学校代号 10532 学 号 S151000891
分 类 号 TP391 密 级 普通

湖南大學
HUNAN UNIVERSITY

硕士学位论文

SDN/NFV网络架构可生存性算法研究

学位申请人姓名 陶恒
培 养 单 位 信息科学与工程学院
导师姓名及职称 谢鲲 教授
学 科 专 业 计算机科学与技术
研 究 方 向 计算机网络
论 文 提 交 日 期 2018年 5月 8日

学校代号: 10532
学　　号: S151000891
密　　级: 普通

湖南大学硕士学位论文

SDN/NFV网络架构可生存性算法研究

学位申请人姓名: 陶恒
导师姓名及职称: 谢鲲 教授
培　养　单　位: 信息科学与工程学院
专　业　名　称: 计算机科学与技术
论文提交日期: 2018年 5月 8日
论文答辩日期: 2018年 5月 20日
答辩委员会主席: 邝继顺 教授

The Research on SDN/NFV Network Architecture Survivable Algorithm

by

Heng TAO

B.E. (Hunan University of Science and Technology) 2015

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of engineering

in

Computer Science and Technology

in the

Graduate school

of

Hunan University

Supervisor

Professor Kun Xie

May, 2018

湖 南 大 学

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名:　　签字日期:　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权湖南大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

1、保密□，在____年解密后适用于本授权书

2、不保密□。

(请在以上相应方框内打"√")

作者签名:　　签字日期:　　年　　月　　日

导师签名:　　签字日期:　　年　　月　　日

摘　　要

现如今网络整体规模和网络功能类型的复杂度越来越大，新技术对网络性能需求不断提高，传统的网络架构已经无法满足按需调动、快速配置等需求，软件定义网络（SDN）的提出就是为了应对现有网络架构无法解决的问题和突破网络性能瓶颈的限制。利用网络功能虚拟化(NFV)技术将网络资源进行虚拟化，在很大程度上可以解决现有网络无法解决的问题和到达现有网络无法达到的网络性能需求。网络服务运行在实际的物理设备上，这将必然产生不可避免的网络故障，SDN/NFV新型的网络架构在网络的可生存性领域比传统网络架构更加高效的检测、处理和恢复网络故障。

光网络和Overlay网络中为了保证网络的可生存性，基于SDN的架构能高效实现不相交路径路由功能，快速的在源节点和目的节点之间寻找满足一定QoS约束的不相交路由。当主用路径出现故障时，将其承载的业务流转换到备用路径上，从而实现快速的业务恢复，因此快速不相交路径算法的研究如今具有很高的研究价值。网络虚拟化的过程中，如何为虚拟网络在底层物理网中找到满足约束条件且最优的嵌入，并且提供可生存性保护需求，来保证底层物理网资源失效情况下原本虚拟网络的业务正常运行，可生存性虚拟网络嵌入问题的研究已经是NFV研究中的重要邻域。

本文结合当今研究热点，基于SDN/NFV网络架构下以可生存性算法作为课题的研究方向，主要研究了以下三个方面的内容：

研究网络可生存性技术，对网络故障失效环境进行了研究，并且探讨了已有各种情形的路径保护算法和不相交路径算法。

研究共享风险链路组不相交的约束条件，提出冲突边集合的概念，通过容量设置来获得冲突边集合，以冲突边集合来分而治之原问题，并行解决SRLG不相交路由问题，并且与已有算法进行理论分析和实验对比，结果表明我提出的算法优于现有其它算法。

研究节点和链路映射以及可生存性需求之间的关系，在可生存性虚拟网络嵌入算法的设计中，考虑到节点带有特定功能约束条件，本文结合已有算法的不足，提出了星型分解动态规划节点映射的可生存性虚拟网络嵌入算法，并且与已有算法进行实验对比，实验结果表明我提出的算法优于现有其它算法。

关键词：软件定义网络；　网络功能虚拟化；　不相交路径；　虚拟网络嵌入；　可生存性；　风险共享链路组

Abstract

Nowadays, the complexity of the whole network scale and the network function type is more and more large, the new technology is increasing the demand for network performance, and the traditional network architecture has been unable to meet the needs of on-demand mobilization, rapid configuration, and so on. Software defined Network (SDN) is proposed to deal with the existing network architecture can not solve the problem and break through the network performance bottlenecks. Using network function virtualization (NFV) technology to virtualize network resources can to a large extent solve the problems that the existing network cannot solve and reach the network performance requirements that the existing network cannot achieve. Network services run on actual physical devices, which will inevitably lead to inevitable network failures. SDN / NFV new network architecture can detect, process and recover network failures more efficiently than traditional network architecture in the field of network survivability.

In order to ensure the survivability of optical network and Overlay network, the architecture based on SDN can efficiently realize disjoint path routing function, and find disjoint routing between source node and destination node quickly. When the main path fails, the traffic flow is converted to the standby path, so the fast disjoint path algorithm has a high research value. In the process of network virtualization, how to find the best embedding in the underlying physical network, and how to provide survivability protection for the virtual network. To ensure the normal operation of the original virtual network under the condition of the failure of the underlying physical network resources, the research of the survivability virtual network embedding problem has become an important neighborhood in the research of NFV.

In this paper, based on the research direction of survivability algorithm under the SDN/NFV network architecture, we mainly study the following three aspects:

In this paper, the survivability technology of network is studied, and the fault failure environment of network is studied, and the existing path protection algorithms and disjoint path algorithms are discussed.

In this paper, the constraint conditions of disjoint shared risk link group are studied, and the concept of conflict edge set is proposed. The conflict edge set is obtained by capacity setting, the original problem of dividing and dominating the conflict edge set is obtained, and the SRLG disjoint routing problem is solved in parallel. The theoretical analysis and

experimental results show that the proposed algorithm is superior to other existing algorithms.

This paper studies the relationship between node, link mapping and survivability requirements. In the design of survivability virtual network embedding algorithm, considering that nodes have specific functional constraints, this paper combines the shortcomings of existing algorithms. A survivabal virtual network embedding algorithm based on star decomposition dynamic programming node mapping is proposed and compared with the existing algorithms. The experimental results show that the proposed algorithm is superior to other existing algorithms.

Key Words: Software Defined Network(SDN); Network Function Virtualization(NFV); Disjoint Path; Virtual Network Embedding(VNE); Survivability; Shared Risk Link Group(SRLG)

目　　录

插图索引

附表索引

第1章　绪论

1.1　研究背景及意义

1.1.1 课题研究背景

1.1.1.1 软件定义网络

众所周知，相比发展迅速的计算机产业，网络产业的创新十分缓慢。每一个创新都需要等待数年才能完成技术标准化。为了解决这个问题，软件定义网络（SDN）创始人Nick McKeown 教授对网络产业的创新模式和计算机产业的创新模式进行了对比和研究。在分析了计算机产业的创新模式之后，他总结出支撑计算机产业快速创新的三个主要因素。

(1) 计算机工业找到了一个面向计算的通用硬件底层：通用处理器，使得计算机的功能可以通过软件定义的方式来实现。

(2) 计算机软件的开源模式，加速了软件开发的进程，催生了大量的开源软件，推动了整个计算机产业的快速发展，Linux开源操作系统就是最好的证明。

(3) 计算机功能的软件定义方式带来了更加灵活的编程能力，使得软件应用的种类得到爆炸式的增长。

相比之下，传统的网络设备与上世纪60年代的IBM大型机类似，网络设备硬件、网络应用和操作系统三部分紧耦合在一起组成一个封闭的系统。这三部分相互依赖，通常隶属于同一家网络设备厂商，每一部分的演进和创新都要求其余部分做出同样的升级。这样的架构严重阻碍了网络创新进程的开展。如果网络产业能像当今计算机产业一样，也具备通用硬件底层、开源模式和软件定义功能三要素，一定能获得更快的创新速度，最终像计算机产业一样取得空前的发展。

正是在这种思路的影响下，McKeown教授团队提出了个新的网络体系结构SDN[1]：在SDN 架构中，网络的数据平面与控制平面相分离，数据平面将变得更加通用化，变得与计算机通用硬件底层类似，不再需要具体实现各种网络协议的控制逻辑，而只需要接收控制平面的操作指令并执行即可。网络设备的控制逻辑转而由软件实现的SDN应用和SDN 控制器来定义，从而实现网络功能的软件定义化。随着开源SDN开放接口和开源SDN控制器的出现，网络体系结构也拥有了通用底层硬件、开源模式和支持软件定义三个要素。

现代电信网络提供多种高速通信，通过大规模的面向连接和分组交换的网络提供服务运行在光网络，数字用户线路(DSL)，电缆连接甚至无线地面和卫星连

接的网络。因为社会很大程度上依赖于现代电信网络，已经做了很多工作来防止网络故障，例如，通过改善设备环境和材料的物理方面。然而，过去一个世纪的网络故障情形显示，高速骨干网络的链路即使是在短暂时间内，此时发生的故障也会造成大量的数据包丢弃[2]，严重影响了通信质量。根据对Internet服务提供者（ISP）的观察，骨干网链路一年内大约有30% 的概率会出现故障[3]。不论采取了那些预防性保护措施，网络节点和链路将最终概率性的失灵并停止运作。链路故障是网络中较为普遍的现象。因此，加快故障的恢复速度，降低故障造成的业务丢弃已成为当前研究亟待解决的问题。

在面向连接的网络中，例如波长路由网络中，由于网络节点或链路的故障而造成的网络服务中断通常可以通过从源节点到每个网络连接的目的节点分配至少两条不相交的路径来防止中断[4]。然后监视从源节点到目标节点的连接状态，当网络连接的主路径故障失效时，可以重新配置连接以使用其备份路径继续保持工作。还可以同时在连接的主路径（工作路径，AP）和备份路径(备用路径，BP)上发送通信信息，因此不需要在主路径故障失效时进行重新配置。虽然找到从源节点到目的节点的一对最小和（Min-Sum）不相交路径是多项式可解的[5,6]，但由于存在陷阱拓扑[7]，返回的主路径可能远远大于节点之间的最短路径[7]。另一种方法是找到一对Min-Min不相交的路径，其中主路径的权重最小化，而不是两条路径的权重之和最小化，网络业务主要运行在主路径上，当出现故障时备份路径主要是保证业务非中断，而不一定保证原来主路径一样的服务质量，所以对主路径的权重代价尽量小，另外必须保证存在备份路径，但对于备份路径的权重不做严格的需求，然而Min-Min不相交路径问题是NP-hard[8]。

在分组交换网络中，流量可以沿着主要路径的一部分重新路由，这在面向连接的网络中是不可能的。主路径上的每个中间节点在必要时具有通过另一个链路接口转发数据包的能力。此外，在分组经过故障重路由后，分组被定向到达目的地的最短剩余路径，可能的方法是跟踪未受故障影响的(初始)主路径的其余部分。然而，配置全网路由需要复杂的转发规则构造，传统的分布式路由协议在计算和内存容量较低的嵌入式系统上难以实现。而新兴的软件定义的网络(SDN)方法可以促进这种网络功能的实现。

1.1.1.2 网络功能虚拟化

目前的网络服务依赖于专有设备和不同的网络设备，这些设备是多种多样和专门创建的[9–11]，这种情况引发了所谓的网络僵化问题[12]，阻碍了业务的增加和网络的升级。为了解决这一问题并减少资本支出和业务支出，虚拟化已成为一种将网络软件功能/应用程序与其支持的硬件分离的方法[13–15]，并允许网络服务作为软件加以实施。利用虚拟化技术，ETSI 工业规范组提出了网络功能虚拟

化NFV，虚拟化以前的一些专有专用硬件执行的网络功能[16,17]，通过将网络功能与底层硬件设备分离，NFV在优化共享的物理基础设施之上提供了基于软件的网络功能的灵活配置。它通过利用低成本商品服务器来解决管理和控制这些封闭和专有设备的运营成本问题。

另一方面，随着软件定义网络SDN的发展，随着网络体系结构[18–20]的引入，将SDN 与NFV(软件定义的NFV体系结构)集成以实现各种网络控制和管理目标的趋势得到了明显的发展。当将SDN应用于NFV 时，可以帮助应对动态的挑战，资源管理，智能服务编排和故障恢复。通过SDN/NFV可以动态地为特定类型的服务链创建虚拟服务环境，从而避免了专用硬件和复杂的工作来提供新的服务请求。在使用SDN 的同时，NFV进一步实现了实时动态功能配置和灵活的业务转发。

网络虚拟化[15]技术通过对物理网络络资源进行抽象、隔离和分配，可支持多个虚拟网络（Virtual Network, VN)共存于同一物理网络中。虚拟网络嵌入[21](Virtual Network Embedding, VNE)是实现网络虚拟化的关键，旨在研究如何合理地将租户定制的虚拟网络请求（Virtual Network Request，VNR)映射到底层物理网络(SubstrateNetwork, SN) 中，获取满足虚拟网络服务所需的物理网络资源，从而提高虚拟网络请求接受率和网络资源利用率。对于不同的虚拟网络和物理网拓扑，设计不同的映射算法，会得到不同效率的映射方案，如何映射以确保网络资源的高效利用以及网络的负载均衡，也是现在和未来网络研究的方向。在虚拟网络映射算法中，默认分配给租户的虚拟网络服务可以一直正常运行，但现实中，由于设备自身或者环境等问题有时会引发物理网络故障，进而影响租户的虚拟网络服务的可生存性。因此，为向租户提供一个高可生存性的虚拟网络服务，一些研究者提出了多种可生存性虚拟网络嵌入算法[22]。

可生存性虚拟网络嵌入问题主要研究节点故障、链路故障或者同时考虑这两种故障的情况，在物理网络中，节点故障对于虚拟网络可用性的影响更大，所以本文将研究基于物理节点故障的可生存性虚拟网络映射（Survivable Virtual Network Embedding, SVNE)算法。

1.1.2 课题研究目的和意义

共享风险链路组(SRLG)是网络生存性分析中的一个逻辑概念，SRLG是一组共享公共物理资源(电缆、管道、节点或子结构)的网络链路，其故障将导致该组所有链路同时失效，现如今SDN/NFV网络架构Overlay网络体系结构可分为两层：逻辑层和物理层。物理层包括电缆、管道、路由器和光纤跨度节点(即光纤跨越的位置)。逻辑层由链路和节点(物理层中节点的子集)组成。逻辑层中的链路是物理层中的连接，它可以穿越多个光纤跨度和/或光纤跨越节点。此外，几个这样的链路可能通过相同的光纤跨度和光纤跨度节点。通过同一光纤跨度的多个链路因此

构成了SRLG，物理层中的光纤跨度等单一故障可能导致多条链路故障。

为了保护网络的两个节点之间的逻辑连接不受单个SRLG故障的影响，通常为连接分配两条不同的路径，其中一条路径称为主路径，在正常情况下使用。当主路径失败时，将使用第二条路径，称为备份路径。假设故障之间的失效是相互独立，这两条路径不应该同时失效，主路径上的链路不能与备份路径上的链路共享任何公共SRLG。在这种情况下，这两条路径是完全不相交的简称SRLG不相交，如果有条主路径，不存在完全SRLG不相交的备份路径，可以找到一条最大的SRLG不相交的备份路径，其是网络中与主路径共享最少公共SRLG的路径[23]。同样，这两条路径可以是链路不相交的，也可以是节点不相交的。由于一条网络链路可能属于几个SRLG，所以找到一对SRLG不相交路径的复杂度规模比找到一对链路/节点不相交路径要更复杂。

在SDN控制器中，往往需要考虑在多约束下的路由问题[24]。特别在业务发放的场景中，为了到达抗故障的效果，需要为业务寻找工作与保护两条路径。而且为了保证网络的通信质量，通常需要考虑时延，跳数，以及工作与保护路径间的不相交（分离）约束。所以多约束下的路由问题对实现网络资源实现高效快速调配和利用具有重要的意义。

现如今有关SRLG完全不相交路径的研究还有很大的空间，其中对两条路径的限制条件的研究将适应多种实际网络应用场景，特别是限制第一条路径的权重尽量小，对在业务中可以更加提高业务的Qos服务质量，和当故障发生时必定保证存在另一条备份路径来维持原先业务基本服务质量。研究**Min-Min SRLG完全不相交路径对问题**是现有网络可生存性算法领域的重要方面。

为了支持具有不同拓扑结构的虚拟网络，需要一种嵌入方法将虚拟网络映射到物理设施，并将物理网故障如链路或交换机故障映射到虚拟网络组件，任何虚拟化解决方案都必须快速执行这些映射操作，以便每个租户可以对其虚拟网络提供实时控制。而由于SDN的集中控制性能，相比传统的分布式算法，集中控制器可以更好地执行最优虚拟网络嵌入算法的计算和提供特殊QoS服务需求，这是因为：

(1) 可以考虑更多的因素，包括当前带宽的负载情况等
(2) 有一个更稳定的网络全局视图
(3) 可以在服务器性能较高的内存和处理器上执行计算，而不是在网络设备能力有限的可用内存和处理器上执行

在将虚拟网络映射到物理网络时，嵌入算法是其中的关键，在将虚拟网络的节点和链路进行映射时，需要找到最佳分配完成资源的配置，现有的嵌入算法已经很多，对于不同的应用场景，很难评判算法性能的好坏，较好的解决办法是针对不同的应用场景设计满足要求的算法，使算法的性能和效率能够优化。另外，

在虚拟网络的映射过程中，为了防止底层物理资源出现故障失效的情况，可以从用户层面和网络层面进行可生存性的研究。

研究单物理节点故障**可生存性虚拟网络嵌入问题(Survivable Virtual Network Embedding SVNE)**，有利于提高网络的可靠性，容错性，鲁棒性和可生存性，当物理层的网络节点出现故障时，能快速完全的保证原网络服务需求正常运行，这对网络的可生存性研究领域有重要的研究含义。

1.2 相关的研究及发展趋势

1.2.1 共享风险链路组完全不相交路径对问题

网络故障主要表现为链路故障，链路故障恢复策略可分为主动式和被动式两种[25,26]。被动式策略在网络故障后自适应动态地进行全网资源重分配，但路由重新收敛花费较多的时间而不可接受。因此目前故障快速恢复研究以主动式策略为主，通过提前对网络进行资源规划和预留，使得故障时能迅速切换，如基于备份路径和基于多拓扑[27]的故障恢复技术。基于备份路径的故障恢复技术提供端到端路径重路由，在全局范围内进行流量分配，易于基于现有协议实现；多拓扑技术需要配置多个拓扑子层，路由存储消耗大。因此，备份路径技术是当前故障恢复领域研究的热点。

网络服务质量需求是多约束不同属性的条件，可以对这些条件加权值和归一化统一成单一约束条件，网络大部分工作流量集中主路径上，备份路径是当主路径出现故障时，临时使用来恢复故障的路径，所以对备份路径的服务质量需求并不需要特别苛刻，而应该尽量提高主路径的服务质量需求。而网络故障一般为链路故障，SDN/NFV架构下多租户数据中心Overlay 网络和传输层光网络的故已经不是单独一条链路的故障，而是拓扑图逻辑上几条链路同时故障，此时延伸到风险链路组SRLG的故障。

共享风险链路组(SRLG)是一组链路共享相同的一个组件，该组件的故障会导致在这个组里所有链路同时发生故障。就路径保护而言，尽管某些链路或者节点不相交路径算法已经提出来，SRLG不相交路径问题是比较棘手的，而且这些原有研究是限制在一定领域的。比如当每个SRLG 只包含一条链路时，这个SRLG不相交路由问题可以简化为链路不相交路径问题，而通过节点分裂方法(node split method)[28]节点不相交路径问题可以转化为链路不相交路径问题。因为SRLG组通常包括的链路超过一条，并且网络中的链路通常可以属于多个SRLG组里，以至于求一对SRLG 不相交路径问题比求一对链路或者节点不相交路径问题要困难得多。

文献[29–31]是获得完全不相交的SRLG路径对的启发式算法。一种链路不相

交算法被扩展到SRLG不相交算法[29]，从而产生了冲突的SRLG冲突集算法。该算法将SRLG 不相交路径问题转化为Min-Min问题。文献[32]提出了一种新的CoSE启发式算法，用于求解CoSE不相交的最小和问题。文献[30]的SRLG不相交路径问题也被描述为一个Min-Sum问题，并通过迭代启发式来求解，即迭代修正的Suurballe启发式(IMSH)。文献[31]提出了一种陷阱避免方案TA算法，其中考虑到的算法试图避免陷入陷阱(即算法无法找到SRLG不相交路径对的情况)。文献[33]提及的段保护用于避免陷阱。文献[34]考虑了一种基于k-最短路径的算法(加权SRLG 或WSRLG)，其中根据与链路相关的SRLG 成员数将费用分配给链路。文中还考虑了根据链路与其他链路分担的风险分配链路成本的思想，其中提出了一种主动SRLG-多路径选择(ASPS)算法，该算法允许提高资源利用率(因为带宽资源在备用路径之间共享)，并考虑到不同用户的区分可靠性要求。文献[35]中也考虑到共享资源保护的背景，文中提出了一种混合保护方案，该方案将共享路径保护和共享链路保护联合应用于具有风险共享链路组的WDM网络中。文献[36]提出了两种不同的算法来计算给定的原点对和目标对的最优路径对，在这种情况下，路径对的最优性可以用两种不同的方式定义，从而导致两个不同版本的缺陷感知最佳路径对问题。文献[37]中考虑了一个不同的上下文，其中作者试图从一个预计算的路由表中找到一个AP和几个BP，并以最小的总附加带宽来承载特定的连接。文献[29]提出了一种改进型CoSE 的变体-modified-CoSE，在该算法中，如果不存在SRLG不相交路径对，则使用的SRLG路径对(其中AP最短且BP共享较少的SRLG)。文献[38]提出了TA算法[31]的一个变体指定的TA-Max。在TA-Max中迭代地考虑候选AP(为了增加成本)，并为每个候选AP计算一个具有最小成本的BP，最后的解决方案是共享最少数目的公共SRLG的路径对。

现如今研究SRLG完全不相交路径问题的精确算法只有[29,30,39]。算法[29]提出一种基于SRLG冲突集的分而治之方法，其分拆的子问题数规模，为得到最优解需消耗大量时间。算法[39] 提出SRLG完全不相交问题的线性规划方程解法。算法[30]求解的不相交路径对，其优化的目标是路径对的权重和最小化。而其他研究SRLG 不相交路径问题的算法[31] 都是启发式的非精确算法，即求解的路径对可能非最优路径对。算法[30] 获得的路径对的优化目标是Min-Sum，即两条两路的权重和最小化，不保证主路径的权重最小化。在实际网络上，主路径承担主要的业务就需要其路径权重最小化。综上，求一对Min-Min SRLG完全不相交满足服务质量需求的路径对问题是现今网络故障可生存性算法的研究重点。

1.2.2 可生存性虚拟网络嵌入问题

在考虑虚拟网络的嵌入过程中，常见的算法主要由两类：首先第一种是节点映射和链路映射彼此独立进行处理，将虚拟节点映射到物理节点上，然后由于此

时已经映射的节点在底层拓扑的位置是已知的，因此接下来的链路映射就可以概述为多商品流问题[40](Multi-commodity Flow Problem，MCF)，这种算法比较简单，但是由于节点和链路映射是独立进行的，只考虑到局部最优解无法获得全局最优解：第二种是节点和链路映射同时考虑，被称为虚拟网络嵌入协调过程，虽然它们都考虑到了全局情况，但是由于问题本身原理的限制，问题的复杂度规模是NP-hard，存在着成本比较高和结果不够精确的缺点。在虚拟网络的嵌入算法中，一般会考虑动态映射和静态映射两种情况[21]，有效的嵌入算法在降低成本，提高资源使用率这一方面就显得尤为重要。

现有基于物理节点故障恢复的可生存性虚拟网络嵌入算法主要采用主动保护[41] 和被动恢复[42]两类策略。主动保护策略在进行虚拟网络请求映射前为虚拟网络预置备份资源以供后续故障恢复使用，被动恢复策略在进行虚拟网络请求映射前不预置备份资源，在后续故障发生后再使用节点迁移或者现有的物理网络资源进行故障恢复。采用主动保护策略的算法包括FI-EVN[41]、FD-EVN[43] LC-SVNE[44]等，采用被动恢复策略的算法包括NB-SVNE[45]、MR-SVNE[46]等。

