物理网络基础设施提供商租赁的虚拟网络的资源需求，包括节点的 CPU、内存、磁盘容量、网卡的需求和链路的带宽、延时要求。拓扑图中的节点表示一个虚拟网络节点，其物理实体可以是一个虚拟路由器，也可以是虚拟主机。拓扑图中的边表示一条虚拟链路，其物理实体是多条物理链路的集合，比如一条物理路径。

协同：虚拟网络的协作是指不同网络服务商的多个不同虚拟网络可以共存于由一个或多个物理网络基础设施提供商组成的物理网络上。

递归：虚拟网络的递归性是指可以从网络服务商的一个已租赁和配置完成的虚拟网络上划分出一个新的层叠网或次级虚拟网络，并且可以将次级虚拟网络提供给另一个网络服务商，而自己充当着基础设施提供商的角色。

重游：重游性允许同一个虚拟网路中的一个以上虚拟节点部署到同一个物理网络基础设施上的节点上。但是，往往由于需要对虚拟网络链路的资源需求的保障，以及为了对物理网络链路进行链路虚拟化，相邻的虚拟网络节点往往不能部署到统一物理网络节点上。著名的 X-Bone 网络系统首先支持物理网络节点的重游性。

(3)虚拟网络资源分配中的关键问题

在虚拟网络的应用过程中，存在着一个物理基础设施提供商必须解决的基本问题：如何将给定资源需求和 QoS 要求的虚拟机网络映射（部署）到物理网络上。这种虚拟网络映射[1]请求是完全随机的产生的，基础设施提供商需要按照某种机制接受

[1]本文中“映射”和“部署”两词通用

并将这些虚拟机网络部署到公共的物理网络基础设施上，并且在满足虚拟网络资源需求和 QoS 要求的前提条件下尽量减小基础设施维护成本，以及最大化收益。这里所指的虚拟网络资源需求和 QoS 要求包括节点和链路两种，如节点的 CPU、内存、磁盘容量、类型等属性，链路的带宽、延时等属性。

如下例所示，图1.1指出了一个虚拟网络（左边小图）和一个物理网络（右边大图），图中节点上的数字表示 CPU 容量，链路上的数字表示带宽，对于虚拟网络而言节点和链路上的数字表示资源请求，对于物理网络而言节点和链路上的数字表示现有资源容量。虚拟网络资源分配，即虚拟网络映射问题即是将坐标小图所示的虚拟网络映射到右边大图所示的物理网络上，使得虚拟网络节点和链路的资源需求能得到满足。图1.2所示即是一个可行的虚拟网络映射。

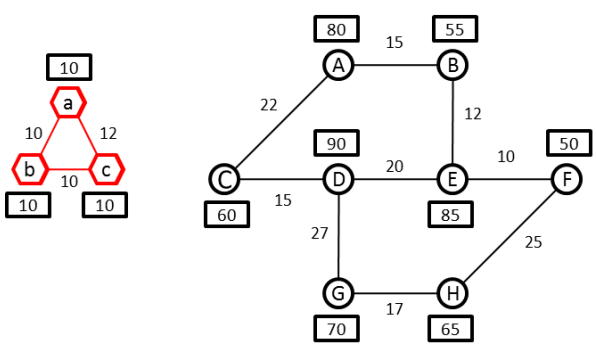

图 1.1: 映射前的虚拟网络和物理网络

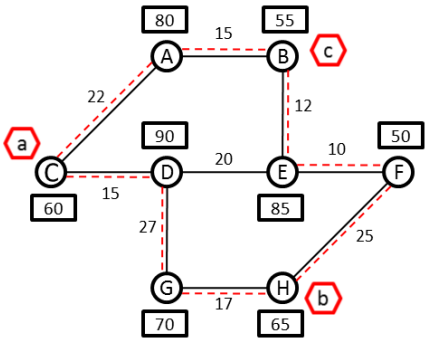

图 1.2: 映射后的虚拟网络和物理网络

这个问题可以分解为以下子问题：

第一，用户提出虚拟网络映射请求的如何表达的？

第二，虚拟网络映射请求对节点和链路的资源约束情况如何？是独占还是共享（虚拟化程度：节点虚拟化与链路虚拟化）？

第三，如何对虚拟网络映射进行数学模型？

第四，根据模型，如何判定用户需求是否具有一致性和最小性？需要设计相应算法对模型中用户虚拟网络需求进行最小化；

第五，建立的数学模型的复杂度如何？需要进行详细的复杂度分析；

第六，模型如何求解？是否存在有理论近似度保证的近似算法，并针对特种特

定样例给出个性化近似算法及理论近似度分析与证明；

第七，如何进行长期优化：即针对在线的密集的虚拟网络映射（部署）请求，如何在长时间周期内确保较高的物理网络资源利用率（基础设施成本优化）。

1.2 国内外研究现状

1.2.1 相关项目

过去，虚拟网络往往指虚拟专用网、层叠网、主动可编程网。到近两三年来，出现了网络虚拟化的概念，而虚拟网络也随着被用来表示虚拟机网络。表1.1[2]给出若干与虚拟网络相关的项目。

按照虚拟化的层次来分，可已将典型的虚拟网络相关项目分为应用层虚拟化、路由层虚拟化、链路层虚拟化、物理层虚拟化。

应用层虚拟化是指通过软件在一个层叠网上构建隔离的虚拟网络，典型项目如VIOLIN。VIOLIN 中，以虚拟机形式将虚拟主机和虚拟路由器运行在物理层叠网上。通过使用虚拟交换机将多个虚拟主机连接成虚拟局域网，然后通过虚拟路由器连接成正个 VIOLIN 虚拟网络。VIOLIN 提供了管理域、地址空间和协议、攻击和故障影响、资源供四个方面的隔离。

路由层虚拟化是指直接在 IP 网络之上构建能提供端到端 QoS 保障的虚拟网络，典型项目如 AGAVE。AGAVE 通过一种基于网络平面(NPs)的新的域间互联结构，能由多个 IP 网络提供商(INPs)共同构建并行的互联网(PIs)以提供端到端的 QoS 需求。

链路层虚拟化的典型项目有 VNET。VNET 其通过一种 Layer 2 的隧道协议(L2TP)，在广域网范围内通过虚拟机互联形成一个 Layer 2 的层叠网，即一个虚拟网络。在 VNET 中，出于不同局域网内的 VM 对外（互联网应用）不会呈现为不同的网络管理域，而是以同一局域网形式对外体现的。

物理层虚拟化通过将物理网络资源以软件包、编程接口的形式呈现给外界网络用户，使得用户能动态配置 Layer 1 光纤网络。典型项目是 UCLP。

按照网络虚拟化的程度来看，可以将虚拟网络相关项目分为节点虚拟化、链路

[2]此图摘自参考文献 [1]

表 1.1: 不同网络虚拟化项目的特点

	Architectural Domain	Networking Technology	Layer of Virtualization	Level of Virtualization
X-Bone	Automating the deployment of IP overlays	IP	Application	Node & Link
Tempest	Enabling alternate control architectures	ATM	Link	
Genesis	Spawning virtual network architectures		Network	Node & Link
UCLP	Dynamic provisioning and reconfiguration of lightpaths	SONET	Physical	Link
VNET	Virtual machine Grid computing		Link	Node
AGAVE	End-to-end QoS-aware service provisioning	IP	Network	
VIOLIN	Deploying on-demand value-added services on IP overlays	IP	Application	Node
VNRMS	Virtual network management	ATM/IP		Node & Link
Darwin	Integrated resource management and value-added services	IP		
NetScript	Dynamic composition of services	IP	Network	Node
PlanetLab	Deployment and management of overlay-based testbeds	IP	Application	Node
VINI	Evaluating protocols and services in a realistic environment		Link	
GENI	Creating customized virtual networks	Heterogeneous		
CABO	Deploying value-added end-to-end services on shared infrastructure	Heterogeneous		Full

虚拟化、节点链路虚拟化、全虚拟化四种类型。其中节点链路虚拟化与全虚拟化的区别在于，全虚拟化将传统的 ISP 的功能分割为基础设施管理和网络服务管理两部分，即不仅仅要提供节点、链路对网络终端用户的虚拟化效果（透明视图），而且需要对网络纵向功能架构通过虚拟化分离开；与之相比节点、链路虚拟化则停留在层叠网的概念领域，仅仅是对终端网络用户提供节点和链路的虚拟透明视图。

根据以上相关项目的分析，虚拟网络部署（映射）过程中，节点的资源分配约束均被视为具有部分独占性的，即对于所分配给节点相应的容量属性（如 CPU、Mem、Storage）不能被分配到统一物理网络节点上的其他虚拟网络节点上的资源共用（如

VIOLIN，X-Bone），而对于分配给虚拟网络链路的资源而言，部分具有类似于虚拟网络节点资源的部分独占性（如 VMRMS, UCLP),部分则不具有，即没有进行链路虚拟化（如 AGAVE，PlanetLab）。

1.2.2 相关研究成果

针对虚拟网络资源分配中的基本问题--虚拟网络部署问题。下面将介绍与虚拟网络部署问题相关的基础研究领域成果，并对国内外虚拟网络部署的相关研究成果按照部署请求处理方式以及算法类型和特点进行分类介绍。

（一）相关基础研究 虚拟网络部署从数学描述角度理解，可以看做是将一个带有权向量的无向图映射到另一个带权无向图上，并且满足权向量之间的约束关系。与之相关的基础研究问题有子图同构、图嵌入、网络流等经典图论问题。

[2]中指出子图同构问题是一个 NP-Hard 问题。 [3]给出了子图同构问题的一个启发式算法。基于该算法， [4]给出了一种虚拟网络部署搜索算法。 [5]使用图嵌入来建模网络服务资源分配问题，并且给出了问题复杂度分析，证明其实一个 NP-Hard 问题。 [6]一文中提出用 Multi-Commodity Flow 来对虚拟网络资源分配进行建模，这里的资源是指链路资源。

（二）offline 虚拟网络映射

离线虚拟网络部署是指在所有虚拟网络部署请求信息均以事先知道的情况下部署虚拟网络。事先知道虚拟网络的拓扑结构、资源约束、QoS 需求等信息，可以达到更高的底层基础设施资源利用率。

[7]最早提出并描述了 Testbed 中的网络映射问题，并且采用映射方案其问题描述中指出节点有资源请求限制和节点类型，给出了相关图论问题如 multi-way cut、sparse cut 的算法调研，有间接指导意义。

[8]中给出了在物理网络资源无限的情况下，离线接收虚拟网络部署请求，并逐个部署到物理网络上的启发式算法。由于该文中假设物理网络资源充足无限制，因此没有提供请求控制功能。该算法通过优先往物理网络节点剩余容量多的节点上部署虚拟网络节点，尽量达到物理网络负载均衡。

[9]中首次对虚拟网络部署请求过程中引入和物理网络节点和链路的费用权值，

将映射的过程建模成一个优化问题。文中离线接收并处理虚拟网络映射请求，通过对物理网罗节点链路引入费用，将映射建模成优化问题后，给出了一种针对特定拓扑--Backbone-Star 结构的启发式网络映射算法，算法的优化目标是最小部署费用。不过，该文中并没有给出如何设定屋物理网络节点和链路的费用权值。

（三）online 虚拟网络映射

由于目前虚拟网络应用于大规模网络实验床、云计算数据中心网络虚拟化等应用场景中，这些应用场景的特点是虚拟网络部署请求均是在线请求的，难以实现预知所有虚拟网络部署请求信息。虽然离线虚拟网络部署能方便的优化并提高物理网络基础设施资源的利用效率，但是与应用场景有所出入。于是，研究人员们纷纷研究在线接收并处理虚拟网络部署请求的算法。

[10]给出了仅考虑虚拟网络的链路 QoS 需求而不考虑节点资源需求，并且在物理资源无限的情况下的虚拟网络映射算法。算法延续了[15]一文中引入物理网络节点、链路资源的费用权值的思想，将在线部署请求转换为一个针对部署费用最小化的优化问题。

[11]一文中给出了 Emulab 的虚拟网络部署方案。Emulab 中物理网络节点被虚拟网络节点所独占，即在整个生命周期中，一个物理网络最多只能承载一个虚拟网络节点。同时 Emlab 还考虑了链路的带宽资源约束，给出了基于模拟退火算法的虚拟网络映射算法。

[12]从另一个角度考虑虚拟网络映射问题。该文首先给出了一个基于[15]研究成果的贪心基本算法。然后提出了使物理网络增加对映射过程的 multi-path splitting 和 path-migration 的支持功能。基于这两种技术假设，作者对前面给出的贪心基本算法进行了改进，并通过实验证明引入这两种技术假设后，物理网络所能接收的虚拟网络部署请求数目有所增加。

[13]一文中，基于 [8]中针对 backbone-star 结构的虚拟网络映射算法，给出了一种贪心的分布式虚拟网络分解算法，将虚拟网络分解为一些列 backbone-star 状的子虚拟网络和一些列将这些子虚拟网络互联的虚拟链路。然而，该文的算法结合[15]算法后，不能保证所有的链路资源 QoS 要求，且没有任何映射优化。

[14]一文中，作者同时考虑节点资源约束和链路资源约束，在支持 multi-path splitting 的假设前提下，建立 MIP（混合整数规划模型）描述映射过程，并且通过利用 LP-rounding 算法，近似求解出映射方案。MIP 模型中，通过将模型所要优化的目标函数中引入与节点和链路负载相关参数，结合多元商品流算法，成功在节点映射过程与链路映射过程之间建立联系，一定程度提高了资源利用效率。

[4]一文中，作者基于子图同构算法 [3]，使用链路延时作为搜索深度限制，给出了一种考虑节点、链路资源请求，同时不需要 multi-path 支持的虚拟网络映射回溯搜索算法。算法的目的也是负载均衡，以期通过负载达到提高资源利用率的效果，继而接收更多的虚拟网络部署请求。

1.3 论文研究内容和目标

1.3.1 现有研究成果存在的问题

第一，没有合适、严格的数学模型描述虚拟网络特性，体现虚拟网络用户定义的特性。比如虚拟网络用户往往不关注虚拟节点之间拓扑连接，而只是关注节点之间的通信服务质量，而目前的虚拟网络描述模型（简单的图模型）无法表达。

第二，对于虚拟网络链路的资源约束，现有研究成果要么不予以考虑，要么则以链路资源部分独占为基础，对与既有虚拟网络链路通信质量需求有没有链路虚拟化的情形不适用。

第三，没有对虚拟网络部署（映射）问题进行详细的复杂度分析，且严格依赖诸多假设（比如 mult-path，path-splitting 等），对具体实际问题不能直接应用，且对未来工作也没有太大指导意义。

第四，所给出的算法都比较初级，且限制条件很多。没有针对各种情况的低复杂度且具有性能保障的近似算法。

第五，现有算法还不能处理像云计算数据中心虚拟化等应用场景中，虚拟网络部署请求的在线、密集、随机的特性，算法均集中于一次映射，对于在线密集型虚拟网络请求的处理，由于算法的复杂度会造成部署效率不够而产生物理基础设施服务性能波动。需要设计一种能有效处理在线密集型的虚拟网络部署请求处理的方法。

1.3.2 研究目标及内容内容

针对上面分析的现有研究存在的五点不足，本文研究内容相应地列举如下：

第一，针对虚拟网络映射问题的若干特点，对其分类并建立统一的、合适的、严格的数学模型。

第二，针对虚拟网络映射问题的特点及分类，为所建立的数学模型（语法）定义相关语义操作，并分析和严格证明各语义操作的复杂度。

第三，提出并解决模型相应的最小化问题，包括问题定义、复杂度分析、（“伪动规”）算法设计与证明。

第四，给出模型最优解求解复杂度分析，给出相应组合优化问题的不可近似性分析与证明，并给出相应最优解求解算法框架。

第五，基于所建立的数学模型，提出如何处理在线虚拟网络映射请求问题，分析问题特点并对初始数学模型的改进，基于策略博弈给出相应的解决方案，并通过实验与现有方案进行对比。

1.4 论文组织结构

论文组织如下：

第二章给出相关理论背景知识简介；

第三章给出 graph pattern 模型的语法定义，两种语义操作及其复杂度分析与证明。并提出若干模型求解的优化问题并给出相应结论；

第四章提出 graph pattern 模型中的最小化问题，给出问题定义、分析以及多项式时间“伪动规”立方时间算法，并分析最小化问题的若干扩展问题的复杂度；

第五章针对 graph pattern 模型最优解求解中的优化问题，分析优化问题类型，给出复杂度证明、不可近似性证明，以及相应算法设计框架；

第六章提出在线密集型虚拟网络映射请求处理问题，并扩充 graph pattern 模型的语法定义，提出一种策略博弈 Local Move Search Game(LMSG)实现 graph pattern 模型中优化问题求解的算法框架，分析并证明 textbfLMSG策略博弈模型及其变种的 Nash Equilibrium 存在性判定问题的复杂性。基于 LMSG，给出一种“半在线”方法

处理该应用问题，并通过仿真实验将该半在线方法与现有研究工作进行对比与分析。

第七章总结全文并基于现有工作进一步提出新的研究点。

2 相关知识简介

内容提要：本节将简单介绍本论文所使用的相关理论基础知识、相关理论研究前沿成果，具体包括：复杂性理论、近似算法及可近似性理论、图论相关等。

2.1 复杂性理论

复杂性理论是计算机科学领域里产生最早、研究最深入的领域之一，其最初始来源和理论基础是 1936 年问世的图灵机理论，其主要目的是分析和界定问题的计算复杂度，即任意定义明确的计算任务的固有复杂程度。其附加目的是分析不同计算任务之间的关系。然而，事实上复杂性理论在其主要目的上的成果并不多，而是在附加目的，即分析不同问题之间的关系上有了成形理论。即可以分析并判定两个完全不同的问题的计算过程及其复杂程度是否是等价的，或者判定哪个更难。本文正是使用复杂性理论的这一功能，通过比较虚拟网络映射问题与现有问题的难易程度而分析其固有计算复杂程度的。

在复杂性理论中，所考察的问题被限定在判定问题类中（事实上，所有优化问题均存在一个对应的判定问题，可以等价地表达优化问题的计算难度）。最基本的判定问题复杂度分类即为 P 和 NP 类（还有其他分类，如 PSPACE, NL, coNP, RP, BPP, ZK, PH, IP 等）。事实上，P 是指能通过确定性图灵机通过有限步输出判定结果的问题，NP 是指能通过不确定性图灵机在有限不输出判定结果的问题。在 NP 类问题中，存在一部分问题，其他 NP 类中所有问题均可规约到这一部分问题中，这里的“规约”指如果可以给出这一部分问题的计算过程及方法，则可以其他所有 NP 类问题的计算过程及方法。这类问题称为 NP 完全问题。NP 完全问题不存在多项式时间算法。

本文需要分析虚拟网络映射问题及其各种变种问题的复杂度，及分析这些问题是否属于 NP 完全问题。证明一个（判定）问题属于 NP 完全问题，需要先证明该问题属于 NP 类，然后证明该问题是 NP 难的。由于本文的第 3 章将进行虚拟网络映射问题模型的计算复杂度分析与证明，这里对基本的证明 NP 完全问题的思路进行说明。

首先，在证明问题的复杂度之前，我们需要分析该问题是否是判定问题，并且对

其进行标准化描述。一个判定问题有两部分组成——实例(instance)和问题(question)。比如对于判定问题：两个奇数之和是否是奇数？其实例描述为：两个奇数，问题描述为：这两个奇数之和是否是奇数？问题的答案只有是、否两种。一个判定问题属于NP类，当且仅当对于给定的一个实例，存在多项式时间算法能输出在该实例下问题的答案（是或否）。对于上例而言，给定一个实例（两个奇数），我们能在多项式时间内判定两者之和是否是奇数，因此该问题属于NP类的。（事实上，其是NP类的，但不是NP难的，因此不是NP完全问题）。

在证明了一个问题是NP类问题后，欲证明其是NP完全问题，还需证明其是NP难。证明方法是构造一个从NP完全问题 Π_{NC} 到当前研究问题 Π_0 的实例集合之间的一个多项式时间完成的规约 R，使得对任意 Π_{NC} 问题的一个实例 $I_{\Pi_{NC}}$,其判定输出结果为“是”当且仅当对应的 π_0 问题的实例 $R(I_{\Pi_{NC}})$ 输出结果为“是”。这种规约被称为Karp规约(Karp-规约),P_1 可以karp规约到 P_2 记作 $P_1 \leq_m P_2$. 如果 $P_1 \leq_m P_2$，则如果 P_2 存在多项式时间算法，则 P_1 能通过多项式时间的Karp规约使用 P_2 的算法，从而存在多项式时间算法。也就是说，$P_1 \leq_m P_2$ 表明 P_1 至少与 P_2 一样简单，其逆否命题表明如果 P_1 不存在多项式时间算法，则 P_2 也不存在多项式时间算法，及 P_1 是NP完全问题，则 P_2 也是NP完全问题。

本文的第三章将使用Karp规约证明虚拟网络映射问题及其变种问题的复杂度。

2.2 组合优化，近似算法，可近似性

（一）优化问题

近似算法和可近似性理论是针对优化问题，特别是组合优化问题而设计的。前面提到过，每个优化问题存在这一个对应的判定问题。因此，根据判定问题的复杂度理论及其问题复杂度类型划分，为优化问题建立类似的计算理论。

一个优化问题可以表示为一个四元组 $(I_P, SOL_P, m_P, goal_P)$,其中:

INSTANCE I_P ：问题 P 的实例集合；

SOLUTION SOL_P ：问题实例到可行解的函数: $\forall x \in I_P$，$SOL_P(x)$ 表示 x 对应的可行解集合；

MEASURE m_p ：度量函数：$\forall x \in I_P, y \in SOL_P(x), m_P(x,y)$ 是一个正整数表示可行解 y 的优劣程度；

GOAL $goal_P$ ：$goal_P \in \{MIN, MAX\}$，表明 P 是最大化问题还是最小化问题；

一般情况下，对于问题的一个实例 x，我们使用 $SOL_P^*(x)$ 表示 x 的最优解，即 $\forall y^*(x) \in SOL_P^*(x)$:

$$m_P(x, y^*(x)) = goal_P\{v|v = m_P(x,z) \wedge z \in SOL_P(x)\}$$

实例 x 的任意最优解 $y^*(x)$ 的度量函数值用 $OPT_P(x)$ 表示。

优化问题对应的判定问题可以表示为二元组 (I_{P_D}, Q)，其中：

INSTANCE I_{P_D} ：$I_{P_D} = I_P$, 度量函数 $m_p : I_{P_D} \to SOL_P(I_{P_D}))$, 整数 $K, goal_{P_D} \in MIN, MAX$;

QUESTION ：$goal_{P_D} > K$（$goal_{P_D} = MAX$）或 $goal_{P_D} < K$（$goal_{P_D} = MIN$）是否成立？

为了刻画优化问题的复杂度并对其进行分类，在过去四十年内产生了许多方法。最直接衡量优化问题复杂程度的方法是考察解决所给优化问题所需的时间，并对应于判定问题为优化问题建立一套类似的复杂度理论。与判定问题类似，优化问题根据其求解复杂程度被被划分成为若干类，其中最基本的是 NPO 类和 PO 类，分别对应于判定问题中的 NP 类和 P 类。与判定问题中 NP-P 类问题的复杂度无法准确定位一样，对于属于 NPO-PO 类问题的固有复杂度还不能准确描述。

（二）近似算法

由于优化问题中，存在许多问题在 $P \neq NP$ 的条件下是不存在多项式时间复杂度的算法能求出其最优解的。在这种情况下，退而求其次，我们可以寻找一个多项式时间算法输出一个离最优解相差不太大的可行解。严格来说，给定一个优化问题的输入实例 x，我们称一个可行解 $y \in SOL(x)$ 是原优化问题的一个近似解；对任意永远输出优化问题可行解的算法，我们称之为该优化问题的近似算法。显然，为优化问题寻找一个多项式时间的近似算法并不难。但是，对于设计近似算法而言，我们最关

心的是近似算法输出的近似解的质量，即是否能保证近似解离最优解的“距离”在某一个预定的可接收的区间内。

为此，人们定义了衡量近似解优劣程度的定量指标——可行解的近似度(approximation factor)。其严格定义如下：

对于给定的优化问题 P，对 P 的任意实例 x 以及相应的任意可行解 y，y 相对于 x 的近似度定义为

$$R(x,y) = \max\{\frac{m(x,y)}{OPT(x)}, \frac{OPT(x)}{m(x,y)}\}$$

显然，无论对于最大化问题还是最小化问题，最优解对应的近似度为 1，且近似度始终大于或等于 1。有了可行解的近似度，相应的近似算法的近似度可以定义如下：

给定一个优化问题 P 和一个 P 的近似算法 A，我们称 A 是一个 P 的 $r-$ 近似算法（或近似度为 r 的近似算法），当对任意 P 的实例 x，A 输出的近似解 $A(x)$ 的近似度具有下界 r，即

$$R(x, A(x) \leq r)$$

我们称算法 A 的近似度为 r（注意区别近似算法的近似度和可行解的近似度的概念）。

常见的多项式时间近似算法设计技巧有贪心方法、序列化方法、局部搜索方法、线性规划方法、动规方法、随机化方法。

我们称那些近似解具有下界（近似度）的近似算法为具有性能保障的近似算法。人们往往关注与近似算法的近似度的特性。根据优化问题的近似算法近似度的特点，NPO 类的优化问题还可以被划分成几个子类。

Class APX 对任意优化问题 P，如果 $P \in NPO$，且存在常数 $r > 1$，P 存在一个多项式时间的具有近似度 r 的近似算法，则 $P \in APX$。所有符合前面所述条件，即所有具有常数近似度近似算法的 NPO 类问题成为 APX 类问题。

Class PTAS 对任意属于 NPO 类的优化问题 P，算法 A 被称为 P 的一个多项式时间近似模式(PTAS)，对于任意给定的实数 $r > 1$，对于 P 的任意实例 x，A 能在关于 $|x|$ 的多项式时间内输出具有近似度为 r 的近似解。所有具有多项式时间近似模式(PTAS)的 NPO 类问题成为 PTAS 类。

Class FPTAS 对任意属于 NPO 类的优化问题 P，算法 A 被称为 P 的一个完全多项式时间近似模式(FPTAS)，对于任意给定的实数 $r > 1$，对于 P 的任意实例 x，A 能在关于 $|x|$ 和 $\frac{1}{r-1}$ 的多项式时间内输出具有近似度为 r 的近似解。所有具有完全多项式时间近似模式(FPTAS)的 NPO 类问题成为 FPTAS 类。

其中对于 APX 类优化问题，人们还进行了扩充，提出了 F-APX 类，定义如下：

Class F-APX 给定一类函数 F，F-APX 定义为由所有满足如下特点的 NPO 问题 P 的集合：$\forall f \in F$，P 存在一个多项式时间的近似算法，且具有近似度 r(n)。

根据 F-APX 的定义，APX、$log - APX$、$poly - APX$、$exp - APX$ 分别表示 F 为常数函数集合、$O(\log n)$ 函数集合、$\cup_{k>0} O(n^k)$ 函数集合、$\cup_{k>0} O(2^{n^k})$ 函数集合。显然

$$PTAS \subseteq APX \subseteq log - APX \subseteq poly - APX \subseteq exp - APX \subseteq NPO$$

（三）可近似性

由近似算法的定义可以看出，并非所有优化问题均有近似度保障，或者说有满足某些条件的近似度保障。因此，有必要研究判定一个优化问题是否存在符合某些条件近似算法的方法。对应于判定问题中最核心的研究判定问题复杂程度的方法——规约（Karp 规约，Turing 规约等），优化问题也存在相应的规约。由于 Karp 规约不能确保两个优化问题之间 m 的一致性，以及可行解的近似度的关系。为此，人们提出了若干种优化问题之间的规约（较 Karp 规约强），不仅能将 P_1 的实例映射到 P_2，同时也能将 P_2 中较优的可行解映射回到 P_1。

由于后文将使用到 NPO 问题之间的一种确保近似度的规约——L 规约 $(L - reuduction)$，下面将简单给出 L 规约的定义。

P_1 和 P_2 是两个 NPO 问题，P_1 被称为可 L 规约到 P_2，记作 $P_1 \leq_L P_2$，当存在两个多项式时间函数 R、S，及两个正常数 α 和 β，满足：

- 如果 x 是 P_1 的一个实例，且该实例具有最优解 $OPT(x)$，则 $R(x)$ 是 P_2 的实例，且相应的最优解 OPT(R(x))满足：

$$OPT(R(x)) \leq \alpha \cdot OPT(x)$$

- 如果 s 是 $R(x)$ 的任一可行解，则 $S(s)$ 是实例 x 的一个可行解，且满足

$$|OPT(x) - m(S(s))| \leq \beta \cdot |OPT(R(x)) - m(s)|$$

L 规约的关键属性是其能保留近似度。如果存在一个从 P_1 到 P_2 的 L 规约 (R, S, α, β)，i.e.$P_1 \leq_L P_2$，如果 P_2 具有多项式时间近似度为 ε 的近似算法，则 P_1 存在一个多项式时间近似度为 $\alpha\beta(\varepsilon - 1)$ 的近似算法。

要证明一个优化问题 P_0 是 APX 完全问题，只需证明存在一个从任一 APX 完全问题到 P_0 的 L 规约即可。

2.3 图论相关研究成果

（一）图论相关问题

由于虚拟网络映射问题直观上来看是研究两个带属性图之间的关系，与子图同构、斯特纳树、最多不相交路径有一定联系，这里简单介绍以下这三个图论相关问题及，后文中会以这些图论研究成果为基础进行讨论、证明和算法设计与比较。

(1)子图同构(graph isomorphism)

图同构问题可以描述如下：

INSTANCE ： 两个图 $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$.

