摘　要

传统 Internet 仅提供“尽力而为”的数据报发送服务，面对网络上日益增长的多媒体应用，现有路由机制已经逐渐不能满足新的需求。如何实现路由协议的扩展，使其提供有效的服务质量路由（QoSR），是现代网络必须考虑和值得研究的问题。

本文研究了开放式最短路径优先（OSPF）协议工作机制，实现了基于遗传-蚁群融合算法的 OSPF 协议上的 QoS 扩展。

论文首先分析了 QoS 路由机制研究现状，详细讨论了现有各种 QoSR 算法及其存在的问题，将遗传-蚁群融合算法应用于解决多约束 QoSR。该算法以基本遗传算法和蚁群算法为基础，克服各自缺陷，通过二者的“融合”——即以遗传算法所得优化解初始化蚁群算法的信息素值，循环迭代，从而求得多约束 QoSR 问题的最优解。

为了实现 OSPF 协议上的 QoSR 扩展，论文还详细探讨了 OSPF 协议的工作过程及其使用的路由算法。作为一种典型的链路状态协议，OSPF 基于 Dijkstra 算法，但是该算法要求以某一固定的链路状态信息来计算，这就使得当前的 OSPF 协议不支持多约束 QoSR 机制，本文的任务就是实现 OSPF-QoSR。

论文提出了 OSPF-QoSR 的具体实施方案，其基本思路是在对当前 OSPF 协议报文格式和工作机制做最小改动的前提下，最大程度地支持多约束 QoSR，实现基于遗传-蚁群融合算法的 OSPF-QoSR。本文路由算法是控制在一个自治域（AS）范围内的 OSPF 网络中，使用分布式路由策略，采用预先计算的方式，扩展 OSPF 报文格式使其包含网络资源信息，改进 LSA 发送机制，利用融合算法进行最优路径选择。

论文最后利用网络仿真软件 OPNET 构造了一个支持 QoS 的 OSPF 网络，模拟仿真实现本文所提出的基于融合算法的 OSPF-QoSR 机制，并将其在某些网络性能上与 RFC2676 所推荐的扩展 Bellman-Ford 算法进行比较，说明本文算法是可行的、有一定优越性的，为今后大型 OSPF 网络中多约束 QoSR 机制的研究提供了新的思路，并指出了下一步研究的工作方向和重点。

关键词：服务质量路由（QoSR），开放式最短路径优先（OSPF），遗传-蚁群融合算法，OSPF-QoSR，OPNET 仿真

Abstract

Traditional Internet only provides "Best-Effort" services for data transmission. Facing to the increasingly growing up of multimedia applications, existing routing mechanisms gradually can't satisfy new demands. How to extend routing protocols in order to provide effective QoSR is becoming a considerable and investigable problem for modern Internet.

This dissertation studies the working mechanisms of OSPF Protocol and implements the QoS routing extensions based on Genetic-Ant Combination Algorithm of OSPF.

Firstly, the dissertation gives a survey about the current researching situation of QoS routing mechanisms, bats around the existing QoSR algorithms, discusses the main problems of them and applies the Genetic-Ant Combination Algorithm to solve multi-constrained QoSR. The Combination Algorithm is based on basic Genetic Algorithm and Ant Algorithm, overcomes respective disfigurement of them. Synoptically, it initializes the pheromone value of Ant Algorithm through the optimization results of Genetic Algorithm and accordingly figures out the multi-constrained QoSR problems by iterative loop to get the best result.

In order to achieve the QoSR extension of OSPF, the dissertation detailedly probe into the working process and routing algorithms of OSPF. As a typical link-state protocol, OSPF is based on Dijkstra Algorithm, which requests a certain fixed link state to compute the shortest path. This brings the result that the current OSPF Protocol can't support multi-constrained QoSR mechanism. And the mission of this dissertation is to implement OSPF-QoSR.