可生存性虚拟网络嵌入算法的研究从基于主动保护和被动恢复两个方面提出了相应的故障恢复方法，但是依旧存在下面的一些不足：

(1) 现有基于传统网络的SVNE算法中的主动保护策略比较固化，在物理网络无法为VNR提供全部所需备份资源的时候，会拒绝该VNR，进而导致虚拟网络请求接受率和网络运营收益偏低，而SVNE算法中的被动保护策略易造成故障恢复效率低和故障恢复成功率抖动大的情况。

(2) 现有基于SDN网络的SVNE相关算法中采用的也是主动保护策略，也存在主动保护策略的请求接受率和网络运营收益低的问题。虽然算法在此基础上增加了备份资源重分配机制[47]和虚拟网络映射资源来增网络运营收益加和网络资源利用率，但是资源动态调整机制不仅会增加算法的时间复杂度，还易造成虚拟网络抖动，影响虚拟网络服务质量。考虑到算法时间复杂度问题，本文暂不考虑使用资源重映射机制。

综上所述，目前的研究对于虚拟网络的保护策略不够灵活，或多或少都会出现性能短板问题，如主动保护策略中的虚拟网络请求接受率和收益低，被动恢复策略故障恢复效率低，传统网络下的SVNE算法无法对物理资源进行集中管控, 需要在物理设备间进行大量的通信，这会带来大量的通信开销，因此提出一个根据物理网络资源现状灵活实现虚拟网络冗余备份的可生存性虚拟网络嵌入算法，可以来提高虚拟网络请求接受率、故障恢复效率和物理网络资源利用率，这非常有意义。

因此，为提高故障恢复成功率和虚拟网络请求接受率，使运营商的物理网络能够为租户提供更为可靠的虚拟网络服务，本文提出了星型分解动态规划节点映

射的可生存性虚拟网络嵌入算法，此算法可以应用在主动保护和被动恢复两种策略的情形，与现有算法的比较，具有更高的可生存性接受率，资源利用率和收益，并且故障恢复效率更高，并且能拓展到应有于地域受限的虚拟网络嵌入问题和物理网络多节点故障的情形。

1.3 论文的主要研究内容和组织结构

1.3.1 论文的主要研究内容

SDN作为新型网络架构，NFV作为未来网络研究的重要领域，两者的结合具有新的研究意义。在网络可生存性领域中，当网络故障发生时对业务主路径提供备份路径是一个重要基础研究领域，在网络虚拟化的过程中，不可避开的一个问题就是虚拟网络嵌入算法的研究。

1.3.1.1 Min-Min SRLG 完全不相交路径对问题

基于图论寻找SRLG完全不相交路径时，对陷阱问题提供了新的见解。具体来说，对于带有陷阱问题的AP，我们观察到AP路径上存在一组链路集，任何通过所有这些“问题”链路AP都不能找到一个SRLG完全不相交的BP。我们称之为SRLG冲突链路集。一旦遇到陷阱问题，建议寻找SRLG冲突链路集，而不是搜索所有可能的替代路径，基于最大流最小割定理[28]。进一步提出了一种分而治之的min-min SRLG完全不相交路由算法，将原路由问题划分为多个子问题并行执行，找到可行的AP和BP对。研究的主要内容如下：

(1) 通过巧妙地设置链路容量来构造一个新的流图，以便于发现SRLG 冲突链路集。

(2) 提出了一种寻找最小SRLG冲突链路集的算法，这有助于减少搜索替代SRLG完全不相交路径对的复杂性。

(3) 根据SRLG的风险共享的特性，将SRLG冲突链路集最小查找问题转化为子集覆盖问题，使我们能够应用通用算法在不同的复杂SRLG场景(包括一个或多个SRLG模式的链路)中寻找最小SRLG冲突链路集。

(4) 提出了一种新的分而治之算法，该算法能在遇到陷阱问题时将原有的min-min SRLG完全不相交路由问题划分为多个并行执行子问题。与现有技术相比，这样的解决方案搜索过程可以利用现有的AP搜索结果和并行执行来加快的路径查找。

(5) 在一个多核CPU平台上进行了广泛的仿真，以评估所提出的算法。仿真结果表明，该算法能够以更快的速度在不同的网络场景中找到最佳解。

1.3.1.2 单物理节点故障可生存性虚拟网络嵌入问题

嵌入算法主要考虑的是虚拟网络节点和链路向底层物理网的节点和链路的映射，考虑如何为虚拟网络在底层物理网中找到满足条件的最优嵌入，并且尽可能地提高网络资源的利用率。

本课题将结合位置约束的概念，以虚拟网络请求的接受率、物理网络资源利用率和故障恢复率等指标作为算法性能评价指标，提出一个星型分解动态规划节点映射的可生存性虚拟网络嵌入算法。为了实现课题目标，已进行以下研究工作：

(1) 调研现有的可生存性虚拟网络映射算法，以及基于SDN的虚拟网络映射算法和可生存性虚拟网络映射算法，分析不同算法的优缺点，以及存在的共性问题。

(2) 提出一种星型分解动态规划节点映射的可生存性虚拟网络嵌入算法，提出虚拟星型图和物理星型图的概念，给出虚拟星型图与物理星型图之间节点映射的权值设定，转换节点映射的问题为多背包问题，通过动态规划方法解决这个多背包问题，理论分析了动态规划方法时空间复杂度。

(3) 设计和搭建仿真平台，对提出的嵌入算法进行仿真，并实现相关的对比方案。对实验结果进行分析，评估本文方案和对比方案的性能优劣得出，提出的算法比其他算法在接受率和资源利用率等指标上结果更优。

1.3.2 论文的组织结构

本文依据网络可生存性、不相交路径以及虚拟网络嵌入等方面的研究现状与发展趋势，并结合自身对以上的探索和研究，将本文分为五章，各章主要内容如下，其由于硕士论文篇幅限制，本文有关不相交路径算法、虚拟网路嵌入算法和可生存性虚拟网络嵌入算法的总结将不在本文中，可在github网址[48]得到论文更详细版本：

(1) 第一章为绪论部分，主要介绍了本课题的研究背景及其意义并且简要地分析了研究现状和发展趋势。

(2) 第二章介绍了网络可生存性相关原理，概述了网络的常见故障和网络故障恢复的两大机制，接着详细介绍了网络故障保护策略。

(3) 第三章主要是Min-Min SRLG完全不相交路由算法的设计与实现，在存在陷阱问题的情况下，提出了一种求解Min-Min SRLG完全不相交路由问题的有效算法。为了降低搜索复杂性，提出了一种分而治之的解决方案，将原Min-Min风险共享链路组不相交路由问题划分为多个子问题，该子问题基于从AP路径遇到陷阱问题导出的SRLG冲突链路集，在一个多核CPU平台上使用拓扑上进行了广泛的仿真。

(4) 第四章主要是单物理网络节点故障可生存性虚拟网络嵌入算法的设计与实现，当一个虚拟请求已经嵌入和运行在底层物理网络中时，突然一个底层物理节点随机独立的出现故障，提出的星型分解动态规划节点映射的可生存性虚拟网络嵌入算法可以预先分配备用资源来预防故障失效，进行多种算法指标度量的仿真。

(5) 第五章对现有的工作和研究成果进行了总结，并进一步对其中的某些技术问题探讨了未来研究的方向。

第2章　网络可生存性基本原理

为了增强网络性能、保证网络的健壮性，需要对SDN网络中的许多问题进行优化设计，这些问题主要包括：网络可生存性设计问题，业务恢复问题等。随着SDN 网络的兴起，在各个领域涌现出来的网络业务在广泛的增长，网络的可生存性问题已成为SDN网络关注的热点。目前业内提出了许多关键词，比如可靠性、容错性、抗毁性、鲁棒性和可信赖性等，它们的本质都是以网络可生存性的研究为出发点。在本章中我们主要就网络可生存性问题展开讨论。

2.1　网络的可生存性概述

软件定义网络中控制层中心控制器管理每条信道，而当某条信道发生故障时，则必须在短时间内恢复。因此在软件定义网络的网络控制层路由方面，研究路由的可生存性变得尤为重要。

考虑到今天通信系统和基础设施的重要性，网络应该在设计和操作上考虑到系统或者设施出现故障能够被解决。例如，由于恶意攻击，自然灾害，突然间的电缆断裂，计划维修，设备故障等等，网络节点/链路可能出现故障。弹性，容错性，可生存性，可靠性，鲁棒性和可信赖性，是已经被使用的不同术语。这些术语是面对网络故障时网络维护运行保持通信能力的术语。文献[49]指出不同的术语有重叠的意思和一定的歧义。在本文中，我们将使用**可生存性**网络一词是指**在网络中当一个网络组件出现故障时，可以通过找到替代路径来避免出现故障的网络组件**。

随着人们对可生存性技术的关注，国内外的许多研究学者和机构都开展对可生存性问题的研究，其研究重点可概括为可生存性的基本概念[4]、可生存性体系结构和系统模型[50]、可生存性风险评估[51]，容错容侵机制模型[52]等。

现阶段，网络可生存性研究已经有一定的开展，许多知名学者就网络可生存性问题的不同方面不同领域作了深入研究，问题涉及故障分类、可生存性问题建模分析和故障恢复技术等许多子领域，并取得了一些成果。

2.2　网络故障分类

由于网络遭受故障的因素及出现故障的网络元素不同，网络故障也是多种多样，按照网络中故障的表现方式不同，将网络故障分为软故障和硬故障[53]；按照故障产生的元素不同可分为信道故障、节点故障和链路故障。概念解释如下：

所谓软故障是指网络信息传输过程中由于信号逐步衰减，而造成信息丢失的事件，如光纤损耗增大等；硬故障是指某些意想不到的随机事件致使传输信道出现中断，如地震、光纤断裂、收发元器件老化失效等。

不难看出，硬故障对网络业务影响较大，但容易检测出来并且处理方便；软故障对网络业务的影响范围比较小，但出现机率极高，不容易被立即发现和处理，并且难以对故障精确定位。

因而，从科研角度出发，网络中的故障按拓扑位置区分是更利于研究：

(1) 信道故障是由于该信道对应的底层物理设备出现故障而引起。

(2) 链路故障主要是由于两物理节点间的通信链路（如光纤）断裂而引起。

(3) 节点故障主要是由于底层物理节点（如路由器）断电或者物理破坏而引起。

2.3 网络可生存性度量指标

对网络进行可生存性设计时，要评价是否达到的最理想的结果或者状态，对于给定的网络拓扑结构,能够在最短的时间内使故障元素获得最大程度的恢复，并同时也要保证最大的资源利用率[54]。然而由于任何事件都是此消彼长的互斥性，很难同时达到所有的需求标准，所以需要根据不同的业务特性或者用户需求甚至网络本身的特点，采取提高网络可生存性的措施，从而满足网络的可生存性指标需求。

常用的网络可生存性度量指标主要有:

(1) 故障恢复时间：指从网络故障发生时，到故障被处理业务恢复正常传输所需的时间。该项指标是最直接也最能体现网络的可生存性能。众所周知，对于网络的用户来说，传输过程是完全透明的，他们所能感受到的服务质量就是网络提供的业务服务质量，而其中最敏感的就是是故障恢复时间。

(2) 业务请求接收率：是指被接收的网络业务数与总体业务请求数的比值。可生存性网络研究的目标是最大化请求接收率。

(3) 网络资源利用率：这是衡量任何一个网络优劣的关键指标，提高网络利用率是网络运营商追求利益降低成本的目标。

(4) 平均网络负载：网络负载是指网络节点或者链路中的流量负荷，一旦节点或者链路负荷过重，将直接影响整体网络的连通性能。因此，最小化网络负载将有利于网络可生存性的研究。

(5) 业务平均跳数：某些网络有时以步长跳数来衡量代价成本，因此最小化路径长度是网络优化指标之一。

(6) 鲁棒性：即健壮性或者可生存性，是指经历过一次网络故障后，网络再次承受故障不受故障影响的能力。它主要用来衡量网络业务的可持续生存性。

2.4 网络故障处理

网络的可生存性实现机制根据是否进行重路由计算和是否预留备用资源，通常把故障处理策略分为保护(protection，主动式) 和恢复(restoration，被动式)。保护措施和恢复措施均是在网络故障情况下，使停止的业务得以重新运行。原理上，两者都要利用重路由方式（重新选择新的路由来代替故障路由），继续保证故障业务数据的传输。但就具体实施方法而言，保护和恢复方法又有所不同。两种方案各有长短，保护机制可以提供更快的恢复时间，有利于网络服务质量的保障；恢复机制能够提高网络资源的利用效率降低网络资源的消耗。网络恢复策略是在网络发生故障后动态自适应地进行全网资源重分配，但分布式网络路由重新收敛需要花费较多的收敛时间，这样服务质量是不可接受，因此目前故障快速恢复研究以网络保护策略为主，它通过对网络故障的快速检测、定位和恢复使其可生存性提高。

保护策略是指事先为业务（如虚拟网络请求、服务链请求等）分配好预留的备用资源，主要是通过节点之间预留的备用资源来实现网络保护。当故障发生时，将工作通路上的通信信号切换到备份通路上，使工作信号通过预留的备份通路维持业务正常传输；而恢复策略指，并不事先为网络业务分配预留的备份资源，在检测到故障时，动态地从网络中寻找替代路由，来承载受故障影响的业务。如果不可避免的找不到符合需求的路由，则该工作路径上携带的业务就会丢弃。

保护策略和恢复策略两者不是互斥关系，因此，我们在实际网络的可生存性设计过程中，往往会同时考虑保护和恢复策略折中的策略，取两种策略中的优点合并，从而为客户提供多种服务级别和服务质量，尽力做到在恢复时间、可生存性保障、资源效率及成本收益之间取得平衡。在实际的网络操作中，由于网络资源的不足，通常以保护策略机制为基础，來保障一些可预料的故障，（如光纤断裂等交换机故障〉，然后再使用恢复策略进行加强，保障整网范围内的服务质量。同时保证故障发生的情况下，及时性和网络资源合理化利用，即保证在一定的约束条件下为工作通路预留好事先准备的保护通路，这就是网络可生存性基于约束条件的资源优化问题。

2.5 网络保护策略

根据不同需求的业务可以选择不同类型的网络保护策略以保证网络的可生存性，比如对于实时业务，可以使用节点/链路保护，即预先建立预留资源和保护通道的保护方式；而对于“best in effort”的业务，就按需建立保护通道或者依靠高层的恢复机制。保护机制与恢复机制相比，保护机制具有更快速的恢复能力、更高的可靠性和更好的可生存性，这更适用于规模大的传输网络，因此本文主要研究的是保护策略机制。其中提高网络可生存性主要的保护策略机制三种方法如下所示。

(1) 网络连通，即可生存性好的网络应该是具有图论中高连通性的。

(2) 网络增强，即可能需要新的链路/节点以增加网络的连通性。

(3) 路径保护，即寻找替代方案的过程，失效故障时的替代路径。

现实网络中网络故障是不可避免的，因此必须不断加强网络可生存性研究，可以采取对网络故障进行快速准确的检测、定位和恢复的方法，来保证网络的可生存性能。

2.5.1 网络连通

网络通常被表示为图$G = (\mathbb{V}, \mathbb{E})$，其中$\mathbb{V}$是$|\mathbb{V}|$个节点的集合（例如表示路由器），$\mathbb{E}$是$|\mathbb{E}|$条链路的集合（例如表示光纤线路或无线电信道）。链路可以用它们的容量、延迟、长度、成本和/或失效概率的权重为特征。如果图中的每对节点之间存在路径则图是连通(connected)，否则该图被称为非连通。在可生存性的含义中，连通性的概念可以进一步拓展为k-连通(k-connected)，其中在每对节点之间存在至少k 条不相交路径。根据这些路径是节点不相交还是链路不相交，我们区分为节点连通性和链路连通性。图G的边连通度$\lambda(G)$ 是删除G的最小边数使得图边不连通。相应地，图的节点连通性$\kappa(G)$是删除G的最小节点数使得图节点不连通。

2.5.2 网络增强

网络的连通度显示出网络可能是不是够健壮(连通)。重新布线(覆盖)网络可以提高其健壮性[55]。因此通常通过向网络添加新的链路和可能的节点来提高网络的性能或网络的健壮性。添加链路或节点可能代价很高(这能通过链路/节点权重来反映)，因此新的链路/节点应该被明智地放置，以便以最少的链路/节点数量获得所需的网络属性，或者添加固定数量的链路/节点使所需的网络属性最大化。这类问题被称为网络增强(network augment) 问题，然而在这类问题中，问题仅在其目标上有所不同。例如，k- 连通性是网络健壮性的一个重要属性，通过添加链

路来达到k- 连通性就是这样的目标之一。代数连通性增强是一个NP-hard问题[56]。类似地，添加最小数量的链路使一个图是弦（chordal）也是NP-hard[57](如果一个图的四个或更多个节点之间都有一条边相连则称这个图为弦)。

2.5.3 路径保护

在Internet上部署诸如OSPF这样的网络协议，以获得正确的拓扑视图，并在发生变化如链路故障时，将路由收敛到新的不受干扰的情况。但是这个过程并不快，应用程序在性能上可能仍然面临不可接受的干扰。结合MPLS技术，可以使用MPLS快速重路由机制，它提供了从失效的主路径在亚秒时间切换到备份路径的能力。这个快速重路由机制在RFC 4090[58]，其被提出并且已经被几个供应商实现了。这一概念也已扩展到纯IP网络，并被称为IP快速重路由[27]。RFC 4090 定义了RSVP-TE扩展来建立备份标签交换路径(LSP)隧道，用于对LSP隧道进行本地修复。备份路径可以为防止链路或节点故障而配置。由于备份路径是预先计算的，所以在计算备份路径发生故障时执行信令是不会浪费时间。因此，迫切需要有效的算法来计算不相交的路径。根据备份路径是在主路径失效之前还是之后计算，可生存性技术可以广泛地分为恢复技术或保护技术。目前，许多文献对于路径保护技术进行了大量的研究，根据不同的方式或功能总结如下:

(1) 从重路由的角度：基于路径（通道）保护、基于链路的保护及区段保护。

(2) 根据备用资源的预留方式：共享保护和专用保护。

(3) 按照路由的计算方式：实时计算和预计算。

2.5.3.1 通道保护

通道(路径)保护是指业务故障恢复由通道两端的终端节点来实现。具体来说，基于通道保护机制是指对工作路由事先预留一条备份路由，在故障发生后，用预留备份通道来传输故障通道中的业务数据流，从而取代原先的故障通道，实现业务的重路由。当发生故障时，其切换过程只涉及源、目的节点，与中间节点无关，由于是源节点和目的节点启动保护切换，因而对故障的具体定位要求并不高。

通道保护又分为共享通道保护（备用资源能同时为多条工作通道提供保护）和专用通道保护（备用资源为某条工作通道专用），共享保护是指1: N保护方式，专用保护是指1+1 通路保护和1: 1通路保护。

2.5.3.2 链路保护

链路保护是指业务请求经过的每一条链路，都有一条备用保护路径对这条链路进行保护，一旦链路出现故障，业务将跳过故障点，直接切换到保护路径上。在与故障点邻接的两点间，为该故障链路寻找一条可不经过该故障点的备用路径。显然，链路保护方案中参与保护切换的节点数较少，因而具有更快的恢复速

度，同时由于为每条链路进行保护，资源浪费过高资源利用低。

链路保护方案中，对于不同链路中的业务只要不同时刻发生网络故障，不同链路共享相同的保护路径，因此也分为共享链路保护和专用链路保护两种。前者是指对于某一条链路，提供专门的保护路径，其它链路则不得使用该条链路的专用保护路径；而后者允许不同的链路保护路径在其重叠的链路或者节点部分实现资源共享，后者比前者资源利用率要高，而前者较后者的抗故障性更好。

2.5.3.3 区段保护

区段保护是对通路保护和链路保护折中的保护机制，考虑了通路保护和链路保护各自的特点而得到的一种保护方式。区段保护是指在一对节点之间出现故障时，对该段链路中的业务切换到这两个节点之间的另一段路径中去。如果两个相邻节点之间发生故障，则类似于链路保护。

2.6 小结

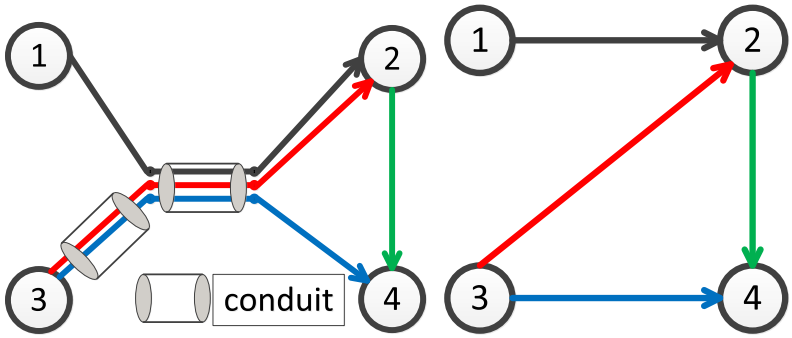
网络可生存性设计的目的是提高网络的健壮性，由于网络故障不可避免，那么故障后的及时修复成为网络性能的一个重要方面，本章简要的概述了网络生存性的故障分类，度量指标和故障处理方法。

第3章　共享风险链路组完全不相交路径对问题

3.1　问题描述

3.1.1 问题定义

共享风险链路组(SRLG)是一组链路共享同一个组件，该组件的故障会导致所有链路的故障。一条链路可以属于多个共享风险链路组里。举个例子，在光网络的导管中[59]可以放置多条光链路，如图3.1 所示，链路(1,2),(3,2) 和(3,4) 放置在一个导管中，同时链路(3,2)和(3,4)也共同放置在另一个导管中。如果某个导管被切断，相应导管的链路将失效。每个导管对应一个共享风险链路组。其它共享风险链路组的应用是交通网络的相关拥塞或电网的级联故障[60]。



a) 带有conduit的物理网络　　b) 逻辑网络

图 3.1 共享风险链路组(SRLG)实例

设$\mathbb{R}$为网络中的风险集（故障）。每个风险可能对应于导管断开、光纤断裂、在一个节点上的驱动故障、软件故障或这些因素的任何组合。对每一个共享风险链路组$r_i \in \mathbb{R}$ 是指与其风险r_i 相关的链路集合$\mathbb{R}_{r_i}$，$1 \le i \le \chi$ 和$\chi = |\mathbb{R}|$是共享风险链路组集合的个数。如图3.2(a) 所示，该图包含五个共享风险链路组集合$\mathbb{R}_{r_1} = \{e_1, e_9\}, \mathbb{R}_{r_2} = \{e_2, e_3, e_{19}\}, \mathbb{R}_{r_3} = \{e_2, e_4, e_{11}, e_{17}\}, \mathbb{R}_{r_4} = \{e_5, e_{13}\}, \mathbb{R}_{r_5} = \{e_{15}, e_{18}\}$。在这个例子中，链路$e_2$同属两个共享风险链路组集合里$\mathbb{R}_{r_2}$ 和$\mathbb{R}_{r_3}$。

SRLG不相交路径在它们之间没有任何共同的风险资源，也就是说，由于风险而导致的路径失败不会影响其他路径。图3.2(b)显示两条SRLG 不相交路径对，表示为AP 和BP。因为这两条路径没有共同的风险资源，如果AP 失败，BP仍然可以工作。本章主要讨论了两条不相交的路径，即可以描述如下。

Min-Min SRLG完全不相交（分离）路径对问题。给定一个图$G(V, E)$，每条链路$e_i \in \mathbb{E}$ 相关联一个权重w_{e_i}，一个源节点s和一个目的节点d，找到一对从s 到d的SRLG不相交路径对(表示为AP和BP)，而且要求这两条不相交路径中路径权重

较小的那条路径权重最小化，形式化如下：

$$\begin{aligned} &\underset{AP,BP}{minimize} && \min(w_{AP}, w_{BP}) \\ &subject\ to && r_{AP} \cap r_{BP} = \phi \\ & && \mathbb{AP} \cap \mathbb{BP} = \phi \end{aligned} \tag{3.1}$$

即w_{AP}和w_{BP}是AP和BP的路径权重，$\mathbb{AP}$和$\mathbb{BP}$分别是路径AP和BP上的链路集，r_{AP}和r_{BP}分别是影响路径AP和BP的SRLG集。