QUESTION ： G_1 是否具有一个子图与 G_2 相似，即是否存在 $V' \subseteq V$，$E' \subseteq E$，使得 $|V'| = |V_2|$，$|E'| = |E_2|$，且存在一个一一映射 f:$V_2 \to V'$，使得 $u, v \in E_2$ 当且仅当 $\{f(u), f(v)\} \in E'$?

通过限制子图同构问题的实例为 G_2 是完全图，得到 CLIQUE 问题，从而该问题是 NP 完全问题。事实上，当 G_1 和 G_2 均是正规图、二分图、L-1 时，问题任然是 NP 难的。

(2)斯特纳树(Steiner Tree)及其变种

斯特纳树在网络设计中经常用到，虚拟网络映射问题，特别是其中的模式 matching 约束下 non-blocking 物理网络上进行虚拟网络映射时，与斯特纳树的变种有一定关系。斯特纳树描述如下：

INSTANCE：图 $G=(V,E)$，边的权值函数 $w(e)\in Z_0^+(\forall e\in E)$，$V$ 的子集 R，正整数 B

QUESTION：是否存在 G 的子树包含所有 R 中的节点，且该树的边的权值之和不超过 B?

该问题可以从 NP 完全问题 XC3 多项式(Karp)规约得到，因此是 NP 难的。

后文将使用的该问题的变种——定点和边均有权值的情况。

(3)最多不相交路径(MAXIMUM DISJOINT CONNECTING PATHS)及其变种

MDCP问题是网络流问题中的经典问题，在分析与证明 embedding 约束条件下虚拟网络映射(模式 embedding)问题时会用到。这里给出简单介绍：

INSTANCE 给定多重图 $G=(V,E)$，定点对集合 $T=\{(s_1,t_1),(s_2,t_2),...,(s_k,t_k)\}$；

SOLUTION 一个由连接 T 中部分节点对的边不相交路径组成的集合；

MEASURE T 中被边不相交路径连接的节点对数目。

该优化问题已存在近似度为 min$\{\sqrt{|E|},|E|/OPT\}$ 的近似算法,但是该问题已被证明不可能存在 $O(2^{\log(1-\varepsilon)|V|})(\forall\varepsilon>0)$ 近似度的近似算法。

3 数学建模及复杂度分析

内容提要：第一，对虚拟网络映射问题的应用需求特征进行分类抽象，进行数学建模：建立了图模式模型及相应的两种（语义）约束（matching 和 embedding）；第二，分析并证明模型的几个重要性质；第三，对模型在两种约束下求解的计算复杂度进行分析与证明；第四，列出在图模式模型下，针对虚拟网络映射问题的几个优化问题，并对优化问题的复杂度进行分析与证明

3.1 虚拟网络映射数学建模：图模式模型

3.1.1 虚拟网络映射示例

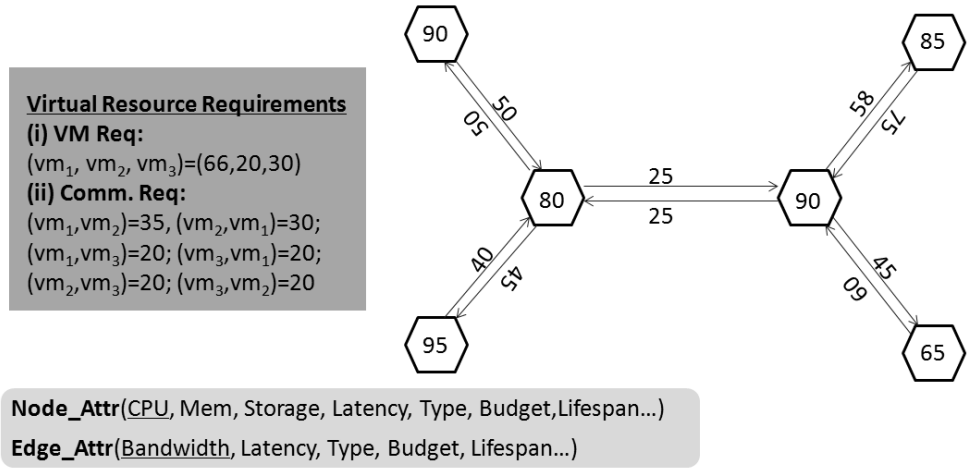

图 3.1: 虚拟网络映射实例

图3.1是一个虚拟网络映射示例。图中右边的图表示物理网络(Substrate Network，简称 SN)，左边表示用户的虚拟网络需求，包括节点资源需求和链路资源及 QoS 需求。如图中所示，节点具有属性 CPU，内存，磁盘，延时，类型，费用，生存周期等，链路具有属性带宽，延时，类型，费用，生存周期等。这些属性对于物理网络(SN)而言，表示的是相应的空闲可利用的资源容量或固有属性，如空闲 CPU 容量、空闲内存容量等；对于虚拟网络(Virtual Network，简称 VN)而言，表达的是用户的虚拟网络资源请求或 QoS 要求，如用户所需的虚拟机节点的 CPU 分配量下限，两虚拟机之间

带宽下限，两虚拟机之间延时的上限等。为了表达清晰，图3.1中节点和链路（节点对）上的数字仅仅只取了属性中的 CPU 和 Bandwidth[3]。

前面绪论中已经提到，目前网络虚拟化相关项目[4] 中，借助于虚拟化技术和虚拟机的支持，节点虚拟化比较普遍，链路虚拟化有部分项目提供一定程度的支持。即节点上虚拟机的资源分配时完全隔离的，对于链路而言，不同虚拟网络之间在公用物理网络链路时，所分配到的链路资源有两种可能，一种是与其他虚拟网络共享的,如 PlanetLab，Darwin 等，还有一种是独占所分配的链路资源，即物理网络链路资源支持虚拟化，如 X-Bone，Genesis 等。

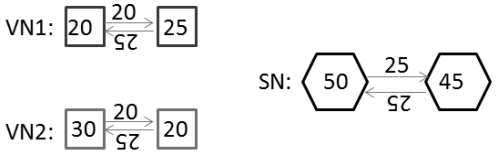

图 3.2: 链路虚拟化特点示例

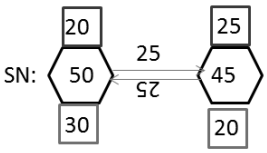

图 3.3: 无链路虚拟化的虚拟网络映射

对于仅支持节点虚拟化的项目及应用，往往关键资源在于节点资源，链路资源可以重用。这种应用常见的有虚拟机集群、虚拟主机等。在这种应用场景下，虚拟网络节点之间的通信往往并非连续的，或者不要求连续的性能保证。图3.2 能体现出链路虚拟化与没有链路虚拟化的区别。在无链路虚拟化时，VN_1 和 VN_2 能同时映射到右边的物理网络 SN 上去，如图3.3所示，但是在支持链路虚拟化时，由于一旦物理网路中的带宽被分配给某一个虚拟网络后，其他虚拟网络只能使用剩余的链路带宽，这时 VN_1 与 VN_2 只能有一个能被映射到 SN 上。同样，对于图3.1中的示例而言，当需要支持链路虚拟化时，左边用户需求对应的虚拟网络不能被映射到右边的物理网络上，而对于仅有节点虚拟化的情况，则可以映射。

事实上，支持链路虚拟化是技术发展的趋势，但是目前还没有达到普及应用的程度。链路虚拟化能够使得各个虚拟网络之间的完全隔离独立，但是一定程度上会造成物理网络链路资源的空闲浪费（在虚拟机之间通信并非连续或密集的情况下）。因此，更进一步的，如果能实现自适应性的链路虚拟化，就不仅能为虚拟网络提供网

[3]现有网络大都是全双工，因此用有向图；对于单工或上下行速率相同的情形，也能表达
[4]见图1.1

络级别的隔离功能，同时还能提高链路的资源利用率。从理论建模上来说，这也是一个新的挑战，是本文可扩展进一步研究的方向之一。

3.1.2 基本数学模型

首先，分析上节中物理网络和用户提出的虚拟网络请求中，网络节点和链路的（资源、QoS）属性往往可以分为下界(Lower Bound)、上界(Upper Bound)，以及固有属性（如 Type、lifespan 等）。因此在位虚拟网络映射问题建立数学模型过程中，我们而已将物理网络、虚拟网络节点和链路的属性抽象成以上三个代表属性，这里我们首要考虑的是下界，比如虚拟网络节点的 CPU 资源请求下界、链路的带宽请求下界，物理网络节点 CPU 资源空闲量下界、链路的带宽空闲量下界等。

基于对虚拟网络映射问题中涉及到的属性的抽象与划分，下面给出虚拟网络映射问题的基本数学模型的描述，及模型的语法部分（语义约束部分见下一小节）。

图模式模型：

定义 3.1 **基图*(Base Graph)***: 一个物理网络*(SN)*在图模式模型中表示为一个基图，即一个四元组 $G_B = (V_G, E_G, A_{V_G}, A_{E_G})$，其中 V_G 是物理网络节点的集合；E_G 是物理网络边的集合；A_{V_G} 是物理网络节点属性向量，该向量中每个元素是一个二元组，分别表示该元素对应节点的下界属性*(Lower Bound)*和上界属性*(Upper Bound)*，分别表示为 $A_{V_G}(v_0)_{.L}$ 和 $A_{V_G}(v_0)_{.U}(\forall v_0 \in V_G)$；$A_{V_E}$ 是物理网络边的属性向量，定义类似于 A_{V_G}，分为下界属性 $A_{E_G}(e)_{.L}$ 和上界属性 $A_{E_G}(e)_{.U}(\forall e \in V_P \times V_P)$.

□

定义 3.2 **模式*(Pattern)***: 一个用户提出的虚拟网络请求*(VN Request)*可用一个模式表示。一个模式是一个三元组 $P = (V_P, A_{V_P}, M)$，其中 V_P 是用户提出的虚拟网络请求中虚拟机节点集合，A_{V_P} 是虚拟机节点属性向量，该向量中每个元素是一个二元组，分别表示该元素对应的虚拟机节点资源请求属性中的下界属性*(Lower Bound)*和上界属性*(Upper Bound)*，分别表示为 $A_{V_P}(v_0)_{.L}$ 和 $A_{V_P}(v_0)_{.U}(\forall v_0 \in V_G)$；$M$ 是一个

$|V_P| \times |V_P|$ 的矩阵，对该矩阵中的任一项 (i,j)*(*其中 $i,j \in V_P$*)*，$M[i,j]_{.L}$ 和 $M[i,j]_{.U}$ 分别表达虚拟机（节点）对 (i,j) 的用户资源及 *QoS* 需求属性中的下界和上界。

□

为了表述简单，后文中，如果没有特殊说明，图模式模型中节点和节点对的上、下界属性均为定义在实数域上的单一属性。事实上，对于上、下界属性是向量的情况，后文中相关定义可类推，习惯结论类似可证。

从上面图模式模型的基本（语法）描述来看，物理网络实质上是用一个带权（节点和链路）有向图来描述的。对于单工或上下行流量完全同步相同的物理网络来说，仍可以用这种带权（Lower Bound 和 Upper Bound）来描述，只需令该有向图中每个节点对的两条边属性值始终同步相等即可。对于虚拟网络，确切来说，是用户虚拟网络请求，模型使用三维矩阵 M 来描述任意节点对之间的资源需求及 QoS 要求属性（Lower Bound 和 Upper Bound）。这种对虚拟网络（请求）的描述较一般用图来建模更贴近实际应用需求，特别是对于数据中心网络或者是 Geni 这种用户或应用自建虚拟网络的网络虚拟化场景而言。因为用户并不关心虚拟机之间的拓扑连接，用户关注的只是虚拟机被分配到的资源以及虚拟机之间通信的 QoS，之于虚拟机之间的拓扑连接，应该有基础设施提供商在进行虚拟网络映射（部署）过程中来决定。

对图3.1中的例子而言，使用图模式模型来描述其虚拟网络（请求）如下:

$$P = (\{vm_1, vm_2, vm_3\}, ((66,+\infty),(20,+\infty),(30,+\infty)), \begin{bmatrix} 0 & 35 & 20 \\ 30 & 0 & 20 \\ 20 & 20 & 0 \end{bmatrix})$$

这里属性矩阵和向量中的 $+\infty$ 表示相关属性项（本示例中是所有的上界属性）为空。

3.1.3 两种语义约束：matching，embedding

前面提到，由于网络虚拟化中虚拟化层次的不同，即节点虚拟化和链路虚拟化的应用程度不同，造成有两种不同的对待虚拟网络映射请求中链路的下界属性的方法。因此，图模式模型中也给出两种相应的语义约束：matching 和 embedding。

定义 3.3 ***Matching(▷)***: 给定一个模式 $P = (V_P, A_{V_P}, M)$ 和一个基图 $G = (V, E, A_V, A_E)$，我们称 G 上存在一个 P 的 *matching*（或称 P 可以被 *match* 到 G 上），当存在单射 $f_V : V_P \to V$，和函数 $p : f_V(V_P) \times f_V(V_P) \to \texttt{PathSet}(f_V(V_P), f_V(V_P))$（其中 $\texttt{PathSet}(f_V(i), f_V(j))$ 是指基图中由连接 $f_V(i)$ 和 $f_V(j)$ 的所有路径组成的集合），满足

1. $\forall i \in V_P, A_{V_P}(i)_{.L} \leq A_V(f(i))_{.L}$
2. $\forall i, j \in V_P, M(i,j)_{.L} \leq A_{p(f_V(i),f_V(j))_{.L}}$，其中 $A_{p(f_V(i),f_V(j))_{.L}} = \min\{A_E(e)_{.L} | \forall e \in p(f_V(i), f_V(j))\}$，表示 *Base Graph* 中路径 $p(f_V(i), f_V(j))$ 的下界属性（*Lower Bound*）。

如果模式 P 能被匹配*(match)*到基图 G 上，则用 $P \rhd_{(f,p)} G$ 表示，在不引起歧义情况下常简写为 $P \rhd G$。

□

Matching 语义约束适用于仅支持节点虚拟化的网络虚拟化应用，如 iVIC，vLab，Planetlab 等。

定义 3.4 ***Embedding(⊵)***:给定模式 $P = (V_P, A_{V_P}, M)$ 和基图 $G_B = (V, E, A_V, A_E)$，我们称 G 上存在一个 P 的 *embedding*（或称 P 可以被 *embed* 到 G 中），当存在单射 $f_V : V_P \to V$，和函数 $p : f_V(V_P) \times f_V(V_P) \to \texttt{PathSet}(f_V(V_P), f_V(V_P))$（其中 $\texttt{PathSet}(f_V(i), f_V(j))$ 是指基图中由连接 $f_V(i)$ 和 $f_V(j)$ 的所有路径组成的集合），满足

1. $\forall i \in V_P, A_{V_P}(i)_{.L} \leq A_V(f(i))_{.L}$
2. $\forall i, j \in V_P, M(i,j)_{.L} \leq A_{p(f_V(i),f_V(j))_{.L}}$，其中 $A_{p(f_V(i),f_V(j))_{.L}}$ 定义与 *matching* 中相同。
3. $$\sum_{\forall (i,j) \in V_P \times V_P : (s,t) \in p(f_V(i), f_V(j))} M[i,j]_{.L} \leq A_{E.L} \qquad (\forall (s,t) \in E)$$

如果模式 P 能被嵌入*(embed)*到基图 G 上，则用 $P \trianglerighteq \rhd_{(f,p)} G$ 表示，在无歧义情况下常简写为 $P \trianglerighteq G$。

□

Embedding 语义约束适用于同时支持节点虚拟化和链路虚拟化（Networking 层）的应用，如 X-Bone 等。

3.2 模型性质分析

图模式模型具有若干性质，本节将给出这些性质。这些性质有助于我们理解虚拟网络映射问题的本质，有利于对相关问题进行复杂度分析与算法设计。

命题 3.5 任意给定模式 $p = (V_P, A_{V_P}, M)$，P 的初始实例 $G_P = (V_{G_P}, E_{G_P}, A_V, A_E)$ 是非多重图。

定义 3.6 实例*(Instantiation)*:给定一个模式 $P = (V_P, A_{V_P}, M)$，我们称图 $G_P = (V_{G_P}, E_{G_P}, A_V, A_E)$ 是 P 的一个实例*(instantiation)*，当且仅当

1. $V_{G_P} = V_P$；
2. $A_V = A_{V_P}$；
3. 存在函数 $p : V_P \times V_P \to \texttt{PathSet}(V_P, V_P)$（即函数 p 将 P 中节点对属性矩阵 M 的每个项 (i, j) 映射到在 G 中连接节点 i、j 的一条路径），同时满足 $M(i,j)_{.L} \leq A_p(p(i,j))_{.L}$，其中 $A_p(p(i,j))_{.L} = \min\{A_E(e)_{.L} | \forall e \in p(i,j)\}$，表示路径 $p(i,j)$ 的下界属性（*lower bound*）；

$P \ltimes G_P$ 表示 G_P 是 P 的一个实例。称以 P 中 M 为邻接矩阵的图 $G_0 = (V_P, E_0, A_V = A_{V_P}, A_E = M)$ 为 P 的初始实例*(initial instantiation)*；

□

定义 3.7 等价关系*(Equivalence)*：给定一个模式 $p = (V_P, A_{V_P}, M)$，定义 $Mod(P) = \{G | P \ltimes G\}$. 对于两个模式 P_1 和 P_2，我们称 $P_1 \equiv_{.L} P_2$ 当且仅当 $Mod(P_1) = Mod(P_2)$.

□

定义 3.8 增广模式*(Augment Pattern)*：给定一个模式 $P=(V_P,A,M)$，定义 P 的增广模式为 $P_{Aug}=(V_P,A,M_{Aug})$，且满足*:*

1. $P_{Aug}\equiv_{.L}P$

2. $\forall(i,j)\in M_{Aug},M_{Aug}(i,j)_{.L}=\max\{M'(i,j)_{.L}|\forall P'=(V_P,A,M'),P'\equiv_{.L}P\}$

简单来说，P 的增广模式 P_{Aug} 是与 P 等价，且所有其节点对的下界属性值均是所有与 P 等价的模式中对应节点对下界属性值的最大者。

□

定理 3.9 如果 $P_{Aug}=(V_P,A_{V_P},M_{Aug})$ 是某一模式的增广模式，则 P_{Aug} 对应的初始实例*(initial instantiation)*是一个完全图 $G=(V=V_P,E=V_P\times V_P,A_V=A_{V_P},A_E=M_{Aug})$，且满足：$\forall(i,j)\in E,A_E(i,j)_{.}=\max\{A_p(p_{ij})|\forall p_{ij}\in \texttt{PathSet}(i,j)\}$

□

定义 3.10 偏序关系*(Partial Order)*:给定模式 P_1 和 P_2，称 $P_1\preceq P_2$ 当且仅当 $Mod(P_1)\subseteq Mod(P_2)$.

□

定理 3.11 *:*给定模式 P_0 和基图 G_B，如果 $P_0\rhd G_B$，则 $\forall P\preceq P_0$，$P\rhd G_B$

□

注：以上定义和性质适用于 matching 语义约束，不能直接应用于 embedding 语义约束。

3.3 模型计算复杂度分析

上节中对虚拟网络映射问题进行数学建模，刻画了两种类型的虚拟网络应用，给出了模型的语法描述和语义约束。本节将针对两种语义约束下——matching 和 embedding——模型求解进行复杂度分析。

3.3.1 matching 约束下模型复杂度分析

对于 matching 约束下模型的求解，可以表述为如下判定问题：

问题 3.1 *matching* 约束下模型求解的判定问题描述：

INSTANCE ：模式 $P = (V_P, A, M)$, 基图 $G = (V, E, A_V, A_G)$

QUESTION ：$P \triangleright G$ 是否成立？

□

定理 3.12 问题*3.1*属于 *NP* 完全问题类，即使问题的实例被约束为：模式的初始实例和基图均是 *DAG* 图，且仅仅只考虑模式中 M 中项、基图中的边的下界属性（即忽略节点下界属性的影响），问题仍然是 *NP* 完全问题。

□

（注：简记在不考虑节点属性限制、且模式的初始实例和基图均是 DAG 的情况下的模式 matching 判定问题为**PMEB**问题——pattern matching with edge/entry lower bounds only）

证明：首先，通过给出**PMEB**问题的一个 NP 算法来证明其属于 NP 类。NP 算法简单描述如下：首先猜测一个单射 $f : V_P \to V_G$ 和一个函数 $p : f(V_P) \times f(V_P) \to \texttt{PathSet}(f_V(V_P), f_V(V_P))$，然后根据 $\triangleright$ 定义检测(f,p)是否是一个正确有效的 matching。显然能在多项式时间内检查 f 和 p 是否分别是单射和函数，以及检查 (f, p) 是否是一个 P 在 G 上的一个 matching。因此，**PMEB**是属于 NP 类的。

我们接下来通过构造一个从 X3C 问题（经典的 NP 完全问题）到**PMEB**的 Karp 规约。

X3C 问题的实例描述如下：给定一个有限集 $X = \{x_1, x_2, ..., x_3q\}$，一个由 X 的三元子集组成的集合 $C = \{C_1, C_2, ..., C_n\}$，其中 $C_i = \{x_{i1}, x_{i2}, x_{i3}\}(\forall j \in \{1, 2, 3\}, x_{ij} \in X)$. X3C 问题即是判定 C 是否包含 X 的完美覆盖(*exact cover*)，即是否存在 $C' \subseteq C$ 且 X 中的每个元素 x_i 在 C' 中仅出现一次。

任给一 X3C 问题的实例 I，我们构造出一个**PMEB**问题的实例，即一个模式 $P_I = (V_I, A_P, M_I)$ 和一个基图 $G_I = (V_{G_I}, E_{G_I}, A_{V_G}, A_{E_G}$，使得 P_I 在 G_I 上存在一个 matching 当且仅当实例 I 存在一个完美覆盖(*exact cover*)。

根据 X3C 的实例 I 构造的**PMEB**问题的实例如下：

1. 模式 $P_I = (V_I, A_P, M_I)$ 的定义：

   - $V_I = \{X_{11}^P, X_{12}^P, X_{13}^P, X_{21}^P, ..., X_{q1}^P, X_{q2}^P, X_{q3}^P, C_1^P, C_2^P, ..., C_q^P\}$;
   - $A_P = [0]_{|V_P|\times|V_P|}$(即模式的节点上没有下界属性限制)
   - M_I 定义: $\forall i \in \{1, 2, ..., q\}, M[C_i^P, X_{ij}^P]_{.L} = a$，其中 a 是一个任意的正实数，且 $j \in 1, 2, 3$. 对 M_I 的任意其他项(entry)*ent*, $M_I[ent]_{.L} = 0$. $\forall (i, j) \in V_I \times V_I, M(i, j)_{.U} = 0$.