The dissertation puts forward the idiographic implementary scheme of OSPF-QoSR. It furthest implements multi-constrained QoSR on the basis of Genetic-Ant Combination Algorithm by improving the current OSPF message format and work mechanism with minimal modification. This routing algorithm is operating in an Autonomous System, makes use of distributing routing strategy, adopts pre-computation method, extends OSPF message format to append network resource information inclusively, ameliorates LSA transmission mechanism and utilizes combination algorithm to choose the best routing path.

Lastly, an OSPF network supporting QoS is constructed by OPNET Modeler to simulate the OSPF-QoSR mechanism based on the Combination Algorithm which is presented in the dissertation. The Genetic-Ant Combination Algorithm is compared in some network capabilities with the Extended Bellman-Ford Algorithm which is recommended by RFC2676. The simulation results show that Combination Algorithm is feasible and superior in contrast with the RFC-recommended algorithm. All these offer some new thoughts of

multi-constrained QoSR research in large-scale OSPF networks and indicate further research orientations and emphases.

Key words: QoSR, OSPF, Genetic-Ant Combination Algorithm, OSPF-QoSR, OPNET Simulation

论文独创性声明

本人声明：本人所呈交的学位论文是在导师的指导下，独立进行研究工作所取得的成果。除论文中已经注明引用的内容外，对论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。本论文中不包含任何未加明确注明的其他个人或集体已经公开发表的成果。

本声明的法律责任由本人承担。

论文作者签名：王凤琳　　2008 年 5 月 30 日

论文知识产权权属声明

本人在导师指导下所完成的论文及相关的职务作品，知识产权归属学校。学校享有以任何方式发表、复制、公开阅览、借阅以及申请专利等权利。本人离校后发表或使用学位论文或与该论文直接相关的学术论文或成果时，署名单位仍然为长安大学。

（保密的论文在解密后应遵守此规定）

论文作者签名：王凤琳　　2008 年 5 月 30 日

导 师 签 名：　　2008 年 5 月 30 日

第一章 绪论

1.1 课题研究的背景和意义

伴随多媒体技术的飞速发展，Internet 上多媒体应用层出不穷，多媒体信息的数量与日俱增。Internet 已逐步由单一的数据传送网向数据、语音、图像等多媒体信息的综合传输网演化，对服务质量（Quality of Service，QoS）提出了更高的要求。QoS 是网络在传输业务流时，业务流对网络服务需求的集合，其中业务流是指与特定 QoS 相关的从源到目的地的分组流[1]。也就是说，QoS 是应用业务对网络传输服务提出的一组可度量的要求，主要包括带宽、端到端延迟、分组丢失率、花费等。网络在传输相应的数据业务时，必须满足这组要求。然而，当前 Internet 只能提供“尽力而为”的发送服务，即网络层不区分用户业务的种类，将网络资源公平地提供给各类业务，以最大速度尽力转发数据分组作为单一目标，在分组丢失概率、延迟等方面公平地对待各类业务。这种尽力发送的机制使网络层无法提供网络传输参数的保证，但是带宽、端到端的延迟、丢失率等参数对于多媒体类应用业务往往是至关重要的。这就要求网络能够区别对待各种业务，并提供不同的服务质量。于是，保证 QoS 的路由机制应运而生，网络路由协议也被逐渐改进和完善以期支持 QoS 要求。

例如当前应用最为广泛的路由协议——开放式最短路径优先 OSPF（Open Shortest Path First）协议，它是 IETF（Internet Engineering Task Force，Internet 工程任务组）为 IP 网络开发的 IGP（Interior Gateway Protocal，内部网关协议）。作为一种典型的单播链路状态（Link-State）路由协议，OSPF 是基于 SPF（Shortest Path First，最短路径优先）算法即 Dijkstra 算法的。OSPF 协议允许管理员给每一条路由指派一个代价，即吞吐量、延时等度量，然后利用 Dijkstra 算法计算“最短路由”——具有最小延时、最大吞吐量或最少跳数的路径。该算法是基于一个固定的链路状态信息进行计算的，可是用户业务流提出的 QoS 要求往往需要在多个约束目标下选择优化路径，这样就使得 OSPF 协议无法支持多约束 QoS 路由选择。为了实现 OSPF 协议的 QoSR 功能，RFC2676 定义了 OSPF 的扩展并对最短路径算法进行了推荐，建议采用扩展的 BF（Bellman-Ford）算法进行路径计算。