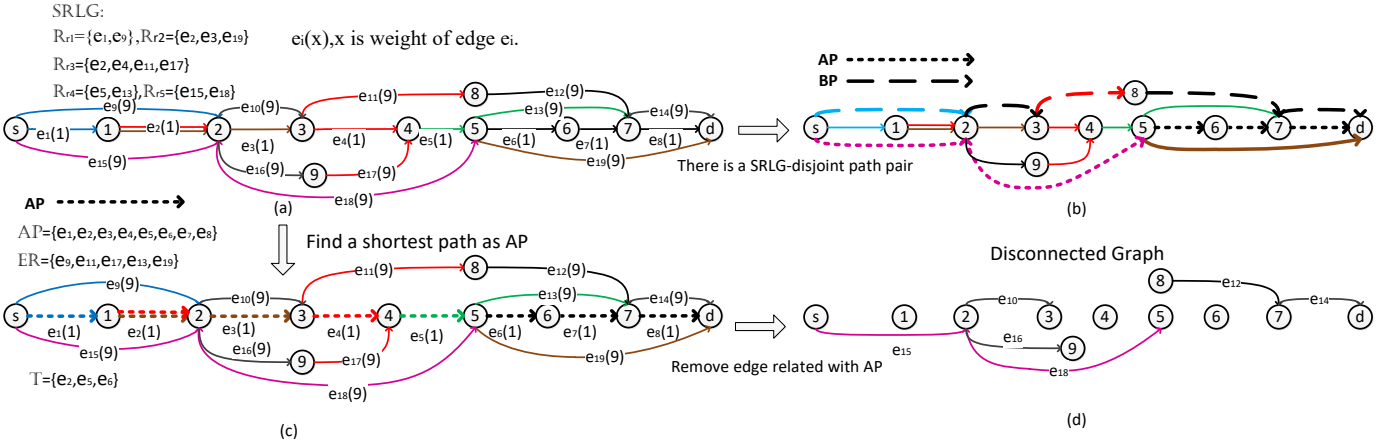


图 3.2 SRLG不相交路径对实例

3.1.2 问题的线性规划方程

首先介绍线性规划方程有关的一些必要的定义：

(1) 零列向量：$\mathbf{0} = [0, 0, \ldots, 0]^T$

(2) 单位列向量：$\mathbf{1} = [1, 1, \ldots, 1]^T$

(3) 边权重的行向量：$w = [w_e]_{e \in \mathbb{E}}$，其中边的权重$w_e (e \in \mathbb{E})$

(4) 源目标列向量：$u = [u_v]^T_{v \in \mathbb{V}}$，其中$\mathrm{u_v} = \begin{cases} 1 & if\ v = s \\ -1 & if\ v = d \\ 0 & otherwise \end{cases}$

(5) 节点边关联矩阵：$A = [a_{v,e}]_{\mathbb{V} \times \mathbb{E}}$ 其中

$$\mathrm{a_{v,e}} = \begin{cases} 1 & if\ edge\ e\ originates\ from\ node\ v \\ -1 & if\ edge\ e\ terminates\ at\ node\ v \\ 0 & otherwise \end{cases}$$

(6) 风险边关联矩阵：$H = [h_{r,e}]_{R\times E}$，其中$\mathrm{h_{r,e}} = \begin{cases} 1 & if\ edge\ e \in \mathbb{R}_r \\ 0 & otherwise \end{cases}$

(7) 边决策变量的列向量包含在路径$i = AP, BP$：$x_i = [x_{i,e}]^T_{e\in\mathbb{E}}$，其中

$$x_{i,e} = \begin{cases} 1 & if\ edge\ e \in path\ p_i \\ 0 & otherwise \end{cases}$$

从s到d的两条SRLG不相交路径最小路径权重最小化的问题，整数线性规划ILP方程表示如下：

$$\begin{aligned} \underset{AP,BP}{minimize} \quad & \min\left(\sum_{e\in\mathbb{E}} w_e * x_{AP,e}, \sum_{e\in\mathbb{E}} w_e * x_{BP,e}\right) \\ subject\ to \quad & A \times x_i = u, (i = AP, BP) \\ & x_{AP} + x_{BP} \le \mathbf{1} \\ & (h_r \times x^T_{AP}) + h_r \times x^T_{BP} \le \mathbf{1} (1 \le r \le \mathbb{R}) \end{aligned} \tag{3.2}$$

从s到d的两条SRLG不相交路径最小路径权重最小化的问题，整数二次规划方程IQCP 表示如下：

$$\begin{aligned} \underset{AP,BP}{minimize} \quad & \min\left(\sum_{e\in\mathbb{E}} w_e * x_{AP,e}, \sum_{e\in\mathbb{E}} w_e * x_{BP,e}\right) \\ subject\ to \quad & A \times x_i = u, (i = AP, BP) \\ & x_{AP} + x_{BP} \le \mathbf{1} \\ & H \times x_{AP} \times x^T_{BP} \times H^T = \mathbf{0} \end{aligned} \tag{3.3}$$

约束$A \times x_i = u, (i = AP, BP)$保证了基于$x_i$选择的边组成一条从s到d的路径。

约束$x_{AP} + x_{BP} \le \mathbf{1}$保证这两条路径是边不相交的。

约束$(h_r \times x^T_{AP}) + h_r \times x^T_{BP} \le \mathbf{1}$和$H \times x_{AP} \times x^T_{BP} \times H^T = \mathbf{0}$保证这两条路径是SRLG不相交的。

3.1.3 复杂度规模

定理 3.1 Min-Min SRLG-不相交路径对问题复杂度是NP-complete.

证明： Min-Min链路不相交路径对问题是NP-complete[61]。Min-Min链路不相交路径问题是Min-Min SRLG不相交路径问题的子问题。设Min-Min SRLG不相交路径

问题的复杂性为C(A)，则NP-complete≤C(A).

为了求Min-Min SRLG不相交路径问题的时间复杂度，首先假设了一个问题B(问题B的复杂度表示为C(B))，问题B定义为找到两条SRLG不相交路径并且路径较小的路径其权重小于或等于M(M 是大于零的整数)。Min-Min SRLG不相交路径问题A与问题B等价，显而易知，M必须大于零且小于$\sum_{e_i \in \mathbb{B}} w_{e_i}$。例如，假设0≤M≤10 和m=6 是最优解，通过经典的二分法（binary search method)其时间复杂度为O(log(N))，如图3.3所示，通过二分法得到了两条不相交路径，较小的路径其权重为m，因此问题A与问题B等价。 假设程序X在输入问题B时，如果问题B没

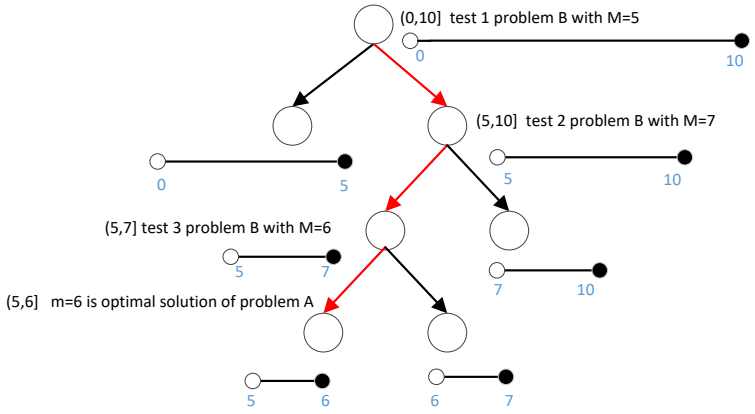


图 3.3 二分法求最优解实例

有解，则程序Y立即停止，否则程序B继续执行并获得问题B的解，因此问题B 可以归约为NP-hard问题。因此C(B)≤NP-hard。

此外，给定任意两条路径，很容易在多项式时间内判别这两条路径是否为SRLG不相交路径，较小的路径其权重小于或等于M，从而使得C(B)≤NP-complete。当B的复杂度等于A时，有C(A)=C(B)≤NP-complete。因此，A=NP-complete。 □

3.2 原有算法概述

为了解决SRLG不相交路径问题，一种可能的方法是0-1整数线性规划(ILP)[39]，通过分支限界法（branch-and-bound）来搜索选择最优的主路径和备份路径。该方法时间复杂度高，不适用于大型网络。为了降低算法的复杂度，基于APF的启发式算法[34,62,63]能够求Min-Min SRLG 不相交路径问题的近似最优解。首先使用Dijkstra算法(或任何其他最短路径算法)求出主路径，求主路径时不考虑其相应的备用路径情况，在删除AP沿线的链路并且与AP共风险的节点和链路后，再利用最短路算法求的备用路径。

然而，使用APF启发式算法的有一个主要缺陷，一旦求得路径AP后也可能无法找到相对的SRLG不相交路径BP，即使网络中确实存在一对不相交路径。这就是所谓的“陷阱”问题，在节3.3 将详细介绍陷阱问题产生的原因，即使稠密网

络[64]中这也是可能发生，在一个稀疏连接的网络中当然是不能被忽略。陷阱问题分为两种：不可避免的陷阱和可避免的陷阱。不可避免的陷阱是受拓扑约束的，任何算法都无法解决。如果网络不是2-边连通度的，则没有算法可以保证在拓扑中存在两个SRLG不相交路径。另一方面，当两个节点之间存在SRLG不相交路径对，但由于路由算法的缺陷而找不到时，就会出现一个可避免的陷阱，在本章中只考虑了可避免的陷阱。

对简单的APF算法扩展，提出了K-最短路径(KSP)算法来处理节点/链路不相交路径的陷阱问题。虽然它是处理陷阱问题最有效的算法之一，但它在大型网络中的性能受到影响，因为KSP 可能会涉及多路径搜索测试，直到它找到不相交路径。当前候选的路径AP遇到陷阱问题后，仅根据路径长度选择下一个要测试的候选AP，而不考虑当前候选AP 的那条链路(或那些链路)导致查找不相交路径BP 失败。因此，为了找到一对不相交路径对，需要对大量的路径进行测试，这就引入了KSP算法中与K 相关的时间复杂度。对于遇到陷阱问题的AP，应用从AP 路径导出的SRLG冲突链路集来指导将来的AP路径测试。这在很大程度上有助于减少寻找替代路径的时间复杂度。

其它的SRLG不相交路径算法，搜索最大SRLG不相交路径对，并且路径间共享最小数目的公共链路。由于AP 和BP 可能具有相同的风险资源组，通过这种方法找到的解决方案是不可靠的。我提出的算法目标是寻找SRLG完全不相交路径对。Xu[31]试图找到SRLG完全不相交路径。但他的算法减少了问题的搜索空间，加快了路径搜索的速度。然而，它可能会以较大的代价返回路径，因为在削减后的搜索空间可能会失去最优解。相反，为了大大加快搜索过程，利用SRLG冲突链路集将原问题划分为多个子问题，这些子问题可以并行执行。因此，提出的算法可以运行得更快，返回主路径成本非常低。

Datta[65]提出方法是将SRLG不相交路径问题转化为链路不相交路径问题，然后利用链路不相交路径算法来解决。然而，只有特殊的SRLG 星型模式可以转换为链路不相交，这样就限制了该算法的广泛应用。当AP遇到陷阱问题时，CoSE[29]算法试图找到一个SRLG集合，任何AP 路径包含了这个SRLG集合里的所有SRLG，则必定找不到任何的与其对应的SRLG不相交路径BP。CoSE 首先通过多轮搜索查找多个AP共享的SRLG，并且组成一个SRLG集合，然后根据SRLG 集合来划分原始问题以搜索SRLG不相交路径对。而不使用SRLG 中链路之间共享风险的特性，CoSE方法的这种穷尽搜索需要非常高的计算开销。

3.3 陷阱问题

基于APF的启发式算法可能会陷入“陷阱”问题。也就是说，当一个AP被确

定时，即使网络中确实存在一对不相交路径对，它也可能无法找到SRLG不相交的BP路径。图3.2(c),(d)说明了陷阱问题，虚线表示一条AP路径，其链路集为$\mathbb{AP} = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$，在删除AP上的链路和与AP共享风险的链路后，图3.2.(d) 所示的不存在从s到d的路径，因此找不到相对应的BP。

虽然KSP算法被认为是解决陷阱问题的有效算法，但它可能面临着效率低下的问题。如图3.4所示假设e_1, e_2, e_3, e_4的链路权重比其他链路大得多。此外，在e_1, e_2, e_3, e_4中，e_1和e_2的链路权重远小于e_3，e_4。然后，在KSP算法多次找从s到d的K短路时，总是包含e_1, e_2（虚线表示）。则最短AP总会遇到陷阱问题，因为e_1和e_4具有相同的风险，因此无法找到BP。为了避免陷阱问题，必须将K设为一个大值，这给KSP带来了很高的时间复杂度。

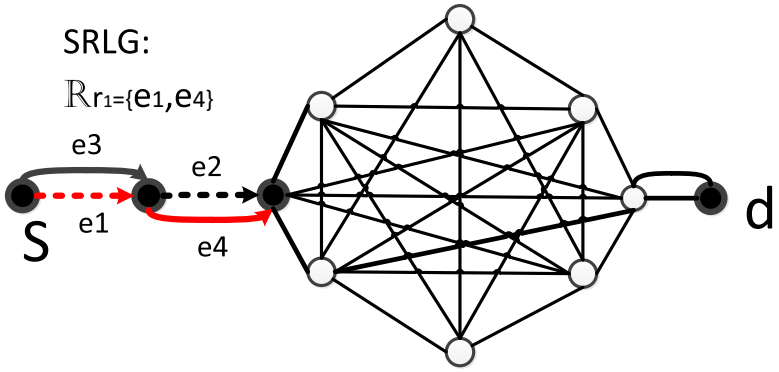


图 3.4 演示KSP算法时间效率低的实例

3.4 分而治之的快速SRLG完全不相交路径对算法

当一个陷阱问题发生，并且对于给定的AP没有SRLG不相交路径BP时，AP中可能存在一个子链路集，这样任何通过这个链路集里所有的这些“问题”链路的AP都不能找到一条相对应SRLG不相交BP路径，称之为**SRLG冲突链路集**。与KSP不同，当最短AP路径遇到陷阱问题时，将通过两个主要步骤来解决这个问题。如图3.4所示的例子中，将首先找到图3.4中的SRLG冲突链路集合，然后应用分而治之算法将原问题划分为两个子问题$\mathcal{P}(\emptyset, \{e_1\})$和$\mathcal{P}(\{e_1\}, \{e_2\})$。这两个子问题可以在多核CPU平台上并行执行，快速得到SRLG不相交路径对。

3.4.1 分而治之

在得到SRLG冲突链路集后，设计了一种分而治之的算法，将原Min-Min SRLG不相交路径问题划分为多个子问题，并行执行，加快求SRLG不相交路径对的过程。

为了便于问题划分，首先定义了两个不相交的链路集合$\mathbb{I}$和$\mathbb{O}$，其中称$\mathbb{I}$为包含集集合，$\mathbb{O}$称为排除集集合。由$\mathcal{P}(\mathbb{I}, \mathbb{O})$表示的Min-Min SRLG不相交问题，用于寻找一对AP和BP，其中AP是所有可能的AP中最短的，其中路径AP必须经过$\mathbb{I}$集合

里的所有链路和不经过$\mathbb{O}$集合里的所有链路。

最初，让$\mathbb{I}=\emptyset$ $\mathbb{O}=\emptyset$，原来的Min-Min SRLG不相交路径对问题可以用$\mathcal{P}(\emptyset,\emptyset)$表示。给定SRLG冲突链路集$\mathbb{T}=\{e_1,e_2,\cdots,e_{|\mathbb{T}|}\}$，原问题$\mathcal{P}(\emptyset,\emptyset)$可按以下步骤划分。

(1) 首先，$\mathcal{P}(\emptyset,\emptyset)$能被划分成两个子问题$\mathcal{P}(\emptyset,\{e_1\})$ 和$\mathcal{P}(\{e_1\},\emptyset)$。

(2) 类似，$\mathcal{P}(\emptyset,\{e_1\})$能被划分成两个子问题$\mathcal{P}(\{e_1,e_2\},\emptyset)$ 和$\mathcal{P}(\{e_1\},\{e_2\})$。

(3) 这个划分步骤持续直到步骤$|\mathbb{T}|$，问题$\mathcal{P}(\{e_1,e_2,\cdots,e_{|\mathbb{T}|-1}\},\emptyset)$ 进一步的拆分成两个子问题$\mathcal{P}(\{e_1,e_2,\cdots,e_{|\mathbb{T}|-1},e_{|\mathbb{T}|}\},\emptyset)$ 和$\mathcal{P}(\{e_1,e_2,\cdots,e_{|\mathbb{T}|-1}\},e_{|\mathbb{T}|})$。注意到，子问题$\mathcal{P}(\{e_1,e_2,\cdots,e_{|\mathbb{T}|-1},e_{|\mathbb{T}|}\},\emptyset)$是无解的。

除了子问题$\mathcal{P}(\{e_1,e_2,\cdots,e_{|\mathbb{T}|}\},\emptyset)$外，将试图求其它每个子问题的最优解。然后选择最好的路径对(即主路径最短的路径对)，作为原问题$\mathcal{P}(\emptyset,\emptyset)$的最终(最优)解。如果这些子问题都没有解，则可以得出原问题没有任何解，因为子问题包括了所有的可能的不相交路径对。

就时间复杂性而言，解决子问题所需的时间比原来的问题应该花费的更少。因为一条链路(来自集合$\mathbb{T}$) 将在计算AP的路径时被去除，这也确保了不同的AP 路径将被测试且是否存在一个SRLG不相交BP 路径。

当遇到陷阱问题时，提出的解决方案将划分原来的问题，并测试每个子问题以寻找到最终的解。在提出的分而治之方法中，子问题是由SRLG冲突链路集而得，而这个SRLG冲突链路集却是当AP路径遇到陷阱问题生成的。与现有的算法相比较，该算法在不考虑现有结果和问题的情况下，可以在很大程度上降低算法的计算量。对于图3.5中的例子所示，SRLG冲突链路集是$\mathbb{T}=\{e_2,e_5,e_6\}$。拆分过程过程如图3.5 所示。根据SRLG冲突链路集，应该测试总共3个子问题$\mathcal{P}(\{e_2,e_5\},\{e_6\}),\mathcal{P}(\{e_2\},\{e_5\})$ 和$\mathcal{P}(\emptyset,\{e_2\})$，其中选择AP 路径权重最低的最优子问题作为原问题$\mathcal{P}(\emptyset,\emptyset)$最终的(最优)解。

注意，不需要解决子问题$\mathcal{P}(\{e_2,e_5\},\emptyset)$ 和$\mathcal{P}(\{e_2\},\emptyset)$，因为它们的解已经包含在其它的子问题中。第一个解空间由两个子问题$\mathcal{P}(\{e_2,e_5,e_6\},\emptyset)$ 和$\mathcal{P}(\{e_2,e_5\},\{e_6\})$ 组成。由于SRLG冲突链路集为$\mathbb{T}=\{e_2,e_5,e_6\}$，显然，子问题$\mathcal{P}(\{e_2,e_5,e_6\},\emptyset)$是没有解的。因此，$\mathcal{P}(\{e_2,e_5\},\{e_6\})$的解空间等于$\mathcal{P}(\{e_2,e_5\},\emptyset)$的解空间。同样，$\mathcal{P}(\{e_2\},\emptyset)$的解空间包括$\mathcal{P}(\{e_2\}\{e_5\},\emptyset)$ 和$\mathcal{P}(\{e_2\},\{e_5\})$的解空间。

$$\mathcal{P}(\emptyset,\emptyset)\begin{cases}\mathcal{P}(\{e_2\},\emptyset)\begin{cases}\mathcal{P}(\{e_2,e_5\},\emptyset)\begin{cases}\mathcal{P}(\{e_2,e_5,e_6\},\emptyset)\\ \boxed{\mathcal{P}(\{e_2,e_5\},\{e_6\})}\end{cases}\\ \boxed{\mathcal{P}(\{e_2\},\{e_5\})}\end{cases}\\ \boxed{\mathcal{P}(\emptyset,\{e_2\})}\end{cases}$$

图 3.5 分而治之的解决方案

3.4.2 SRLG冲突链路集合

在本节中，我将描述了如何找到一个SRLG冲突链路集合，当在网络G中给定一个AP 路径并且没有SRLG 不相交的BP路径。

3.4.2.1 边容量设置准则

如3.4.2.2节介绍的，如果在图G中去除所有在边割集$\mathbb{L}_\Phi$ 的所有边，则$|f|=0$。也就是说，不存在任何流能从s到d。在本文中，试图基于割集的概念找到SRLG冲突链路集合。如果AP从s到d流通，经过的链路与割集$\mathbb{L}_\Phi$ 共享风险，则找不到任何与其对应的SRLG不相交路径BP，因为没有一条在割集中的链路可以被BP 选择。

基于割集基础为了找到SRLG 冲突链路集，构造了一个新图G^* ，如下所示。

(1) G^*与G的节点和链路拓扑关系一样。

(2) 跟每条链路e_i相关的链路权重w_{e_i}是跟其相对应图G中边的权重一样的。

(3) 使用公式3.4的准则设置每条边$e_i \in \mathbb{E}$相关的容量c_{e_i}。

$$c_{e_i}=\begin{cases}1 & e_i \in \mathbb{AP}\\ |\mathbb{AP}|+1 & e_i \in \mathbb{ER}\\ |\mathbb{AP}|+(|\mathbb{AP}|+1)\times|\mathbb{ER}|+1 & otherwise\end{cases} \tag{3.4}$$

$\mathbb{AP}$指在图G中较小权重路径AP上所有链路的集合，和$\mathbb{ER}$指不属于路径AP上的边但是与路径AP上的边共享风险的链路集合。

如图3.2(c)所示，路径AP的边集合$\mathbb{AP}=\{e_1,e_2,e_3,e_4,e_5,e_6,e_7,e_8\}$, $\mathbb{ER}=\{e_9,e_{11},e_{17},e_{13},e_{19}\}$。$|\mathbb{AP}|=8$, $|\mathbb{ER}|=5$, $|\mathbb{AP}|+1=9$ 和$|\mathbb{AP}|+(|\mathbb{AP}|+1)\times|\mathbb{ER}|+1=54$。产生一个新图$G^*$如图3.6所示，在图$G^*$中边的容量是根据公式(3.4)所设置。

r_P表示影响路径P上的风险集合，即$r_P=\{r\in\mathbb{R}:$ 路径P包含的链路在$\mathbb{R}_r$ 中$\}$。如图3.2(c) 所示，在路径AP上的边集$\mathbb{AP}=\{e_1,e_2,e_3,e_4,e_5,e_6,e_7,e_8\}$，并且$e_1\in\mathbb{R}_{r_1}$, $e_2\in\mathbb{R}_{r_2}$, $e_2\in\mathbb{R}_{r_3}$, $e_3\in\mathbb{R}_{r_2}$, $e_4\in\mathbb{R}_{r_3}$, $e_5\in\mathbb{R}_{r_4}$，路径AP的风险集合是$r_{AP}=\{r_1,r_2,r_3,r_4\}$。$\mathbb{ER}$代表不属于$AP$上的链路但是与$AP$共享相同的风险集合的链路。如图3.2(c) 所示，$\mathbb{ER}=\{e_9,e_{11},e_{17},e_{13},e_{19}\}$。

3.4.2.2 最小割最大流定理

设$G=(\mathbb{V},\mathbb{E})$是一个网络(其中$\mathbb{V}$是$|\mathbb{V}|$个节点的集合，$\mathbb{E}$是$|\mathbb{E}|$条链路的集合)，其中$s\in\mathbb{V}$和$d\in\mathbb{V}$分别指源节点和终节点。链路$e_i$的**容量**表示该条链路的最大流量。链路的流$f_{e_i}$应该满足以下两个限制：

(1) 容量限制: $\forall e_i\in\mathbb{E}$: $f_{e_i}\le c_{e_i}$.