2. 基图 $G_I = (V_{G_I}, E_{G_I}, A_{V_G}, A_{E_G})$ 定义：

   - $V_{G_I} = \{X_{11}^G, X_{12}^G, X_{13}^G, X_{21}^G, ..., X_{q1}^G, X_{q2}^G, X_{q3}^G, C_1^G, C_2^G, ..., C_n^G\}$;
   - $E_{G_I} = \{(C_i^G, X_{i\{1,2,3\}}) | \forall i \in \{1, 2, ..., q\}\}$;
   - $A_{V_G} = [0]_{|V_G|\times|V_G|}$;
   - $A_{E_G} : E_G \to \{a\}$;

从以上构造过程中, 我们可以发现 $\forall i \in \{1, 2, ..., p\}, j \in \{1, 2, ..., n\}$，$P_I$ 中的 C_i^P 只能被映射到 G_I 的 C_j^G，P_I 的 X_{ik}^P 只能被映射到 G_I 的 $X_{jl}^G(k, l \in \{1, 2, 3\})$，$P_I$ 节点对中的源节点只能被映射到 G_I 中有向边的源节点，目的节点映射规律也同样如此。事实上，P_I 编码了 I 的一个完美覆盖*exact cover*，而 G_I 对 I 进行了编码.

很显然，以上构造过程可以在多项式时间内完成(PTIME)。下面将证明以上构造是一个 Karp 规约过程，即 I' 中 $P_I \rhd G_I$ 当且仅当实例 I 存在完美覆盖(*exact cover*).

一方面，如果 I' 存在一个 matching，即存在一个从 P_I 到 G_I 的单射 f，使得 $P_I \rhd G_I$。则必然有 $\{f(C_i^P) | \forall i \in \{1, 2, ..., q\}\}$ 是 I 的一个完美覆盖(*exact cover*)。因为如果 $\{f(C_i^P) | \forall i \in \{1, 2, ..., q\}\}$ 不是 I 的一个完美覆盖(*exact cover*)，则

有 $card(\{f(X_{ik}^G)|\forall i \in \{1,2,...,q\}, k \in \{1,2,3\}\}) < card(\{X_{ik}^G|\forall i \in \{1,2,...,q\}, k \in \{1,2,3\}\})$，其中 $card(S)$ 表示集合 S 的基数。这与 f 是单射相矛盾。

另一方面，如果 I 存在一个完美覆盖(*exact cover*)，则我们称该完美覆盖(*exact cover*) 同样制订了一个从 P_I 的节点到 G_I 的节点的单射 f。我们使用 $S^C = \{C_{j_k}^G|k \in \{1,2,...,q\} and j_k \in \{1,2,...,n\} \subseteq C = \{C_1, C_2, ..., C_n\}$ 表示 I 的完美覆盖(*exact cover*)。则单射 f 定义如下：

- 对 $\{C_1^P, C_2^P, ..., C_q^P\}$ 中的节点 C_i^P，$f(C_i^P) = C_{j_i}^G (\forall i \in \{1,2,...,q\})$；
- 对 P_I 中 $X_{ik}^P (i \in 1,2,...,q, k \in 1,2,3)$，$f(X_{ik}^P) = X_{j_i k}^G$.

显然 f 定义了 P_I 在 G_I 上的一个 matching，即 $P_I \rhd G_I$.

综上，**PMEB**的实例存在一个 matching 当且仅当 NP 完全问题 X3C 的实例具有一个完美覆盖(*exact cover*). 因此**PMEB**问题是 NP 完全问题. □

3.3.2 embedding 约束下模型复杂度分析

对于 embedding 约束下模型的求解，可以表述为如下判定问题：

问题 3.2 *embeding* 约束下模型求解的判定问题描述：

INSTANCE *：*模式 $P = (V_P, A, M)$, 基图 $G = (V, E, A_V, A_G)$

QUESTION *：* $P \trianglerighteq G$ 是否成立？

□

定理 3.13 问题*3.2*属于 *NP* 完全类，即使问题的实例被约束为*:*

(a) 模式对应的初始实例与基图均是 *DAG*，且（模式或基图）仅有边的下界属性的限制；

或者

(b) 模式中的矩阵 M 是对称矩阵，且对应的初始实例与基图均是树；

仍都是 *NP* 完全类问题。

□

（注：记（b）情形下 embedding 问题为**TPE**——Tree-shape pattern embedding problem）

证明：对于（a）情形，其证明与定理3.12的证明类似，任可由 X3C 问题 Karp 规约到（a）情况的 embedding 问题，从而证明其是 NP 完全问题。下面给出（b）情形的复杂度证明。

首先，与前面类似，易知**TPE**问题属于 NP 类的。定理3.12证明中的 NP 算法同样适用。下面将建立一个从六大基本 NP 完全问题中的 PARTITION 问题到**TPE**的 Karp 规约。

回顾 PARTITION 问题，其任意一实例 I 可描述为：一个有限集 $C = \{a_1, a_2, ..., a_n\}$，以及一个作用于 C 中每个元素的权值函数 $s(a_i) \in \mathbb{Z}^+$。PARTITION 问题既是需要判定实例 I 是否存在一个分割，及是否存在 $C' \subseteq C$，使得 $\sum_{a \in C'} s(a) = \sum_{a \in C - C'} s(a) = \frac{A_0}{2}$，其中 $A_0 = \sum_{i=1}^{n} s(a_i)$。

下面将根据 PARTITION 的任意实例 I，构造一个**TPE**问题的实例 I'，即一个模式 $P_I = (V_I, A_P, M_I)$ 和一个基图 $G_I = (V_{G_I}, E_{G_I}, A_{V_G}, A_{E_G}$，使得 P_I 在 G_I 上存在一个 embedding 当且仅当实例 I 存在一个分割。

根据 PARTITION 的实例 I 构造的**TPM**问题的实例如下：

1. 模式 $P_I = (V_I, A_P, M_I)$ 的定义：

   - $V_I = \{A_1, A_2, ..., A_n, T_P\}$;
   - A_P 定义：$\forall A_i \in V_I, A_P(A_i)_{.L} = z_i$，$A_P(T_P) = z_{i+1}$。其中 $\forall i, j \in \{1, 2, ..., n\}, i > j, z_i > z_j (z_i, z_j \in Z^+)$.
   - M_I 定义: $\forall i \in \{1, 2, ..., n\}, M[A_i, T_P]_{.L} = s(a_i)$，对 M_I 的任意其他项(entry)*ent*, $M_I[ent]_{.L} = 0$. $\forall (i, j) \in V_I \times V_I, M(i, j)_{.U} = 0$.

2. 基图 $G_I = (V_{G_I}, E_{G_I}, A_{V_G}, A_{E_G})$ 定义：

- $V_{G_I} = \{A_1^L, A_2^L, ..., A_n^R, A_1^R, A_2^R, ..., A_n^R, T_G\}$;
- $E_{G_I} = \{(A_i^L, A_{i+1}^L), |\forall i \in \{1, ..., n-1\}\} \cup \{(A_n^L, T_G)\} \cup \{(A_{i+1}^H, A_i^H), |\forall i \in \{1, ..., n-1\}\} \cup \{(T_G, A_n^H)\}$;
- A_{G_I} 定义：$\forall A_i^L, A_i^H \in V_I, A_P(A_i^L)_{.L} = A_P(A_i^R)_{.L} = z_i$，$A_P(T_G) = z_{i+1}$。
- A_{E_G} 定义：$A_E(T_G, A_n^L) = A_E(T_G, A_n^R) = \frac{A_0}{2}$，对于 E_{G_I} 中的任意其它边 e，$A_E(e)_{.L} = +\infty$.

由以上构造知，I' 中 P_I 的节点 V_I 中，T_P 必然被映射到 G_I 中的 T_G，$\forall i \in \{1, ..., n\}$，$A_i$ 必然被映射到 G_I 中的 A_i^L 或 A_i^H.

显然该构造能在多项式时间内完成，下面说明其是一个 Karp 规约，即 I 存在一个分割当且仅当 I' 存在一个 embedding。一方面，如果 PARTITION 的实例 I 存在分割，即存在 $C' \subseteq C$，使得 $\sum_{a \in C'} s(a) = \sum_{a \in C - C'} s(a) = \frac{A_0}{2}$，则将 I' 中分别于 C 与 C' 对应的节点分别映射到 G_I 中的 $\{A_1^L, A_2^L, ..., A_n^R\}$ 和 $\{A_1^R, A_2^R, ..., A_n^R\}$ 即为 I' 的一个 embedding。

另一方面，如果 I' 存在一个有效的 embedding，则 I 中与 I' 中的 P_I 分别被映射到 $\{A_1^L, A_2^L, ..., A_n^R\}$ 和 $\{A_1^R, A_2^R, ..., A_n^R\}$ 上的节点对应的元素作为 C' 和 $C - C'$ 中的元素，即成为 I 的一个分割。

综上，**TPE**的实例存在一个 embedding 当且仅当 NP 完全问题 PARTITION 的实例具有一个分割。因此**TPE**问题是 NP 完全问题。 □

3.4 组合优化问题

3.4.1 相应优化问题规范描述

问题 3.3 最小实例*(instantiation minimization，IM)*

INSTANCE *pattern* $P_0 = (V_{P_0}, A_0, M_0)$

SOLUTION *pattern* $P = (V_P, A, M) \equiv_{.L} P_0$，$P$ 及 P 的初始实例 G_P

MEASURE G_P 的边的上界属性之和，亦即：$\sum_{\forall (i,j) \in V_P \times V_P : M[i,j]_{.L} > 0} M[i,j]_{.U}$

GOAL *MINIMIZATION*

□

相应的变种问题 1：当基图和模式的初始实例均是无向图时，求给定模式的最小实例。

相应的变种问题 2：（考虑节点延时时的总最小延时问题）即求 $P \equiv_L P_0$，使得 G_P 的边的上界属性和中间传输节点的上界属性之和最小。

问题 3.4 最小 *matching:*

INSTANCE 模式 $P = (V_P, A, M)$, 基图 $G = (V, E, A_V, A_E)$

SOLUTION P 在 G 上的 *matching:* (f, p),其中 f 是从 V_P 到 V 的单射，p 是从 $V_P \times V_P$ 到 $PathSet(f(V_P) \times f(V_P))$ 的函数

MEASURE $\sum_{\forall e \in E: \exists i,j \in V_P, e \in p(f(i),f(j))} A_E(e)_{.U}$

GOAL *MINIMIZATION*

□

相应的变种问题：（考虑节点延时时的总最小延时问题）即求 P 的一个 matching，使得在考虑传输节点的上界时 matching 的总上界属性值之和最小。

问题 3.5 最小 *embedding*：

INSTANCE 模式 $P = (V_P, A, M)$, 基图 $G = (V, E, A_V, A_E)$

SOLUTION P 在 G 上的 *embedding:* (f, p),其中 f 是从 V_P 到 V 的单射，p 是从 $V_P \times V_P$ 到 $PathSet(f(V_P) \times f(V_P))$ 的函数

MEASURE $\sum_{\forall e \in E: \exists i,j \in V_P, e \in p(f(i),f(j))} A_E(e)_{.U}$

GOAL *MINIMIZATION*

□

问题 3.6 最大冗余带宽 *Embedding:*

INSTANCE 模式 $P = (V_P, A, M)$, 模式 $G = (V, E, A_V, A_E)$

SOLUTION P 在 G 上的 *embedding:* (f, p),其中 f 是从 V_P 到 V 的单射，p 是从 $V_P \times V_P$ 到 $\texttt{PathSet}(f(V_P) \times f(V_P))$ 的函数

MEASURE $\sum_{\forall (i,j) \in V_P \times V_P} A_p(p(f(i), f(j))_{.U}$

GOAL *MAXIMIZATION*

□

3.4.2 优化问题复杂度

下面给出以上优化问题及相关变种问题的计算复杂度结论。相关证明在第 4 章、第 5 章中解决相应问题时详细给出。

定理 3.14 问题*3.3*对应的判定问题存在 $O(|V_P|^3)$ 多项式时间精确算法。

□

定理 3.15 问题*3.4*对应的判定问题是 *NP* 完全问题，即是模式和基图均是 *DAG*。其变种问题——基图和模式的初始实例均是无向图，且均为树时，求最小 *matching* 仍是 *NP* 难的。

□

定理 3.16 问题*3.5*对应的判定问题是 *NP* 完全问题，即是模式和基图均是 *DAG*。其变种问题——基图和模式的初始实例均是无向图，且均为树时，求最小 *embedding* 仍是 *NP* 完全问题。

□

定理 3.17 问题*3.6*对应的判定问题是 *NP* 完全问题。

□

定理3.14的证明在第 4 章中给出。定理3.15、3.16、3.17的证明在第 5 章中给出。

3.5 本章小结

本章首先总结了网络虚拟化中虚拟网络的特点及不同层次的虚拟化，然后提出一个严格的数学模型（图模式模型）对虚拟网络和虚拟网络映射问题进行建模，给出语法定义和两种语义操作——matching 和 embedding，分别对应为两种不同层次的虚拟化。最后分析并严格证明了基本语义操作的复杂度（均是 NP 难的），并提出了若干后文待分析解决的优化问题。

4 模式最小化

内容提要：本节将主要分析、证明模式最小化问题及其各种扩展问题的计算性质，并设计相应最小化算法，包括最大-最小化方法（增广模式多项式算法和基于此的“伪动规”模式最小化算法）。

注：本节中的最小化均是在 matching 语义下定义的。

4.1 模式最小化问题示例

回顾前面为虚拟网络映射问题的图模式模型中，模式表示用户的虚拟网络需求。例如，在图3.1中，模式表示了用户申请的三个虚拟机 vm_1, vm_2, vm_3 的节点下界属性（$(vm_1, vm_2, vm_3 = (66, 20, 30)$），以及任两节对的下界属性（$(vm_1, vm_2) = 35, (vm_2, vm_1) = 30; (vm_1, vm_3) = (vm_3, vm_1) = 20; (vm_2, vm_3) = (vm_3, vm_2) = 20$）。可以发现，对任意 matching，在其满足了节点 $vm_i (i = 1, 2, 3)$ 的下界属性后，如果其满足了 (vm_1, vm_2) 和 (vm_2, vm_3) 的下界属性，则一定满足 (vm_1, vm_3) 的下界属性。这种现象表明，对于只支持节点虚拟化的虚拟网络映射问题中，用户的需求往往存在冗余（链路）资源请求. 因此在已给的 base graph 中为直接从用户需求得到的模式寻找 matching 的过程中，有些节点对的下界属性的 matching 计算过程是多余的。因此，在基图中为模式寻找 matching 的过程中，首先对其预处理去掉模式里的多余节点对的下界属性要求，能减少冗余的计算。这种预处理即是模式的最小化过程。

4.2 对称模式最小化问题：最大-最小化方法

第 3 章中的问题3.3给出了基本的模式最小化问题的描述。即对于一个给定的模式 $P_0 = (V_{P_0}, A_0, M_0)$，求出一个与其等价的模式 $P = (V_P, A, M)$，使得 P 的初始实例 G_P 中所有边的上界属性之和最小，即最小化

$$\sum_{\forall (i,j) \in V_P \times V_P : M[i,j]_{.L} > 0} M[i,j]_{.U}.$$

定理3.14给出了其复杂度的一般结论——存在多项式时间算法可求出精确解。下面将通过给出该问题的多项式时间算法来证明定理3.14。该多项式时间算法能在 $O(\frac{1}{2}(|V_P|^3 - |V_P|^2))$ 时间内求出给定模式的最小等价模式，是一种最大-最小化方法：先在 $O(|V_P|^3)$ 时间内求出与所给模式等价的最大模式——增广模式（见定义3.8），然后基于增广模式多项式时间内求出最小化模式。

为了便于理解，我们先处理当模式 P 对应的初始实例 G_P 是无向图的情况（此时 P 中的节点对属性矩阵 M 为对称矩阵，我们称这类模式为对称模式），然后扩展到一般化的非对称模式情况（见下节）。

定义 4.1 给定模式 $P = (V_P, A_V, M)$，我们称 P 为对称模式，当 P 的初始实例 $G_P(V_P, E_{G_P}, A_V, A_{G_E})$ 是无向图，即：

- $\forall i, j \in V_P, M[i, j]_{.L} = M[j, i]_{.L}$；
- 在 *matching* $\triangleright_{f,p}$ 作用下有 $p(f(i), f(j))$ 和 $p(f(j), f(i))$ 是无向路径，且 $p(f(i), f(j)) \equiv p(f(j), f(i))$；

□

问题 4.1 对于对称模式 $P = (V_P, A_V, M)$，其初始实例为无向图 $G_P(V_P, E_{G_P}, A_V, A_{G_E})$，最小化问题等价于给定 P 求图 $G_P^{min} = (V_P, E^{min}, A_V, A_{E^{min}}^{min})$，满足 G_P^{min} 中 $\forall i, j \in V_P, \exists p(i, j)$ 连接节点 i 和 j，且 $A_{p(i,j)} = \min\limits_{\forall e \in p(i,j)} A_{E^{min}}^{min}(e)_{.L} \geq M[i, j]_{.L} = M[j, i]_{.L}$。

□

我们称 G_P^{min} 是 P 的最小等价实例，称以 G_P^{min} 为初始实例的模式是 P 的最小化模式，记作 P^{min}（等价的最小模式）。

4.2.1 增广模式的 $O(|V_P|^3)$ 算法

算法**AUGP** 构造了给定模式的一个完全图实例（增广模式对应的初始实例）。在算法运行过程中，每一步通过往当前已构造完的增广模式的初始实例的完全子图中

AUGP:Computing Augment Pattern of a given pattern:
Input

- a pattern $P = (V_P, A, M)$ (symmetric)

Output:

- Augment Pattern of $P_{Aug} = (V_P, A, M')$

//$V_P = 1, 2, ..., |V_P|$
1. $M_1 = M(1, 1), V_1 = \{1\}$
2. for(i=2; i$\leq |V_P|$; i++)
3. $\quad M_i =$ **Update**(M_{i-1})
4. $\quad V_i = V_{i-1} \cup \{i\}$
5. Output $P_{Aug} = (V_P, A, M' = M_i)$

图 4.1: **Main frame of algorithm AUGP**

新增节点，并更新相应节点对之间下界属性，经过 $O(|V_P|)$ 次循环后输出的即是所求的增广模式的初始实例，亦即求出了给定模式的增广模式。

子过程**Update**将一个新节点加入以当前已经计算出的子增广模式中，并更新节点对之间的下界属性。换句话说，其将一个新节点加入当前已经计算出的完全图中，并更新相应边的下界属性。更新过程分为为两部分：

第一，更新引入新节点所新增的边的下界属性。具体过程为：首先将新增边的原始下界属性按从大到小排序，然后根据下界属性定义依次更新（增大或不变）其属性值（具体见算法描述图4.2），使得每一个新引入的边的下界属性值是连接该边两端点的所有路径中下界属性的最大值。

第二，更新原来完全图中的边的下界属性。因为新引入的连接新节点与当前完全图中所有节点的边可能改变已有边的下界属性值。可以证明，对原来完全图中边下界属性值的更新不会影响到上一步中更新完的边的下界属性值。

我们将证明算法**AUGP**能在 $O(|V_P|^3)$ 时间内正确输出增广模式。

定理 4.2 给定任意对称模式 P，算法***AUGP***能在 $O(\frac{1}{2}(|V_P|^3 - |V_P|^2))$ 时间内正确输出 P 的增广模式 P_{Aug}。

□

Update(M'_{m-1})
Input:

- $(m-1)$-dimensional matrix M'_{m-1}
- $|V_p|$-dimensional matrix M of P

Output:

- i-dimensional matrix M_m

1. sort the entries M[m,1],...M[m,m-1] descending.
//denote the descendent entries by M[m,m-1],...M[m,1]
2. for($i = m-1; i > 0; i--$)
3. temp=$M[m,i]$; M'[m,m]=0
4. for($j = m-1; j > i; j--$)
5. if($M'_{m-1}[i,j] > M[m,j]$)
6. temp=max{$M'_{m-1}[i,j]$, temp}
7. $M'_m[m,i] = M'_m[i,m]$=temp
8. for($i = 1; i \leq m-1; i++$)
9. for($j = 1; j < i; j++$)
10. if($M'_m[i,j] < M'_m[i,m]$&&$M'_m[i,j] < M'_m[j,m]$)
11. $M'_m[i,j] = \min\{M'_m[i,m], M'_m[j,m]\}$
12. Output M'_m

图 4.2: **Procedure** Update **of algorithm AUGP**

证明 （**1**）**正确性**:我们将通过以下三条性质来证明算法的正确性：

性质 1: 在第 i^{th} 轮循环执中，记当前已经计算出的完全子图为 K_i，对于每条连接新引入节点与 K_i 中 i 个节点的新边而言，其下界属性值仅受下界属性值比它大其它新边的影响。

性质 2: 在更新 K_i 中 $i \times i$ 条边的下界属性值过程中，$\forall(p,q) \in K_i$，边 (p,q) 的下界属性值均需从三角形 (p, n_{new}, q) 中求出即可（其中 n_{new} 指当前新加入的节点），不会涉及到其他边。也即是说，更新 K_i 中边的下界属性值过程中，K_i 中各条边的更新是相互独立的。

性质 3: 在更新了当前完全图 K_i 中的 $\times i$ 条边的下界属性值后，不需要再次对新引入连接 K_i 和新节点之间的边的下界属性值进行更新。即每次循环中，K_i 中边的更新不会影响已经更新过的新边的下界属性。

以上三条性质成立的原因在于，更新前的 K_i 和更新后的 K_{i+1} 中每条边的下界属性值均是当前所有连接该边两端点的路径的下界属性值的最大值。

（**2**）**复杂度**: 我们用 t_i 表示第 i^{th} 次循环过程中**Update**函数的时间（计算次数）。从图4.2**Update**子过程可以看出 t_i 是更新新引入边的下界属性值所耗时间（t_i^1）与更新 K_{i-1} 中边的下界属性值所耗时间（t_i^2）之和。其中 $t_i^1 = \frac{i(i-1)}{2}$，$t_i^2 = frac(i-1)(i-2)2$。因此，算法**AUGP**的时间复杂度为 $O(\sum_{i=1}^{|V_P|}\{\frac{i(i-1)}{2} + \frac{(i-1)(i-2)}{2}\}) = O(\frac{|V_P|^3}{3} - \frac{{V_P}^2}{2} + \frac{N}{6})$。

□

4.2.2 $O(|V_P|^3)$“伪动规”算法

引理 4.3 给定一个对称模式 $P = (V_P, A_V, M)$，P 的等价最小模式对应的初始实例 G_P^{min} 是树，但是不一定是 P 的初始实例 G_P 的生成树。

□

下面给出对称模式的最小化算法。算法基于引理4.3设计的。主要过程分为两个阶段：首先求出所给模式的增广模式 P_{Aug}；然后针对增广模式，通过”伪动规“算法求解出最小化等价模式。我们将首先给出算法描述，然后证明和分析算法的正确性和复杂度。

在我们深入**APBA**算法的**UpdateTree**函数之前，我们先给出在**UpdateTree**中将使用到的替换边的定义。

定义 4.4 在一个环 C 中，一个边 e 的**替换边**是 C 中具有与 e 相同下界属性且不大于 e 的上界属性的边，用 $Alt_C(e)$ 表示，包含有 e 的替换边的连接 e 的两端点的非 e 路径用 $Alt(e)$ 表示。

□

定义 4.5 对任意模式 $P = (V_P, A, M)$，$G = (V_P, E, A_V = A, A_E)$，$P \ltimes G$，$\forall (s,t) \in E$，我们称 G 中任意连接节点 (s,t) 的路径 $p(s,t)$ 为 (s,t) 的**替代路径**，当且仅当 $p(s,t)$ 满足以下四点：

- 不包含 (s,t);

(APBA)Augment Pattern Based Algorithm:
Input

- a pattern $P = (V_P, A, M)$
  //(undirected/symmetrical: $M(i,j) = M(j,i)$)

Output:

- a minimum graphism graph $G_G = (V_P, E)$ of P

//AGA construct a minimum spanning graph across the augment pattern with greedy strategy **incrementally**.
1.Compute the augment pattern $P_{Aug} = (V_P, A, M_{Aug})$ of P using the algorithm in Fig. 1;
2.Construct a undirected complete graph $G_C = (V_P, V_P \times V_P)$ from P_{Aug}
//$V_P = 1, 2, ..., |V_P|$
3.Tree $T = (V_T, E_T) = (\{1, 2\}, (1, 2))$
4.for $(i = 3; i \leq |V_P|; i++)$
5. **UpdateTree**(T,i,G_C)
6.Output T

图 4.3: **Algorithm(APBA)**

- $p(s,t)$ 中无环；
- $\forall (i,j) \in p(s,t)$， $M[i,j] \geq M[s,t]$;
- $\exists (p,q) \in p(s,t), M[p,q] = M[s,t]$;

我们称 (p,q) 为 (s,t) 的**替代边**。(s,t) 的替代路径记作 $Alt^p(s,t)$，替代边记作 $Alt^e(s,t)$。如果 $\sum_{\forall e \in Alt^p(s,t) \cup (G-(s,t))} A_E(e) \leq \sum_{\forall e \in G - Alt^e(s,t)} A_E(e)$，则称 $Alt^p(s,t)$ 为 (s,t) 的**较小替代路径**，用 $mAlt^p(s,t)$ 表示，相应的 $Alt^e(s,t)$ 为 (s,t) 的**较小替代边**，用 $mAlt^e(s,t)$ 表示。

□

引理 4.6 任给一对称模式 $P = (V_P, A_V, M)$，P 的增广模式 P_{Aug} 的初始实例是一个由 $\frac{|V_P|(|V_P|-1)(|V_P|-2)}{6}$ 个等腰三角形组成的完全图。且每个等腰三角形的底的上界属性值不小于腰的上界属性值。

□

根据引理4.6，在 P 的初始实例中，对任意等腰三角形（由任意三个节点连接而成）中的两条腰而言，其各自是另一条腰的替代边。在一棵树中，没有任何边具有替代边。

UpdateTree(T,m,G_C):
Input

- Tree $T = (1, ..., m-1, E_T)$
- new node m
- Complete Graph G_C corresponding to the augment pattern P

Output:

- $T = (1, ..., m, E_T \cup m)$ such that the total upper bound of T is minimized.