(2) 流量守恒: $\forall u\in\mathbb{V}-\{s,d\}$: $\sum_{v\in\mathbb{V}} f_{(v,u)}=\sum_{v\in\mathbb{V}} f_{(u,v)}$, (v,u)和(u,v)代表链路$e(v,u)$

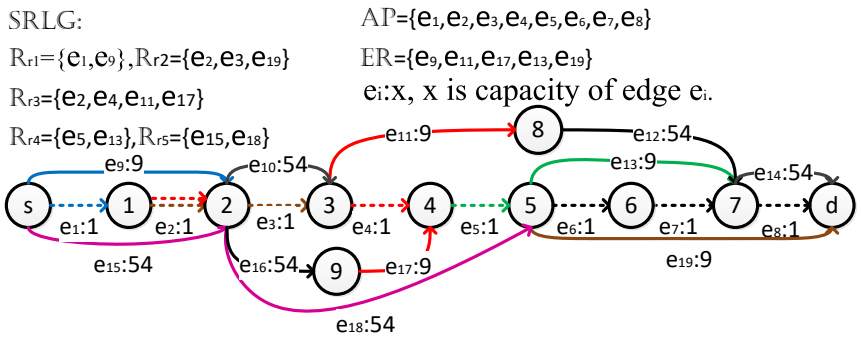


图 3.6 图G^*实例

和$e(u,v)$.

流的值定义为$|f| = \sum_{v\in\mathbb{V}} f_{(s,v)}$，其中s是源节点。它表示从s节点到d 节点的流量。**最大流量问题：**尽可能的求从s 节点到d 节点的最大流量值$|f|$。

一个s-d 割$\Phi = (\mathbb{S},\mathbb{D})$ 是节点$\mathbb{V}$的划分且$s \in \mathbb{S}$ 和$d \in \mathbb{D}$，Φ 的割集合$\mathbb{L}_\Phi$ 是一个包含边的集合。

$$\mathbb{L}_\Phi = \{(u,v) \in \mathbb{E} : u \in \mathbb{S}, v \in \mathbb{D}\}. \tag{3.5}$$

如果在割集合$\mathbb{L}_\Phi$中的边被去除，那么在原图中的流值$|f| = 0$。即没有流能从s节点到d 节点。一个s-d 割$\Phi = (\mathbb{S},\mathbb{D})$ 的容量被定义成$c(\Phi) = \sum_{e_i\in\mathbb{L}_\Phi} c_{e_i}$。**最小s-d 割$\Phi$问题**，最小化$c(\Phi)$即决定点集$\mathbb{S}$ 和$\mathbb{D}$使得s-d 割$\Phi = (\mathbb{S},\mathbb{D})$的$(c(\Phi))$ 最小化。

最小割最大流定理：一个s-d流的最大值等于s-d割的最小割。如图3.7所示，在图G 中的流量$|f| = f_{(s,v_1)} + f_{(s,v_2)}$。这个割$\Phi(\mathbb{S},\mathbb{D})$ 是$\mathbb{S} = \{s, v_1, v_2, v_4\}$ 和$\mathbb{D} = \{v_3, d\}$，它是最小的割其容量为$c(\Phi) = c_{(v_1,v_3)} + c_{(v_4,v_3)} + c_{(v_4,d)} = 12 + 7 + 4 = 23$. 显然，$|f| = c(\Phi)$, 即s-d 最大流等于所有s-d割中最小的容量。

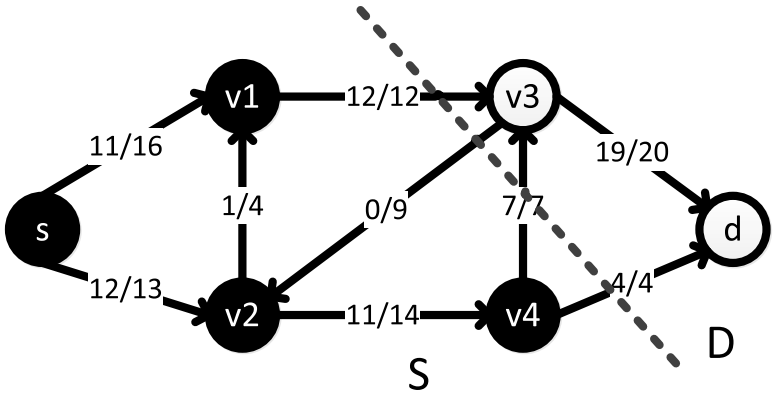


图 3.7 最大流最小割定理实例

3.4.2.3 新图G^*中最小割集性质

最大流最小割定理指出，在网络中，从源节点s到目的节点d的最大流值等于最小边割集的总容量，即最小的链路总容量，如果删除最小边割集上的边，将断开目的节点d 与源节点s的连通。提出的算法基于图的最小割集的链路集来求得最

小SRLG 冲突链路集。将首先展示重建新图G^*的一些优良特性为求得最小SRLG冲突链路集。

引理 3.1 在图G^*中任何从s到d的路径必须经过在$\mathbb{AP}$ 或者$\mathbb{ER}$中边集合的一条边。

证明： 用矛盾法证明这条引理。假设对于某条路径AP，有另一条路径从s到d 在图G^*中不与AP共享风险，即这条路径不通过$\mathbb{AP}$ 或者$\mathbb{ER}$的任何一条链路。可以很容易得出这样的结论，这条路径是与路径AP对应的SRLG不相交路径BP。这与AP没有对应的SRLG不相交路径的说法相矛盾。 □

引理 3.2 图G^*的任何一条最大流的流值最多为$|\mathbb{AP}| + (|\mathbb{AP}| + 1) \times |\mathbb{ER}|$。

证明： 假设G^*的最大流f为$|f| = k$。在G^*中，f可以被划分为k个从s到d的1 单元流。根据引理.3.1，这些1-单位流中的每一个都必须通过通过$\mathbb{AP}$ 或者$\mathbb{ER}$的至少一条链路。请注意，$\mathbb{AP}$ 或者$\mathbb{ER}$ 中链路的容量分别为1或者$|\mathbb{AP}|+1$。根据$\mathbb{AP}$ 和$\mathbb{ER}$中链路的容量设置，因为$\mathbb{AP}$中的链路只能承载1-单位流量，而在ER中一条链路在$\mathbb{ER}$中能最多承载$|\mathbb{AP}|+1$单位流量。因此，最多只能有$|\mathbb{AP}| + (|\mathbb{AP}| + 1) \times |\mathbb{ER}|$ 个从s到d 的单位流量。 □

引理 3.3 图G^*最小割Φ的边割集$\mathbb{L}_\Phi$上所有链路都在$\mathbb{AP}$ 或者$\mathbb{ER}$ 中。

证明： 根据最大流最小割定理，$c(\Phi)$表示的最小切割Φ的的容量，其应该等于最大流量值，根据引理3.2 最大流量最多为$|\mathbb{AP}| + |\mathbb{ER}| \times (|\mathbb{AP}| + 1)$。根据容量设定原则如公式3.4所示，一条链路既不在$\mathbb{AP}$ 也不在$\mathbb{ER}$ ($e_i \notin \mathbb{AP}$ 和$e_i \notin \mathbb{ER}$), 则这条链路的容量为$c_{e_i} = |\mathbb{AP}| + (|\mathbb{AP}| + 1) \times |\mathbb{ER}| + 1$，这条链路是大于$|\mathbb{AP}| + (|\mathbb{AP}| + 1) \times |\mathbb{ER}|$。因此，这条链路不可能在$\mathbb{L}_\Phi$ 中。因此，割集$\mathbb{L}_\Phi$中的所有链路都必须属于边集合$\mathbb{AP}$ 或者$\mathbb{ER}$。 □

定理 3.2 如果在图G中一个单元流阻塞了全部边割集$\mathbb{L}_\Phi$的所有边，则在原图中不会存在任何流经过这个图的割Φ。

证明： 如果在图G中一个单元流阻塞了全部边割集$\mathbb{L}_\Phi$的所有边，则在原图中没有流能使用边割集$\mathbb{L}_\Phi$ 里的边和没有任何流能经过这个图的割Φ。 □

定理3.2提供了找到SRLG冲突链路集的可能性。即当AP路径遇到陷阱问题时，找到$\mathbb{AP}$ 路径边集的子集能够阻塞原有在边割集$\mathbb{L}_\Phi$ 的所有边，所以这个子集合即为SRLG 冲突链路集。当一条路径包含SRLG冲突链路集里的所有边，则没有任何流能经过割集Φ，因此没有与其对应的SRLG不相交路径BP。

3.4.2.4 SRLG冲突链路集的集合覆盖问题

虽然AP路径上的所有链路一起也可以构成SRLG冲突链路集，感兴趣的是尽量规模较小的SRLG冲突链路集，SRLG 冲突链路集的大小确定相互间互斥的子问题规模。

根据定理3.2，最小SRLG冲突链路集问题可以描述为：查找AP 链路集上的最小链路子集，这些链路可以阻塞边割集$\mathbb{L}_\Phi$。

对于任何链路e_i，$\mathbb{SR}_{e_i}$表示与链路e_i共享风险的链路集。显然，$\mathbb{SR}_{e_i}$包括e_i本身和所有包含链路e_i风险贡献链路组SRLG集合的所有边。例如，如图3.8所示，e_i是在两个SRLG中$\mathbb{R}_{r_2} = \{e_2, e_3, e_{19}\}$，$\mathbb{R}_{r_3} = \{e_2, e_4, e_{11}, e_{17}\}$，因此，$\mathbb{SR}_{e_2} = \{e_2, e_3, e_{19}, e_4, e_{11}, e_{17}\}$。

对每条在AP上的路径e_i，定义每条边的cut-block-link集合为$\mathbb{B}_{e_i} = \mathbb{SR}_{e_i} \cap \mathbb{L}_\Phi$，这个集合是边割集$\mathbb{L}_\Phi$的子集，能通过$e_i$而堵塞这个集合的所有边。

因此，最小SRLG冲突链路集问题可以定义为一个集合覆盖问题：给定$\mathbb{AP}$(路径AP上的链路集合)、边割集$\mathbb{L}_\Phi$和cut-block-link集合$\mathbb{B}_{e_1}, \mathbb{B}_{e_2}, \cdots, \mathbb{B}_{e_{|\mathbb{AP}|}}$。求出最小cut-block-link集合集，其交集是边割集$\mathbb{L}_\Phi$，即最小规模的$\mathbb{T} \subseteq \{e_i | e_i \in \mathbb{AP}\}$以至于$\cup_{e_i \in \mathbb{T}} \mathbb{B}_{e_i} = \mathbb{L}_\Phi$。

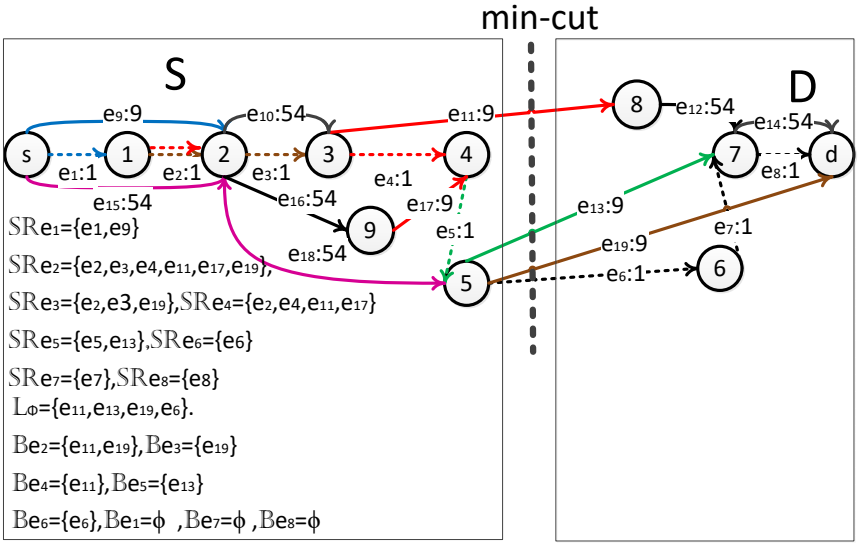


图 3.8 图G^*的最小割实例

集合覆盖问题通常是一个NP-hard问题。它的复杂性取决于元素的大小(表示(N)。在提出的最小SRLG 冲突链路集问题中，$n = |\mathbb{L}_\Phi|$，即割集$\mathbb{L}_\Phi$的边数。因为本文的重点不是改进集合覆盖问题算法，应用[66]中提出的算法。在复杂度$O(log(|\mathbb{L}_\Phi|))$的情况下求出最小的SRLG 冲突链路集，通常即使是在大规模的网络，最短的路径作为AP都没有大跳数$n = |\mathbb{L}_\Phi|$。因此，用集合覆盖问题来求最小值SRLG冲突链路集代价不是很大。

为了说明如何找到最小SRLG冲突链路集，如图3.8所展示了一个例子，

求最小$\Phi(\mathbb{S},\mathbb{D})$，$\mathbb{S}=\{s,1,2,3,4,5,9\}$和$\mathbb{D}=\{d,6,7,8\}$，边割集$\mathbb{L}_\Phi=\{e_{11},e_{13},e_{19},e_6\}$。对于AP路径上的所有链路，cut-block-link集合是：$\mathbb{B}_{e_1}=\emptyset,\mathbb{B}_{e_2}=\{e_{11},e_{19}\}$, $\mathbb{B}_{e_3}=\{e_{19}\},\mathbb{B}_{e_4}=\{e_{11}\},\mathbb{B}_{e_5}=\{e_{13}\},\mathbb{B}_{e_6}=\{e_6\},\mathbb{B}_{e_7}=\emptyset$ and $\mathbb{B}_{e_8}=\emptyset$。为了覆盖$\mathbb{L}_\Phi$，最少的cut-block-link集合是$\mathbb{B}_{e_2}=\{e_{11},e_{19}\}$,$\mathbb{B}_{e_5}=\{e_{13}\}$,$\mathbb{B}_{e_6}=\{e_6\}$。因此，最小SRLG冲突链路集是$\mathbb{T}=\{e_2,e_5,e_6\}$。

如图3.8所示的例子中，虽然$|\mathbb{L}_\Phi|=4$，但是最小SRLG冲突链路集$|\mathbb{T}|=|\{e_2,e_5,e_6\}|=3$是小于$|\mathbb{L}_\Phi|=4$。这是因为$e_2$属于$\mathbb{R}_{r_2}$和$\mathbb{R}_{r_3}$，并能阻塞在割集$\mathbb{L}_\Phi$中的两条链路$e_{11}$和$e_{19}$

根据SRLG的拓扑类型[65]，SRLG的拓扑类型：星型类型和非星型类型。对星型类型SRLG，所有链路都从同一个节点开始或结束在同一个节点。例如，如图3.8所示，e_1和e_9来自相同的节点s，$\mathbb{R}_{r_1}$是星型SRLG。对非星型类型，并不是SRLG中的所有链路都是从相同节点开始或结束于同一节点。如图3.8所示，$\mathbb{R}_{r_2},\mathbb{R}_{r_3},\mathbb{R}_{r_4}$和$\mathbb{R}_{r_5}$是非星型类型。而且，即使在图3.8所示包括星型SRLG和无星型SRLG，提出的算法高效且有效地通过解决集合覆盖问题来解决冲突集。

因此，与现有的一些研究不同的是，文献[65]只能处理单个SRLG类型，这样的简单场景中一条链路只属于一个SRLG，提出的算法能更有效的处理多种情形。在更一般的情况下，链路可以属于一个或多个SRLG具有更多不同的SRLG类型。

3.4.3 算法步骤

算法3.1显示了完整的min-min SRLG不相交路由算法。算法中的输入参数包括网络图(G)、源节点(s)、目的节点(d)、包含链路集应包括在AP($\mathbb{I}$)中，而排除链路集应不包含在AP($\mathbb{O}$)中。算法的输出结果是SRLG不相交路径对(AP,BP)。

为了寻找SRLG不相交路径，首先通过步骤2中的FIND_AP$(G,s,d,\mathbb{I},\mathbb{O})$搜索网络中最小权重AP路径，其中$\mathbb{I}=\phi,\mathbb{O}=\phi$，然后在步骤4中通过FIND_SRLG_Disjoint_BP (G,s,d,AP)搜索BP。特别是，可以通过Dijkstra算法找到AP路径。为了计算BP，FIND_SRLG_Disjoint_BP (G,s,d,AP)包括两个步骤。首先，对于AP上的所有链路，删除与这些链路有共同风险的链路。第二，Dijkstra的算法再次运行在网络其余的链路上，计算从s到d的第二条最短路径BP。

如果能找到一个SRLG不相交的BP路径，则解决Min-Min SRLG不相交的路由问题，并如步骤6所示返回找到的路径对。否则，就会出现陷阱问题。为了处理陷阱问题，步骤8首先找到SRLG冲突链路集$\mathbb{T}$，第10步将原Min-Min SRLG不相交路由问题划分为基于冲突集$\mathbb{T}$的$|\mathbb{T}|$个子问题。所有子问题都可以并行执行。步骤11使用集合$\mathbb{F}$存储满足$AP_i\neq\phi$和$BP_i\neq\phi$的可行解。在$\mathbb{F}$中的所有可行解中，选择AP路径权重最低的路径对作为原Min-Min SRLG不相交路由问题的最优解。

以图3.2中的例子来说明提出的算法3.1。根据步骤2，提出的算法首先通

过Dijkstra算法搜索路径权重最小的路径$\mathbb{AP}=\{e_1,e_2,e_3,e_4,e_5,e_6,e_7,e_8\}$，路径显示为图3.2(c)中的虚线。在删除AP上的链路以及与AP共享共同风险的链路之后，得到图3.2(d)，这是一个不连通图没有BP路径。然而，如图3.2(b)所示，拓扑中实际存在一对SRLG不相交路径。因此，陷阱问题就会发生。在步骤8中找到SRLG冲突链路集$\{e_2,e_5,e_6\}$之后，采用分而治之的方法将原问题$\mathcal{P}(\emptyset,\emptyset)$划分为三个子问题，根据步骤8，并行执行这些子问题$\mathcal{P}(\emptyset,\{e_2\})$, $\mathcal{P}(\{e_2\},\{e_5\})$和$\mathcal{P}(\{e_2,e_5\},\{e_6\})$后，返回具有最小AP权重的路径对。

算法 3.1 Min-Min SRLG不相交路径对算法

Input: G: 网络图
s: 源节点
d: 目的节点
$\mathbb{I}$: 必过链路集
$\mathbb{O}$: 必不过链路集
Output: AP: 主路径
BP: 备用路径

$AP=\emptyset$, $BP=\emptyset$, $\mathbb{I}=\emptyset$, $\mathbb{O}=\emptyset$
$AP \leftarrow$ FIND_AP($G,s,d,\mathbb{I},\mathbb{O}$)
if $AP \neq \emptyset$ **then**
  return $BP \leftarrow$ FIND_SRLG_Disjoint_BP(G,s,d,AP)
  if $BP \neq \emptyset$ **then**
    return 路径对(AP,BP)
  else
    找到SRLG冲突链路集$\mathbb{T}$
    $\mathbb{T} \leftarrow \mathbb{T}-(\mathbb{I}\cup\mathbb{O})$
    分而治之的并行执行

$$\begin{cases} (AP_1,BP_1)=Min-Min(G,s,d,\mathbb{I},\mathbb{O}\cup\{t_1\}), \\ (AP_2,BP_2)=Min-Min(G,s,d,\mathbb{I}\cup\{t_1\},\mathbb{O}\cup\{t_2\}), \\ (AP_3,BP_3)=Min-Min(G,s,d,\mathbb{I}\cup\{t_1,t_2\},\mathbb{O}\cup\{t_3\}), \\ \cdots \\ (AP_{|\mathbb{T}|},BP_{|\mathbb{T}|})=Min-Min(G,s,d,\mathbb{I}\cup\{t_1,t_2,\cdots,t_{|\mathbb{T}|-1}\},\mathbb{O}\cup\{t_{|\mathbb{T}|}\}) \end{cases}$$

$F \leftarrow$ FIND_FEASIBLE$((AP_1,BP_1)),\cdots,(AP_{|\mathbb{T}|},BP_{|\mathbb{T}|})$
    if $F \neq \emptyset,\emptyset$ **then**
      return 路径对(AP,BP) 满足条件$AP=\arg\min_{AP}\{F\}$
    end if
  end if
end if

3.4.4 算法时间复杂度

当AP遇到陷阱问题时为了找到SRLG不相交路径对，我的算法首先计算出SRLG冲突链路集，然后通过把原问题划分成T个子问题来求解原问题。如3.4.2.4节所述边割集$\mathbb{L}_\Phi$的边数规模通常不是很大，因此，本算法寻找SRLG冲突链路集不会带来太多的时间成本。因此，关注的是路径查找过程的计算开销。

一般来说，对于一个有$|\mathbb{E}|$条链路和$|\mathbb{V}|$个节点的网络，求最小权重路径问题的时间复杂性是$(|\mathbb{E}|+|\mathbb{V}|)\times log(|\mathbb{V}|)$。为了解决陷阱问题，提出的路径查找问题与最

初的最小权重路径问题有点不同。在路径查找的过程中引入了一些约束条件。例如，查找AP路径必须通过必过链路集$\mathbb{I}$和必不过链路集$\mathbb{O}$。由于这些链路集通常并不大，这些约束在成本计算上几乎没有差别。因为不同的子问题有不同的链路集，为了使描述简单明了，仍然使用$(|\mathbb{E}|+|\mathbb{V}|)\times log(|\mathbb{V}|)$作为一次路径搜索时间复杂度。而算法将原问题分成$|\mathbb{T}|$个子问题，算法复杂度为$|\mathbb{T}|\times(|\mathbb{E}|+|\mathbb{V}|)\times log(|\mathbb{V}|)$。

对于不同算法复杂性的比较，也展示了在CoSE[29]和KSP[63]在路径查找过程中的复杂性。

CoSE试图找到一个冲突的SRLG集合，而不是一个冲突链路集。但是他们寻找冲突的SRLG集的方法是穷尽的查找而且成本很高。在这主要分析研究路径查找过程时间成本。由于提出的SRLG冲突链路集是由最小割和集合覆盖问题导出的，$|\mathbb{T}|$是最小的SRLG冲突链路集规模。因此，在CoSE中的冲突SRLG集至少是$|\mathbb{T}|$，并且表示SRLG集合为$\{SRLG_1, SRLG_2, \cdots, SRLG_{|\mathbb{T}|}\}$，由于每个SRLG路径包含多条链路，因此CoSE的子问题应该比提出的要大得多。在AP路径上的一个SRLG的必过链路集合和不过链路集会产生$|SRLG|$个子问题。所以一个SRLG集合$\{SRLG_1, SRLG_2, \cdots, SRLG_{|\mathbb{T}|}\}$将引入$\prod_{i=1}^{|T|}|SRLG_i|$个子问题。因此CoSE的复杂性$\prod_{i=1}^{|\mathbb{T}|}|SRLG_i|\times(|\mathbb{E}|+|\mathbb{V}|)\times log(|\mathbb{V}|)$是比我提出的算法复杂度规模更大。

对于KSP算法[63]，路径查找复杂度为$K\times((|\mathbb{E}|+|\mathbb{V}|)\times log(|\mathbb{V}|))$，其中K是在发现SRLG不相交路径对之前应该测试的路径次数。然而，由于KSP没有从前面的路径搜索过程利用前面的信息，在最坏的情况下，KSP可能尝试从源节点s到目的节点d的所有路径。因此，最糟糕的K值是$2^{|\mathbb{E}|}$，这会带来很大的计算成本。

3.5 实验配置与评价指标

ubsection实验配置

因为找不到任何拓扑带有SRLG链路属性，所以通过注入SRLG信息生成一个合成的数据集。拓扑数据有7种不同的拓扑，具有不同的节点数目、链路数目、链路权重，如表3.1所示显示了拓扑的基本属性。这7 个拓扑中的节点、链路。

表 3.1 SRLG拓扑数据

拓扑	1	2	3	4	5	6	7
点	527	521	521	2023	451	521	449
边	4158	4052	4152	4142	2780	4052	2778
No.SRLG	132	86	89	207	210	128	88
SRLG边比率	9.66%	6.16%	6.18%	14.94%	22.55%	9.65%	9.53%

为了进行性能比较，除了我提出的算法(SCLS)，还实现了另外四个SRLG完全不相交路径对算法。算法如下：

(1) ILP[39]：中的工作旨在寻找SRLG不相交路径对。通过整数线性规划方程使

这两条路径总的权重最小化。没有找到其他通过整数线性规划方程寻找Min-Min SRLG不相交路径对的方法的研究。因此，从[39]的整数规划方程，通过改变目标函数构造Min-Min SRLG不相交路径对问题的整数线性规划方程。

(2) IQCP[39]：因为任意0-1整数线性规划方程，其中所有变量为0或1，原问题的整数线性规划方程可以表示为一个二次约束方程，设计了Min-Min SRLG不相交路径问题成一个整数二次约束规划(IQCP)[39]。

(3) KSP[63]：它在源节点s和目的节点d中找到第K短路作为候选AP路径，一个接一个地测试候选的AP路径是否有相应的SRLG不相交路径BP，直到发现了这样的BP，算法才结束。

(4) CoSE[29]：当AP遇到陷阱问题时，CoSE尝试进行简单而详尽的搜索，以找到一个SRLG集合。任何AP路径通过这个SRLG集都无法找到SRLG不相交的BP路径。基于这个SRLG集合，它划分原问题并设计算法来求SRLG不相交路径对。

3.5.1 评价指标

ILP和IQCP是基于整数规划模型的。在提出的实现中，工具GUROBI 7.0[67]用于解决这两个整数规划问题。六种性能指标用于评估不同的SRLG分路径算法：

- 路径权重：路径中链路权重的总和。
- 路径跳数：路径中的跳数。
- 运行时间：查找SRLG不相交路径对的归一化平均时间。
- 算法加速比：给定两种不同的算法(alg_1 和alg_2)的计算时间，表示为T_1和T_2，算法alg_1对于算法alg_2计算时间上的加速比是alg_1: $S_{1-2} = T_1/T_2$。
- 核加速比：并行程序的核心加速比[68] 通常定义为$S_P = \frac{T_1}{T_p}$，其中p是处理器内核和T_1和T_p表示在1核和p核上的运行时间。
- 效率[68]：定义为$E_p = \frac{S_p}{p} = \frac{T_1}{pT_p}$，它是百分比(0，1)的范围内。

所有实现都是在linux服务器上运行的，这个服务器配置Intel(R) Xeon(R) CPU E5-2620 2.00GHz 24 核和32.00GB内存。为了测量计算时间，在所有实现的算法中插入一个定时器。

通过在拓扑数据集里注入SRLG链路属性产生了两种SRLG类型，星型类型和非星型类型。在光纤网络中，SRLG是星型类型，而在其他网络类型中，例如一个Overlay网络中，SRLG 可以是非星型的。每个SRLG组是通过随机选择2-5链路组成的。在五种SRLG不相交路径对算法中，只有CoSE和SCLS是并行算法。尽管ILP、IQCP和KSP不是并行算法，仍然在实现它们来体现出我算法设计所获得的速度增益，平均化所有拓扑数据的结果为最终结果。

3.5.2 算法性能评估及比较

从节3.4.4分析，在KSP下求第一条K最短路的计算复杂度是$2^{|\mathbb{E}|}\times((|\mathbb{E}|+|\mathbb{V}|)\times log(|\mathbb{V}|))$，最坏的情况下这将是$K\times((|\mathbb{E}|+|\mathbb{V}|)\times log(|\mathbb{V}|))$。与分析一致，当使用7个拓扑运行KSP，没有仿真结果能在1 小时内返回，而其他算法则可以在11 秒内。计算时间长使得KSP难以在实践中使用。因此，不在结果显示KSP。

(1) 路径权重：如图3.9所示，显示AP路径权重、BP路径权重和AP和BP的总和路径权重。显然，所有的算法SCLS、CoSE、ILP和IQCP实现了相同的AP路径权重。但是不同的算法有不同的BP权重，因此它们有不同的路径权重和。因为所有算法都解决了SRLG不相交路径对问题，尽管他们发现不同的SRLG不相交路径对，它们都能达到找到最小权重相等的AP路径。然而，这两个基于整数线性规划的算法，ILP 和IQCP，主要是找到最小化AP的权重但是其随意找到其它任何SRLG不相交路径BP，因此这两个算法搜索到的BP路径是不同的。

(2) 路径跳数：图3.9显示AP路径跳数、BP 路径跳数和AP和BP路径跳数之和，因为所有算法的目标都是最小化SRLG不相交路径对的最小路径权重。如图3.9 所示在路径跳数中，它们具有相同的AP权重，但是如图3.10 所示他们有不同的AP 路径跳数。尽管所有算法的AP路径权重都小于BP 路径权重如图3.9所示，在图3.10中，AP跳数可能并不总是少于BP路径跳数。

(3) 运行时间：如图3.11所示，通过改变使用的CPU核数来展示不同算法下的运行时间。由于CoSE下的运行时明显大于其他算法的运行时，为了更清楚地展示其他算法的结果，在图3.12 中通过排除CoSE来进一步绘制运行时间的结果。由于ILP和IQCP不是并行算法，这些算法在不同核数下的运行时间大致相等。提出的SCLS和CoSE的运行时间随着处理器核数的增加而减少，因为这两种算法可以将原问题划分为多个子问题来并行执行，并利用多核CPU的并行性来加快路径搜索的速度，虽然CoSE 是一种并行算法，但计算时间比ILP和IQCP 还要大。一些可能的原因包括：1)在CoSE中冲突SRLG集合的搜索过程效率不高；2)由于一个SRLG 通常包含多条链路，基于冲突SRLG问题的划分将带来大量的子问题需要解决，这也将带来大量的计算量。与CoSE不同的是，当AP上遇到陷阱问题时提出的SCLS根据图中的最小割集理论来求SRLG冲突链路集，并达到图3.11中所示的最低时间消耗。这说明了提出的冲突链路集查找算法是有效的，并且提出的分治算法和基于SRLG冲突链路集的智能AP搜索过程，可以大大降低计算量。

(4) 算法加速比：如图3.14所示，进一步比较了它们的计算速度。特别地，为了找出使用不同算法寻找所需路径时所获得的加速比，使用CoSE作为基准算法，并设置alg_1 =CoSE。与图3.11中的结果相似，在图3.14 中SCLS的加速度是CoSE的600倍以上。在图3.14 中由于CoSE的运行速度明显小于其他算法，很难在图3.14中观察到，在图3.15 中排除了最大的SCLS 数据来进一步绘制了算法加速比结果。

(5) 核加速比:与使用算法加速比来比较所有算法的总体运行速度不同，这个度量“核加速比”是来评估CPU中的核数如何影响给定算法的运行速度。图3.13 绘制了所有实现的算法的核加速比。算法ILP和IQCP下的核加速比在任意核数下近似等于1，因为它们不是并行算法。当核数小于4 时，提出的SCLS的核心加速比随着核数的增加而增加，当超过4核时，SCLS 的核心加速比保持稳定，说明4 核对SCLS是足够的。这一结果与Amdahl定律[69]是一致的，即理论核加速比确定的上界限定于问题的规模大小。但是，即使核心数等于8，CoSE下的核加速比也继续增加，这是数字4 的两倍。结果表明，即使是8核CPU也不能满足COSE的并行性要求.这是因为CoSE发现的冲突SRLG集包含了大量的链路，这进一步导致了大量的子问题，从而导致了较大的问题规模和计算成本。

(6) 效率：与图3.13相似，随着更多的核增多效率值会降低，因为一个大的核心数会带来更多的成本来协调进程。如图3.16所示，所有算法的效率值都随着核心数的增加而降低。与图3.13中的结果一致，由于CoSE 引入的子问题比提出的SCLS多，在核心数达到4之后，CoSE下的效率大于SCLS。然而，如图3.14所示提出的SCLS实现了更大的算法加速比。

全部仿真结果表明，在搜索速度较快的情况下，SCLS算法的性能优于其他算法，因为算法发现的冲突链路集可以方便有效的执行并行算法并且计算量小。

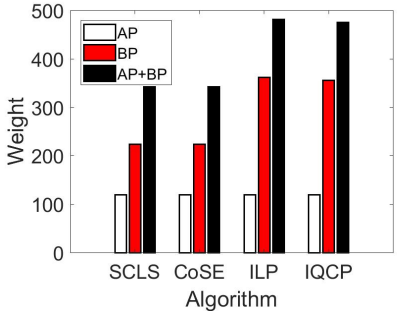


图 3.9 路径权重

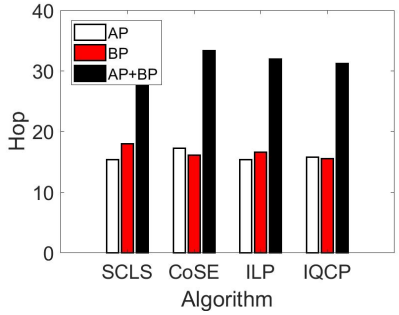


图 3.10 路径跳数

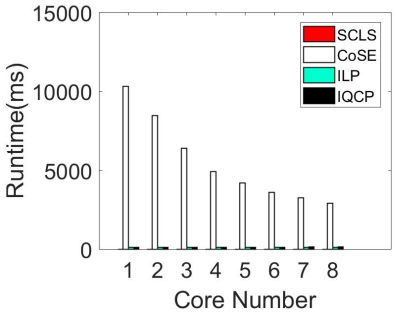


图 3.11 运行时间

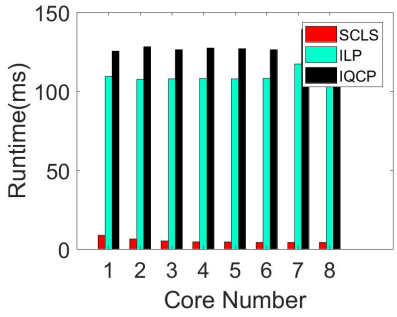


图 3.12 运行时间(无CoSE)

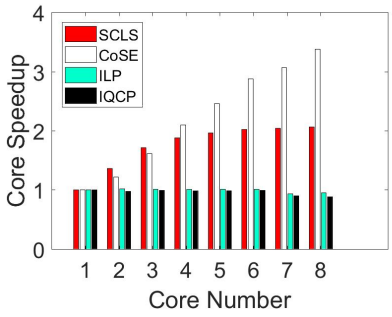


图 3.13 核加速比

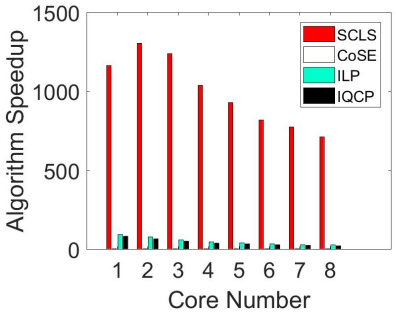


图 3.14 算法加速比

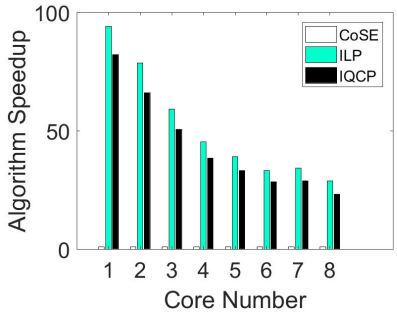


图 3.15 算法加速比(无SCLS)

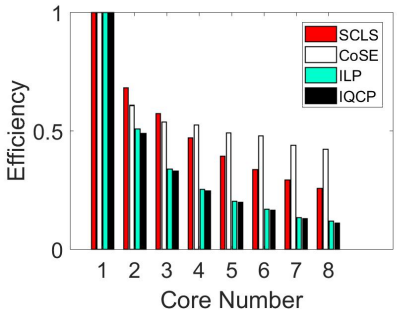


图 3.16 效率

3.6 小结

本章提出了一种分而治之算法，该算法能在遇到陷阱问题时将原有的min-min SRLG 不相交路由问题划分为多个并行执行的子问题。与现有其它算法相比，提出的解决方案在搜索过程中利用现有的AP 搜索结果来加快求最优结果的速度，并且并行执行来进一步加快路径查找的速度。

第4章　单物理节点故障可生存性虚拟网络嵌入问题

4.1　问题描述

在这一部分中，我们首先介绍了虚拟网络嵌入问题的基本概念，然后提出了我们的可生存性虚拟网络嵌入问题。

4.1.1 虚拟网络嵌入问题

我们将**虚拟网络**VN表示为无向图$G(V,E)$，其中V和E分别是虚拟节点和虚拟链路的集合。每个虚拟链路e_{ij}具有带宽需求d_{ij}。每个虚拟节点v_i具有计算容量需求d_i。对于虚拟节点v_i，需要在虚拟节点上执行的虚拟功能表示为$f(i)$。如图4.1所示虚拟网络$G(V,E)$具有虚拟节点集$V=\{v_1,v_2,v_3,v_4\}$和虚拟边集$E=\{e_{12},e_{13},e_{14},e_{23}\}$。需要在这些虚拟节点上执行的虚拟函数是$f(1)=f_1$，$f(2)=f_2$, $f(3)=f_3$, $f(4)=f_4$。

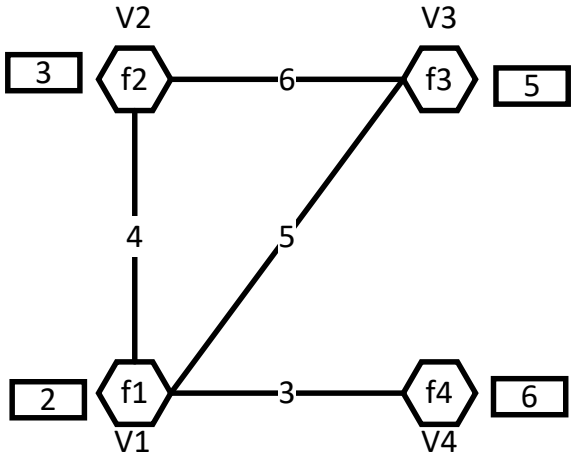


图 4.1 虚拟网络请求$G(V,E)$

我们将底层**物理网络**建模为一个无向图$G(S,L)$，其中S和L分别是物理节点和物理链路的集合。对于物理节点s_i，我们使用$F(i)$和c_i分别表示可以在该节点上执行的一组可行的虚拟功能和可用的计算能力。每个物理链路l_{ij}都有可用带宽b_{ij}。如图4.2所示的物理网络$G(S,L)$，物理节点集合$S=\{s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8\}$，链路集合$L=\{l_{12},l_{13},l_{14},l_{15},l_{23},l_{25},l_{35},l_{36},l_{37},l_{47},l_{58}\}$，每一个物理节点的可行的虚拟功能$F(1)=\{f_1\}$, $F(2)=\{f_2,f_3\}$, $F(3)=\{f_3\}$, $F(4)=\{f_4\}$, $F(5)=\{f_1,f_2\}$, $F(6)=\{f_1,f_4\}$, $F(7)=\{f_2,f_3\}$,$F(8)=\{f_2\}$。

在给定VN请求$G(V,E)$的情况下，虚拟网络嵌入问题的目的是将该请求映射到物理网络$G(S,L)$上，同时提供所需的足够资源。一个可行的嵌入应该满足节点容量约束、链路带宽约束和功能类型约束这三个约束条件。对于一个虚拟节点v_i，

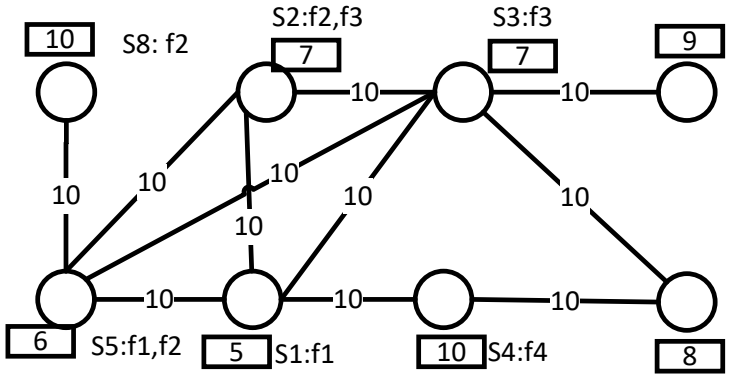


图 4.2 底层物理网络$G(S,L)$

物理节点s_j只在节点容量约束和功能类型两种情况满足约束条件（即$f_i \in F_j$）下被映射这个虚拟节点。即节点的容量请求应满足$d_i \le c_j$的物理节点，虚拟节点上需要执行的虚拟功能属于物理节点s_j上可以执行的虚拟功能集$f_i \in F_j$。如果物理节点s_j满足这两个约束，则节点映射是可行的，并且我们表示这样的映射为$\phi(v_i) = s_j$。

对于在两个已经映射到两个物理节点$s_{i'}$和$s_{j'}$(即$\phi(v_i) = s_{i'}$ 和$\phi(v_j) = s_{j'}$) 的虚拟节点之间的虚拟链路e_{ij}，在链路带宽约束下$d_{ij} \le b_{i'j'}$(其中$b_{i'j'}$是连接物理节点$s_{i'}$和$s_{j'}$的可用带宽)，这条虚拟链路e_{ij}可以映射到物理路径$p_{\phi(v_i)\phi(v_j)}$，则表示可行的链路映射为$\rho(e_{ij}) = p_{\phi(v_i)\phi(v_j)}$。

显然，为了找到一个可行的虚拟网络嵌入，我们需要找到两个映射函数ϕ和ρ来将所有虚拟节点映射到物理节点，以及将所有虚拟链路映射到物理路径。

如图4.3所示一个可行的虚拟网络嵌入，将图4.1中的虚拟网络嵌入到图4.2 中的物理网络中，其中虚拟节点v_1嵌入到物理节点s_1，虚拟节点v_2嵌入到物理节点s_2，虚拟节点v_3 嵌入到物理节点s_3，虚拟节点v_4嵌入到物理节点s_4。在图4.2 中，我们还展示了在这样映射之后物理网络中已经占用的和可用的资源。例如，对于物理节点s_1，其占用的计算容量为2，可用容量为5。

给定VN请求$G(V,E)$和物理网络$G(S,L)$，对于可行映射，我们将映射的物理图表示为$G\left(\hat{S},P\right)$，其中$\hat{S}$是容纳虚拟节点的物理节点集，其中$\hat{S} = \{s_{i'} : \phi(v_i) = s_{i'}, for\ \ all\ v_i \in V, s_{i'} \in S\}$和$P$是路径集，其中每个路径都持有一个虚拟链路$P = \{p_{\phi(v_i)\phi(v_j)} : \rho(e_{ij}) = p_{\phi(v_i)\phi(v_j)}, for\ all, e_{ij} \in E\}$。由于每个虚拟链路对应一个物理网络路径，该路径由多个物理链路组成，因此我们还将$G\left(\hat{S},\hat{L}\right)$表示为占领的物理网络$\hat{L} = \{l_{pg} : l_{pg} \in p_{s_{i'}s_{j'}}, \rho(e_{ij}) = p_{\phi(v_i)\phi(v_j)}, for\ all\ e_{ij} \in E, l_{pg} \in L\}$。

已经有了许多关于虚拟网络嵌入问题[70]的研究，文献[71]指出虚拟网络嵌入问题复杂度是NP-hard，因为本文的重点不是虚拟网络嵌入算法，我们采用了文献[72]中的算法作为基本的虚拟网络嵌入算法。

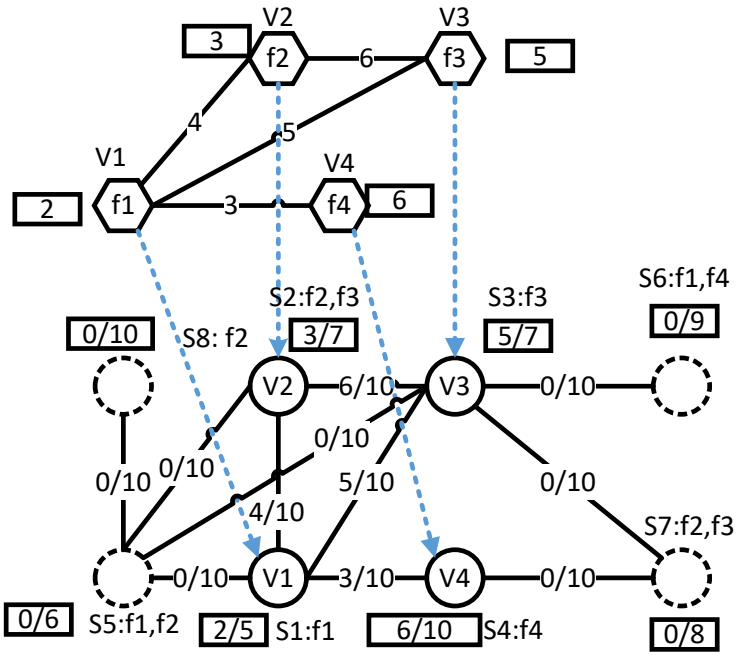


图 4.3 节点映射和链路映射的实例

4.1.2 可生存性虚拟网络嵌入问题

由于恶意攻击、自然灾害、无意中断电缆、计划维护、设备故障、物理节点承载的虚拟节点可能会遭受不可避免的故障。当单个或多个物理网络组件发生故障时，VN 中可能会发生故障，从而造成财务损失。一般来说，多个物理节点的同时失效是相互独立的，单个节点的失效通常发生在大多数时间[73]。本文研究了单节点失效的可生存性虚拟网络嵌入问题。在本章中，我们将讨论如何将我们的算法扩展到多节点故障的场景中。

对于VN请求$G(V,E)$和物理网络$G(S,L)$，给出了其占用物理网络的可行映射，在任何一个物理节点发生故障时，增加最小备份物理资源以提供可生存性的网络服务。

节点故障不仅影响运行在失效的物理节点上的可视化服务，而且会终止通过该节点的所有通信。物理节点$s_i \in S$的失效导致相邻物理链路的失效$L_i = \{l_{ik} : k \in N(i)\}$，其中$N(i)$是节点$s_i$的邻居节点。物理链路的失效场景可以通过增加中间节点的方法等价转换为节点故障的场景。

由于我们无法预测哪个节点将失效，即使我们知道多个节点不会同时失效，为了处理单节点故障，一个直接的方式是为VN请求中的每个虚拟节点提供专用备份资源，也称为1+1 保护方案。如图4.4所示利用一个例子来说明这种直截了当的方法。在该示例中，将具有4个虚拟节点的虚拟网络映射到物理网络，其中4 个物理节点参与了这样的嵌入。为了提供1 +1 保护方案，在图中添加了4 个备份节点、8 个备份链路。

1+1保护方案虽然实现简单，但需要大量的备份资源提去算法的目的是以最小的备份资源成本提供可生存性的网络服务。

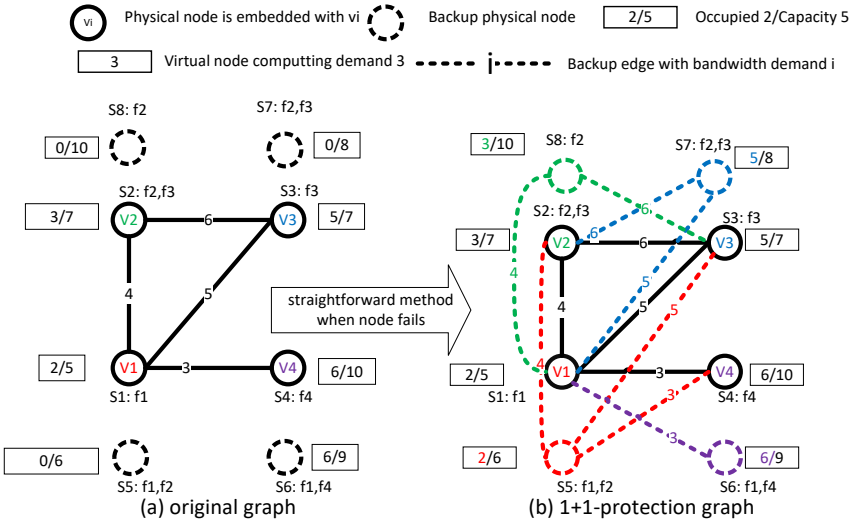


图 4.4 1+1-保护机制

4.2 星型图分解动态规划节点嵌入算法

可生存性虚拟网络嵌入需要添加备份资源，以保证当任何一个物理节点失效时，剩余的物理资源加上备份资源仍然可以支持原先的虚拟网络请求。为了便于在节点失效时找到可行的嵌入，本节首先对虚拟网络进行分解，物理网络以星型结构为基础的局部组件。在此基础上，提出了一种新的二分图，并将具有最小备份资源的可生存性虚拟网络嵌入问题转化为一个基于定义良好的二分图的虚拟星图分配问题。