//**UpdateTree** procedure add the new node m to T with edges inG_C, and then delete alternative edges to gain a minimum spanning tree *without break the restriction of lower bounds*.
// for an equilateral triangle, we regard the edge with smallest upper bound as the base edge, the other two are waists.

1. connect node m and 1 with edge $(1, m)$ of G_C
2. for($i = 2; i \leq m-1; i++$)
3. connect m and i with edge (i, m) of G_C
4. if i and m are in an isoceles triangle
5. delete the waist with larger upper bound
//delete $mAlt_{\triangle}$(waist)
6. else //there is one cycle C
7. delete $mAlt_C(waist)$ in the cycle C**(if it exists)** or (i, m)(if no $mAlt_C(waist)$)
8. output T

图 4.4: **Procedure** UpdateTree **of Algorithm(APBA)**

定理 4.7 给定一对称模式 $P = (V_P, A, M)$，***APBA***算法能在 $O(|V_P|^3)$ 时间内计算出 P 的最小化模式 $P^{min} = (V_{P^{min}}, A^{min}, M^{min})$，即 $\forall P' = (V_{P'}, A', M'), P \equiv_{\cdot} P'$, $\sum_{\forall i,j \in V_{P^{min}}} \{M^{min}[i,j]\} \leq \sum_{\forall i',j' \in V'} \{M[i',j']\}$.

□

证明：（1）正确性：由引理4.6，P 的增广模式 P_{Aug} 的初始实例 $G_{P_{Aug}}$ 中的任意三点组成一个等腰三角形，且其腰的下界属性值不大于底边的下界属性之。同时，由引理4.3知，任何对称模式的最小模式对应的初始实例一定是树。**APBA**算法正是基于这两条性质，通过对给定模式的增广模式的初始实例进行”去替代边（$Alt_{\triangle}(waist)$ 或 $Alt_C(waist)$）“处理得到最小化模式。具体来讲，对于每条从 $G_{P_{Aug}}$ 中删掉的边 e（连接顶点 i 和 j），均存在一条（下界属性值不小于 e 的）替代路径连接 i 和 j。因此，以所称成的树为初始实例的模式与 P 等价，即算法输出的解是正确的。

（2）最小性：我们首先证明模式最小化问题具有”伪最优子结构“性质，即：$\forall e \in E_{P_{Aug}}$($P_{Aug}$ 的初始实例的边的集合)，如果 e 不在以 $K_i(i \geq 2)$为初始实例的模式 P_i 经过最小化后对应的初始实例 T_i 中（K_i 定义见定理4.2证明），则 $e \notin T_{i+1}$。

我们利用反证法来证明模式最小化具有以上”伪最优子结构“性质。即对假设存在边 $(i,j) \in K_i$，$(i,j) \notin T_i$，在引入新节点 $i+1$ 后，$(i,j) \in T_{i+1}$。

一方面，如果对任意 T_i 中的边 e，e 在 T_{i+1} 中，则在 $T_i \cup (i,j) \subset T_{i+1}$ 中存在环。矛盾，假设不成立，模式最小化问题具有“伪最优子结构”。

另一方面，如果 $\exists (i',j') \in T_i$ 且 $(i',j') \notin T_{i+1}$:

1. 如果在 T_{i+1} 中连接节点 i' 和 j' 的较小替代路径（相对 T_i 中的边 (i',j')）$mAlt_{T_{i+1}}(i',j')$ 不包含 (i,j)，则在 $mAlt_{T_{i+1}}(i',j') \cup T_i \cup (i,j) \subseteq T_{i+1}$ 中存在环路。矛盾。

2. 如果 $mAlt_{T_{i+1}}(i',j')$ 包含 (i,j)，由于 $mAlt_{T_i}(i,j) \subseteq T_i$，我们有 $p(i,i') \cup (i',j') \cup p(j',j) \cup (j,i)$ 是一个环，且子树 $p(i,i') \cup (i',j') \cup p(j',j)$ 中所有边的上界属性之和不大于 $p(i',i) \cup (i,j) \cup p(j,j')$ 中所有边的上界属性之和。这与 T_{i+1} 是以 K_{i+1} 为初始实例的模式最小化后的初始实例这一事实矛盾。

以上两方面表明假设不成立，因此模式最小化问题具有前面所述的“伪最优子结构”。

基于”伪最优子结构“性质，我们很容易证明算法**APBA**能正确输出给定 P 最小化后的 P^{min}。事实上可以证明第 i^{th} 次执行函数**UpdateTree**总能输出以 K_i 为增广实例的模式最小化后的初始实例。

首先我们证明在第 i 次执行**UpdateTree**中，每一次通过 for 循环去掉的边均不可能出现在 T_i 中。假设第 j 次执行 for 循环去掉的边 (p_j, q_j) 在 T_i 中，由于 T_i 是树，其任意子图也均是树，因此在 T_i 的子图 $T_i^S = (V_{T_{i-1}} \cup \{i\}, E_{T_{i-1}} \cup \{(i,k)|k = 1, ...j\} - \{e_k|k = 1, ...j\}, A_V, A_E)$（其中 e_k 表示在第 k 次执行 for 循环时去掉的边）中，存在 (p_j, q_j) 的一条较小替代路径 $mAlt^p(p_j, q_j) \in T_i^S$，记作 $Alt^{p \in T_i^S}(p_j, q_j)$，其对应的较小替代边 $mAlt^{e \in T_i^S}(p_j, q_j)$ 不在 T_i 中。则我们得到一个新的与以 K_i 为初始实例的模式等价且初始实例为树的模式，$T_i' = (V_i' = V_{T_i}, E_i' = (E_{T_i} - (p_j, q_j)) \cup mAlt^{p \in T_i^S}(p_j, q_j) = (E_{T_i} - (p_j, q_j)) \cup mAlt^{e \in T_i^S}(p_j, q_j), A_V', A_E')$，且 $\sum\limits_{\forall e \in E_i'} A_E'(e) \leq \sum\limits_{\forall e \in E_i} A_E(e)$。以上结论成立的原因有二：

- T_i' 树且其对应的模式（以 T_i' 为初始实例）与以 K_i 为初始实例的模式等价：因为对于任意 $e \in E_{T_i} - E_{T_i^S}$，若 $mAlt^e(e) = (p_j, q_j)$，有 $mAlt^{e \in T_i^S}(p_j, q_j)$ 是 e 的较小替代边且上界属性值不大于 (p_j, q_j)；若不存在 $e \in E_{T_i} - E_{T_i^S}$，若 $mAlt^e(e) = (p_j, q_j)$，则 $mAlt^{ei}(e) = mAlt^{ei'}(e)$。又由于 for 循环执行了 $i-2$ 遍，因此 T_i' 是树；

- T_i' 的所有边的上界属性之和小于 T_i 所有边的上界属性之和：因为 $M(mAlt^{e \in T_i^S}(p_j, q_j)) \leq M(p_j, q_j)$；

这与 T_i 是最小化后模式的初始实例矛盾，因此假设不成立。因此在第 i 次执行**UpdateTree**中，每一次通过 for 循环去掉的边均不可能出现在 T_i 中。故在第 i 次执行**UpdateTree**时总能输出 T_i（以 $\{1, 2, ...i\}$ 为顶点集合的 P_{Aug} 的初始时实例的子图）。因此在第 n 次执行**UpdateTree**是输出正确的 T_n，即 P 最小化后的初始实例。

（3）时间复杂度：在算法**APBA**的主函数中，计算给定模式 P 的增广模式 P_{Aug} 能在 $O(n^3)$ 时间内完成（见定理4.2证明），第 i^{th} 次执行函数**UpdateTree**可在 $O(\frac{i(i-1)}{2})$ 时间内完成，因此算法**APBA**的复杂度为 $O(|V_P|^3 + \sum\limits_{i=1}^{|V_P|} \frac{i(i-1)}{2}) = O(|V_P|^3)$。

□

4.3 非对称模式上的最小化问题

前文所处理的模式最小化问题均是建立在模式对应的初始实例是无向图的前提下的。回顾前面定义的对称模式，其包含两方面的约束：第一，节点对属性约束；对称模式中，节点对 i,j 对应的两个节点对属性项 $M[i,j]$ 与 $M[j,i]$ 相等。
第二，实例化约束；对称模式中，对任意两个节点 i、j，$M[i,j]$ 与 $M[j,i]$ 所描述的用户对虚拟机的需求及在映射到物理网络后对应的物理实体是同一条物理网络路由路径，是等价的。简而言之，对称模式的（初始）实例被限定为无向图而非 (i,j) 与 (j,i) 属性相等的有向图。
与之对应，将没有以上两条约束的一般化的模式（在图模式模型中定义的模式）称之为非对称模式。（注意，这里的“对称”与“非对称”不同于矩阵或图的“对称”概念）。

本小节主要解决一般性的非对称模式的最小化问题。其主要思想与对称模式类似，均是通过去除较小替代边求出增广模式的初始实例最小化后的图，以该图为初始实例的模式即为所求最小化模式。下面给出具体说明与处理方法的正确性和最优性证明。

非对称模式最小化问题与对称情况下描述类似：

问题 4.2 对于非对称模式 $P=(V_P,A_V,M)$，其初始实例为有向图 $G_P(V_P,E_{G_P},A_V,A_{G_E})$，最小化问题等价于给定 P 求图 $G_P^{min}=(V_P,E^{min},A_V,A_{E^{min}}^{min})$，满足 G_P^{min} 中 $\forall(i,j)\in V_P\times V_P,\exists p(i,j)$ 从 i 到 j，且 $A_{p(i,j)}=\min\limits_{\forall e\in p(i,j)} A_{E^{min}}^{min}(e)_{.L}\geq M[i,j]_{.L}$。

□

4.3.1 非对称模式相关性质

首先，我们将上文中“替代路径”的说法推广到一般的非对称模式中。

定义 4.8 对任意模式 $P=(V_P,A,M)$，$G=(V_P,E,A_V=A,A_E)$，$P\rtimes G$，$\forall(s,t)\in E$，我们称 G 中任意以 s 为源点、t 为终点的的有向路径 $p(\widehat{s,t})$ 为 $(\widehat{s,t})$ 的**替代路径**，当且仅当 $p(\widehat{s,t})$ 满足一下三点：

- 不包含 (s,t);
- $p(\widehat{s,t})$ 中无环；
- $\forall(i,j) \in p(\widehat{s,t})$，$M[i,j] \geq M[s,t]$;
- $\exists(p,q) \in p(\widehat{st}), M[p,q] = M[s,t]$;

我们称 (p,q) 为 (s,t) 的**替代弧**。记弧 (s,t) 的替代路径记作 $Alt^p(s,t)$，替代弧记作 $Alt^{arc}(s,t)$。如果 $\sum_{\forall(i,j)\in Alt^p(s,t)\cup(G-(s,t))} A_E(i,j) \leq \sum_{\forall(i,j)\in G-Alt^{arc}(s,t)} A_E(i,j)$，则称 $Alt^p(s,t)$ 为 (s,t) 的**较小替代路径**，用 $mAlt^p(s,t)$ 表示，相应的 $Alt^{arc}(s,t)$ 为 (s,t) 的**较小替代弧**，用 $mAlt^{arc}(s,t)$ 表示。

□

下面两个定理给出了非对称模式最小化的基本思想。

引理 4.9 对任意模式 $P = (V_P, A, M)$，记 P 最小化后（记作 P_{min}）的初始实例为 $G_{min} = (V_P, E, A_V = A, A_E)$，则 $\forall(i,j) \in E$，G_{min} 中不存在 (i,j) 的较小替代路径 $mAlt^p(i,j)$ 和较小替代边 $mAlt^{arc}(i,j)$。

□

证明：用反证法证明。假设 $G_{min} = (V_P, E_{min}, A_V, A_E)$ 中存在弧 (i,j) 具有替代路径 $mAlt^p(i,j)$，$mAlt^p(i,j)$ 中相应的替代边为 $mAlt^{arc}(i,j) = (p,q)$。则根据非对称模式中较小替代路径和较小替代边的定义，存在 $G'_{min} = (V_P, E'_{min} = E_{min} - (i,j), A_V, A_E)$，且以 G'_{min} 为初始实例的模式与 P 等价且权值（所有边的上界属性之和）小于 G_{min}。与模式最小化定义矛盾。

引理 4.10 对任意 *pattern* $P = (V_P, A, M)$，P 最小化后的初始实例 G_{min} 有且仅有 $2|V_P| - 2$ 条弧。

□

证明：用数学归纳法证明。对 $|V_P|$ 进行数学归纳法。不是一般性，我们首先为 V_P 中节点编号为 $V_P = \{1, 2, ..., |V_P|\}$，当 $|V_P| = 2$ 时，$V_P = \{1, 2\}$ 有两个节点，相

应的 G_{min} 中有 $(\widehat{1,2})$ 和 $(\widehat{2,1})$ 两条弧；假设当 $|V_P| = n-1 (n \geq 3$ 时，G_{min} 中弧的数目为 $a_{n-1} = 2n-4$，下面证明当 $|V_P| = n$ 时，G_{min} 中弧的数目为 $a_n = 2n-2$，事实上，最小模式的定义，$\forall (i,j) \in V_P \times V_P$，$G_{min}$ 中必存在唯一一条有向路径 $p(\widehat{i,j})$ 连接 i 和 j，且满足 $A_{p(\widehat{i,j})} \geq M[i,j]$。因此，当 $|V_P| = |\{1,...,n-1,n\}| = n$ 时，只需将新增节点 n 连接到由 $\{1,2,...,n-1\}$ 节点集合组成的模式的最小模式上去即可，只需新增两条弧。因此 $a_n = 2n-2$。结论成立。

□

事实上，更简单的来看，对于如果对于 $|V_P| = n$ 的模式，其 G_{min} 的弧的数目多于 $2n-2$，则必定存在一条弧具有替代弧，继而可以通过除掉较小替代弧而减少 G_{min} 弧数目，同时不影响 G_{min} 中任意两点之间下界属性值。如果 G_{min} 的弧的数目少于 $2n-2$，由于有 $n-1$ 个顶点，任两个顶点之间相互连接必须需要两条有向路径，这需要每个节点出度和入度至少均为 1 才能满足，矛盾，因此 G_{min} 弧的数目不能少于 $2n-2$。

4.3.2 非对称模式最小化算法

通过上节的分析与证明，下面首先给出非对称模式最小化算法，然后通过定理4.12给出算法的正确性和复杂度。

AUAP:Computing Augment Pattern of a asymmetric pattern:
Input

- a pattern $P = (V_P, A, M)$ (asymmetric)

Output:

- Augment Pattern of $P_{Aug} = (V_P, A, M')$

//$V_P = 1, 2, ..., |V_P|$
1. $M_1 = M(1,1), V_1 = \{1\}$
2. for(i=2; i$\leq |V_P|$; i++)
3. $\quad M_i =$ **UpdateD**(M_{i-1})
4. $\quad V_i = V_{i-1} \cup \{i\}$
5. Output $P_{Aug} = (V_P, A, M' = M_i)$

图 4.5: **Main frame of algorithm AUAP**

UpdateD($M^{'}_{m-1}$)
Input:

- $(m-1)$-dimensional matrix $M^{'}_{m-1}$

Output:

- m-dimensional matrix M_m

1. sort the entries M[m, 1],...M[m,m-1] descending. And sort the entries M[1,m],...,M[m-1,m] descending
//denote the descendent entries by M[m,m-1],...M[m, 1], and M[m-1,m],M[m-2,m],...,M[1,m], respectively.
2. for($i = m-1; i > 0; i--$)
3. temp=$M[m, i]$; M'[m,m]=0
4. for($j = m-1; j > i; j--$)
5. if($M^{'}_{m-1}[j, i] > M[m, i]$)
6. temp=max{$M^{'}_{m-1}[j, i]$, temp}
7. $M^{'}_{m}[m, i]$=temp
8. for($i = m-1; i > 0; i--$)
9. temp=$M[i, m]$;
10. for($j = m-1; j > i; j--$)
11. if($M^{'}_{m-1}[i, j] > M[i, m]$)
12. temp=max{$M^{'}_{m-1}[i, j]$, temp}
13. $M^{'}_{m}[i, m]$=temp
//now updating K_{m-1}
14. for($i = 1; i \leq m-1; i++$)
15. for($j = 1; j < i; j++$)
16. if($M^{'}_{m}[i, j] < M^{'}_{m}[i, m] \&\& M^{'}_{m}[i, j] < M^{'}_{m}[m, j]$)
17. $M^{'}_{m}[i, j] = \min\{M^{'}_{m}[i, m], M^{'}_{m}[m, j]\}$
18. if($M^{'}_{m}[j, i] < M^{'}_{m}[j, m] \&\& M^{'}_{m}[j, i] < M^{'}_{m}[m, i]$)
19. $M^{'}_{m}[j, i] = \min\{M^{'}_{m}[j, m], M^{'}_{m}[m, i]\}$
20. Output $M^{'}_{m}$

图 4.6: **Procedure** UpdateD **of algorithm AUAP**

事实上，非对称模式的增广模式与对称情况有着同样的性质。

定义 4.11 任意增广模式 $P_{Aug} = (V_P, A, M_{Aug})$ 的初始实例 $G_{Aug} = (V_P, E, A_V, A_E)$。对任意三个节点 i, j, k，如果存在弧 $(\widehat{i, j}), (\widehat{i, k}), (\widehat{k, j})$，则称有这三条弧组成的图为 *parallel-triangle*。对任给节点 i, j，将由弧 $(\widehat{i, j})$ 和一条有 i 到 j 的有向路径组成的图成为 *parallel-cycle*。

□

(AAPBA)Asymmetric Augment Pattern Based Algorithm:
Input

- a pattern $P = (V_P, A, M)$

Output:

- the initial instantiation $G_G = (V_P, E)$ of the minimum pattern equivalent to P

//AGA construct a minimum spanning graph across the augment pattern with greedy strategy **incrementally**.
1.Compute the augment pattern $P_{Aug} = (V_P, A, M_{Aug})$ of P using the algorithm in Fig4.5;
2.Construct the initial instantiation of P_{Aug}: $G_C = (V_P, V_P \times V_P)$
//$V_P = 1, 2, ..., |V_P|$
3.$G_m = (V_m, E_m) = (\{1, 2\}, (1, 2))$
4.for $(i = 3; i \leq |V_P|; i++)$
5. **UpdateDGraph**(G_m, i, G_C)
6.Output G_m

图 4.7: **Algorithm(AAPBA)**

定理 4.12 算法***AUAP***能在 $O(|V_P|^3)$ 时间内求出给定非对称模式的增广模式。基于***AUAP***，算法***AAPBA***能在 $O(|V_P|^3)$ 时间内求出给定非对称模式的最小等价模式（即完成最小化）。

□

定理4.12的证明与对称模式中定理4.2的证明类似。非对称模式最小化过程同样具有“伪最优子结构”性质，利用基于“伪动规”的算法4.5进行非对称模式最小化与对称模式利用算法4.1过程及性质类似。

4.4 pattern 最小化若干扩展及复杂度分析与证明

本节给出若干扩展的模式最小化问题，作为本节基本问题的引申。同时还将给出问题的复杂度证明，及若干近似性分析和近似算法。

问题 4.3 *(Pattern Minimization with upper bound constraints)*
给定模式 $P = (V_P, A, M)$，求 $P_0 = (V_P, A_0, M_0)$ 满足一下两个条件：

UpdateDGraph(G_0,m,G_C):
Input

- $G_0 = (1, ..., m-1, E)$
- new node m
- G_C(initial instantiation of P_{Aug}

Output:

- $G_0 = (1, ..., m, E_0)$ such that the total upper bound of G_0 is minimized.

//**UpdateDGraph** procedure add the new node m to G_0 with arcs inG_C, and then delete alternative edges to gain a minimum spanning directed graph with no alternative pathes and *without break the restriction of lower bounds*.
// for an equilateral triangle, we regard the edge with smallest upper bound as the base edge, the other two are waists.

1. connect node m and 1 with edge $(\widehat{1,m})$ and $(\widehat{m,1})$ of G_C
2. for($i = 2; i \leq m-1; i++$)
3. connect m and i with arc $(\widehat{m,i})$ of G_C
4. if i and m are in an *directed isoceles parallel-triangle*
5. delete the waist-arc with larger upper bound
//delete $mAlt_{\triangle}$(waist-arc)
6. else //there is one parallel-cycle C
7. delete $mAlt_C(arc)$ in the parallel-cycle C **(if it exists)** or arc $(\widehat{m,i})$ **(if no** $mAlt_C(arc)$)
8. connect i and m with edge $(\widehat{i,m})$ of G_C
9. if i and m are in an *directed isoceles parallel-triangle*
10. delete the waist-arc with larger upper bound
//delete $mAlt_{\triangle}$(waist-arc)
11. else //there is one directed-cycle C
12. delete $mAlt_C(arc)$ in the cycle C **(if it exists)** or arc $(\widehat{i,m})$ **(if no** $mAlt_C(arc)$)
13. output G_0

图 4.8: **Procedure** UpdateDGraph **of Algorithm(AAPBA)**

- $P_0 = (V_P, A_0, M_0) \equiv_{.U} P$，即 $\forall (i,j) \in V_P \times V_P, M_0[i,j]_{.U} \leq M[i,j]_{.U}$；
- $\forall P' = (V_P.A', M'), P' \equiv_{.L} P$，有 $\sum_{\forall (i,j) \in V_P \times V_P} M_0[i,j]_{.U} \leq \sum_{\forall (i,j) \in V_P \times V_P} M'[i,j]_{.U}$；

□

命题 4.13 问题*4.3*是 *NP* 完全问题。

□

问题 4.4 *Upper Bound Pattern Minimization(MST:PTIME)*

给定没有节点对的下界属性约束的对称模式 $P^U = (V_P, A, M)$，即 $\forall i, j, p, q \in V_P, M[i, j]_{.L} = M[p, q]_{.L}$，求与 P^U 等价的最小化模式。

□

事实上，问题4.4等价于求 P^U 的增广模式的初始实例的最小生成树，因此可在多项式时间内完成。

问题 4.5 *Upper Bound Pattern Minimization with Intermediate Node Latency*

给定没有节点对下界属性约束（比如对于任意两对节点 $i, j, p, q \in V_P, M[i, j]_{.L} = M[p, q]_{.L}$）的对称 *pattern* $P^U = (V_P, A, M^U)$，求其最小传输延时 *pattern* $P_m = (V_P, A, M_m)$*(*其初始实例表示为 G_{P_m}，即 $\forall P = (V_P, A, M) \equiv_{.L} P^U$*:*

$$\sum_{\forall i,j \in V_P} M[i, j]_{.U} + \sum_{\forall i \in V_P : Degree_{G_{P_m}}(i) \geq 2} A(i)$$

$$\leq \sum_{\forall i,j \in V_P} M[i, j]_{.U} + \sum_{\forall i \in V_P : Degree_{G_{P_m}}(i) \geq 2} A(i)$$

□

命题 4.14 问题*4.5*是 *NP* 完全问题，且如果存在近似度小于 *1.463* 的近似算法，则 $NP \subseteq DTIME[n^{O(loglogn)}]$

□

问题 4.6 *Upper Bound Pattern Minimization with total latency threshold*

给定模式 $P = (V_P, A, M)$，任意两点之间直接通信的传输延时 $w : V_P \times V_P \to \mathbb{R}^+$，

一个模式中非零下界属性节点对的上界属性之和的阈值 ζ(可能是分布式系统或网络中端到端性能，如延时要求等)，求 $G=(V_P,E)$，使得 $P \prec G$ 且最小化 G 的权值，即 $\min \sum_{\forall(i,j)\in E}\{w(i,j)\}$，同时满足上界属性门槛限制 ζ 即 $\sum_{\forall(i,j)\in V_P\times V_P}\{rout_G(i,j)_{.U}\}\leq\zeta$，其中 $rout_G(i,j)_{.U}=\min_{\forall pathp(i,j)\in 2^E} p(i,j)_{.U}=\min_{\forall pathp(i,j)\in 2^E}\{\sum_{\forall e\in p(i,j)} M(e)_{.U}\}$

□

虽然一般化的模式最小化问题可以在立方时间内求解，但是当为节点对引入新一维的权值时且引入相应的门槛限制时，问题就变得复杂了。事实上，问题4.6是 NP 完全问题。

定理 4.15 问题*4.6*是 *NP* 完全问题，即是模式被限定为对称模式。

□

证明: 问题4.6的判定形式描述即是：任意给定一个模式 $P=(V_P,A,M)$，一个权值函数 $w:V_P\times V_P\to\mathbb{R}^+$，所有节点对对应路径的上界属性之和的门槛限制：正实数 ζ，以及另一个正实数 η，需要判定是否存在与 P 等价的模式的初始实例图 $G=(V_P,E)$ 使得 $P\succ G$ 且 $\sum_{\forall(i,j)\in E}\{w(i,j)\}\leq\eta$ 且 $\sum_{\forall(i,j)\in V_P\times V_P}\{rout_G(i,j)_{.U}\}\leq\zeta$.