4.2.1 基于星型图的分解

将虚拟网络分解为虚拟局部星型图。每个虚拟局部星型图与一个虚拟节点相关联。给出虚拟节点v_i，相应的虚拟局部星型图被定义为属性化单层根树，并如公式4.1 所示。

$$VS(v_i) = (v_i, \phi(v_i), d_i, f_i, D_i, N_i) \tag{4.1}$$

其中，N_i是虚拟网络中的相邻节点集，并与节点v_i相关联的带宽需求集$D_i = \{d_{ij} | v_j \in N_i\}$。注意，VS($v_i$) 包括节点映射信息$\phi(v_i)$。因为我们希望最小化备份资源以提供可生存性的网络服务，在节点失效前重复使用映射可能减少额外资源，使系统保持稳定的一个很好的选择。在虚拟星型图结构中，根节点和它的邻居节点存在链路，但是根节点的邻居节点之间不存在链路。

物理网络被分解为物理局部星型图。同样，每个物理局部星型图与一个物理

节点相关联。给定物理节点s_j，对应的物理局部星型图被定义为属性化单层根树，并表示如公式4.2所示。

$$PS(s_j) = (s_j, \phi^{-1}(s_j), c_j, F(j), \phi(N(\phi^{-1}(s_j))), a) \tag{4.2}$$

其中$\phi^{-1}(s_j)$是映射到物理节点s_j的虚拟节点集，$N(\phi^{-1}(s_j))$是$\phi^{-1}(s_j)$中虚拟节点的相邻节点，$\phi(N(\phi^{-1}(s_j)))$是承载这些邻居的物理节点，c_j是节点容量，$F(j)$是s_j支持的虚拟功能，a是一个1 位的单比特，它的值为0 或1，以指示这个物理节点是否开启和设置了虚拟机。与虚拟星型图类似，在物理星型图结构中，根节点与相邻节点之间存在边，相邻节点之间不存在边。定义在4.1 和4.2 中的虚拟星型图和物理星型图很好地捕获了隐藏在VN 请求中的局部结构，以保持节点与其相邻节点之间的关系。基于虚拟局部星型图和物理虚拟星型图，虚拟网络和物理网络可以分解为多个组件。

4.2.2 构建星型二分图

构造了一个二分图$G = \{V_1, V_2, E\}$来表示虚拟网络与物理网络之间的关系。V_1和V_2分别表示虚拟星型图和物理星型图的集合。如果VS(v_i)的虚拟功能f_i能在具有$f_i \in F_j$的物理节点s_j上执行，则在边集E中添加边$e(i, j)$，以连接VS(v_i)和PS(s_j)。

我们的目标是最小化备份资源以提供可生存性的服务。为了服务于目标，给定边$e(i, j)$，我们定义了边权$w(i, j)$作为备份资源成本。当节点发生故障时，将虚拟星型图(v_i)映射到物理星型图(s_j)。根据虚拟节点v_i映射到物理节点s_j节点是否失效，如公式4.3所示在两种不同的情况下定义了边权重$w(i, j)$。

$$w(i,j) = \begin{cases} \alpha \sum\limits_{\phi(v_k) \notin \phi(N(\phi^{-1}(s_j)))} d_{ik} & v_i \in \phi^{-1}(s_j), v_k \in N(i) \\ \alpha \sum\limits_{k \in N(i)} d_{ik} + \beta M_m + \lambda c_i + \theta & v_i \notin \phi^{-1}(s_j), v_k \in N(i) \end{cases} \tag{4.3}$$

在公式.(4.3)中，θ被定义如下.

$$\theta = \begin{cases} C_s & a = 0 \\ 0 & a = 1 \end{cases} \tag{4.4}$$

在第一种情况下$(v_i \in \phi^{-1}(s_j))$，由于虚拟节点v_i映射到的物理节点s_j没有失效，节点容量需求已经得到满足，因此当将虚拟星(v_i)映射到物理星型图(s_j)时只需要带宽备份成本。对于每个相邻的$v_k \in N(i)$，如果物理节点$\phi(v_k) \notin \phi(N(\phi^{-1}(s_j)))$包含失效虚拟节点$v_k$，则应添加具有带宽$d_{ik}$的新路径作为备份资源。因此，在这种情况下，备份成本只包括带宽成本，并表示为$\alpha \sum\limits_{\phi(v_k) \notin \phi(N(\phi^{-1}(s_j)))} d_{ik}$，其中$\alpha$是单位带宽

成本。

在第二种情况下($v_i \notin \phi^{-1}(s_j)$)，由于虚拟节点$v_i$在节点失效之前没有映射到物理节点$s_j$，因此需要将虚拟节点$v_i$迁移到物理节点$s_j$。因此，边权重$w(i,j)$)包括节点容量成本、路径带宽成本和迁移成本，以便在物理节点失效时将虚拟节点从物理节点迁移到另一个物理节点。此外，如果备份物理节点s_j之前不包含任何虚拟节点，则将虚拟节点迁移到该物理节点也会引入虚拟机启动成本，其表示为C_s。

如图4.5所示显示了当物理节点s_1失效时这种二分图的一个例子。这种二部图的边权如下矩阵4.5 表示。

	R_{S_1}	R_{S_2}	R_{S_3}	R_{S_4}	R_{S_5}	R_{S_6}	R_{S_7}
L_{V_1}	∞	∞	∞	∞	$\boxed{C_s + M_m + (2) + 12}$	$C_s + M_m + (2) + 12$	∞
L_{V_2}	∞	$\boxed{4}$	∞	∞	$C_s + M_m + (3) + 10$	∞	$C_s + M_m + (3) + 10$
L_{V_3}	∞	$M_m + (5) + 11$	$\boxed{5}$	∞	∞	∞	$C_s + M_m + (5) + 11$
L_{V_4}	∞	∞	∞	$\boxed{3}$	∞	$C_s + M_m + (6) + 3$	∞

(4.5)

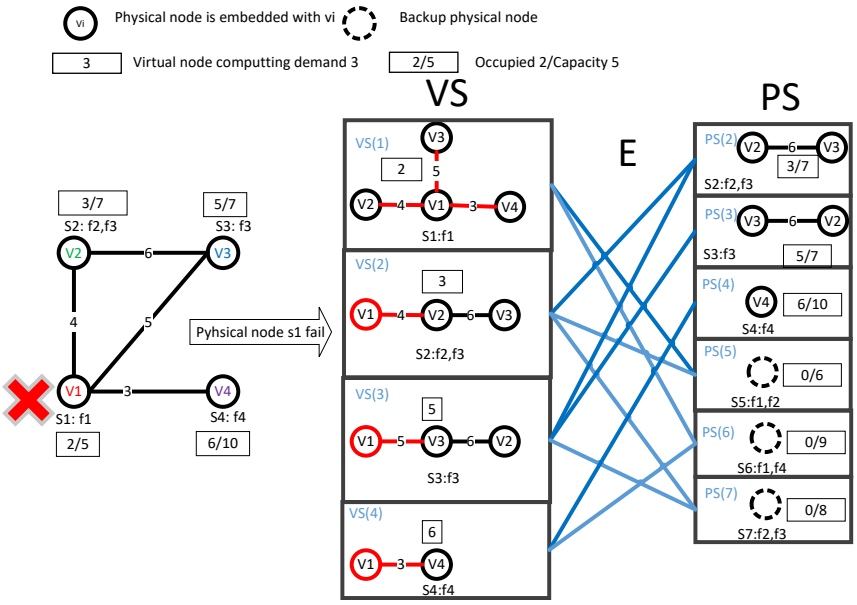
对于虚拟节点v_i，如果它的虚拟功能不能在具有$f(i) \notin F(i)$的物理节点s_j中执行，则没有边连接在V_1中的$VS(v_i)$和在V_2中的$PS(s_j)$。为了便于问题表述，我们将边权设为∞。例如，由于v_1的虚拟功能是f_1，所以它不能在物理节点s_2中执行。因此，与$VS(v_1)$和$PS(s_2)$连接时，不添加任何边，我们设置$w(1,2) = \infty$。

对于$VS(v_2)$，因为v_2最初由s_2持有，但是，当物理节点s_1(最初持有v_1)失效时，虚拟链路e_{12}无法映射到物理网络。我们应该找到一条连接$\phi(v_2)$和$\phi(v_1)$满足带宽需求$d_{12} = 4$的新路径。因此，边权系数$VS(v_2)$和$PS(s_2)$的值为4。

当节点s_1失效时，它直接影响虚拟节点v_1。由于s_5可以执行虚拟功能f_1，我们可以添加一个边来连接$VS(v_1)$和$PS(s_5)$。然而，在由于s_5以前没有设置虚拟机，因此还引入了虚拟机的启动成本C_s。因此，链路权重为C_s(启动成本)+M_m(迁移成本)+3(节点容量成本)+10(带宽成本)。

4.2.3 星型二分图匹配问题定义

为了提供可生存性的服务，每个虚拟星型图(根虚拟节点和连接根节点及其相邻节点的虚拟链路)都应该映射到物理星型图。假设虚拟网络由n 个虚拟节点组成，因此有n个虚拟星型图。物理网络由m 个物理节点组成，从而构成m 个物理星型图。在等式4.6中，我们使用二进制位M_{ij}来表示第i 个虚拟星型图是否映射到



	$PS(1)$	$PS(2)$	$PS(3)$	$PS(4)$	$PS(5)$	$PS(6)$	$PS(7)$
$VS(1)$	∞	∞	∞	∞	$C_s + M_m + \lambda(2) + \alpha 12$	$C_s + M_m + \lambda(2) + \alpha 12$	∞
$VS(2)$	∞	4	∞	∞	$C_s + M_m + \lambda(3) + \alpha 10$	∞	$C_s + M_m + \lambda(3) + \alpha 10$
$VS(3)$	∞	$M_m + \lambda(5) + \alpha 11$	5	∞	∞	∞	$C_s + M_m + \lambda(5) + \alpha 11$
$VS(4)$	∞	∞	∞	3	∞	$C_s + M_m + \lambda(6) + \alpha 3$	∞

图 4.5 当物理节点s_1 失效时VS(v_i)和PS(s_j)

第j 物理星型图。

$$M_{ij} = \begin{cases} 1 & map\ v_i\ to\ s_j \\ 0 & otherwise \end{cases} \tag{4.6}$$

当物理节点发生故障时，为了使备份资源成本最小化，可生存性虚拟网络嵌入问题可以定义如下：

$$\begin{aligned} \min_{M_{ij}} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} M_{ij}w_{ij} \\ s.t., \quad & \sum_{i=1}^{n} d_i M_{ij} \le c_j \\ & \sum_{j=1}^{m} M_{ij} \le 1 \\ & M_{ij} = \{0,1\} \end{aligned} \tag{4.7}$$

其中$\sum_{i=1}^{n}\sum_{j=1}^{m} M_{ij}w_{ij}$ 表示当物理节点失效时将所有虚拟星型图映射到物理星型图的总备份资源成本。在公式4.7 中，$\sum_{i=1}^{n} d_i M_{ij} \le c_j$是物理节点的容量约束，也就是说，即使允许将多个虚拟节点映射到一个物理节点，总容量需求不应大于物理节

点的容量。$\sum_{j=1}^{m} M_{ij} \leq 1$ 表示一个虚拟星型图只映射到一个物理星型图。

显然，公式4.7中定义的问题是一个二进制ILP 问题，根据Karp 的21 个NP- 完全问题[74]，它是一个一般的NP-complete 问题。

定理 4.1 公式4.7是一个NP-complete 问题。

证明： 如果只有一个物理节点m=1，则我们的可生存性虚拟网络嵌入问题可以退化为单背包问题，即NP-完全问题。在实际应用中，m 通常大于1，单背包问题是可生存性虚拟网络嵌入问题的子问题，给出的可行解在多项式时间内很容易验证，根据计算机复杂性领域上的可归约性定理[75]，得出我们在公式4.7 中定义的问题是NP-complete 的结论。 □

4.2.4 动态规划节点嵌入方法

虽然求解ILP公式会得到最小成本的可生存性虚拟网络嵌入，但其指数时间复杂度使得这种方法在大型物理网络中嵌入虚拟网络是不可行的。在这一部分中，我提出了一种基于动态规划的算法，该算法仅具有多项式时间复杂度，因此对于实际的网络系统是可行的方法。

为了描述基于动态规划的算法，我们定义了$dp[i][x_1][x_2]\ldots[x_m]$ 表示以最小备份资源代价将前面$i(0 \leq i \leq n)$个虚拟星型图放置到物理网络中的m个物理星型图中，其容量限制为$x_1, x_2, \ldots, x_m$。

第i个虚拟节点可以选择放置在任何一个存活的的物理星型图上。设$\theta(i, j)$ 表示已经将原先$i-1$个虚拟节点最佳放置后再将第i 个虚拟节点放置到第j 个物理节点的备份资源成本。$\theta(i, j)$表示如下:

$$\theta(i,j) = \begin{cases} dp[i-1][x_1][x_2]\ldots[x_j - d_i]\ldots[x_m] + w_{ij} & (x_j \geq d_i, f_i \in F_j) \\ \infty & otherwise \end{cases} \tag{4.8}$$

在等式4.8中，如果物理节点x_j的容量限制大于容量需求d_i 并且在物理节点$f_i \in F_j$能执行虚拟功能f_i，则$\theta(i, j)$ 是已经放置前面i-1 个最佳虚拟星型图的代价和(即$dp[i-1][x_1 - d_i][x_2]\ldots[x_m]$) 和将虚拟星型图$(v_i)$ 映射到物理星型图(s_1)的成本(即w_{i1})。基于$\theta(i, j)$，$dp[i][x_1][x_2]\ldots[x_m]$可通过以下动态规划函数计算。

$$dp[i][x_1][x_2]\ldots[x_m] = min\{\theta(i,1), \theta(i,2), \ldots, \theta(i,j), \ldots, \theta(i,m)\} \tag{4.9}$$

基于动态规划的算法如伪码4.2.4所示。我们还以如图4.6所示中的一个例子来说明该算法。

在本例中为了清晰地表示，需要将三个虚拟星型图放置到两个可用的物理星型图以实现最小的备份资源成本。物理节点下的可用容量分别为$c_1 = 5$ 和$c_2 = 4$。

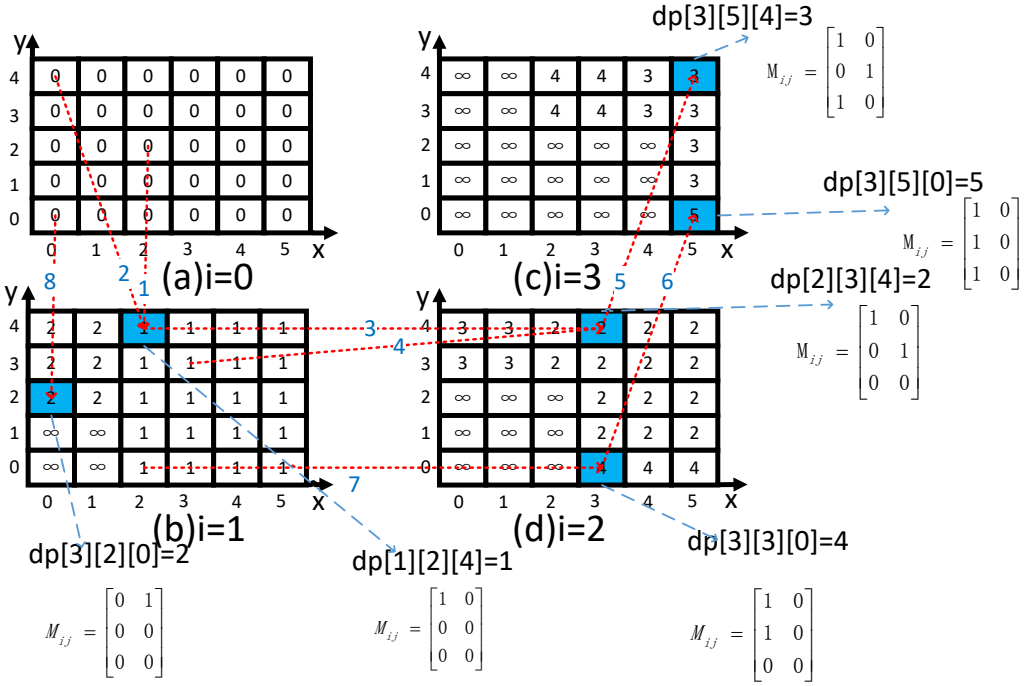


图 4.6 动态规划解法演示

这三个虚拟星型图的容量需求分别为$d_1 = 2$，$d_2 = 1$和$d_3 = 2$。在这些虚拟星型图中需要执行的虚拟功能是$f(1) = f_1, f(2) = f_1, f(3) = f_2$。这两个物理节点支持的虚拟功能是$F(1) = \{f_1, f_2\}$ 和$F(2) = \{f_1\}$。

如图4.6所示，x和y 轴分别表示物理星型图s_1 和s_2 的容量极限。虚拟星型图与物理星型图的连接边的权重$w_{11} = 1, w_{12} = 2$，$w_{21} = 3$，$w_{22} = 1$，$w_{31} = 1$，$w_{32} = \infty$。最初，在图4.6(a) 中，由于没有对任何物理星型图放置虚拟星型图，所有容量限制情况下的备份成本(x_1=0,1,2,3,4和x_2=0,1,2,3,4,5) 都是0。特别是，即使设置了4和5 的容量限制，dp[0][4][5]=0。如图4.6(b)所示，当将容量要求为$d_1 = 1$ 的第一个虚拟节点v_1 放置到这两个物理节点时，可以在两个物理节点中执行f_1。因此，我

算法 4.1 基于动态规划方法二分星型图匹配算法

Input: $dp[i][x_1][x_2]\ldots[x_m] = 0(1 \le i \le n, 0 \le x_1 \le c_1, 0 \le x_2 \le c_2, \ldots, 0 \le x_m \le c_m)$ 首先被定义成无穷大∞，$dp[0][x_1][x_2]\ldots[x_m] = 0(0 \le x_1 \le c_1, 0 \le x_2 \le c_2, \ldots, 0 \le x_m \le c_m)$, m：物理节点的数量。$M[x_1][x_2]\ldots[x_m] = \mathbf{0}_{n\times m}$ 每个节点的映射矩阵

Output: 当节点容量为$c_1, c_2, \ldots, c_m$时最小代价的节点映射

1: **for all** i such that $1 \le i \le n$ **do**
2: **for all** $x_1, x_2, \ldots, x_m$ such that $c_1 \ge x_1 \ge d_i$, $c_2 \ge x_2 \ge d_i$,$c_3 \ge x_3 \ge d_i$,$c_m \ge x_m \ge d_i$ **do**
3: $dp[i][x_1][x_2]\ldots[x_m] = min\{\theta(i,1), \theta(i,2), \ldots, \theta(i,j), \ldots, \theta(i,m)\}$
4: $j' = \arg\min_j \{\theta(i,1), \theta(i,2), \ldots, \theta(i,j), \ldots, \theta(i,m)\}$, .
5: $M[x_1][x_2]\ldots[x_m] = M[x_1][x_2]\ldots[x_{j'} - d_i]\ldots[x_m]$
6: $M[x_1][x_2]\ldots[x_m]_{ij'} = 1$
7: **end for**
8: **end for**
9: **return** $dp[i][c_1][c_2]\ldots[c_m]$ and $M[c_1][c_2]\ldots[c_m]$

们有

$$dp[1][x_1][x_2] = \min\{\theta(1,1), \theta(1,2)\} \tag{4.10}$$

特别是，如果这两个物理节点的容量极限分别为x_1=2和x_2=0，则$dp[1][2][0] = dp[0][0][0] + w_{11} = 2$。如果这两个物理节点的容量极限分别为$x_1$=2和$x_2$=4，则$\theta(1,1) = dp[0][0][2] + w_{11}$和$\theta(1,2) = \infty$，从而$dp[1][2][4] = min\{dp[0][0][4] + w_{11}, dp[0][2][2] + w_{12}\} = 1$。

同样，当将容量要求$d_2 = 2$的第二个虚拟节点放置到这两个物理节点时，所有容量限制下的成本结果如图4.6(d)所示。因为f_2可以在两个物理节点中执行，因此v_2可以放置在两个物理节点中，所以我们有

$$dp[2][x_1][x_2] = \min\{\theta(2,1), \theta(2,2)\} \tag{4.11}$$

特别是，如果这两个物理节点的容量极限分别为x_1=3和x_2=4，则$dp[2][3][4] = min\{dp[1][2][4] + w(2,1)$，$dp[1][3][3] + w(2,2)\} = 2$。如果这两个物理节点的容量极限分别为$x_1$=3和$x_2$=0，则$dp[2][3][0] = dp[1][2][0] + w(2,1) = 4$。

如图4.6(d)显示了将容量要求为$d_3 = 1$的第三个虚拟节点放置到这两个物理节点时的最小资源成本结果。由于f_3只能在物理节点s_1中执行，所以我们有$\theta(3,2) = \infty$。特别是，如果这两个物理节点的容量极限分别为x_1=5 和x_2=0，则$dp[3][5][0] = dp[2][3][0] + w(3,1) = 4$。由于这两个物理节点的节点能力分别为5 和4，我们得到了$dp[3][5][4] = dp[2][3][4] + w(3,1) = 3$，如图4.6(d)所示，在dp[3][5][4]=3处得到了最佳位置，节点映射为$v_1 \rightarrow s_1, v_2 \rightarrow s_2, v_3 \rightarrow s_1$。

例如，如图4.5和等式4.5 所示，其最小资源成本结果的最优节点映射矩阵如矩阵4.12 所示。

$$M_{ij} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.12}$$

4.2.5 嵌入备份资源

如果一个节点失败，则在应用算法4.2.4 计算最小额外资源后，添加新的物理节点和物理路径来维护网络可生存性服务。如果不包含任何虚拟节点的物理节点在节点失效前就已经添加为备份节点，我们应该将这个物理节点设置为a=1，以指示该物理节点已经容纳过虚拟节点。

由于我们的可生存性虚拟网络嵌入问题是为了在任何一个节点失效时最小化

增加的备份资源，而不是给定的一个节点故障，我们应该逐一测试每个节点的故障，并添加足够的备份资源。作为备份资源只需要当节点失效时，备份资源应在不同节点失效时共享。如果我们直接应用公式4.3中定义的成本来计算另一个物理节点失效时的备份资源，因为公式4.3中定义的成本不考虑备份资源共享，则会导致重复添加备份资源的问题。

为了解决这一问题，我们将$M(j)$加入到物理星型结构中，表示在节点失效的情况下迁移到物理节点s_j中的虚拟节点集。

$$PS(s_j) = (s_j, \phi^{-1}(s_j), c_j, F(j), \phi(N(\phi^{-1}(s_j))), a, M(j)) \tag{4.13}$$

由于备份资源应该只添加一次，为了便于表示该约束，我们定义了以下函数。

$$\mu(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{4.14}$$

当我们测试另一个节点失效时，而不是等式4.3 中定义的代价，我们定义了一个新的代价函数，考虑到不同节点失败时的备份资源共享。

如等式4.15所示，资源成本是根据两种情况设置的。如果虚拟节点v_i最初由物理服务器s_j持有，或者v_i被迁移到物理服务器，即$v_i \in \left(\phi^{-1}(s_j) \cup M(j)\right)$，当它的邻居$v_k \in N(i)(\phi(k) \notin \phi\left(\phi^{-1}(s_j)\right))$失效时，路径带宽成本为$\sum_{k \in N(i) \cap N(i')} \mu(d_{ik} - d_{i'k}) + \sum_{k \in (N(i) - N(i'))} d_{ik}$。

否则，在原先节点失效的测试前这个虚拟节点v_i没有被一个物理节点持有和迁移到其他物理节点，除了这个带宽成本为$\sum_{k \in N(i) \cap N(i')} \mu(d_{ik} - d_{i'k}) + \sum_{k \in (N(i) - N(i'))} d_{ik}$，节点的容量成本为$\lambda * \mu(d_i - \max_{v_{i'}}(d_{i'}))$，虚拟服务器迁移的代价为$\beta * M_m$和$\theta$是从映射虚拟星型图$v_i$到物理星型图$s_j$的代价。如$\mu(d_i - \max_{v_{i'}}(d_{i'}))$而言，如果备份容量已分配的最大值$\max_{v_{i'}}(d_{i'})$大于当前映射(即$d_i$)，不需要任何资源，否则，备份资源还缺少$d_i - \max_{v_{i'}}(d_{i'})$。

对于$VS(v_2)$，由于v_3最初由s_3持有，但是当物理节点s_1失效时，我们可以将虚拟节点v_3迁移到物理节点s_2，因为s_2是以前已经安装过虚拟机，因此，成本包括虚拟链路带宽成本11，节点容量成本5，虚拟机迁移成本M_m。

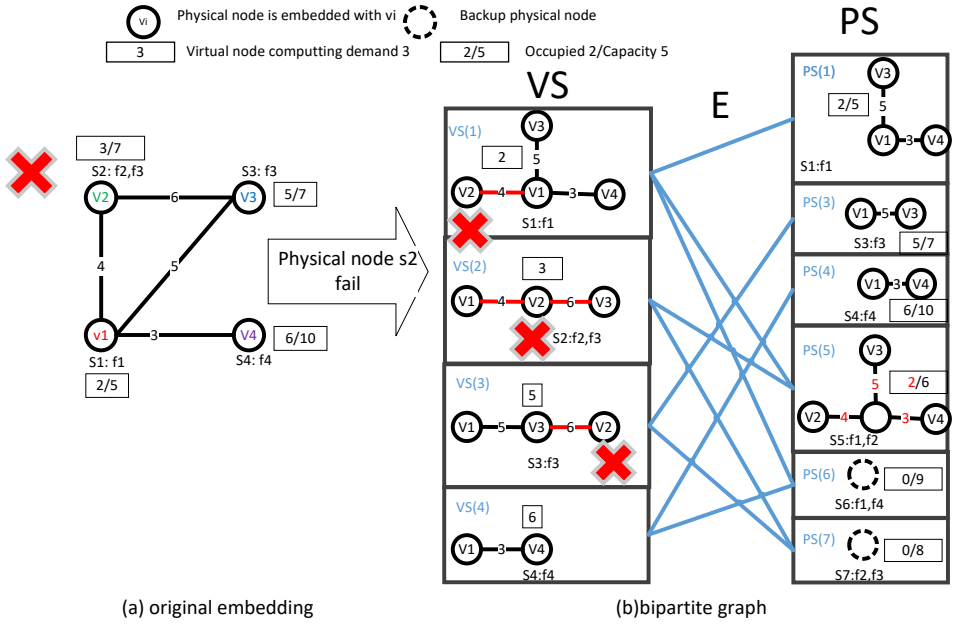
基于节4.2.4的方法，在物理节点s_1失效时得到最佳节点映射：$v_1 \rightarrow s_5$, $v_2 \rightarrow s_2, v_3 \rightarrow s_3, v_4 \rightarrow s_4$。物理节点$s_5$开始为虚拟节点$v_1$建立和应用2个节点计算资源，并分别找到带宽约束为4、5、3 的对应的虚拟链路$(v_1, v_2), (v_1, v_3), (v_1, v_4)$的

$$w(i,j) = \begin{cases} \sum_{k \in N(i) \cap N(i')} \mu(d_{ik} - d_{i'k}) + \sum_{k \in (N(i) - N(i'))} d_{ik} & v_i \in \left(\phi^{-1}(s_j) \cup M(j)\right), v_{i'} \in M(j), k \notin \phi\left(\phi^{-1}(s_j)\right) \\ \sum_{k \in N(i) \cap N(i')} \mu(d_{ik} - d_{i'k}) + \sum_{k \in (N(i) - N(i'))} d_{ik} + \lambda\mu(d_i - \max_{v_{i'}}(d_{i'})) + \beta M_m + \theta & v_i \notin \left(\phi^{-1}(s_j) \cup M(j)\right), v_{i'} \in M(j) \end{cases} \tag{4.15}$$

三个物理路径。

如图4.7展示了二分图的一个例子，物理节点s_2在物理节点s_1失效之后再失效。这个二分图的边权矩阵可以用矩阵4.16表示。

$$\begin{array}{c|ccccccc} & R_{S_1} & R_{S_2} & R_{S_3} & R_{S_4} & R_{S_5} & R_{S_6} & R_{S_7} \\ L_{V_1} & \boxed{4} & \infty & \infty & \infty & M_m & C_s+M_m+(2)+12 & \infty \\ L_{V_2} & \infty & \infty & \infty & \infty & \boxed{M_m+(1)+5} & \infty & C_s+M_m+(3)+10 \\ L_{V_3} & \infty & \infty & \boxed{5} & \infty & \infty & \infty & C_s+M_m+(5)+11 \\ L_{V_4} & \infty & \infty & \infty & \boxed{3} & \infty & C_s+M_m+(6)+3 & \infty \end{array} \tag{4.16}$$



$$\begin{array}{c|ccccccc} & R_{S_1} & R_{S_2} & R_{S_3} & R_{S_4} & R_{S_5} & R_{S_6} & R_{S_7} \\ L_{V_1} & 4 & \infty & \infty & \infty & 0 & C_s+M_m+\lambda(2)+\alpha 12 & \infty \\ L_{V_2} & \infty & \infty & \infty & \infty & M_m+\lambda(1)+\alpha 5 & \infty & C_s+M_m+\lambda(3)+\alpha 10 \\ L_{V_3} & \infty & \infty & 5 & \infty & \infty & \infty & C_s+M_m+\lambda(5)+\alpha 11 \\ L_{V_4} & \infty & \infty & \infty & 3 & \infty & C_s+M_m+\lambda(6)+\alpha 3 & \infty \end{array}$$

(c)

图 4.7 当物理节点s_2失效时VS(v_i)和PS(s_j)

如图4.8所示给出了当物理节点s_1和s_2同时失效时星型二分图的一个例子。这种二分图的边权矩阵可以用矩阵4.17表示。

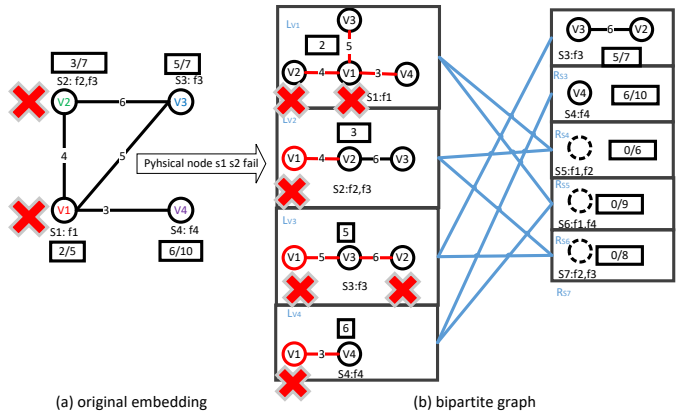


图 4.8 当物理节点s_1和s_2同时失效时VS(v_i)和PS(s_j)

$$
\begin{array}{c|ccccccc}
 & R_{S_1} & R_{S_2} & R_{S_3} & R_{S_4} & R_{S_5} & R_{S_6} & R_{S_7} \\
L_{V_1} & \infty & \infty & \infty & \infty & \boxed{C_s + M_m + (2) + 12} & C_s + + M_m + (2) + 12 & \infty \\
L_{V_2} & \infty & \infty & \infty & \infty & C_s + M_m + (3) + 10 & \infty & \boxed{C_s + M_m + (3) + 10} \\
L_{V_3} & \infty & \infty & \boxed{5} & \infty & \infty & \infty & C_s + M_m + (5) + 11 \\
L_{V_4} & \infty & \infty & \infty & \boxed{3} & \infty & C_s + M_m + (6) + 3 & \infty
\end{array}
\tag{4.17}
$$

4.2.6 算法步骤

在这一部分中，我们首先给出了SVNE图的增广资源分配过程。下面将详述更多细节。

此外，对于节点嵌入和虚拟链路嵌入，由于单个节点失效时并不是所有的虚拟链路或它们的带宽都会同时使用，所以如果将虚拟链路嵌入在同一底层物理链路上，则一些虚拟链路可以共享底层物理资源。减少总底层物理带宽所需。简单地说，底层物理链路对应不同虚拟网络节点失效时可生存性请求的备份带宽可以相互共享。

从节4.2.4中，我得到了节点v_i失效的节点映射和链路映射关系，和计算出每个节点需要重新分配的计算量，以及每个链路在底层物理网络SN中应该重新分配的带宽，如图4.9所示。分别在物理节点$s_1, s_2, s_3, s_4, s_5, s_6, s_7$中增加计算资源0、2、0、0、3、6、0。查找带宽分别为1，4，6，3，6，6与物理路径$p_{s_1s_2}, p_{s_1s_3}, p_{s_1s_5}, p_{s_1s_6}, p_{s_2s_5}$对应的五条路径。尽管虚拟网络嵌入算法很多，但相对于我们的算法，节点映射阶段已经完成，接下来的步骤描述了链路映射阶段。我们使用标准最短路径算法Dijkstra 算法[76]来请求底层物理节点的扩展路径，并为这种虚拟网络生存的请求的部分带宽重新分配给这些路径。一些研究[47]集中在路径嵌入和路径分割嵌入，以实现高的链路利用率、链路压力和其他目标，但本文没有重点讨论链路映射问题。

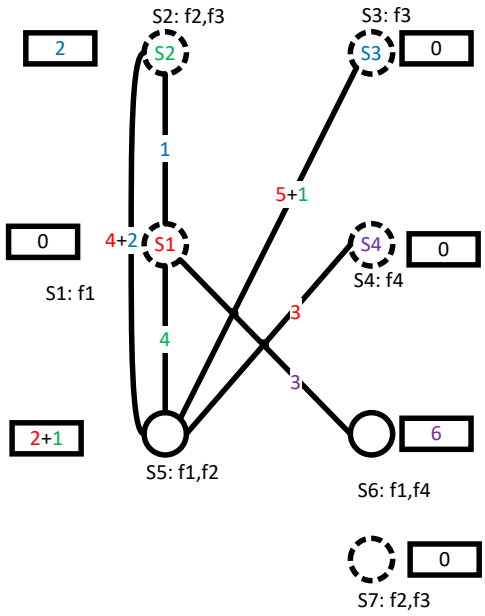


图 4.9 增广的备份资源图

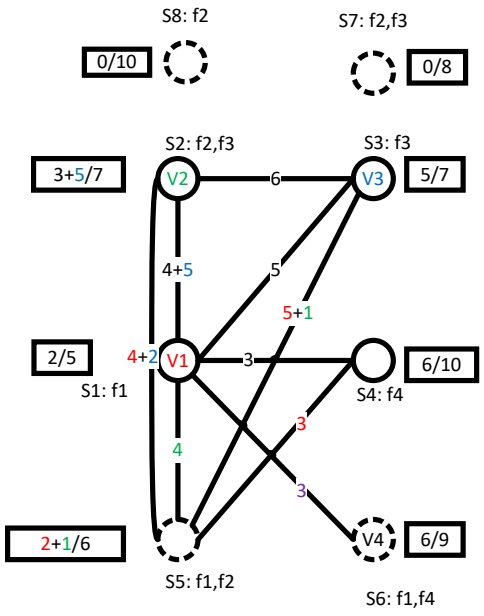


图 4.10 节点s_4失效实例

最后，迭代的重复图分解过程和多背包问题求解过程，处理每个嵌入虚拟节点的物理节点失效情况，每个物理节点失效顺序将影响备份资源的增广过程，这是一个网络的动态优化问题，文献[77]综述了网络动态规划常用的方法，本章我们依物理节点的标号顺序来依次处理资源增广的过程，重新构造增强图，以保证虚拟网络的生存请求。如图4.10 所示，连续使用不同颜色标记SVNE 的部分增广资源并且连续显示每个节点故障。值得注意的是，部分增广资源的每一步都是基于前一步的资源增强。在本节中，我们描述了SVNE问题完整的算法步骤，如伪码4.2.6 所示。对SVNE请求问题分析，该问题是优化领域中的二次规划问题，

算法 4.2 星型图分解动态规划节点嵌入算法

Input: $G(V,E)$：虚拟网络请求; $G(S,L)$：物理网络。
Output: 生成SVNE并增加资源将SVNE嵌入到底层物理网络中。

1: 将虚拟网络G^V 嵌入[70]到底层网络G^S 中
2: 从与此VN嵌入请求对应的SN中提取嵌入的虚拟网络eVN$G\left(\hat{S},\hat{L}\right)$
3: **for all** v_i 并且$v_i \in G\left(\hat{S},\hat{L}\right)$ **do**
4: 从图$G\left(\hat{S},\hat{L}\right)$中分解eVN $G\left(\hat{S},\hat{L}\right)$ 成两个星型结构集$VS(v_i)$ 和$PS(s_j)$
5: 构建items根据星型结构集$VS(v_i)$.
6: 构建knapsacks根据星型结构集$PS(s_j)$
7: 构建边权值矩阵4.15.
8: 根据节4.2.4所描述的动态规划方法解决多背包问题
9: 添加新节点，连接新的边，重新分配节点计算资源和链路的带宽资源到$G\left(\hat{S},\hat{L}\right)$中构建新的$G\left(\hat{S},\hat{L}\right)$
10: **end for**
11: 从$G\left(\hat{S},\hat{L}\right)$中嵌入增广资源到物理网络$G(S,L)$

而SVNE问题解的实质是一系列排列矩阵，决定了每个节点失效后的迁移重分配

过程。虚拟网络$|V|$节点全部都依次失效，每个节点的迭代时间复杂度为$O[|V|*|S|*\prod_{i=1}^{|S|}c^i]$。因此，其总体计算复杂度为$O[|V|*|S|*|V|*\prod_{i=1}^{|S|}c^i]$，其中$|V|$为失效节点数。

4.3 实验配置与评价指标

本节首先描述了物理网络和虚拟网络的配置，然后给出了各种可生存性评价指标。

4.3.1 实验配置

对于任何VN请求，VN节点的数目是由3到10之间的均匀分布随机得到的，每一对虚拟节点都是以概率0.5 随机连接的。VN节点的计算需求从1到5 之间随机分布，并且VN 上的带宽从1 到10之间随机分布。

VN请求的到达服从泊松过程(平均每1时间单位请求15次)。请求的持续时间服从指数分布，平均100个时间单位，高的请求率和较长的租用时间保证了物理基础设施的高利用率。

根据文献[78]，即$\lambda/\alpha=3$，节点的计算资源与链路的带宽资源的相对成本为3。使用的SN 拓扑是SNDlib拓扑数据[79]如表4.1所示。底层物理节点(链路)的计算(带宽)资源都是整数的。分别分布在10至20(50 至100)之间。为了对设备节点失效场景进行建模，我们选择了底层物理网络中的所有底层物理设备节点逐个失效，并统计了每12个时间单元的迁移频率。

表 4.1 SNDlib拓扑数据

数据集	节点数	链路数
cost266	37	57
geant	22	36
germany50	50	88
giul39	39	172
janos-us-ca	39	122
janos-us	26	84
nobel-eu	28	41
norway	27	51
pioro40	40	89
ta1	24	55
ta2	65	108
zib54	54	81

为了进行性能比较，除了我提出的方案(命名为$STAR$)之外，我还实现了其他

可生存的算法*RVN*[73]如下：

(1) *RVN*[73]：算法将备份节点与关键节点之间、备份节点与备份节点之间的连接起来，而不连接关键节点与关键节点之间，关键节点表示嵌入物理节点中的虚拟节点，备份节点表示的增广节点。

(2) *oVN*：当与虚拟节点映射的底层物理上的每个节点失效时，一种简单的方法是增加一个新的物理节点，用于迁移失败的虚拟节点，并将新的链路与失败的物理节点连接起来。

此外，我们还与虚拟网络嵌入算法[70]进行了比较，后者是虚拟网络请求第一次到达时的嵌入方法，其中VN没有可生存性要求，虚拟网络嵌入算法请求(标记为*bVN*)作为衡量可生存性所消耗的扩展资源量的标准算法，在实验评估图中标记为“algorithm shared”或者“algorithm Noshared”。我们假设当一个虚拟网络嵌入到底层物理网络中时，对于*bVN*算法来说，就需要有成功的可生存性保证。

基于底层物理网络的资源，无论是共享的还是非共享的(专用的)[80]，我们在资源分配模式上独立地实现了这两种情况。每个虚拟网络请求的相应的备份需求相应的底层物理资源可以相互共享，并具有可生存性保证。“专用的”意味着对每个虚拟网络是一个完整的备份网络，备份资源完全用到虚拟网络中，相互独立。但是这是资源效率低下的，因为对于每个获得嵌入式的虚拟资源，都需要一个专用的底层物理实体。在某些情况下，共享和重用备份资源也是可以接受的，以便减少附加备份资源在底层物理网络上的占用。通常，较高程度的重用备份资源会导致可靠性降低，反之亦然。

每个虚拟网络的请求都存在一个生命周期，统计记录虚拟或底层物理板网络在实时情况下的性能指标。由于每个算法的性能度量的实验数据波动，我们将每个虚拟或底层网络性能度量的实验数据逐个记录下来，然后通过成功接受的虚拟网络嵌入或可生存的嵌入式虚拟网络作为分母来对实验数据进行平均。

所有仿真都运行在服务器上，服务器配置为Intel(R) Xeon(R) CPU E5-2620 2.00GHz (24 Cores) 和32.00GB RAM。

4.3.2 评价指标

本节介绍了启发式算法在平均接受率、活动节点和其他度量方面的不同度量[21]如下所述。

(1) 接受率：与阻塞概率的概念相反。虚拟网络请求的比率是成功嵌入物理网络的虚拟网络请求数数除以虚拟网络嵌入请求数，表示为A_{eVN}/A_{VN}。为可生存性保证而成功增广和嵌入到物理网络的虚拟网络请求数表示为A_{SVNE}/A_{VN}，当虚拟网络嵌入到底层网络中时该虚拟网络成功可生存性保护的比率被表示为A_{SVNE}/A_{eVN}。

(2) 活动节点：表示底层物理网络开启的节点数和可生存性虚拟网络的节点数，这种度量在节能的VNE算法中特别有用。

(3) 链路或路径长度：链路长度度量虚拟网络中链路的数量，和路径长度度量是两个互连虚拟节点映射到两个底层节点之间的物理链路数量。

(4) 成本、收入和成本/收益：成本：使用的物理网络节点计算能力或链路带宽资源。收益：虚拟网络的节点计算能力或链路带宽需求。成本/收益：这个比率表示虚拟化开销。

A_{VN}表示单位时间t时虚拟网络请求数，嵌入到物理网络中的虚拟网络请求的成功数目表示为A_{eVN}，以及成功增广资源并且嵌入到物理网络中表示为A_{SVNE}。

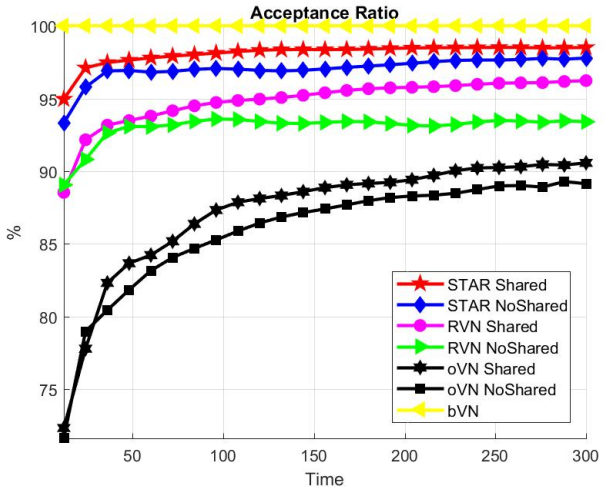


图 4.11 接受率

考虑所有虚拟节点的情况下，评估了SVNE问题启发式方法$STAR$在分配资源时的性能。我们特别关注物理网络使用的资源和虚拟网络请求的资源情况，以及可生存虚拟网络请求的可接受率。

4.3.3 算法性能评估及比较

4.3.3.1 接受率

在这一部分中，如图4.11所示比较了SVNE问题不同算法可生存性虚拟请求的接受率。此外，bVN是所有其它算法前期的虚拟嵌入算法，在可生存性增广嵌入过程中，我们假设算法bVN没有可生存性资源增广而已经达到可生存性功能需求。以bVN作为标准比较算法来度量其它算法的性能，特别是可生存性需求导致的对底层物理网络额外资源消耗量。如图4.11所示，当系统中运行到200单元时间后接受率相对稳定和保持一致，$STAR$ 算法比RVN 算法有更高的接受率，接受率提高了近4%，这是因为$STAR$算法当可生存性需求时资源增广分配过程中比RVN算法消耗更少的底层资源。$STAR$-NoShared 比$STAR$-Shared的接受率低，直观地说，这是因为$STAR$-NoShared为了可生存性的保证而需要更多的资源，备份资源

之间都相互不共享，为了可生存性需求而资源增广都需要申请大量的额外资源，由于冗余资源消耗的原因，接受率至少比*STAR*-Shared 损失了2%。

4.3.3.2 活动节点

为了承载所有虚拟网络而需要使用的逻辑节点数如图4.12所示，*STAR* 算法比其它算法使用更少的虚拟节点数这将使得*STAR* 算法比其它算法消耗更少的底层资源。如图4.13所示，为了承载所有虚拟网络而需要启动和被嵌入虚拟节点的底层物理网络节点数，底层物理网络需要启动节点数与底层物理网络上承接的虚拟网络路径的平均长度有关，因为使用额外的节点来转发端节点之间的通信数据，因此，选择未启动的物理节点来转发数据的概率增加了。在能源效率方面，嵌入所需的底层物理网络节点数可以粗略估计能源消耗。

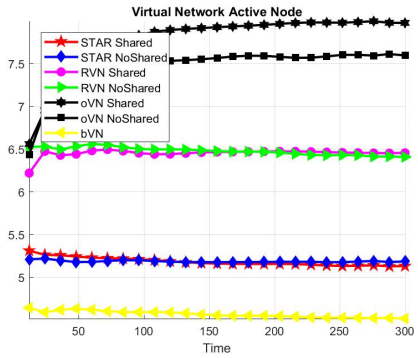


图 4.12 虚拟网络的平均虚拟节点数

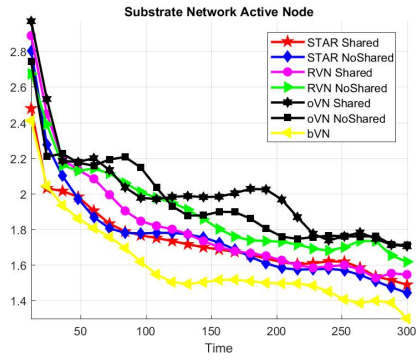


图 4.13 底层物理网络平均启动的节点数

4.3.3.3 链路与路径长度

为了承载所有虚拟网络而需要使用的虚拟链路数如图4.14所示，虚拟网络链路对应的物理路径越长，就需要为嵌入虚拟链路预留更多的资源。由于作为路径的一部分所以每个底层物理网络节点(接收节点除外)都需要一些时间来转发通过该路径发送的包，因此服务质量受路径长度的影响。通常，包延迟随着路径长度的增加而增加。虚拟网络请求需要增广的链路数如图4.14所示，为了承载所有虚拟网络而需要使用的底层物理网络链路数如图4.15 所示。 提出的算法*STAR* 获得