我们首先通过给出问题4.6的一个简单的 NP 算法来说明其是属于 NP 类的。该 NP 算法如下：首先随机猜测一个与 P 等价的任一模式的初始实例图 G，然后检验是否有 $\sum_{\forall(i,j)\in E}\{w(i,j)\}\leq\eta$ 且 $\sum_{\forall(i,j)\in V_P\times V_P}\{rout_G(i,j)_{.U}\}\leq\zeta$。这种检查可以再多项式时间内完成。

现在我们通过构造一个从 PARTITION 问题到问题4.6的 Karp 规约，来证明其是 NP 难的。

回顾经典的六大 NP 完全问题之一的 PARTITION 问题的一个实例形如：给定一有限集合 $C=\{a_1,a_2,...,a_n\}$ 和集合元素的权值函数 $s(a_i)\in\mathbb{Z}^+$ $(\forall a_i\in C)$.PARTITION 问题既是判定是否存在集合 C 的一个划分，即是否存在结合 $C'\subseteq C$ 使得 $\sum_{a\in C'} s(a)=\sum_{a\in C-C'} s(a)=\frac{A_0}{2}$，其中 $A_0=\sum_{i=1}^{n} s(a_i)$。

给定一个 PARTITION 问题的实例如上所述，我们现在构造一个对称模式 P，一个权值函数 $w:V_P\times V_P\to\mathbb{R}^+$，两个正实数 ζ 和 η，使得 PARITITON 问题的实例输

出“是”，即 $\sum_{a\in A'} s(a) = \sum_{a\in A-A'} s(a)$，当且仅当我们构造的问题4.6的实例输出“是”，即存在图 G,使得以 G 为初始实例的模式 P_G 满足 $P \equiv_L P_G$，且 $\sum_{\forall(i,j)\in E} \{w(i,j)\} \leq \eta$ 以及 $\sum_{\forall(i,j)\in V_P\times V_P} \{rout_G(i,j)_{.U}\} \leq \zeta$。

相应与 PARTITION 问题的实例，我们构造问题4.6的实例，即 $P = (V_P, A, M)$, 一个下界阈值门槛 ζ,一个正实数 η 和一个权值函数如下：

- $V_P = \{1, 2, ..., 2n+1\}$;
- A 随意定义。
- 对称矩阵 M: 对 P 中 $(2n+1)$-维矩阵 M 的任意项定义如下：$M[2n+1, i]_{\{.L,.U\}} = M[i, 2n+1]_{\{.L,.U\}} = s(a_{\lfloor\frac{i+1}{2}\rfloor})(\forall i \in \{1, 2, ..., 2n\})$, $M[i, i+1]_{\{.L,.U\}} = M[i+1, i]_{\{.L,.U\}} = s(a_{\lfloor\frac{i+1}{2}\rfloor})(\forall i \in \{1, 3, ...2n-1\})$,且对于任意其他项*ent* $M[\textit{ent}]_{.L} = 0$ 且 $M[\textit{ent}]_{.U} = +\infty$;
- $\forall$ 节点对 $(i,j) \in V_P \times V_P$, $w(i,j) = M(i,j)_{.U}$;
- $\zeta = (4n - \frac{1}{2})A_0$;
- $\eta = \frac{5}{2}A_0$;

显然以上构造过程是 PTIME 可完成的。下面我们验证其是一个从 PARTITION 问题实例 C 到以上实例的 Karp 规约，即问题4.6构造的实例中存在 G，使得 $P \equiv_L P_G$，且满足 $\sum_{\forall(i,j)\in E} \{w(i,j)\} \leq \eta$ 且 $\sum_{\forall(i,j)\in V_P\times V_P} \{rout_G(i,j)_{.U}\} \leq \zeta$，当且仅当那个 PARTITION 问题的实例 C 存在一个有效分割。

从 M 的构造过程中可以看出，如果我们使用 $G^* = (V_P, E^*)$ 表示由边 $(2n+1, i)$ $(\forall i \in 1, 2, ..., 2n)$ 组成的图，则显然 $P \equiv_L P_{G^*}$，且对任意其他图 G 满足 $P \equiv_L G_P$，G^* 必是 G 的子图且具有相同的节点集合。否则 G 的路径的上界属性之和将会是 ∞。对于 G^*，显然 $\sum_{\forall(i,j)\in E} \{w(i,j)\} = 2A_0 = \eta - \frac{1}{2}A_0$ 且 $\sum_{\forall(i,j)\in V_P\times V_P} \{rout_G(i,j)_{.U}\} = 4nA_0 = \zeta + \frac{1}{2}A_0$。如果我们往 G^* 中加入有限上界属性值的边，则 G^* 权值的增量与其所有上界属性值之和的递减量相等。这种均衡揭示了存在一个图 G 满足前面构造问题4.6实例的约束，

当且仅当 PARTITION 问题的实例 C 具有一个分割。因此问题4.6是 NP 难的。因此是 NP 完全问题。

□

4.5 本章小结

本章提出并严格定义了 graph pattern 模型中的最小化问题，并阐述了其实际应用需求及意义。随后，本章给出了一种“最大-最小化”思想，并设计了一个“伪动规”算法在立方时间内解决最小化问题。最后，本章分析并证明了若干最小化问题的变种问题的复杂度。

5 模型求解：组合优化复杂度、不可近似性及求解算法

内容提要：本节将研究第三章中提出的图模式模型的求解过程中产生的若干组合优化问题，本章将给出其复杂度、不可近似性相关证明及（无近似度）近似算法，同时也分析并解决这些组合优化问题的特殊情况及相关扩展问题。

5.1 matching 语义约束模型求解优化问题的不可近似性证明

5.1.1 问题规范描述与分析

问题 5.1（最小 *matching-1*）

给定模式 $P = (V_P, A, M)$，基图 $G = (V_G, E_G, A_{V_G}, A_{E_G})$，判断是否存在单射 $f : V_P \to V_G$ 和函数 $p : f(V_P) \times f(V_P) \to \texttt{PathSet}(f_V(V_P), f_V(V_P))$（见图模式模型予以定义），使得 $P \rhd_{f,p} G$，且最小化 $\sum_{\forall e \in \bigcup_{\forall i,j \in V_P} p(f(i),f(j))} A_{E_G}(e)_{.U}$，若存在，则输出 f, g（即 $\rhd_{f,p}$）

□

5.1.2 复杂度、不可近似性证明

首先分析优化问题5.1，其对应的判定形式问题表述如下：

问题 5.2 *matching* 约束下模型求解的判定问题描述：

INSTANCE ：模式 $P = (V_P, A, M)$, 基图 $G = (V_G, E_G, A_{V_G}, A_{E_G})$,正整数 K

QUESTION ：是否存在单射 $f : V_P \to V_G$ 和函数 $p : f(V_P) \times f(V_P) \to \texttt{PathSet}(f_V(V_P), f_V(V_P))$，使得 $P \rhd_{f,p} G$,且 $\sum_{\forall e \in \bigcup_{\forall i,j \in V_P} p(f(i),f(j))} A_{E_G}(e)_{.U} \leq K$

□

定义 5.1 我们称基图 $G = (V, E, A_V, A_E)$ 满足 *non-blocking* 性质，当且仅当 $\forall e \in E, A_E(e)_{.L} = +\infty$. 称满足 *non-blocking* 性质的基图为 *non-blocking* 图。

□

定理 5.2 最小 *matching* 问题*(*问题*5.2*描述*)*是 *NP* 完全问题，即如果能在多项式时间内求出输入模式在输入基图上的最小 *matching*，则 $P = NP$。即使在对称情况下（模式是对称的，基图是无向图），且基图是 *non-blocking* 图时，优化问题*5.1*对应的判定问题*5.2*仍是 *NP* 完全问题。

□

证明： 问题5.2显然属于 NP 类的。很容易给出一个 NP 算法如下：任意猜想一个单射 $f : V_P \to V_G$ 和一个函数 $p : f(V_P) \times f(V_P) \to \texttt{PathSet}(f_V(V_P), f_V(V_P))$，我们能在多项式时间内检查 f 和 p 是否分别是单射和函数，检测出 $\rhd_{f,p}$ 是否是一个有效的 P 在 G 上的 matching，以及在其是有效 matching 情况下计算出 matching $\rhd_{f,p}$ 的权值是否大于 K，即 $\sum_{\forall e \in \bigcup_{\forall i,j \in V_P} p(f(i),f(j))} A_{E_G}(e) \leq K$ 是否成立。

我们接下来通过构造一个从 X3C 问题（经典的 NP-complete 问题）到问题5.2的 Karp 规约，以证明其是 NP 难的。

X3C 问题的实例描述如下：给定一个有限集 $X = \{x_1, x_2, ..., x_3q\}$，一个由 X 的三元子集组成的集合 $C = \{C_1, C_2, ..., C_n\}$，其中 $C_i = \{x_{i1}, x_{i2}, x_{i3}\}(\forall j \in \{1, 2, 3\}, x_{ij} \in X)$. X3C 问题即是判定 C 是否包含 X 的完美覆盖(*exact cover*)，即是否存在 $C' \subseteq C$ 且 X 中的每个元素 x_i 在 C' 中仅出现一次。

任给一 X3C 问题的实例 I，我们构造出一个问题5.2的实例，即一个对称模式 $P_I = (V_I, A_P, M_I)$，一个无向基图 $G_I = (V_{G_I}, E_{G_I}, A_{V_G}, A_{E_G}$，以及正整数 K，使得 P_I 在 G_I 上存在一个权值大于 K 的 matching $\rhd_{f,p}$ 当且仅当实例 I 存在一个完美覆盖(*exact cover*)。

根据 X3C 的实例 I 构造的问题5.2的实例如下：

1. 对称模式 $P_I = (V_I, A_P, M_I)$ 的定义：

   - $V_I = \{X^P_{11}, X^P_{12}, X^P_{13}, X^P_{21}, ..., X^P_{q1}, X^P_{q2}, X^P_{q3}, C^P_1, C^P_2, ..., C^P_q\}$;
   - A_P 定义：$A_P(X^P_{ij}) = a(\forall i \in \{1, 2, ..., q\}, j \in \{1, 2, 3\})$，$A_P(C^P_i) = a(\forall i \in \{1, 2, ..., q\}$，其中 $a, b \in Z^+$，且 $a > b$;

- M_I 定义：$\forall i \in \{1,2,...,q\}, M[C_i^P, X_{ij}^P]_{.L} = X_{ij}^P], M[C_i^P{}_{.L} = w$；

2. 基图 $G_I = (V_{G_I}, E_{G_I}, A_{V_G}, A_{E_G})$ 定义：

- $V_{G_I} = \{X_{11}^G, X_{12}^G, X_{13}^G, X_{21}^G, ..., X_{q1}^G, X_{q2}^G, X_{q3}^G, C_1^G, C_2^G, ..., C_n^G\}$;
- $E_{G_I} = \{(C_i^G, X_{i\{1,2,3\}}) | \forall i \in \{1,2,...,q\}\}$;
- A_{V_G} 定义：$A_{V_G}(X_{ij}^G) = a(\forall i \in \{1,2,...,q\}, j \in \{1,2,3\})$，$A_{V_G}(C_i^G) = a(\forall i \in \{1,2,...,q\}$ 其中 $a, b \in Z^+$，且 $a > b$；
- A_{E_G} 定义：$\forall e \in E_{G_I}, A_{E_G}(e)_{.L} = +\infty, A_{E_G}(e)_{.U} = 1$。

3. 整数 $K = 3q$

从以上构造过程中, 我们可以发现 $\forall i \in \{1,2,...,p\}, j \in \{1,2,...,n\}$，$P_I$ 中的 C_i^P 只能被映射到 G_I 的 C_j^G，P_I 的 X_{ik}^P 只能被映射到 G_I 的 $X_{jl}^G(k,l \in \{1,2,3\})$。且对称模式 P_I 的初始实例中任意一条边在 (f,p) 定义的 matching 中对应于 G_I 中的一条边，否则 matching $\rhd_{f,p}$ 的权值必定大于 $3q$（因为模式 P_I 的初始实例的边个数恰好是 $3q$ 个）。事实上，P_I 编码了 I 的一个完美覆盖*exact cover*，而 G_I 对 I 进行了编码.

很显然，以上构造过程可以在多项式时间内完成(PTIME)。下面将证明以上构造是一个 Karp 规约过程，即 I' 中 $P_I \rhd G_I$ 当且仅当实例 I 存在完美覆盖(*exact cover*).

一方面，如果 I' 存在一个权值（边的上界属性之和）小于或等于 $K = 3q$ 的 matching，即存在一个从 P_I 到 G_I 的单射 f，使得 $P_I \rhd G_I$ 及函数 $p : f(V_P) \times f(V_P) \to \mathtt{PathSet}(f_V(V_P), f_V(V_P))$，使得满足 $P_I \rhd_{f,p} G_I$ 且 $\sum_{\forall e \in \bigcup_{\forall i,j \in V_P} p(f(i),f(j))} A_{E_G}(e) \leq 3q$（事实上一定是取等号）。则必然有 $\{f(C_i^P) | \forall i \in \{1,2,...,q\}\}$ 是 I 的一个完美覆盖(*exact cover*)。因为如果 $\{f(C_i^P) | \forall i \in \{1,2,...,q\}\}$ 不是 I 的一个完美覆盖(*exact cover*)，由于模式 P_I 中每个节点对 (i,j) 在 matching 中必须对应于一条边（否则 matching $\rhd_{f,p}$ 权值大于 3q），故有 $card(\{f(X_{ik}^G) | \forall i \in \{1,2,...,q\}, k \in \{1,2,3\}\}) < card(\{X_{ik}^G | \forall i \in \{1,2,...,q\}, k \in \{1,2,3\}\})$，其中 $card(S)$ 表示集合 S 的基数。这与 f 是单射相矛盾。

另一方面，如果 I 存在一个完美覆盖(*exact cover*)，则我们称该完美覆盖(*exact cover*) 同样制订了一个从 P_I 的节点到 G_I 的节点的单射 f 及函数 p。我们使用

$S^C = \{C^G_{j_k} | k \in \{1,2,...,q\} and j_k \in \{1,2,...,n\} \subseteq C = \{C_1, C_2, ..., C_n\}$ 表示 I 的完美覆盖(*exact cover*)。则单射 f 定义如下：

- 对 $\{C^P_1, C^P_2, ..., C^P_q\}$ 中的节点 C^P_i，$f(C^P_i) = C^G_{j_i} (\forall i \in \{1,2,...,q\})$；
- 对 P_I 中 $X^P_{ik} (i \in 1,2,...,q, k \in 1,2,3)$，$f(X^P_{ik}) = X^G_{j_i k}$.

函数 p 为：$p(f(C^P_i), f(X^P_{ik})) = (C^G_i, X^P_{ik}) (\forall i \in \{1,2,...,1\}, k \in \{1,2,3\})$ 显然 f 定义了 P_I 在 G_I 上的一个 matching，即 $P_I \rhd G_I$.

综上，问题5.2的实例存在一个 matching $\rhd_{f,p}$ 满足 $\sum_{\forall e \in \bigcup_{\forall i,j \in V_P} p(f(i),f(j))} A_{E_G}(e) \leq K$，当且仅当 X3C 的实例具有一个完美覆盖(*exact cover*). 因此问题5.2是 NP-hard 问题.

因此，问题5.2是 NP 完全问题。

□

下面进一步分析问题5.1的可近似性。首先给出 AP-规约定义和性质，然后利用 AP-规约证明问题5.1的不可近似性（优化问题的近似类型）。

定义 5.3 P_1 和 P_2 是两个 *NPO* 问题（其对应的判定问题是 *NP* 的），我们称 P_1 能 *AP*-规约到 P_2（表示为 $P_1 \leq_{AP} P_2$），如果存在函数 f 和函数 g，正常数 $\alpha > 1$，满足：

1. $\forall x \in I_{P_1}$，$\forall$ 正实数 $r > 1$，$f(x,r) \in I_{P_2}$；
2. $\forall x \in I_{P_1}$，$\forall$ 正实数 $r > 1$，如果 $SOL_{P_1}(x) \neq \emptyset$，则 $SOL_{P_2}(f(x,r)) \neq \emptyset$；
3. $\forall x \in I_{P_1}$，$\forall$ 正实数 $r > 1$，$\forall y \in SOL_{P_2}(f(x,r)), g(x,y,r) \in SOL_{P_1}(x)$；
4. 对任意给定的 r，f 和 g 在多项式时间内可计算；
5. $\forall x \in I_{P_1}$，$\forall r > 1$，$\forall y \in SOL_{P_2}(f(x,r))$，如果 $R_{P_2}(f(x,r),y) \leq r$，则 $R_{P_1}(x, g(x,y,r)) \leq 1 + \alpha(r-1)$；

三元组 (f,g,α) 被称为从 P_1 到 P_2 的一个 *AP*-规约。

□

引理 5.4 如果 P_1 和 P_2 是两个 *NPO* 问题，且 $P_1 \leq_{AP} P_2$，如果 P_1 属于 *NPO* 完全类，则 P_2 也属于 *NPO* 完全类。

□

[15]中给出了引理5.4及 AP-规约的相关性质的介绍和证明。

基于以上引理下面给出问题5.1的不可近似性的分析与证明。

定理 5.5 问题*5.1*属于 *NPO* 完全，即是基图和模式的初始实例均是 *DAG*，且基图是 *non blocking* 图。

□

证明：根据引理5.4，我们只需构造出一个从 NPO 完全问题到问题5.1的 AP-规约即可。下面我们给出从 NPO 完全问题 Minimum Weighted 3-Satisfiability(简记为**MW-3SAT**)到问题5.1的 AP-规约。

MW-3SAT问题的任一实例形如：$\phi = C_1 \wedge C_2 \cdots \wedge C_n$，其中 ϕ 中的变量为 $x_1, x_2, \cdots, x_m$，每个项 $C_j (j \in [1,n])$ 形如 $y_{j_1} \vee y_{j_2} \vee y_{j_3}$，这里 $y_{j_i} (i \in [1,3])$ 是 $x_{p_{ji}}$ 或者是 $x^-_{p_{ji}} (p_{ji} \in [1,m])$。我们使用 $x_{p_{ji}}$ 表示项 C_j 中的第 i 个出现的变量。对于每个变量 $x_i (i \in [1,m])$，存在权值函数 $w : \{x_1, x_2, \cdots, x_m\} \to R^+$。**MW-3SAT**问题即是求 ϕ 的一个真值赋值 $\rho\{x_1, x_2, \cdots, x_m\} \to 0, 1$，使得 ϕ 为真且最小化 $\sum_{\forall i \in [1,m]} w(x_i) \cdot \rho(x_i)$。

下面我们给出一个从**MW-3SAT**问题到问题5.1的 AP-规约 (f, g, α) 如下：

$f : \phi \to$ **问题5.1的实例** 给定**MW-3SAT**问题实例 ϕ，我们通过给出相应的问题5.1实例 $f(\phi)$ 的构造过程来定义函数 f。

1. 模式 $P = (V_P, A, M)$ 定义如下：
   - $V_P = \{C_1, \cdots, C_n, X_1, \cdots, X_m\}$
   - A 定义如下：$A(X_i) = a_X (\forall i \in [1,m]$，$A(C_j) = b_C (\forall j \in [1,n])$，其中 $a_X, b_C \in R^+$，且 $a_X > b_C$。
   - M 定义如下：$M[X_{p_{jk}}, C_j]_{.L} = c (\forall k \in [1,3], j \in [1,n])$，其中 c 是任意正常数。对矩阵 M 中任意其他项*entry*，均有 $M[\textit{entry}]_{.L} = 0$。

P 实际上表达了**MW-3SAT**问题的任一实例 ϕ。

2. 基图 $G=(V_G,E_G,A_V,A_G)$ 定义如下：

   - $V_G=\{X_{T1},...,X_{Tm},X_{F1},\cdots,X_{Fm},0_1,\cdots,7_1,\cdots,0_n,\cdots,7_n\}$.
   - E_G 定义：对 ϕ 中每个项 $C_j=y_{j_1}\wedge y_{j_2}\wedge y_{j_3}$ 而言，存在 7×3 条边，E_G 共存在 21n 条边。下面给出这 21n 条边的定义。

     (a)我们用 ρ 表示对变量 $x_{p_{j1}},x_{p_{j2}},x_{p_{j3}}$ 的真值赋值，将对 C_j 的真值赋值表示为 8 个节点 $C_j(\rho)$。每个节点 $C_j(\rho)$ 是一个 3bit 的常数 $y_{j_1}y_{j_2}y_{j_3}$，由 $\rho(x_{p_{j1}}),\rho(x_{p_{j1}})$ 和 $\rho(x_{p_{j1}})$ 表示。(b)对每个使得 C_j 为真（1）的真值赋值（针对变量 $x_{p_{j1}},x_{p_{j2}},x_{p_{j3}}$），$E_G$ 由以下边组成：$(X_{Tp_{jk}},C_j(\rho))$（当 $\rho(X_{p_{jk}})=true(1)$），或者 $(X_{Fp_{jk}},C_j(\rho))$（当 $\rho(X_{p_{jk}})=false(0)$）。
   - A_V 定义如下：$A_V(X_i)=a_X(\forall i\in[1,m]$，$A(C_j)=b_C(\forall j\in[1,n])$，其中 a_X,b_C 定义与模式实例中相同。
   - A_E 定义如下：$\forall(i,j)\in E_G,A_E(i,j)_{.L}=c$，c 与模式中 M 里的定义相同。$\forall(X_{Ti},t),(X_{Fj},t)\in E_G$，有 $A_E(X_{Ti},t)_{.U}=w(x_i),A_E(X_{Fj},t)_{.U}=w(x_j)$。

G 实际上表达了使 ϕ 为真的全部真值赋值。节点 $X_{Ti}(i\in[1,m],X_{Fi})$ 表示真值分配中变量 x_i 的值为真（假）。节点 $\{0_j,\cdots,7_j\}$ 表示 $C_j(\rho)$，这里 $C_j(\rho)$ 是使用一个 3bit 的常数表达的，该 3bit 常数表达了 C_j 的三个变量的真值赋值。从点 X_{Ti} 或 X_{Fi} 到节点 $\{0_j,\cdots,7_j\}$ 表达了对变量 $(x_{p_{j1}},x_{p_{j2}},x_{p_{j3}})$ 的真值赋值和相应的项 $C_j(\rho)$ 的关系。

函数 g 对问题5.1相应于**MW-3SAT**实例 ϕ 的实例 $f(\phi)$（如上构造），若 $\rhd_{\lambda,p}$ 是 $f(\phi)$ 的任意可行解，即 $f(\phi)$ 中 P 在 G 上的一个有效的 matching，我们通过给出在 ϕ 中与 $\rhd_{\lambda,p}$ 相应的 ϕ 的可行解 ρ 来定义函数 g。ρ 定义如下：

$\forall x_i(i\in[1,m])$,当 $\lambda(X_i)=X_{Ti}$ 时，$\rho(x_i)=1(true)$，当 $\lambda(X_i)=X_{Fi}$ 时，$\rho(x_i)=0(false)$.

$\underline{\alpha}\ \alpha = 1$;

下面证明 (f, g, α) 是 AP-规约，即证明其满足 AP-规约定义中的五条性质。

1. $f(\phi)$ 是问题5.1的实例：这点由构造过程说明，以上即是通过构造**MW-3SAT**问题的任意 ϕ 对应的问题5.1的实例来定义函数 f 的；

2. $SOL(\phi) \neq \emptyset \Rightarrow SOL(f(\phi)) \neq \emptyset$：下面证明，如果存在真值赋值 ρ 满足 ϕ，即使得 ϕ 为真，则问题5.1的实例 $f(\phi)$ 存在一个有效的 matching，即 $P \rhd_{\lambda,p} G$。$\rhd_{\lambda,p}$ 给出如下：(1)当 $\rho(x_i) = 1(true)$ 时，$\lambda(X_i) = X_{Ti}$，当 $\rho(x_i) = 0(false)$ 时，$\lambda(X_i) = X_{Fi}$；(2)$\lambda(C_j) = C_j(\rho)$；(3)$p(i, j)$ 定义为 G 中(i,j)两点间的唯一有向边。很容易验证 $\rhd_{\lambda,p}$ 是 P 在 G 中的一个有效 matching。故 $SOL(\phi) \neq \emptyset \Rightarrow SOL(f(\phi)) \neq \emptyset$。

3. $\forall \rhd_{\lambda,p} \in SOL(f(\phi)), g(x, \rhd_{\lambda,p}) \triangleq \rho \in SOL(\phi)$：根据前面定义 g 的由 $f(\phi)$ 的有效的 $\rhd_{\lambda,p}$ 构造使 ϕ 为真的真值赋值 ρ 的过程，注意到节点 P 中 X_i 不能被同时映射到 G 中的 X_{Ti} 和 X_{Fi} 上，对每个节点 $C_j (j \in [1, n])$，$\lambda(C_j)$ 保证了 ρ 一定能使 C_j 为真。因此，g 中定义的 ρ 能确保 ϕ 为真。

4. f 和 g 能是多项式时间可计算的。

5. 有上面对 $f(\phi)$ 中有向边的上界属性的构造定义，有 $R_{matching}(f(\phi), \rhd_{\lambda,p}) = R_{MW-3SAT}(\phi, g(\phi, \rhd_{\lambda,p}))$，因此当 $\alpha = 1$ 时有 $R_{matching}(f(\phi), \rhd_{\lambda,p}) \leq r \Rightarrow R_{MW-3SAT}(\phi, g(\phi, \rhd_{\lambda,p})) \leq r$.

综上，(f, g, α) 是 AP-规约，因此问题5.1属于 NPO 完全类。

□

回顾第 2 章中对优化问题按照可近似性的分类，NPO 完全类是最难的一类优化问题，具有不可近似性。

定理 5.6 问题*5.1*不存在近似度为 $O(2^{n^k})$*(k* 为任意正整数*)*的近似算法，能在多项式时间内判定问题的任意实例的可行解是否为空。

□

证明：定理5.5证明了问题5.1是 NPO 完全问题，由 NPO 与 exp-APX 类问题的区别及 NPO 完全问题的定义可知以上结论成立。

□

推论 5.7 对于问题*5.1*中节点已经映射完毕（或者节点可行映射方案唯一），即 $\triangleright_{f,p}$ 中的 f 已经唯一确定，寻找有效的 p 使得 $\triangleright_{f,p}$ 是最小 *matching* 仍是 *NP* 难的，且属于 *APX* 完全类。即是 *base graph* 是 *non-blocking graph* 结论仍然成立。

□

证明： 首先，显然可以确定最小 matching 一定是树。因为如果不是树，则最小 matching 中一定村在一个环 C，这样我们可以去掉环 C 中下界属性最小的边，使得剩下的树仍是 matching，但所有边的上界属性之和减小了，与原来存在环 C 的 matching 是最小 matching 矛盾，因此最小 matching 一定是树。这样，对于 $\triangleright_{f,g}$ 中 f 已唯一确定的情况，最小 Steiner 树问题是最小 matching 问题的一个特例（及 base graph 是 non-blocking 的问题5.1的实例）。而最小 Steiner 树问题是 NP 难的，且是 APX 完全类。

□

引理5.7表明，即使在寻找模式的最小 matching 过程中，节点映射时唯一确定且基图是 non-blocking graph，问题还是 NP 难且不可能存在有常数近似度的多项式时间近似算法。

推论 5.8 如果问题*5.1*存在近似度为 $f(n)$ 的近似算法，则对于问题*5.1*中节点已经映射完毕（或者节点可行映射方案唯一），即 $\triangleright_{f,p}$ 中的 f 已经唯一确定的情况，寻找有效的 p 使得 $\triangleright_{f,p}$ 是最小 *matching* 也存在近似度为 $f(n)$ 的近似算法。

□

证明： 显然问题5.1的约束问题是原问题的特例，其实例集合是问题5.1实例集合的子集。因此5.8成立。

□

5.2 embedding 语义约束模型求解优化问题的不可近似性证明

5.2.1 问题规范描述与分析

问题 5.3 *minimum embedding* 给定模式 $P=(V_P,A,M)$，基图 $G=(V_G,E_G,A_{V_G},A_{E_G})$，判断是否存在单射 $f:V_P\to V_G$ 和函数 $p:f(V_P)\times f(V_P)\to \texttt{PathSet}(f_V(V_P),f_V(V_P))$（见图模式模型予以定义），使得 $P\trianglerighteq_{f,p}G$，且最小化 $\sum_{\forall e\in \bigcup_{\forall i,j\in V_P} p(f(i),f(j))} A_{E_G}(e)$，若存在，则输出 f,g（即 $\trianglerighteq_{f,p}$）

□

问题 5.4 *Maximum Total Residual Bandwidth Embedding* 给定 $P=(V_P,A,M)$, *base graph* $G=(V_G,E_G,A_V,A_E)$，求 P 在 G 中的一个 *embedding*，即单射 $f:V_P\to V_G$ 和函数 $p:f(V_P)\times f(V_P)\to \texttt{PathSet}(f_V(V_P),f_V(V_P))$，使得 $\max\sum_{\forall entry(i,j)\in M} res(p_G(i,j))$，其中 $p_G(i,j)$ 表示 M 的项 (i,j) 在 G 中 *embedding* 对应的路径（见 *embedding* 定义），$\forall e\in E_G, res(e)=A_E(e)_{.L}-\sum_{\forall(i,j)\in M,e\in p_G(i,j)} M(i,j)_{.L}$，这里 $res(p_G(i,j))=\min_{\forall e\in p_G(i,j)}\{res(e)\}$.

□

5.2.2 复杂度、不可近似性证明

下面研究问题5.3的近似性特征。

下面结论给出了最小 embedding 问题（问题5.3）的不可近似性结论：

定理 5.9 问题*5.3*属于 *NPO* 完全类，即不存在近似度为 $O(2^{n^k})$(k 为正整数)的多项式时间近似算法，即是基图是 *non-blocking* 的，且基图和模式的初始实例均是 *DAG*。

□

证明过程仍是通过构造出一个从 Minimum Weighted 3SAT 问题到问题5.3的 AP-规约来完成的。AP-规约的构造与定理5.5的证明完全相同。

以上结论指出了最小 embedding 问题属于所有组合优化问题中最复杂的一个子类，除非 P=NP，否则其不存在多项式时间近似算法。其实，当我们将最小 embedding

问题，即问题5.3进行约束和简化后，其仍然是一个非常复杂的问题。下面定理证实了其复杂性。

推论 5.10 在问题*5.3*中，即使节点映射方案唯一确定，即节点已经映射完毕，求最小 *embedding* 仍是 *NP* 难的，且属于 *APX* 完全类。即使基图满足 *non-blocking* 性质，结论仍然成立。

□

事实上，推论5.10的证明过程与 matching 语义约束下相应的定理5.7证明方法相似。其包含了最小 Steiner 树问题作为其特例。

前面分析了最小 embedding 基本问题，即问题5.3的不可近似性，下面给出问题5.4的不可近似性证明。

定理 5.11 如果 $P \neq NP$，问题*5.4*不可能存在一个近似度为 $O(|V_G|^{\frac{1}{2}-\epsilon}+1(\forall\epsilon>0)$ 的多项式时间近似算法。

□

证明： 我们通过给出一个从**EDP**(edge disjoint path)问题到问题5.4的 L-规约。其中**EDP**问题不存在 $O(m^{\frac{1}{2}-\epsilon})$-多项式时间近似算法。

首先，我们简单回顾第 2 章中介绍过的 L-规约。问题 A 和 B 是两个优化问题，如果存在一个从 A 到 B 的 L-规约，即存在 PTIME 的函数 R 和 S，满足：

i) 如果 $x \in I_A$(问题 A 的实例)，用 $OPT(x)$ 表示最优解的权值，则 $R(x)$ 时问题 B 的实例，且该实例的最优解权值满足：$OPT(R(x)) \leq \alpha OPT(x)(\alpha$ 是正常数);

ii) 如果 s 是实例 $R(x)$ 的任意可行解，则 $S(s)$ 是 x 的可行解，且满足 $|OPT(x)-c(S(s)| \leq \beta|OPT(R(x))-c(S)|$，其中 c 表示实例的权值。

L-规约的特点再约其保留了问题的近似性：如果存在一个从优化问题 A 到 B 的 L-规约 (R,S) 以及常数 α and β，且存在问题 B 存在 ϵ-多项式时间近似算法 B($\epsilon>1$)，则问题 A 具有 $\alpha\beta(\epsilon-1)$-多项式时间近似算法。我们可以使用其逆否命题来证明问题的不可近似性。

EDP问题的实例 I_{EDP}：给定有向图 G 及包含 k 个有序节点对集合 $T = \{(s_i, t_i) : 1 \leq i \leq k\}$($s_i, t_i$ 是 G 中节点)。**EDP**问题是在 G 中超出最多个数的边不相交的链路连接 T 中的节点对。给定问题**EDP**的实例 I_{EDP}，我们通过给出两个 PTIME 的函数 R 和 S 构建一个问题5.4 的实例，即一个模式 P_{MRPBE} 和一个基图 G_{MRPBE} 满足 L-规约的要求。

函数 R 和 S 定义如下：

a) $R : I_{EDP} \times I_{MRPBE}$。对于一个给定的 I_{EDP}，即一个有向图 $G_0 = (V_0, E_0)$ 和有序节点对集合 $T = \{(s_i, t_i) : 1 \leq i \leq k\}$，我们定义函数 R，即构造 I_{MRPBE} 如下：

模式 $P = (V_P, A, M)$ ：

- $V_P = V_0$
- A: $\forall i \in \{s_i : 1 \leq i \leq k\}, A_{.L}(i) = i$; $\forall i \in \{t_i : 1 \leq i \leq k\}, A_{.L}(i) = |T| + i$; for any other node j in V_P, $A_{.L}(j) = 0$
- M: $\forall (i, j) \in T, M_{.L}(i, j) = 1.5$; for any other pair $(i', j') \in V_P \times V_P, M_{.L}(i', j') = 0$

基图 $G_d = (V_d, E_d, A_V, A_E)$ ：

- $V_d = V_0$
- $E_d = V_d \times V_d$
- A_V: $\forall i \in \{s_i : 1 \leq i \leq k\}, A_{.L}(i) = i$; $\forall i \in \{t_i : 1 \leq i \leq k\}, A_{.L}(i) = |T| + i$; 对任意其他节点 j in V_d, $A_{.L}(j) = 0$
- A_E: $\forall (i, j) \in V_d \times V_d$, if $(i, j) \in E_0, A_E(i, j)_{.L} = 1.5$; 对任意其他边 $(i', j') \in E_d, A_E(i', j')_{.L} = 1$

通过以上构造，我们能发现，对于给定的实例 I_{EDP} 中包含在 T 中的任一节点对，通过 $Sol(I_{MRPBE})$ 中的一个可行解(embedding)，其在实例 $I_{R(I_{EDP})} = I_{MRPBE}$ 的 G_d 中将会被分配到其在 G_0 中同样的节点位置。因为在一个正确的 embedding 中，节点的映射时单射。也就是说，如果在实例 I_{EDP} 的 G_0 中，$s_i = V_0[i]$（我们使用 $V[\ ]$ 表

示 $G(V,E)$ 的一个节点向量），则在 I_{MRPBE} 中，由于构造的节点的下界属性，必有 $f(s_i)=V_P[i]=V_0[i]$。

b) $S:Sol(I_{MRPBE})\times Sol(I_{EDP})$. 从 R 的定义，即实例 I_{MRPBE} 的构造来看，对任意**EDP**问题的实例，问题5.4对应的实例存在非空可行解集合（用 $Sol(I_{MRPBE})$ 表示）。定义在 $Sol(I_{MRPBE})$ 上的函数 S 定义如下：

对任意可行解 $s\in Sol(I_{MRPBE})$，即一个*embedding*（单射 f: $V_P\to V_d$，满足*embedding*定义中 P 在 G 中的*edge-to-path*的映射约束条件，其中 P 和 G 是在构造 I_{MRPBE} 过程中由 R 定义的。我们构造了一个可行解 $S(s)\in Sol(I_{EDP})$，即 G_0 中连接每个 T 中有序节点对的边不相交的路径。$\forall(i,j)\in T$，如果 G_d 中 $p(f(i),f(j))$ 由下界属性为 1.5 的边组成，则在实例 I_{EDP} 中，我们定义 G_0 中连接 (i,j) 的边不相交路径 $p_dis(i,j)$ 是由在问题5.4实例中组成 $p(f(i),f(j))$ 的顶点组成的。很容易验证由 $p_dis(i,j)(\forall(i,j)\in T$ 组成的集合是 $Sol(I_{EDP})$ 中的一个可行解。

下面我们证明函数 R 和 S 满足 L-规约的约束条件，即：

1) 函数 R 和 S 均是 logspace/PTIME 可计算的；

2) $OPT(R(x))\le\alpha\cdot OPT(x)(\forall x\in Sol(I_{EDP})$；其中 α 是正常数；

3) $|OPT(x)-c(S(s))|\le\beta\cdot|OPT(R(x)-c(s)|(\forall x\in Sol(I_{EDP}),\forall s\in Sol(I_{MRPBE})),\beta$ 是正常数；

显然上面构造的函数 R 和 S 是 PTIME。由于每个解 x 对应的边不相交路径，在 G_d 中均存在相应的边。事实上，$OPT(R(x))=(1.5-1)\cdot|OPT(x)|$。因此条件 2)成立，其中 $\alpha=0.5$。对于条件 3)，由于 T 中任两对有序节点对在 G_d 中对应的路径是边不相交的，因此条件 3)成立（事实上取等号），其中 $\beta=2$。综上，函数 R 和 S 是一个从**EDP**问题到问题5.4的 L-规约。[16]中已给出证明，如果**EDP**具有近似度为 $|V_0|^{\frac{1}{2}-\epsilon}(\forall\epsilon>0)$ 的多项式时间近似算法，则 P=NP。因此，根据 L-规约的性质，定理5.11成立。

□

推论 5.12 对于问题*5.4*，即是在节点映射方案唯一确定的情况下，任然不可能有近似度为 $O(m^{\frac{1}{2}-\epsilon})$ 的多项式时间近似算法。

□

5.3 构造型和枚举型启发式算法框架

由于模式求最小（优）matching 和最小（优）embedding 均是 NPO 完全问题，是所有组合优化问题中最难的一类，不可能存在多项式时间近似算法($O(2^{n^k})$ 判定其优化问题的实例的可行解是否为空，因此不可能存在有近似度保证（甚至是指数级的）的近似算法。鉴于此，只能求助于没有近似度保证的近似算法——启发式算法。

我们从构造性启发式算法和枚举型启发式算法两个角度出发分析图模式模型的最优解求解问题。

我们首先分析枚举类算法框架的设计。事实上，对于一般化的最优 matching 问题，我们可以通过以下步骤通过枚举型启发式算法模式求解：

第一步：（构建索引）利用增广模式算法求出以基图 G_B 为初始实例的模式 P_{G_B} 的增广模式 $P_{G_B}^{Aug}$.

第二步：（启发式枚举）参照第一步中的索引，对模式 P 中每个节点进行回溯搜索（深度优先或广度优先搜索），剪枝条件是

- 存在任两个以搜索的模式中的节点对无法满足下界属性需求（包括两个节点的下界属性和节点对的下界属性）；
- 存在一个局部更优的 matching；

第三步：如果模式中所有节点均通过回溯搜索得到唯一可行分配，则将其输出作为枚举型启发式算法的输出。

对于最优 embedding 问题的枚举型启发式算法框架，与 matching 基本思想相同，只不过第一步中 embedding 无法建立基图中节点对下界属性索引，只能按照模式中节点与基图中节点之间符合下界属性要求的匹配关系进行回溯搜索。

对于利用构造性的启发式算法求解最优 matching 和 embedding 的基本算法框架，两种语义的处理过程的算法框架是相同的，均形如如下形式：

第一步：优先根据模式和基图间节点的下界属性约束，在仅考虑节点约束情况下先随机将模式中节点通过单射映射到基图上；

第二步：对基图上每个模式节点实施 Move 操作:一个模式节点会从基图中节点 A 移动到 B，当且仅当该移动不会降低模式和基图的匹配度，即符合下界属性约束的节点对个数（类似的定义均视作此类构造性启发式算法框架）。

事实上，在第 6 章中，这是基于这种 Local Better Move 的构造性启发式求解思想，从策略博弈角度给出了一种 Local Move Search Game 模型用以优化 matching 及 embedding。

5.4 本章小结

本章分析了图模式模型求最优解过程中的若干优化问题（分别对应于 matching 和 embedding 两种语义），证明了基本优化问题均是 NPO 完全问题，即不存在多项式时间的近似度为指数（$O(2^{n^k})$）的近似算法能判定问题实例的可行解集合是否为空。并且证明了两种语义对应优化问题的几个特例也仍是 APX 完全问题，不可近似的。虽然如此，本章最后还是从构造型启发式算法和枚举型启发式算法的思想出发，给出了两种语义优化问题求解的基本启发式算法框架。

6 在线虚拟网络映射

6.1 在线虚拟网络映射

在虚拟网络部署（映射）场景中，特别是在云计算数据中心网络中，虚拟网络映射请求（对应于图模式模型中的模式）往往是在线随机到来的，且具有生存周期、相应时间要求等。这种应用场景下虚拟网络映射问题的特点有以下几个特点：

1. 虚拟网络请求的产生（到来）是完全随机的，且没有任何规律性。
2. 虚拟网络请求在线到来的密度往往比较大，因为同时在线申请租赁虚拟网络使用数据中心网络的用户数目可能会很多。
3. 相较数据中心中高速以太网链路而言，虚拟网络链路的资源请求（带宽、延时）往往要小得多。
4. 物理网络节点数巨大，其规模远大于虚拟网络规模
5. 虚拟网络映射的目标是降低数据中心运营成本，比如通过提高活动资源的资源利用率等。

这种应用场景对图模式模型提出了新的需求，同时也引入新的约束和前提假设。本章将首先将重新审视前面简化的、一般化的图模式模型，引入多重上、下界属性、第三类属性（见第 3 章）以表达这种在线的虚拟网络映射请求。另一方面，应用场景中的数据中心网络里的高速以太网链路与虚拟网络请求链路资源请求的悬殊，也为图模式模型引入了新的特点。

6.2 在线虚拟网络映射：图模式模型的扩展

6.2.1 模式的固有属性

在第 3 章建立图模式模型及分析其性质时已经提到过，物理网络和用户提出的虚拟网络请求中，网络节点和链路的（资源、QoS）属性往往可以分为下界(Lower

Bound)、上界(Upper Bound)，以及固有属性（如 Type、lifespan 等）。前面主要侧重于单一模式执行 matching 和 embedding 语义操作时，仅考虑下界属性的一些模型特性、复杂度特性分析与证明，指出了一般情况下基图模型的复杂程度，同时分析了模型求解中考虑上界属性时的几个优化问题及变种问题的优化问题类型（两种基本操作的优化问题均是 NPO 完全，对应的简化特例是 APX 完全）。在公有云数据中心这种应用场景下，由于虚拟网络请求具有上小结所述的若干性质，我们必须考虑前几张分析模型一般化计算特点时所忽略的固有属性：如生存周期（lifespan）、密集程度——在线请求率（arriving rate）等。

下面给出引入固有属性后的图模式模型的定义。

模式模型：

定义 6.1 **基图*(Base Graph)***: 一个数据中心网络*(DN)*在图模式模型中表示为一个基图，即一个六元组 $G_B = (V_G, E_G, A_{V_G}, A_{E_G}, R_{V_G}, R_{E_G})$，其中 V_G 是数据中心网络节点的集合；E_G 是数据中心网络链路的集合；A_{V_G} 是数据中心网络节点属性向量，该向量中每个元素是一个二元组，分别表示该元素对应节点的下界属性*(Lower Bound)*和上界属性*(Upper Bound)*，分别表示为 $A_{v_0.L}$ 和 $A_{v_0.U}(\forall v_0 \in V_G)$；$A_{V_E}$ 是物理网络链路属性向量，定义类似于 A_{V_G}，分为下界属性 $A_{E_G.L}$ 和上界属性 $A_{E_G.U}$. R_{V_G} 和 R_{E_G} 分别表示 G_B 中的 V_G 里每个节点的当前下界属性值。

注：下面关于 R_V, R_E 的定义是语义部分的！！定义为 $\forall i \in V_G, R_{V_G}(i) = A_{V_G}(i)_{.L} - \sum_{\forall vnode \quad on \quad i} \{A_{V_P}(vnode)_{.L}\}$. 与此类似，$\forall e \in V_E, R_{V_E}(e) = A_{E_G}(e) - \sum_{\forall i,j \in V_P,} \{\}$。

□

定义 6.2 **模式*(Pattern)***: 一个用户提出的虚拟网络请求*(VN Request)*可用一个模式表示。一个模式是一个五元组 $P = (V_P, A_{V_P}, M, L, RT)$，其中 V_P 是用户提出的虚拟网络请求中虚拟机节点集合，A_{V_P} 是虚拟机节点属性向量，该向量中每个元素是一个二元组，分别表示该元素对应的虚拟机节点资源请求属性中的下界属性*(Lower Bound)*和上界属性*(Upper Bound)*，分别表示为 $A_{v_0.L}$ 和 $A_{v_0.U}(\forall v_0 \in V_G)$；$M$ 是一个 $|V_P| \times |V_P|$

的矩阵，对该矩阵中的任一项 (i,j)*(*其中 $i,j \in V_P$*)*，$M[i,j]_{.L}$ 和 $M[i,j]_{.U}$ 分别表达虚拟机（节点）对 (i,j) 的用户资源及 *QoS* 需求属性中的下界和上界；L 是一个正常数，表示了该模式的生存周期；RT 是一个正常数，表示了该模式的响应时间上界。

□

对于 matching 和 embedding 两种语义约束，在前面更新的图模式模型中主要体现在两个方面：第一，两种语义对应的虚拟网络映射问题的使用环境，前者是仅有节点虚拟化的情况，后者是节点、链路均支持虚拟化的情况，与原始的图模式模型中 matching 和 embedding 定义相同；第二，两种语义对基图中的 R_V 和 R_E 的处理。在 embedding $\trianglerighteq_{f,p}$ 语义下，$\forall i \in V_G, R_{V_G}(i) = A_{V_G}(i)_{.L} - \sum_{\forall vnode \quad on \quad i} \{A_{V_P}(vnode)_{.L}\} = A_{V_G}(i)_{.L} - \sum_{\forall j \in V_P, f(j)=i} \{A_{V_P}(j)_{.L}\}$。

6.2.2 两种语义约束下模型最优解

对于两种语义约束（matching/embedding）下图模式模型的最优解求解，与前面的一般情况下的模型求解侧重点有所不同。前几章已经分析过，图模式模型的最优解求解是 NPO 完全的，复杂度很高且不可近似。但是，在数据中心网络中在线虚拟网络映射的场景下，对应的基图的链路的下界属性远大于模式中节点对的节点对下界属性，因此可以采用相应的启发式规则利用这一特点。更近一步，数据中心网络中在线虚拟网络映射具有相应时间的要求和请求映射效率的稳定性的需求——即需要处理随机的密集的虚拟网络映射请求。这种应用场景的特点使得图模式模型中的基图具有类似于前面给出模型最优解特殊情况下的 non-blocking 性质，虽然求最优解仍是 NP 难的，但已经不是 NPO 完全优化问题了，判定其实例的可行解是否为空也不再是 NP 难的了。因此，在这种场景下，利用图模式模型，我们将侧重于模式节点的约束，映射效率，以及基图中模式相应节点的位置与密度（分别对应于虚拟网络节点映射位置和物理网络节点的资源利用率）。在后文中会详细规范说明。

更重要的，这种在线的、随机的、密集型的虚拟网络映射应用，其根本目标并非是前面的每次执行 matching 语义和 embedding 语义操作时取得最优解，而是要使得数据中心网络资源利用率在长时期内分配最优化，即达到长时期的高资源利用率。

这对应到图模式模型中则表现为需要考虑多个模式如何进行 matching 和 embedding 的情况。

新的图模式模型具有与前几章中一般化分析时不同的特点，具体如下：首先，在这种高速以太网数据中心网络里的在线虚拟网络映射场景下，图模式模型需要处理多个模式同时映射的情况。其次，由于一般情况下的图模式模型的两种语义操作的复杂度很高，而当前应用场景下提出了对映射响应时间的需求，因此需要利用这种场景下的模型中基图的 non-blocking 性质。但是 non-blocking 性质并非长期存在，由于在线虚拟网络请求的到来，虽然基图中边的固有下界属性要远大于模式中节点对的下界属性，但是在长期范围内仍会出现 $R_E(e) < M[i,j](i,j \in V_P, e \in E_G)$ 的情况。第三，模型的两种语义操作的最优解并非应用所需的最优效果，及单次的模式执行 matching 和 embedding 操作达到最有效果并非应用场景的需求，而是需要具有长期的对基图中活动节点下界属性的利用程度（即数据中心网络中资源较高的利用率），这种长期的优化目标基图模型不能形式化表达。第四，模型中基图的节点规模超大，而模式的节点规模往往较小（公有云中应用的分布式规模较小），即 $|V_G| >> |V_P|$。

6.3 针对图模式模型的策略博弈：Local Move Search Game(LMSG)

在前面提到过，虽然根据应用场景的需求更新了图模式模型的定义，但是模型仍不能表达应用场景的优化目标。基于图模式模型，我们提出了一种基于策略博弈的优化方法——Local Move Search Game(LMSG)。LMSG 策略博弈模型基于图模式模型，能对图模式模型的语义操作结果进行阶段性的优化，以达到数据中心网络所要求的长期的较高资源利用率、低成本的目标。本节主要给出 LMSG 策略博弈模型的定义、性质、相关策略搜索算法，后面将给出一种半在线方法，通过组合利用 LMSG 和图模式模型，处理数据中心中在线虚拟网络映射请求，并达到保持较高的长期数据中心网络资源利用效率的效果。

6.3.1 基本策略博弈模型 LMSG 及变种

策略博弈 LMSG 模型是应用于当多个模式经过图模式模型的两种语义操作后，对操作结果的进一步优化。其主要思想是将每一个前段时间执行 matching/embedding

操作的模式视作博弈的参与者（Player），将执行这若干 matching/embedding 操作之前的基图的当前剩余资源（属性）R_V 和 R_E 视作博弈的竞争资源。下面给出严格的描述。

定义 6.3 ***LMSG****(Local Move Search Game)*

给定基图 $G = (V_G, E_G, A_V, A_E, R_V, R_E)$，存在若干已执行 *matching/embedding* 操作的待优化的模式*:* $P_i = (V_{P_i}, A_{V_{P_i}}, M_i)(i \in \{1, 2, ..., m\}$，其中 *m* 是当前待优化的模式数目*)*，在 *LMSG* 中，P_i 是参与者，P_i 能通过移动其节点来改变其映射状态（从状态 *M(f,p)*改变为 *M'(f',p')*），以减小其在博弈模型中的费用 $c(P_i)$。这里状态 *M* 情况下模式 P_i 在 *LMSG* 中的费用定义为 $c_M(P_i) = \sum_{\forall n \in V_{P_i}} \{\frac{A_{V_{P_i}}(n).L}{A_V(f(n)).L - R_V(f(n))} \cdot A_V(f(n)).U\} + \sum_{\forall e \in E_G : e \in p(s,t)(s,t \in V_{P_i})} \{\frac{M_i[s,t].L}{A_E(e).L - R_E(e)} \cdot A_E(e).U\}$.一个 *player* P_i 从有效映射状态 *M* 改变到有效映射状态 *M'*，当且仅当 $c_M(P_i) \leq c_{M'}(P_i)$. 我们定义一个 *LMSG* 的总费用为 $s = \sum_{\forall P_i (i \in \{1,2,...,m\})} c_M(P_i)$.