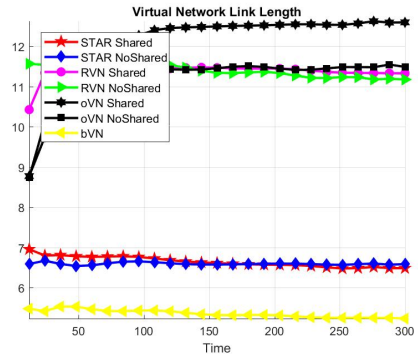


图 4.14 虚拟网络链路数

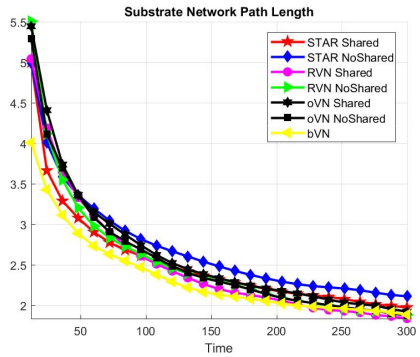


图 4.15 物理网络链路数

的虚拟网络链路数和物理网络链路数与算法*bVN*最接近，越与标注算法*bVN*接近

越体现性这个算法在这个指标上越优，算法*STAR*比其它算法*RVN* 和*oVN* 具有更优的虚拟网络链路数和物理网络链路数。

4.3.3.4 成本、收入和成本/收入比

在本工作中，成本是为满足SEVN可生存性需求而消耗的底层物理资源(即所有底层物理设备节点的节点计算能力、所有光纤链路上的链路带宽)。底层物理网络消耗资源的平均成本如图4.16所示。虚拟请求为了满足可生存性需求会增广一定需要嵌入的虚拟资源，这部分资源相当于虚拟请求的收益，虚拟网络的平均收益如图4.17所示。 提出的算法*STAR* 获得的成本和收益与算法*bVN*最接近，算

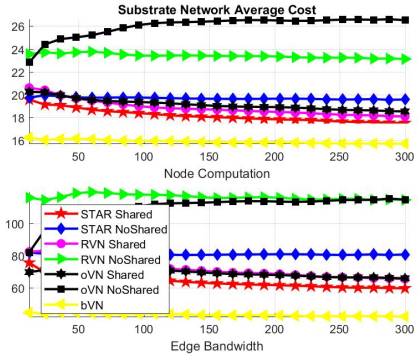


图 4.16 底层物理网络消耗资源的平均成本

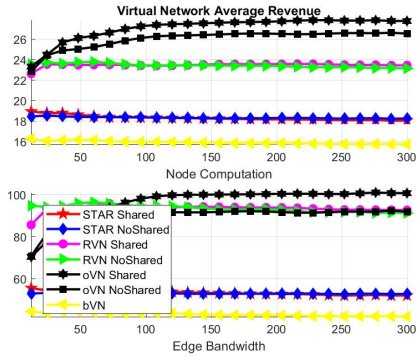


图 4.17 虚拟网络获得需求的平均收益

法*STAR* 比其它算法*RVN*和*oVN* 具有更优的成本和收益。

4.4 小结

本节提出了可生存性虚拟网络嵌入问题的一种有效算法*STAR*，用于解决在具有可生存性保证请求的嵌入虚拟网络中，通过冗余节点和链路来有效地分配SVNE 问题的增广资源(启动新的物理节点、使用链路带宽或使用节点计算)。资源分配方法考虑的是冗余节点是被动的还是主动的，只要在从故障恢复时有充足的资源可用。由于一个物理基础设施(底层网络)承载多个虚拟网络，因此在虚拟网络之间共享冗余节点是更有效的方式。我们引入了一种启发式方法来共享这些独立的故障备用资源，并在星型构造分解和动态规划的基础上实现了资源的近似最优的分配。我们的算法采用了一种新的资源分配方法来减少新的启动主机，减少问题的规模。由于约束条件的规模很大，最直接的线性规划方法是无法计算出最优解。仿真结果表明，提出的算法在实现高的虚拟请求接受率、高资源利用率和低启动节点数量等方面具有显著的效果。

总结和展望

4.5 本文工作总结与贡献

本文针对网络故障的自身特点，对如何在SDN/NFV网络架构中提出时效快，低开销，高可生存性的保护算法，做出了多层次，多角度，多方位的深入研究和分析。本文针对SDN/NFV网络架构中可生存性算法研究做出了以下贡献。

(1) 首先，为了提高主路径传输的可生存性，全面研究了现在网络故障存在的各种故障类型特点，包括主动式和被动式。针对这些故障恢复机制方式，设计了快速的分而治之SRLG不相交路径对算法。

(2) 本文提出了一种拓扑图在存在陷阱问题的情况下求解Min-Min SRLG不相交路由问题的高效算法，为了降低搜索的复杂性，提出了一种分而治之的解决方案，将原Min-Min SRLG 不相交路由问题划分为多个子问题，该子问题基于从AP路径上遇到陷阱问题时导出的SRLG 冲突链路集。提出的算法利用现有的AP搜索结果SRLG冲突链路集，来并行执行实现更快的路径查找。

(3) 本文在一个多核CPU平台上使用合成的拓扑进行了真实的仿真模拟。仿真结果表明，在搜索速度比较下，该算法的查找性能优于其它现有算法。

(4) 分析虚拟网络嵌入问题的本质特征，从而提出适合星型分解动态规划节点嵌入的可生存性虚拟网络嵌入算法，以解决原有可生存性虚拟网络嵌入算法无法解决的时间复杂性和低资源利用率等问题。

(5) 本文在可生存性虚拟网络嵌入问题中，引入网络节点带有特定功能类型的限制条件，提出一种星型分解动态规划节点嵌入的启发式算法，提出虚拟与物理星型图之间协调增广资源增加的权重设置机制，考虑网络资源的利用率和网络节点开启代价。

(6) 提出的算法能快速的实现虚拟网络嵌入的可生存性需求，仿真结果表明，提出的算法与其他现有算法效果比，虚拟嵌入可生存性请求的成功率更高，嵌入的物理资源消耗更低，物理资源的利用率更高。

4.6 未来研究工作与展望

由于自身条件和时间的限制，对网络中可生存性算法的研究工作还不够深入也不够完善。在分析和总结本文工作的同时，未来还可以从如下几个方面继续研究：

(1) 对于SRLG不相交路径问题，可以考虑路径上的附属限制条件不单单只是一个累加性属性的权重值，可以同时考虑多限制条件，并且限制条件是多属性的[81]，对不相交条件的限制不再是单单的共享风险链路组不相交，可以同时考虑点或者边不相交的混合情形。

(2) 本文提出的Min-Min SRLG不相交路径对算法只考虑的是单备份路径，算法可以通过迭代法和延伸SRLG冲突链路集的概念拓展到求多条Min-Min SRLG不相交路径的算法。

(3) SDN/NFV 网络架构的集中式控制器模式，能易于实现过必经点的路由算法，则未来可以考虑路径对有必过节点或者链路的限制条件。

(4) 对可生存性虚拟网络嵌入问题研究中，未来可以考虑多个物理节点发生故障的情形，研究多个故障点发生的概率分布和影响特征，同样，我提出的算法能容易的实现多故障点的情形。未来可以考虑引入节点和链路更多的需求约束条件。

(5) 本文提出的星型分解动态规划节点嵌入的可生存性算法，没有考虑对虚拟链路映射的优化问题，可以在本文的算法的基础上做进一步改进，提出协调节点映射和链路映射的算法，来协调节点和链路映射两部分的资源增广和嵌入。

参考文献

[1] Nick McKeown, Tom Anderson, Hari Balakrishnan, et al. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2):69–74

[2] Gero Dittmann, Andreas Herkersdorf. Network processor load balancing for high-speed links. In: Proc of Proceedings of the 2002 International Symposium on Performance Evaluation of Computer and Telecommunication Systems, volume 735. July, 2002

[3] Christian Doerr, Fernando A Kuipers. All quiet on the internet front? IEEE Communications Magazine, 2014, 52(10):46–51

[4] Fernando A Kuipers. An overview of algorithms for network survivability. ISRN Communications and Networking, 2012, 2012

[5] JW Suurballe. Disjoint paths in a network. Networks, 1974, 4(2):125–145

[6] John W Suurballe, Robert Endre Tarjan. A quick method for finding shortest pairs of disjoint paths. Networks, 1984, 14(2):325–336

[7] D Anthony Dunn, Wayne D Grover, Mike H MacGregor. Comparison of k-shortest paths and maximum flow routing for network facility restoration. IEEE Journal on selected areas in Communications, 1994, 12(1):88–99

[8] Longkun Guo, Hong Shen. On finding min-min disjoint paths. Algorithmica, 2013, 66(3):641–653

[9] Justine Sherry, Shaddi Hasan, Colin Scott, et al. Making middleboxes someone else's problem: network processing as a cloud service. ACM SIGCOMM Computer Communication Review, 2012, 42(4):13–24

[10] Zhaoguang Wang, Zhiyun Qian, Qiang Xu, et al. An untold story of middleboxes in cellular networks. In: Proc of ACM SIGCOMM Computer Communication Review, volume 41. ACM, 2011, 374–385

[11] Michael Walfish, Jeremy Stribling, Maxwell N Krohn, et al. Middleboxes No Longer Considered Harmful.. In: Proc of OSDI, volume 4. 2004, 15–15

[12] Hassan Hawilo, Abdallah Shami, Maysam Mirahmadi, et al. NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC). IEEE Network, 2014, 28(6):18–26

[13] Gregor Schaffrath, Christoph Werle, Panagiotis Papadimitriou, et al. Network virtualization architecture: Proposal and initial prototype. In: Proc of Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures. ACM, 2009, 63–72

[14] NM Mosharaf Kabir Chowdhury, Raouf Boutaba. A survey of network virtualization. Computer Networks, 2010, 54(5):862–876

[15] NM Mosharaf Kabir Chowdhury, Raouf Boutaba. Network virtualization: state of the art and research challenges. IEEE Communications magazine, 2009, 47(7)

[16] Margaret Chiosi, Don Clarke, Peter Willis, et al. Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action. In: Proc of SDN and OpenFlow World Congress. 2012, 22–24

[17] Frank Yue. Network functions virtualization-everything old is new again. F5 Neworks, 2013.

[18] Antonio Manzalini, Roberto Saracco, Cagatay Buyukkoc, et al. Software-defined networks for future networks and services. In: Proc of White Paper based on the IEEE Workshop SDN4FNS. 2014

[19] Soheil Hassas Yeganeh, Amin Tootoonchian, Yashar Ganjali. On scalability of software-defined networking. IEEE Communications Magazine, 2013, 51(2):136–141

[20] Xiaohu Ge, Hui Cheng, Mohsen Guizani, et al. 5G wireless backhaul networks: challenges and research advances. IEEE Network, 2014, 28(6):6–11

[21] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, et al. Virtual network embedding: A survey. IEEE Communications Surveys & Tutorials, 2013, 15(4):1888–1906

[22] Sandra Herker, Ashiq Khan, Xueli An. Survey on survivable virtual network embedding problem and solutions. In: Proc of International Conference on Networking and Services, ICNS. 2013

[23] Mohammad Javad Rostami, Azadeh Alsadat Emrani Zarandi, Seyed Mohamad Hoseininasab. MSDP with ACO: A maximal SRLG disjoint routing algorithm based on ant colony optimization. Journal of Network and Computer Applications, 2012, 35(1):394–402

[24] Ian F Akyildiz, Ahyoung Lee, Pu Wang, et al. A roadmap for traffic engineering in SDN-OpenFlow networks. Computer Networks, 2014, 71:1–30

[25] Amund Kvalbein, Audun Fosselie Hansen, Stein Gjessing, et al. Fast IP network recovery using multiple routing configurations. In: Proc of in INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings. Citeseer, 2006

[26] Ning Qi, Bin-Qiang Wang, Zhi-Ming Wang. Research on reconfigurable service carrying network resilient construction algorithms. Dianzi Yu Xinxi Xuebao(Journal of Electronics and Information Technology), 2012, 34(2):468–473

[27] Mike Shand, Stewart Bryant. IP fast reroute framework. 2010.

[28] Lester Randolph Ford Jr, Delbert Ray Fulkerson. Flows in networks. Princeton university press, 2015

[29] Mohammad Javad Rostami, Siavash Khorsandi, Ali Asghar Khodaparast. CoSE: A SRLG-disjoint routing algorithm. In: Proc of Universal Multiservice Networks, 2007. ECUMN'07. Fourth European Conference on. IEEE, 2007, 86–92

[30] Ajay Todimala, Byrav Ramamurthy. IMSH: An iterative heuristic for SRLG diverse routing in WDM mesh networks. In: Proc of Computer Communications and Networks. Proceedings. 13th International Conference on. IEEE, 2004, 199–204

[31] Dahai Xu, Yizhi Xiong, Chunming Qiao, et al. Trap avoidance and protection schemes in networks with shared risk link groups. Journal of Lightwave Technology, 2003, 21(11):2683

[32] Teresa Gomes, Luís Fernandes. Obtaining a SRLG-disjoint path pair of min-sum cost. In: Proc of International Congress on Ultra Modern Telecommunications and Control Systems. 2010

[33] Hongbin Luo, Lemin Li, Hongfang Yu. Insights for segment protection in survivable WDM mesh networks with SRLG constraints. Photonic Network Communications, 2007, 14(3):361–368

[34] Eiji Oki, Nobuaki Matsuura, Kohei Shiomoto, et al. A disjoint path selection scheme with shared risk link groups in GMPLS networks. IEEE Communications letters, 2002, 6(9):406–408

[35] Xiaoshan Pan, Gaoxi Xiao. Heuristics for diverse routing in wavelength-routed networks with shared risk link groups. Photonic Network Communications, 2006, 11(1):29–38

[36] Sheng Wang, Lemin Li. Impairment aware optimal diverse routing for survivable optical networks. Photonic Network Communications, 2007, 13(2):139–154

[37] Xiaofei Cheng, Xu Shao, Yixin Wang. Multiple link failure recovery in survivable optical networks. Photonic Network Communications, 2007, 14(2):159–164

[38] José Silva, Teresa Gomes, Luís Fernandes, et al. An heuristic for maximally SRLG-disjoint path pairs calculation. In: Proc of Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2011 3rd International Congress on. IEEE, 2011, 1–8

[39] Jian Qiang Hu. Diverse routing in optical mesh networks. IEEE Transactions on Communications, 2003, 51(3):489–494

[40] Shimon Even, Alon Itai, Adi Shamir. On the complexity of time table and multi-commodity flow problems. In: Proc of Foundations of Computer Science, 1975., 16th Annual Symposium on. IEEE, 1975, 184–193

[41] Hongfang Yu, Vishal Anand, Chunming Qiao, et al. Cost efficient design of survivable virtual infrastructure to recover from facility node failures. In: Proc of Communications (ICC), 2011 IEEE International Conference on. IEEE, 2011, 1–6

[42] Muntasir Raihan Rahman, Issam Aib, Raouf Boutaba. Survivable virtual network embedding. In: Proc of International Conference on Research in Networking. Springer, 2010, 40–52

[43] Zhiming Wang, Jiangxing Wu, Yu Wang, et al. Survivable virtual network mapping using optimal backup topology in virtualized SDN. China communications, 2014, 11(2):26–37

[44] Qian Hu, Yang Wang, Xiaojun Cao. Location-constrained survivable network virtualization. In: Proc of Sarnoff Symposium (SARNOFF), 2012 35th IEEE. IEEE, 2012, 1–5

[45] LU Bo, Tao Huang, Xiao-chuan SUN, et al. Dynamic recovery for survivable virtual network embedding. The Journal of China Universities of Posts and Telecommunications, 2014, 21(3):77–84

[46] Zhu Qiang, WangHui Qiang, FengGuang Sheng, et al. Heuristic survivable virtual network embedding based on node migration and link remapping. In: Proc of Information Technology and Artificial Intelligence Conference (ITAIC), 2014 IEEE 7th Joint International. IEEE, 2014, 181–185

[47] Minlan Yu, Yung Yi, Jennifer Rexford, et al. Rethinking virtual network embedding: substrate support for path splitting and migration. ACM SIGCOMM Computer Communication Review, 2008, 38(2):17–29

[48] Tao Heng. my paper complete version. https://github.com/hengfanz/franztao.File.io/tree/master/hnuthesis_master_v4.0

[49] Mohamed Al-Kuwaiti, Nicholas Kyriakopoulos, Sayed Hussein. A comparative analysis of network dependability, fault-tolerance, reliability, security, and survivability. IEEE Communications Surveys & Tutorials, 2009, 11(2)

[50] 卢兴华, 刘增良, 林憬锵. 信息系统生存性评估方法研究. 微计算机信息, 2006, 22(3):39–41

[51] 林雪纲, 许榕生. 信息系统生存性分析模型研究. 通信学报, 2006, 27(2):153–159

[52] 林闯, 汪洋, 李泉林. 网络安全的随机模型方法与评价技术. 计算机学报, 2005, 28(12):1943–1956

[53] Panagiotis Sebos, Jennifer Yates, Gisli Hjalmtysson, et al. Auto-discovery of shared risk link groups. In: Proc of Optical Fiber Communication Conference and Exhibit, 2001. OFC 2001, volume 3. IEEE, 2001, WDD3–WDD3

[54] 王秀君. 网络中可靠路由算法的研究: [Thesis]. 山东师范大学, 2008

[55] Piet Van Mieghem, Huijuan Wang, Xin Ge, et al. Influence of assortativity and degree-preserving rewiring on the spectra of networks. The European Physical Journal B, 2010, 76(4):643–652

[56] Damon Mosk-Aoyama. Maximum algebraic connectivity augmentation is NP-hard. Operations Research Letters, 2008, 36(6):677–679

[57] Mihalis Yannakakis. Computing the minimum fill-in is NP-complete. SIAM Journal on Algebraic Discrete Methods, 1981, 2(1):77–79

[58] Ping Pan, George Swallow, Alia Atlas. Fast reroute extensions to RSVP-TE for LSP tunnels. Technical report, 2005

[59] Ramesh Bhandari. Optimal diverse routing in telecommunication fiber networks. In: Proc of INFOCOM'94. Networking for Global Communications., 13th Proceedings IEEE. IEEE, 1994, 1498–1508

[60] David Coudert, Pallab Datta, Stéphane Pérennes, et al. Shared risk resource group complexity and approximability issues. Parallel Processing Letters, 2007, 17(02):169–184

[61] Randeep Bhatia, Murali Kodialam, TV Lakshman. Finding disjoint paths with related path costs. Journal of Combinatorial Optimization, 2006, 12(1-2):83–96

[62] Guangzhi Li, Bob Doverspike, Chuck Kalmanek. Fiber span failure protection in mesh optical networks. Optical Networks Magazine, 2002, 3(3):21–31

[63] David Eppstein. Finding the k shortest paths. SIAM Journal on computing, 1998, 28(2):652–673

[64] P Laborczi, J Tapolcai, Pin-Han Ho, et al. Solving asymmetrically weighted optimal or near-optimal disjoint path pair for the survivable optical networks. In: Proc of Third International Workshop On Design Of Reliable Communication Networks (DRCN' 01). 2001

[65] Pallab Datta, Arun K Somani. Graph transformation approaches for diverse routing in shared risk resource group (SRRG) failures. Computer Networks, 2008, 52(12):2381–2394

[66] Vasek Chvatal. A greedy heuristic for the set-covering problem. Mathematics of operations research, 1979, 4(3):233–235

[67] Gurobi Optimization, et al. Gurobi optimizer reference manual. URL: http://www.gurobi. com, 2012, 2:1–3

[68] Ananth Grama. Introduction to parallel computing. Pearson Education, 2003

[69] Gene M Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In: Proc of Proceedings of the April 18-20, 1967, spring joint computer conference. ACM, 1967, 483–485

[70] Wenzhi Liu, Yang Xiang, Shaowu Ma, et al. Completing virtual network embedding all in one mathematical programming. In: Proc of Electronics, Communications and Control (ICECC), 2011 International Conference on. IEEE, 2011, 183–185

[71] Edoardo Amaldi, Stefano Coniglio, Arie MCA Koster, et al. On the computational complexity of the virtual network embedding problem. Electronic Notes in Discrete Mathematics, 2016, 52:213–220

[72] Jens Lischka, Holger Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. In: Proc of Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures. ACM, 2009, 81–88

[73] Wai-Leong Yeow, Cédric Westphal, Ulaş Kozat. Designing and embedding reliable virtual infrastructures. In: Proc of Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures. ACM, 2010, 33–40

[74] Richard M Karp. On the computational complexity of combinatorial problems. Networks, 1975, 5(1):45–68

[75] Derick Wood, Derrick Wood. Theory of computation. 1987.

[76] S Skiena. Dijkstra’s algorithm. Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley, 1990. 225–227

[77] 林闯, 万剑雄, 向旭东, et al. 计算机系统与计算机网络中的动态优化: 模型, 求解与应用: [Dissertation]. 中国计算机学会— 中国科学院计算技术研究所, 2012

[78] Hongfang Yu, Chunming Qiao, Vishal Anand, et al. Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures. In: Proc of Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE. IEEE, 2010, 1–6

[79] Sebastian Orlowski, Roland Wessäly, Michal Pióro, et al. SNDlib 1.0 —Survivable network design library. Networks, 2010, 55(3):276–286

[80] Jing Lu, Jonathan Turner. Efficient mapping of virtual networks onto a shared substrate. 2006.

[81] 林闯, 李寅, 万剑雄, et al. 计算机网络服务质量优化方法研究综述: [Dissertation]. 2011

致　谢

转瞬间，三年的研究生学习生涯落下帷幕，在这个三年的苦读学习中我得到了来自我的导师，我的朋友，我的同学们的无私的关怀和无私的帮助，让我在求学的道路上不断前行。值此论文完成之时，谨向帮助和关心过我的人致以最诚挚的谢意！

首先，我向我的恩师谢鲲教授表示我最深的敬意和感谢！非常感谢她在我攻读硕士学位的三年时间里，对我学习上给予的充分全面的指导，让我在个人能力上和学术能力都提升了一个台阶。谢老师宽厚待人的品格，高尚的师德，都作为一种深人格魅力，深深地影响着我对待人生对待学术的态度，这一切必将会让我终身受益，我在此向谢老师表示深深的谢意！谢老师用她渊博的知识对我们进行全面而专业的指导，她对待学术执着，细心，一丝不苟的精神对我形成一种教诫，让我督促自己也要像她那样对待学术，要通过不断地学习来提升自我能力。祝愿谢老师及家人平安，快乐，健康！

感谢基地实验室已经毕业的各位师姐师兄在我研究生一二年级时对我生活上和学习上的指导与帮助。特别要感谢宁雪萍师姐、黄俊师兄、施文师兄、郑哲师兄、陈宇翔师兄、王乐乐师兄和陈振华师兄在我学术刚入门时，对我的热情的帮助，无论是在学科的学习上还是论文的写作，他们的提点总会让我受益匪浅，让我从最初的迷茫期很快地过度到了前进期，非常感谢他们的指导和帮助。感谢彭灿、潘海娜、赵彦彦和李晓灿，我们一起探讨，学习上相互指导，生活上相互帮助，希望我们能够同门之情长存！感谢我的好朋友们胡海洋、何展、黎孟、侯宇凡、秦光睿和王思娟，跟你们一起相处，给我枯燥和压力的硕士学习过程增添了不少的欢笑，愿我们友谊长久！

深深地感谢我的爱人和我的父母，在我漫漫的求学路程中不断的激励我默默地支持我，生活中给我无微不至的关心，你们一直是我不断前进的主要动力。我会以自己的实际成绩向你们感恩和回馈！

感谢信息科学与工程学院的答辩委员会和评审论文的各位教师，感谢您们在百忙之中拿出宝贵的时间对我的论文进行建设性的指导和提出关键性的意见！

最后，我想感谢湖南大学、感谢我的学校。惟楚有才、于斯为盛，我有幸能够在千年学府里不断追溯着先贤的脚步和思想，在这里度过人生中珍贵的三年光阴。难忘的是那一次次呼朋引伴结伴而行的岳麓山之行、那一场场湘江边盛大的美丽烟火，爱晚亭的枫、湘江的水还有那麓山南路的风，将永远吹拂我的这颗恒心。

附录A　发表论文和参加科研情况说明

（一）发表的学术论文

[1] Kun Xie, Heng Tao, Xin Wang, Gaogang Xie, Jigang Wen, Jiannong Cao, Zheng Qin. Divide And Conquer For Fast SRLG Disjoint Routing[C]. DSN 2018: International Conference on Dependable Systems and Networks, Luxerbourg(CCF B).

（二）申请及已获得的专利

[1] 陶恒，谢鲲. 一种求完全风险共享链路组分离路径对的方法及系统：中国。已具有国家知识产权局公开号，并已进入实审阶段。