□

从定义可以看出，LMSG 促使每个模式的节点尽量分布在当前已经在使用的基图节点中，这对应于实际应用场景中则为负载汇聚，即使得在线虚拟网络请求尽量使用当前使用率较高且能满足其资源需求的物理网络节点和链路，从而实现动态的负载汇聚，继而降低公有云计算数据中心运营成本。

根据前面对应用场景的分析，虚拟网络中链路资源请求往往远小于由廉价 PC 通过高速以太网连接而成的云计算数据中心的链路带宽资源，且很多应用场景下虚拟网络映射往往对应于图模式模型中的 matching 语义，因此在线密集型虚拟网络映射请求处理的瓶颈往往在于数据中心网络中的节点资源。这种应用特点在图模式模型对应为 non-blocking 性质，这也产生了 LMSG 的一个特殊情况(non-blocking)的变种策略博弈——**NLMSG**(Nodes Local Move Search Game)。

定义 6.4 ***NLMSG****(Nodes Local Move Search Game)*

模式 P_i 在 *NLMSG* 中的费用仅有节点费用，不记边的费用（相应于图模式模型中的 *non-blocking* 性质），即：

给定基图 $G=(V_G,E_G,A_V,A_E,R_V,R_E)$，存在若干已执行 *matching/embedding* 操作的待优化的模式*:* $P_i=(V_{P_i},A_{V_{P_i}},M_i)(i\in\{1,2,...,m\}$，其中 *m* 是当前待优化的模式数目*)*，在 *NLMSG* 中，P_i 是参与者，P_i 能通过移动其节点来改变其映射状态（从状态 *M(f,p)*改变为 *M'(f',p')*），以减小其在博弈模型中的费用 $c(P_i)$。这里状态 *M* 情况下模式 P_i 在 *LMSG* 中的费用定义为 $c_M(P_i)=\sum_{\forall n\in V_{P_i}}\{\frac{A_{V_{P_i}}(n).L}{A_V(f(n)).L-R_V(f(n))}\cdot A_V(f(n))_{.U}\}$。一个 *player* P_i 从有效映射状态 *M* 改变到有效映射状态 *M'*，当且仅当 $c_M(P_i)\leq c_{M'}(P_i)$. 我们定义一个 *LMSG* 的总费用为 $s=\sum_{\forall P_i(i\in\{1,2,...,m\})}c_M(P_i)$。

□

为了便于分析 NLMSG 的性质，下面给出 NLMSG 的一个特例变种，其保留了 NLMSG 的根本性质，且更简单便于分析。

定义 6.5 ***UNLMSG****(Uniform Nodes Local Move Search Game)*

给定基图 $G=(V_G,E_G,A_V,A_E,R_V,R_E)$，存在若干已执行 *matching/embedding* 操作的待优化的模式*:* $P_i=(V_{P_i},A_{V_{P_i}},M_i)(i\in\{1,2,...,m\}$，其中 *m* 是当前待优化的模式数目*)*，在***UNLMSG***中，我们将经过 *matching/embedding* 操作 (f,p) 所有待优化的模式 $P_i(i\in\{1,2,...,m\}$ 中的节点同等看待，均作为***UNLMSG***的参与者。即在状态 $M(f,p)$ 下，$\forall n\in\bigcup_{\forall i\in[1,m]}V_i, c_M(n)=A_V(f(n))_{.U}\cdot\frac{A_{V_P}(n).L}{A_V(f(n)).L-R_V(f(n))}$.这里的 n 即为***UNLMSG***的参与者。假若每个参与者均是理性的，则参与者 n 从状态 *M(f,g)*转移到 *M'(f',g')* 当且仅当

- 参与者 n 的费用减少，即 $c_M(n)>c_{M'}(n)$；
- 移动后节点 $f'(n)$ 上的其它参与者的费用不会增加，即 $\forall n'\in V_P\cup f'(n')=f'(n), c_{M'}(n')\leq c_{M'}(n)$.

□

6.3.2 模型性质分析

前面给出了针对在线密集型虚拟网络映射请求在经过图模式模型在 non-blocking 条件下的语义处理后的优化模型，三个策略博弈模型——**LMSG**，**NLMSG**，**UNLMSG**。下面分析其性质。

引理 6.6 判定***LMSG***是否存在纳什均衡是 *NP* 完全问题。

□

证明：首先，我们很容易检验判定**LMSG**是否存在纳什均衡是 NP 类的。给出 NP 算法如下：先随机选取**LMSG**的状态 M，然后检验其是否是纳什均衡。显然检验 M 是否是纳什均衡能在多项式时间内完成。因此判定**LMSG**是否存在纳什均衡是 NP 类问题。下面证明其是 NP 难的。

我们通过构造一个从 3-SAT 问题（NP 完全）到**LMSG**的纳什均衡存在性判定问题的 Karp 归约来证明该判定问题是 NP 难的。

给定 3-SAT 问题的实例 ϕ:布尔变量集合 $\{x_1, x_2, ..., x_n\}$，项集合 $\{C_1, C_2, ..., C_m\}$，$\phi = C_1 \cup \cdots \wedge C_m$。每个项 $C_k = l_{k_1} \vee l_{k_2} \vee l_{k_3}$，其中 $l_{k_i} \in \{x_j, \neg x_j\}, x_j \in \{x_1, ..., x_n\}$. 下面根据 3-SAT 问题的实例 ϕ 构造出一个基图 G_B，若干个模式 P_i，使得 3-SAT 问题实例 ϕ 存在真值赋值使其为 1，当且仅当有所购造成的模式和基图组成的**LMSG**实例存在纳什均衡。

相应于 3-SAT 实例 ϕ 的**LMSG**实例构造如下：

基图 G_B**：** $G_B = (V_G, E_G, A_V, A_G)$ 定义如下：

- $V_G = \{C_1, C_2, \cdots C_m, x_1, \neg x_1, x_2, \neg x_2, \cdots, \cdots, t_1, t_2, ...t_{m+n+2}\}$;

- $E_G = E_1 \cup E_2 \cup E_3$.其中 $E_1 = \{(x_i, \neg x_i\}(i \in [1, n])$,
  $E_2 = \{(l_{ki}, C_k)\}(i \in \{1, 2, 3\}, k \in \{1, 2, ..., m\}$,
  $E_3 = \{(t_i, t_{i+1})\} \cup \{(t_{m+n+2}, t_1)\}(i \in \{1, 2, ..., m+n+1\})$;

- $A_V(n)_{.L} = A_V(n)_{.U} = 1(\forall n \in V_G$;
- $A_E(e)_{.L} = A_E(e)_{.U} = 1$;

模式： 分两部分说明：

- （第一类）m+1 个模式 $P_i(i \in \{1,2,...,m+1\})$: 对每个模式，其初始实例形如($\epsilon \to \alpha$)，其中节点的下界属性和上界属性相等，均为 $\alpha > \frac{1}{2}$，$\epsilon < \frac{1-\alpha}{m}$. 相应边的下界属性为 1。
- （第二类）n 个模式 $P_j(j \in \{1,2,...,n\}$: 对每个模式，其初始实例形如($\delta \to 1$);其中节点下界属性等于上界属性，为 $\delta = 1 - m\epsilon$。边的上下界属性均为 1。

下面验证上面构造的**LMSG**的实例 m+n+1 个模式在 G_B 上存在纳什均衡，当且仅当存在真值赋值使 3-SAT 实例 ϕ 为真。

首先，当存在真值赋值使 3-SAT 实例 ϕ 为真时，我们将以上参与者（模式）按照以下方法映射到基图上：为 G_B 中由每个 $C_j(j \in [1,m]$ 组成的部分分配一个第一类参与者(模式 P_i)。其中 ϵ 节点只能被映射到 G_B 中的变量节点(x_i 或者 $\neg x_i$)，α 节点只能被映射到 G_B 中相应的项节点 C_j。将 n 个第二类参与者(P_j)映射到 G_B 中的 n 对节点($x_i, \neg x_i$)，其中 ϵ 节点映射到 δ 节点上。因为 3-SAT 的实例 ϕ 是可满足的，因此以上映射分配均成立，将最后剩下的一个 $(\epsilon \to \alpha)$ 映射到 G_B 中由 t_i 节点构成的环上。很容易看出此时**LMSG**实例是一个纳什均衡。因为明显不存在其他的映射方法（状态）能避免不在环上放置模式中的边（容易验证，多余一个时**LMSG**状态不稳定），也没有任何参与者（模式）能有一个使其费用减少的替代映射方案。

另一方面，假设不存在真值赋值使的 3-SAT 实例 ϕ 可满足，可以证明不存在**LMSG**的实例，及模式的映射方案，使得存在一个以上的第一类 pattern 共存于 G_B 中由 t_i 组成的环上。因此，我们分配一个第一类的参与者到环上，然后将剩余的映射到有 C_j 构造出的部分上。容易看出，我们能映射 n 个第一类模式到形如 $(x_j \to \neg x_j)$ 这样的边上，但是仍然存在 m 个没有被分配。由于 ϕ 不能被满足，因此形如 $(l_{ki} \to C_k)$ 这样的边数目少于 m 个。因此我们必须在环上分配至少两个第一类模式，造成不稳定状态。

因此判定**LMSG**是否存在纳什均衡是 NP 完全问题。

□

定理 6.7 ***UNLMSG***存在纳什均衡。

□

证明：事实上，**UNLMSG**的参与者的费用之间存在偏序关系。我们称参与者的费用向量 P' 小于 P，记作 $L(P') < L(P)$，即存在一些列状态跳变，使得**UNLMSG**能从状态 P 跳变至 P'，使得 $p'^{(i)} \leq p^{(i)} (\forall i \in [1,k])$，且 $p'^{(k)} < p^{(l)}$。这表明存在一个或多个最小的参与者费用向量。由于每次跳变（假设参与者 x_j 从 base graph 中的节点 a 移动至 b，则此次移动所影响的参与者费用只会变小，而原来比其小的某个费用会增大，但是不会超过 x_j 费用在费用向量中的位置。因此，可在有限步跳变（移动）达到费用向量中的最小值，即是纳什均衡。

□

在知道**UNLMSG**存在纳什均衡后，我们有必要知道其 PoA(Price of Anarchy)。

推论 6.8 ***UNLMSG***的 *PoA* 为 *2*。

□

下文中将利用**UNLMSG**即若干约束条件，实现一种在线虚拟网络映射请求处理方法。其中将会对**UNLMSG**的策略移动提供剪枝搜索。对于非 non-blocking 情况下的图模式模型，则可使用**LMSG**，但其移动策略的选择是 NP 完全问题。

6.4 在线虚拟网络映射请求处理：一种半在线方法

本文提供了一种半在线式虚拟网络映射请求处理方法，除了能控制响应时间和适应在线请求到来密度的变化外，基于前面给出的策略博弈模型，还能优化长期的映射效果。

半在线虚拟网络映射请求处理过程包括三大步：

（1）在线虚拟网络映射请求的接受过程；

（2）虚拟网络映射请求预部署；

（3）基于策略博弈(NLMSG)的预部署方案优化；

6.4.1 在线虚拟网络映射请求接收过程：滑动窗口机制

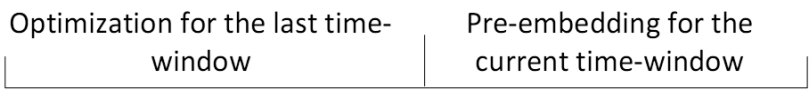

图 6.1: 虚拟网络请求接收窗口

我们使用一个自适应性的滑动时间窗口来接收在线虚拟网络映射请求，即图模式模型中的若干模式。该滑动时间窗口与所接收的模式的响应时间 RT 要求、当前模式产生密度，以及最终映射结果的质量密切相关。如图6.1所示，一个时间窗口包含有两个部分，前一部分表示的是对上一时间窗口中所接收并预部署的模式的优化过程（基于前面给出的三种策略博弈）；后一部分表示预部署当前时间窗口内所接受的pattern。我们将影响滑动窗口大小的影响因素分为两大类，前端因子和后端因子，分别表示当前所接收到的所有 pattern 的固有属性 RT 的平均值 $\bar{RT}$ 和当前 pattern 产生的密度 $AR(arrivalrate = arrivepatterns/timeunit)$。

具体的窗口大小调节机制如下：

1. 初始化时，设置时间窗口宽度为 0；
2. 当所有接收到的模式均在各自的 RT 的限制内完成优化时，逐步增大时间窗口宽度（如：以 AR 的单位）；
3. 当存在接收的模式不能在其固有属性 RT 的限制内完成优化时，逐步减小窗口大小；
4. 如果 AR 增大但当前 $\bar{RT}$ 没有减小，则转向第 2 步；
5. 如果 $\bar{RT}$ 减小，则转向第 3 步；

在实施过程中可对窗口大小限定一个固有数目区间（可接收模式的最小数目和最大数目）以确保窗口调节的稳定性。

6.4.2 预处理及预映射

此阶段将会处理前面接收阶段所接收的模式，具体包括预处理和预映射两个子阶段。由于前面分析过，在数据中心网络应用场景下，基图 G_B 的节点数目远大于在线接收到的模式的节点数目，即 $|V_G| >> |V_P|$。因此，我们在预映射阶段如果要求出最优映射方案则会花费很长时间。鉴于这种应用场景下对 non-blocking 性质的支持，本文提出一种“预处理 + 预映射”的实用方法。其主要思想是将所接收到的模式分解成若干个初始实例为星型的子模式，然后并行地将这些星型子模式映射到（执行 matching 或 embedding）基图上。对于星型结构的 matching 和 embedding，均已有了启发式算法，本文也给出了一种新的具有所有边 QoS 保障（虽然有 non-blocking 性质，但是由于是负载动态汇聚式的映射，因此在 embedding 语义下仍有可能出现边 QoS 无法满足的情况）的星型结构的映射算法。下面详细给出预处理和预映射算法说明。

A. pattern 预处理

下面给出多项式时间算法，能将所接收的模式分解为初始实例为星型的若干子模式，并且尽量较小所去掉边的权值之和。

定义 6.9 定义模式的直径为其初始实例中最长的任意两节点间最短路。

显然，星型图形的直径不大于 2。

下面给出模式预处理算法。算法包含两大步：第一步，求出输入模式的初始实例的最大生成树 T；第二步，利用下面的算法1处理 T。

算法1通过维护数组 $sort[0:p]$ 分治递归地将 T 分割成若干直径不大于 2 的子结构，即星型。

B. 模式预映射

将所有接收的模式预处理后，得到了一些列初始实例为星型的子模式。下面将并行的对这些星型进行预映射。预映射不需要达到最优映射效果，其目的在于寻找一个可行的映射方案（借助与 non-blocking，以负载汇聚为启发式规则）。预映射过程可分为两步：第一，将与处理后的星型模式映射到基图上（可并行）；第二，通过在基图中寻找相应的路径恢复在预处理过程中去掉的（模式的初始实例的）边。

Algorithm 1 minimum star-cut-in-tree algorithm

Input:

maximum spanning tree of a received pattern: T;

sort[0:p]: a sorting of edges which is always in a ascending order;

Output:

rec[0:q]: records of the output;

Backtrace(T)

1: **if** d(T)$\leq$ 2 **then**
2: return;
3: **end if**
/*T_1 and T_2 denote the separated parts of T, respectively*/
4: current=sort[t];/*($sort[t] \in T, \forall i < t, sort[i] \notin T$)*/
5: **if** $d(T_1) \geq 2 \&\& d(T_2) \geq 2$ **then**
6: delete current from record;
7: **Backtrace**(T_1);
8: **Backtrace**(T_2);
9: **else if** $d(T_1) + d(T_2) \leq 5$ **then**
10: i=t;
11: **while** (!($sort[i] \in T \&\& d(T_1) \leq 2 \&\& d(T_2) \leq 2$)) **do**
12: i++;
13: **end while**
14: current=sort[i];
15: add current to record;
16: delete current from sort;
17: return;
18: **else**
19: i=t;
20: **while** (!($sort[i] \in T \&\& d(T_1) > 2 \&\& d(T_2) > 2$)) **do**
21: i++;
22: **end while**
23: add current to record;
24: delete current from sort;
25: **Backtrace**(T_1);
26: **Backtrace**(T_2);
27: **end if**

定义 6.10 一个节点 *N* 被称为与一个星型 *C* 相邻，当且仅当 *N* 与 *C* 的中心节点相邻。我们如果 *N* 与 *C* 相邻，我们称 *N* 是 *C* 的一个邻居节点*(adjacent node)*。

□

已有针对树形、星型的虚拟网映射算法，已有算法往往关注的是单次映射的优化程度，且大都是负载均衡。下面的给出一个负载汇聚式的星型子结构预映射方案，算法2是其伪代码形式。其主要思想是通过星型子结构的中心节点判定星型子结构的位置，然后分别映射其各个叶节点。

由于不同模式之间的星型子结构的预映射过程相互独立，因此以上过程可以并行进行以加快预映射速度。在星型子结构映射完后，我们还需要将属于同一模式的星型子结构连接起来。下面给出了具有节点、链路 QoS 保障的连接算法。

算法3将已经预映射的 star-cluster 连接起来，同时对连接的路径进行 QoS 检测，如果不满足（概率较小，因为在 non-blocking 性质下的基图上进行模式 embedding 时，瓶颈在节点的下界属性上），则进行有深度限制的 QoS 保障性搜索，以寻找有效连接路径。

6.4.3 基于(UN)LMSG 的预部署方案优化

预映射完成后，将给予**(UN)LMSG**对预映射方案进行动态负载汇聚式的优化。前面已经证明，**UNLMSG**存在纳什均衡。我们可以对预映射模式节点进行策略移动，达到全局优化的目的。下面首先给出基于**UNLMSG**优化过程，及策略移动算法。

事实上，基于**UNLMSG**进行预映射方案优化是建立在两个条件的基础之上的：第一，模式中节点的下界属性约束是模式是否存在 embedding 的瓶颈所在，即相对于节点下界属性的严格约束而言，基图在这种应用场景下默认具有 non-blocking 性质；第二，基图中节点支持重游性（见第一章虚拟网络特点简介）。对于第一个要求，正常情况下在数据中心网络中均成立。对于第二个条件，一般对于虚拟机集群环境下也是成立的，不过对于 Internet 虚拟化场景则不成立。不顾可以再其策略移动过程中通过限制同一模式中的节点不共存，对策略移动搜索进行剪枝约束，可以避免依赖于第二个条件，但是会造成性能优化方面的损失。

很明显，**UNLMSG**的最优移动策略搜索问题实际上是一个背包问题，准确来讲，对于每一个可能的策略移动，均可用一个背包问题来建模，其背包是当前考察的移动目的节点，而物品则是当前要进行策略移动的虚拟节点和当前考察的移动目的节点上已有的虚拟节点。下面给出一个基于随机选择与 FPTAS 背包问题算法

Algorithm 2 Pre-embedding star-clusters

Input:

G_B;//substrate network

$G_P[0:m-1]$ denotes the initial instantiations of accepted patterns;

star-cluster[0:m-1][0:n-1] denotes the star-clusters;

Star-Cluster-Embedding1(G_P[]**,star-cluster[][])**

1: **for** (i=0; i<m; i++) **do**
2: **for** (j=0; j<n; j++) **do**
3: if **JudgeQoS1**(star-cluster[i][j])==true
4: embed star-cluster[i][j] onto G_B;
5: update G_B's information;
6: else
7: put $G_P[i]$ into waiting queue;
8: break;
9: **end for**
10: **end for**

/*suppose $V_G[0:n]$ is sorted in descending order according lower bound attribute;*/

JudgeQoS1(cluster)

11: **for** (i=n-1; i>=0; i--) **do**
12: if $V_G[i]$ isn't active
13: continue;
14: **CheckEmbed**(cluster, $V_G[i]$)==false
15: continue;
16: **end for**
17: if cluster isn't embedded
18: **for** each idle nodes $V_G[k]$ which is adjacent to active nodes **do**
19: if **CheckEmbed**(cluster, $V_G[k]$)== true
20: return true;
21: **end for**
22: return false;

/* $snode \in V_G[0:n]$ */

CheckEmbed(cluster, snode)

23: if $(A_{G_V}(snode) \geq A_P(c-cluster))\wedge$ there is no adjacent nodes of cluster already embedded onto snode
24: **for** $i \in$ cluster **do**
25: $\exists n$ adjacet to snodes, $A_P(i)_{.L} \leq A_{G_V}(n)_{.L}$
26: return true;
27: **end for**
28: return false;

Algorithm 3 Connect Clusters

Input:

vlink[0:p] remaining virtual links connecting the star-clusters;
M is the star-cluster pre-embedding solution(M(vlink[i]).left and M(vlink[i]).right denotes the two endpoints)

PreEmbeddingEdges(vlink[], M)

```
1:  for (i=0; i<p; i++) do
2:    if the shortest path between M(vlink[i]).left and M(vlink[i]).right meets the require-
      ment of vlink[i] then
3:      pre-embed vlink[i];
4:    else
5:      if QosGuarranteeSearch(vlink[i], M)==false
6:      put the correspond VN which containts vlink[i] in to waiting queue.
7:      return false;
8:    end if
9:  end for
    /*vlink denotes a virtual link which remains to be pre-embedded;*/
```

QosGuarranteeSearch(vlink, M)

```
10: find two sets of candidate nodes csn1 and csn2, which meet the pre-embedded vnodes
    and vlinks, for M(vlink).left and M(vlink).right.
11: if there exist a path connecting csn1 and csn2 meets the resource reqirement of vlink
    then
12:   return ture;
13: else
14:   return false;
15: end if
```

的**UNLMSG**策略移动选择算法——算法4。

当使用**UNLMSG**时，可直接使用前面给出的基于背包问题的随机伪多项式时间算法。使用**UNLMSG**的前提条件是数据中心网络虚拟化应用场景下，对应的图模式模型中所有模式映射的瓶颈均是节点下界属性的限制。

当是用一般化的**LMSG**时，不需要这种对应用场景特点的依赖。但是，前面已经证明了判定**LMSG**是否存在纳什均衡是 NP 完全问题。因此直接使用不能确保能达到优化效果，且不能判定博弈是否会终止。这里我们结合实际应用特点，对**LMSG**的移动策略加以约束，我们限制在**LMSG**中，参与者 P_i 的一个有效的移动($M \to M'$)必须满足一下三个条件：

Algorithm 4 UNLMSG Local Move Strategy

Input:

- G_B: base graph
- $P_i(i \in [1, m])$: players about to make a move

Local-Move-Search(G_B**,**$p[]$**)**

flag=1
while flag==1 **do**
  flag=0;
  for (i=1;$i \leq m$;i++) **do**
    if FPTAS-knapsack($V_{temp} = random(V_{G_B})$, P_i)==True **then**
      $P_i \rightarrow V_{G_B}$
      flag=1;
    end if
  end for
end while
/*FPTAS for knapsack*/
FPTAS-knapsack(v_G**,**P_i**)**
Given $\varepsilon > 0$, let $K = \frac{\varepsilon \cdot \max\limits_{\forall vnode \in v_G \cup P_i} w(vnode)}{n}$;
For each $vnode_i$ on v_G,define $w'(vnode_i) = \lfloor \frac{w(vnode_i)}{K} \rfloor$
$w'(P_i) = \lfloor \frac{w(P_i) = A_P(P_i).L}{K} \rfloor$
With these as weights of vnodes, using the dynamic programming algorithm, find the most profitable set, say S'
if $P_i \in S'$ **then**
  return True;
end if
return False;

- $c_M(P_i) < c_{M'}(P_i)$；
- 移动仅包含一个节点移动或者一条路径移动；
- 新的状态 M' 必须满足所有参与者的资源需求。

应用这种加强限制的**LMSG**能减少搜索空间大小。在实际应用中，配合博弈时间限制，可以起到一定程度的优化效果。同样，对于**UNLMSG**，同样可以加上节点颜色的约束，使得统一模式的不同节点不能共存于一个基图节点上。

6.5 仿真实验

本节对前面给出的半在线方法进行仿真实验检验。检验半在线方法与已有的虚拟网络映射算法在处理在线密集型虚拟网络映射请求时的效果差异。

6.5.1 实验环境

物理网络：使用 GT-ITM 工具随机生成物理网络拓扑。物理网络节点数表示为 n_S，在生成物理网络过程中，我们取 n_S 在区间[50,500]内。物理网络节点随机互连，每条边产生的概率是 0.5。物理网路节点和边的下界属性值随机分布于区间 $(r_S^N, r_S^N + 100]$ 和 $(r_S^L, r_S^L + 100]$。这里 r_S^N、r_S^L 是用来检验物理网络（对应于基图）的 non-blocking 性质对实验结果的影响大小的，均从区间(0,100]中依照相应的实验目的选取。

虚拟网络映射请求：我们假设虚拟网络映射请求是服从 Poisson 分布在线到来的，该 Poisson 分布的参数为一个正整数 λ（根据实验需求设定）。每个虚拟网络映射请求对应的模式具有 n_V 个节点，其中 $n_V \in [5, 50]$。生存时间服从指数分布，指数分布的参数确定为 1000 单位时间（与现有研究一致）。虚拟网络节点和链路的资源请求分别为 $(r_V^N + 40, r_V^N + 50]$ 和 $(r_V^L, r_V^L + 10]$，其中 r_V^N 和 r_V^L 服从区间(0,50]上的均匀分布。每个虚拟网络映射请求的响应时间要求服从 $(t_r, t_r + 10]$（单位时间）均匀分布，其中 t_r 是正整数。

6.5.2 实验结果

本小节将给出本章提出的半在线虚拟网络映射请求处理方法与现有虚拟网络映射算法之间的对比仿真实验。在前面介绍的仿真实验环境下，主要测试各算法对在线请求、物理网络资源属性下界远大于虚拟网络资源属性下界情况、负载汇聚与负载均衡等等情况下的性能差异。这些测试均通过改变表6.1中所列举的参数来完成，同时表6.2也给出了两种测试度量标准。

下面表6.2给出了各种算法的简称及特点，将会在下面的实验数据图示中使用。

注意到现有的很多算法，如表6.1中的 MIP、Path-splitting 等均是基于 multi-path routing 的假设来设计算法的，即任两个虚拟机之间可以通过多条路径同时连通，这

表 6.1: 试验参数及度量标准

Evaluation Metrics	Act.Util.	物理网络活动节点资源利用率 $Act.Util = \frac{\sum_{each\ active\ node\ n} (1 - \frac{R_V(n)}{A_{G_V}(n)_L})}{\sum_{each\ active\ node\ n} 1}$
	Acc.Rat.	请求接受率
Experimental Parameters	λ	请求平均密度(Patterns / 100time units)
	t_r	请求响应时间分布区间参数
	r_S^N	物理网络节点下界属性分布区间参数
	r_V^N	虚拟网络节点分布区间参数
	n_S	物理网络节点数目
	n_V	虚拟网络节点数目

表 6.2: 对比算法简记表

简记号	算法描述	前提假设
semi-online	论文本章基于 LMSG 的半在线算法	——
semi-online(M)	支持 path-splitting 的 semi-online 算法	Multi-path
2stage [12]	节点贪心选择、链路最短路算法	——
M-2stage	支持 path-splitting 的 2stage 算法	Multi-path
MIP [14]	基于混合整数规划(MIP)的算法	Multi-path
Subgraph [4]	基于图同构的回溯搜索算法	——
M-Subgraph	支持 path-splitting 的 Subgraph 算法	Multi-path

些路径上所分配到的资源之和满足虚拟网络链路资源下界属性（带宽）即可。因此，为了与这些基于 multi-path 假设的算法进行比较，我们修改了算法2，允许 pattern 中一个节点对映射到 multi-path 上（公共端点）。另一方面，由于现有工作均是集中于一次虚拟网络映射，以负载均衡为启发式规则，因此，我们也对修改了算法 subgraph [1]和 2stage[2]算法记性了修改，将贪心选择策略由负载均衡改为负载汇聚，将改后的算法分别简记为 M-subgraph 和 M-2stage（见表6.1）。

我们将从三个方面来比较这些算法在高速以太网连接而成的公有云计算数据中心网络中虚拟网络在线映射场景下的性能和特点：除了表6.1给出了 Act.Util(活动节点资源利用率)、Acc.Rat（请求接受率）两个度量标准外，我们还将比较在在线请求变化的情况下各个算法的性能稳定性，如请求接受率的波动情况。另外，由于已有工

作均针对的是 embedding 语义下的引入各种假设的网络映射问题，因此下面实验均是指在 embedding 语义约束下的映射实验。

A. 单路径映射(Single path)

A.1: 活动节点资源利用率测试

在单路径映射情况下，将本章的 semi-online 方法与表6.2中的 2stage，M-2stage, subgraph,M-subgraph 算法进行比较。 试验参数设置：$\lambda = 10, n_S = 300, n_V = 10, r_S^N =$

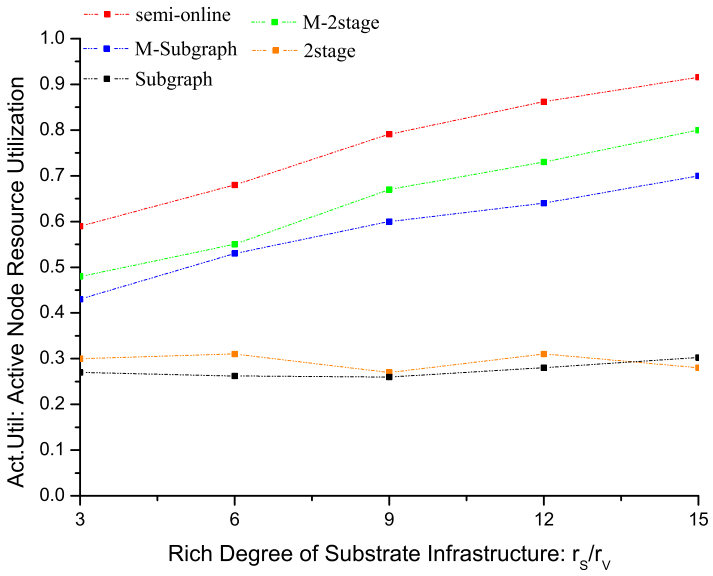

图 6.2: 活动节点资源利用率随 r_s/r_v 的变化关系

$r_S^L = 50, r_V^N = r_V^L = 5, t_r = 80$,runtime=20000 time units。

观察：图6.2各算法对数据中心中物理网络资源相较虚拟网络资源的富余程度（特别是链路资源：non-blocking 属性）的利用程度。可以看出 semi-online(半在线)算法能很好利用这一特点，达到较高的负载汇聚程度。当物理网络与虚拟网络资源下界属性（容量）相差越大时，semi-online 算法负载汇聚的优势就越明显。

A.2: 请求接收率-时间测试

实验参数设置：$\lambda = 10, n_S = 300, n_V = 10, r_S^N = r_S^L = 50, r_V^N = r_V^L = 5, t_r = 80$,runtime=20000 time units.

观察：从图6.3可以看出，在算法运行初期，semi-online 算法的请求接收率最低。但

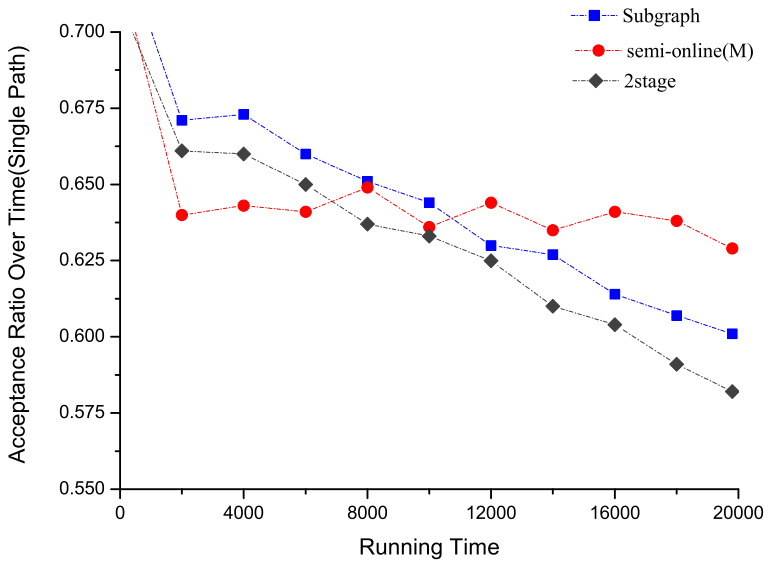

图 6.3: 请求接收率随时间的变化关系

是随着时间的增长，semi-online 的请求接收率降低的速度要小于另两个算法。总体上讲，semi-online 在初期和中期的请求接收率均很低，但是请求接收率相对于运行时间而言很稳定。而其他两个算法则随着运行时间请求接收率较大幅度的下降了。

A.3: 请求接收率-请求密度测试

实验参数设置 $t_r = 30, n_S = 300, n_V = 10, r_S^N = r_S^L = 50, r_V^N = r_S^L = 5$,runtime=10000 time units.

观察：λ 表达了在线虚拟网络映射请求的密度。从图6.4可以看出，当在线请求密度较小时，semi-online 的请求接收率较 subgraph 和 2stage 算法要低，但相差不远。当在线请求密度增大到一定程度后，三者的请求接收率均有所下降，但是 semi-online 受到的影响最小。当 $\lambda > 12$ 时（即没 100 单位时间到来 12 个虚拟网络映射请求），semi-online 算法的请求接收率比 2stage 和 subgraph 要高了。三个对比算法中，2stage 算法下降速度最快。

B. 多路径映射(Multi-path)

对于支持 multi-path 情况，我们将 semi-online 的预映射和优化部分进行修改，提供了对 multi-path 的支持，记作 semi-online(M)(表6.1)。

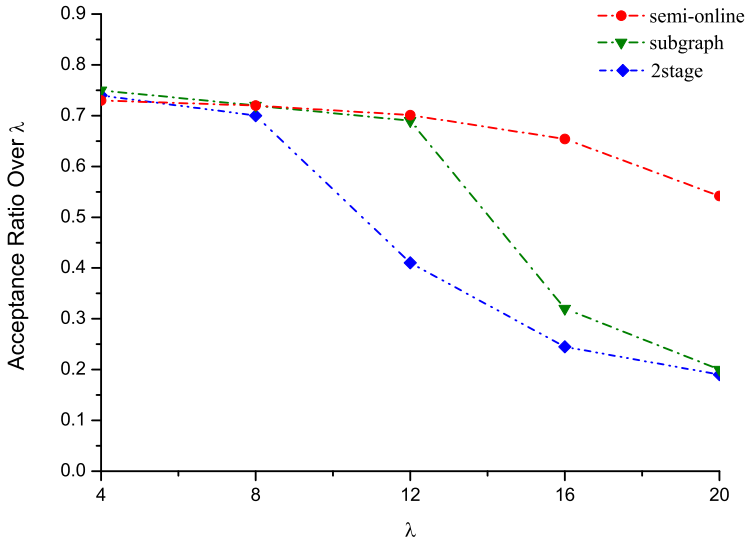

图 6.4: 请求接收率随 λ 的变化关系

B.1: 活动节点资源利用率测试

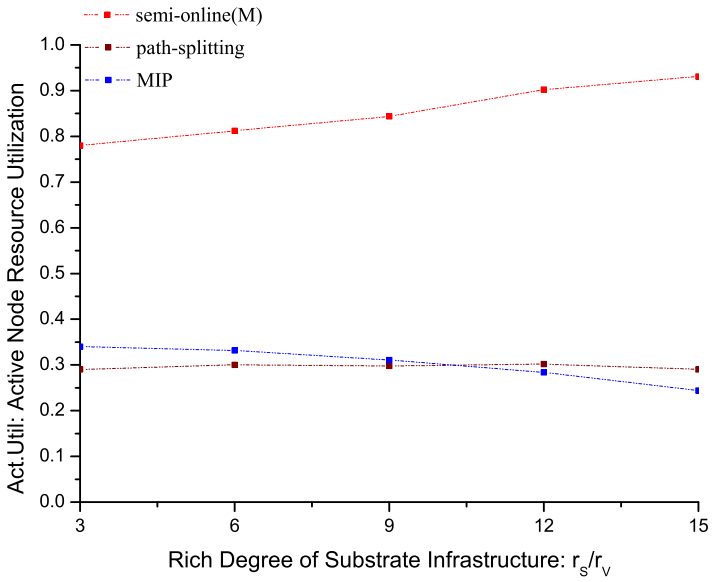

图 6.5: Multi-path 情况下活动节点资源利用率随 r_s/r_v 的变化关系

实验参数设置 $\lambda = 10, n_S = 300, n_V = 10, r_S^N = r_S^L = 50, r_V^N = r_V^L = 5, t_r = 80$,runtime=20000 time units. 观察：由图6.5可以看出，多路径情况与前面的单路径情况

类似，在 r_S/r_V 增大时活动节点资源利用率上升，相反地，算法 MIP 和 path-splitting 略微下降（与 single path 对应情况不同）。

B.2: 请求接收率-时间测试

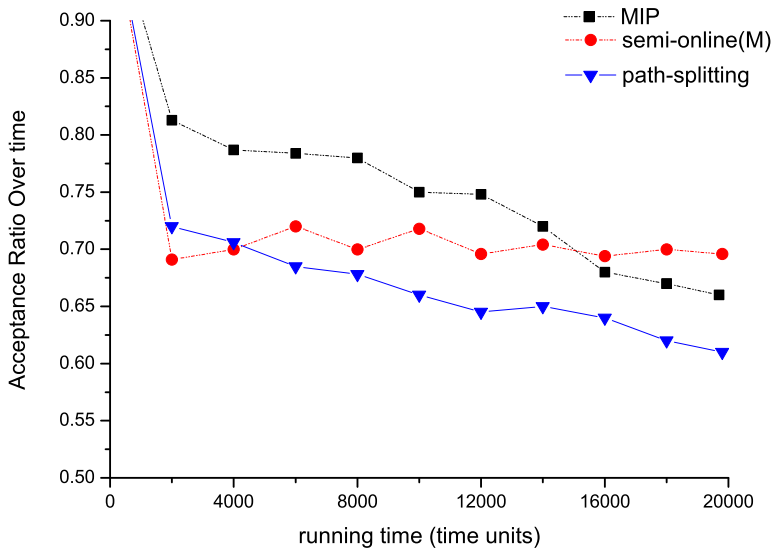

图 6.6: Multi-path 情况下请求接收率随时间的变化关系

实验参数设置：$\lambda = 10, n_S = 300, n_V = 10, r_S^N = r_S^L = 50, r_V^N = r_V^L = 5, t_r = 80$,runtime=20000 time units. 观察：由图6.6可以看出，MIP 算法在初期具有最高的虚拟网络映射请求接收率，直至运行了 14488 单位时间后才将到与 semi-online(M)稍低的情况。semi-online(M)的请求稳定性虽然好，但是一直都处于较低的水平。而 path-splitting 算法则既没有较好的平均请求接收率，也没有较好的稳定性。

B.3: 请求接收率-请求密度测试

实验参数设置：$t_r = 30, n_s = 300, n_V = 10, r_S^N = r_S^L = 50, r_V N = r_V^L = 5$,runtime=10000 time units. 观察：由图6.7可知对于请求接收率随请求密度的变化关系（即请求接收率的鲁棒性），多路径情况与单路径情况类似。semi-online(M)的请求接收率随请求密集程度的变化与 semi-online 在单路径情况类似，但是 MIP 和 path-splitting 的性能下降要比 subgraph 和 2stage 要快。

综合以上现象，我们可以看出，semi-online 算法由于是基于负载汇聚以及利

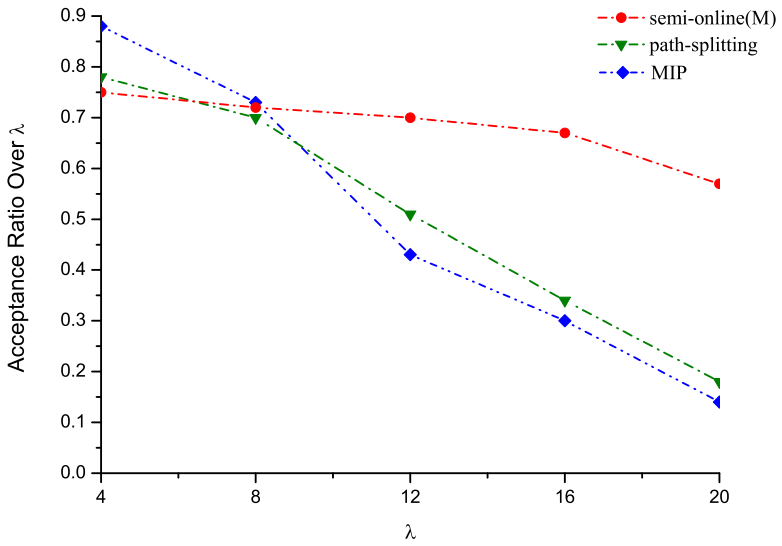

图 6.7: Multi-path 情况下请求接收率随 λ 的变化关系

用数据中心网络里虚拟网络映射场景所具有的 non-blocking 性质，有着较高的活动节点资源利用率。但是由于其请求接收率在请求密度较低时并不比现有的 MIP、path-splitting 等算法要高，不过其稳定性要好，为数据中心提供稳定的虚拟化网络资源服务提供了一定的保证。

6.6 本章小结

本章基于前几种对虚拟网络映射问题本质的理解以及图模式模型，提出了在线虚拟网络映射请求处理这一技术应用问题。随后，本章通过给出三个策略博弈（LMSG 及其变种），分析并证明了博弈模型的纳什均衡存在性判定问题复杂度，基于随机化的背包问题的 FPTAS 近似算法给出了博弈策略搜索算法。并基于策略博弈，提出了一种半在线虚拟网络映射请求处理方式，然后通过仿真实验检验了半在线方式的效果。

结论

网络虚拟化正受到学术界和工业界越来越多的关注。因为其能提供网络规模级别的资源隔离、网络资源按需构建，在新一代互联网、云计算数据中心、网络化软件及服务、GreenIT 中起到越来越大的作用。而虚拟网络映射问题正是网络虚拟化中的基本问题之一。

虚拟网络映射即是要求将用户提出的用来封装网络应用或网络资源的虚拟机网络部署（映射）到一个物理网络上，使得用户提出的虚拟网络节点和虚拟网络链路的资源需求及 QoS 需求，如节点 CPU、节点内存大小，链路带宽、延时属性等要求能得到满足。

过去，虽然有若干虚拟网络映射方面的研究，但均是技术层面的，严重依赖于某些很强的自定义假设设计应用问题的算法，缺乏对问题本身的深入的、系统的研究，包括问题本身的固有复杂度、问题解的特性、问题的理论模型以及一般情况下不依赖于很强的自定义假设的算法（框架）等。本文正是出于这一目的，对虚拟网络映射问题进行有深度的、系统性的理论研究，并结合应用场景，提出了新的虚拟网络映射应用问题，并基于前面对问题固有复杂度和性质的分析，给出应用问题的算法。

本文的主要贡献在于以下几点：

第一，系统性的分析了虚拟网络映射的技术特点、应用需求特点，并对其进行总结、归类，提取本质特征，建立了图模式数学模型严格地刻画虚拟网络及虚拟网络映射问题。

第二，根据虚拟网络技术特点和需求，为图模式模型定义了 matching 和 embedding 两种语义操作（约束），分析并严格证明了两种操作的复杂度，以及两种操作的若干变种问题的复杂度。并分析了模型的相关性质。

第三，基于图模式模型，提出最小化问题。并给出了一种“最大-最小化”方法，通过立方时间的“伪动规”算法解决最小化问题。另外，针对最小化问题的若干变种，给出了严格的分析与复杂度、可近似性证明。

第四，分析了图模式模型求最优解过程中组合优化问题的复杂度，证明了基本优化问题的不可近似性，以及若干变种、特例问题的优化问题近似性分类。尽管问题证明出事不可近似的（NPO 完全），本文给出了两种启发式算法设计框架：构造是启发和枚举式启发。

第五，基于图模式模型，规范化了一个新的虚拟网络映射应用问题——在线密集型虚拟网络映射请求处理问题。并通过策略博弈模型（LMSG）及其变种 UNLMSG 博弈模型，给出了一种半在线处理方法，并通过仿真验证了半在线方法的优势和特点。

总的来说，本文对虚拟网络映射问题进行了系统性的、深入本质的研究，并针对一个新的应用场景，进行了理论联系实际的应用问题解决工作。

在现有工作的基础上，我们还将在以下几个方面开展进一步工作：

第一，目前的虚拟网络映射的处理方法方法是在节点和边的初始映射完成后，在对应虚拟网络生命周期内不在改变。然而，我们可以借助于物理网络中路由的思想，对于虚拟网络节点（虚拟机）映射完后，可以实时动态、自适应性的选择物理网络链路，这样能提高虚拟网络通信可靠性，也能提高物理网络链路资源利用率。同时这也对虚拟网络映射问题的理论建模产生了新的挑战。

第二，由于目前图模式模型的操作复杂度非常高，且紧耦合与虚拟网络映射问题，因此我们下阶段将对图模式模型进一步抽象，以降低其复杂度且增大适用范围，并为虚拟化资源检索与索引建立相关理论模型。抽象的方法是利用 Alternating Atamata 来表达模式。

第三，将模型完善成为一种图模式查询语言，即设计一种 graph pattern language，解决图匹配这一类问题，在虚拟资源查询语言、social network 以及 Large-scale database 中特定的分布式查询中得到应用。

第四，与数据库中 schema discovering 类似，在 graph 中创建 similar graph 发现机制，最终提出一种 graph pattern algebra/schema，准确且快速反应两个模式之间的关系。

致谢

在本文完成之际，谨向所有给予我指导、关心、支持和帮助的老师、领导、同时、同学和亲人们致以衷心的感谢和最崇高的敬意！

衷心感谢我的导师怀进鹏教授！感谢怀老师为我的学习、成长、研究、生活中所给予的无微不至的关怀、悉心的指导、和鼓励关心，尤其是怀老师在百忙之中仍经常抽出时间教导我如何思考问题、如何做研究，为我耐心解答各种问题，提供优秀的科研环境，让我感受到做研究的乐趣。怀老师给予的关怀我不知如何回报，唯有更扎实、更刻苦的专研科学问题，不辜负怀老师的期望。同时，怀老师对科研工作的严谨、细致、求实的作风和敏锐的洞察力、以及对祖国计算机科学事业的热爱和献身精神将永远是我前进的动力和楷模。衷心的感谢怀老师！

非常感谢实验室胡老师、沃老师的悉心指导。胡老师对科研和项目深刻的见解给我留下了非常深的影响，感谢胡老师对我毕设论文的指点，让我受益匪浅，也感谢胡老师对我的实验室工作的关心。非常感谢沃老师对我的指导和帮助，与沃老师的交流让我对毕设工作的应用背景有了更为深刻的理解，感谢沃老师在系统工作非常繁忙之际对我个人研究工作的关心、关注与理解。

特别感谢英国 Edinburgh 大学的樊文飞教授。感谢樊老师对我在 HIT 生活的关注和学习、工作的指导，这为我毕设论文的成型有了直接的作用。也非常感谢樊老师为我打开了界于计算机理论与计算机系统之间的“中间地带”的大门，让我明白了什么叫理论联系实际。

感谢邓婷学姐对我的关怀和照顾，感谢邓婷学姐帮我复印书籍、给我推荐阅读数目，让我少走了许多弯路，衷心感谢学姐！

感谢父母的养育之恩，在我成长、成熟的每一步都有他们的心血和汗水。感谢我的姐姐对我细致的照顾和关爱，让我在北京能安心学习，不用操心生活上的琐事。

参考文献

[1] NM Chowdhury and R. Boutaba. A survey of network virtualization. *Computer Networks*, 2009.

[2] M.R. Garey, D.S. Johnson, et al. *Computers and Intractability: A Guide to the Theory of NP-completeness*. WH freeman San Francisco, 1979.

[3] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. An improved algorithm for matching large graphs. In *Proc. of the 3rd IAPR-TC-15 International Workshop on Graph-based Representations*, pages 149--159. Citeseer, 2001.

[4] J. Lischka and H. Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 81--88. ACM, 2009.

[5] C. Cadere, D. Barth, and S. Vial. Virtualization and allocation of network service resources using graph embedding. In *Computer and Information Sciences, 2008. ISCIS'08. 23rd International Symposium on*, pages 1--6, 2008.

[6] W. Szeto, Y. Iraqi, and R. Boutaba. A multi-commodity flow based approach to virtual network resource allocation. In *Proc. GLOBECOM: IEEE Global Telecommunications Conference*, 2003.

[7] D.G. Andersen. Theoretical approaches to node assignment. *Unpublished Manuscript, http://www. cs. cmu. edu/˜ dga/papers/andersen-assign. ps*, 2002.

[8] Y. Zhu and M. Ammar. Algorithms for assigning substrate network resources to virtual network components. In *Proc. IEEE INFOCOM*. Citeseer, 2006.

[9] J. Lu and J. Turner. Efficient mapping of virtual networks onto a shared substrate. *Department of Computer Science and Engineering, Washington University in St. Louis, Technical Report WUCSE-2006*, 35, 2006.

[10] J. Fan and M. Ammar. Dynamic topology configuration in service overlay networks: A study of reconfiguration policies. In *Proc. IEEE INFOCOM*. Citeseer, 2006.

[11] R. Ricci, C. Alfeld, and J. Lepreau. A solver for the network testbed mapping problem. *ACM SIGCOMM Computer Communication Review*, 33(2):81, 2003.

[12] M. Yu, Y. Yi, J. Rexford, M. Chiang, et al. Rethinking virtual network embedding: Substrate support for path splitting and migration. *COMPUTER COMMUNICATION REVIEW*, 38(2):17, 2008.

[13] I. Houidi, W. Louati, and D. Zeghlache. A distributed virtual network mapping algorithm. In *Proceedings of IEEE ICC*, pages 5634--5640, 2008.

[14] N.M.M.K. Chowdhury, M.R. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. In *IEEE INFOCOM*, 2009.

[15] G. Ausiello. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Verlag, 1999.

[16] V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd, and M. Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems* 1. *Journal of Computer and System Sciences*, 67(3):473--496, 2003.

[17] C. Chekuri, S. Khanna, and F.B. Shepherd. An O ($\sqrt{n}$) approximation and integrality gap for disjoint paths and unsplittable flow. *Theory of Computing*, 2:137--146, 2006.

[18] T.H. Cormen. *Introduction to algorithms*. The MIT press, 2001.

[19] J.H. Drew, D.L. Evans, A.G. Glen, and L.M. Leemis. *Computational probability: algorithms and applications in the mathematical sciences*. Springer Verlag, 2007.

[20] D. Du and P.M. Pardalos. *Handbook of combinatorial optimization*. Springer, 1999.

[21] J. Díaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Computing Surveys (CSUR)*, 34(3):313--356, 2002.

[22] D.S. Hochba. Approximation algorithms for NP-hard problems. *ACM SIGACT News*, 28(2):40--52, 1997.

[23] N. Nisan. *Algorithmic game theory*. Cambridge Univ Pr, 2007.

[24] C.H. Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.

[25] C.H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover Pubns, 1998.

[26] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Springer Verlag, 2003.

[27] V.V. Vazirani. *Approximation algorithms*. Springer Verlag, 2001.