本课题针对多约束单播 QoS 路由选择问题，在 OSPF 协议上进行 QoS 扩展，并深入探讨了基于遗传算法和蚁群算法的遗传-蚁群融合算法，将其应用于解决 QoSR 这一

NP 完全问题，为实际应用中大型网络的 OSPF-QoSR 提供了新思路。

1.2 QoS 路由机制及 OSPF 协议研究现状

1.2.1 QoS 路由研究现状

服务质量路由（QoS Routing，简称 QoSR）是一种基于数据流 QoS 请求和网络可用资源进行路由的机制[1]。或者说，QoSR 是一种动态路由协议，并且在其路径选择标准里可能包含可用带宽、资源消费量、延迟、跳数及抖动等 QoS 参数[2]。由于传统的路由机制在选路时通常只使用单一优化目标（如跳数或花费）、基于“最短路径”进行，即使从源节点到目的节点之间存在“更好路径”，只要不是“最短路径”也不会被使用，这样就会导致某些链路空闲的情况下而在另外的一些链路上造成拥塞。QoSR 就是将传统的最短路径变为一条更好的路径，其主要目标是[1]：(1)为每一个接纳的 QoS 业务连接请求，找到满足其 QoS 要求的可行路径；(2)优化全局资源利用率，平衡网络负载，从而最大化网络接受其他 QoS 请求的能力。因此，QoSR 应包括两方面的内容：(1)测量、收集并维护网络状态信息，即与网络当前状态有关的各种信息；(2)根据维护的网络状态信息计算优化的可行路径。第一个内容涉及到网络状态的获取和状态信息传播的问题；后一方面主要是 QoSR 算法，而各种算法往往需要依据节点所维护的状态信息。

网络状态信息分为本地状态、全局状态和聚集状态三类，以周期方式、触发方式或二者相结合的方式在网络中进行状态传播。当每个节点收集到适当的网络状态信息后，需要据此采用一种相应的路由策略实现路由，路由策略有源路由、分布式路由和层次化路由三种。基于节点获得和维护的网络状态信息，选取适当的路由策略，进行 QoSR。按照所求解的问题类型和求解方法，目前常用的单播 QoSR 算法有以下几类：

1. 多项式非启发类

多项式非启发类算法针对的是多项式可解问题，其求解思路是 QoSR 算法的基础。Wang 和 Crowcroft 使用 Dijkstra 最短路径树算法，实现了带宽延迟受限的源路由求解[3]。对于分布式方法，他们对最短最宽路径提出了预计算方式，每个节点通过链路状态协议维护全局状态，并使用扩展 Bellman-Ford 算法或 Dijkstra 算法为每个可能的连接目的地预先计算出下一跳，这样在连接请求时只需要简单的查找路由表就可以了。

2. 伪多项式非启发类

多重受限的 QoSR 问题通常是 NPC 问题[4]，但可能具有伪多项式时间的求解算法，其复杂度依赖于链路权值的组合[5]。如扩展 Bellman-Ford 算法（EBFA）就是一种典型

的用于求解多重受限 QoSR 问题的伪多项式非启发类算法[6]。EBFA 的核心思想是在偏序集中寻找最小元。

3. 限定 QoS 度量类

如果将除了一类之外的所有 QoS 度量都限定在一个有限集上，则 QoSR 问题是多项式可解的[7,8]。基于这种思想，Yuan 和 Liu 在 EBFA 的基础上设计了粒度受限（Limited Granularity）的启发式算法[9]，通过限制最优路径的数目达到减小复杂度的目的。Chen-Nahrstedt 设计了多重路径受限的源路由算法[7]。算法的思想是将 QoS 度量的取值范围 R 或 Z 映射到一个有限集合C，这样保证多项式可解。Orda 等设计了通过量化花费函数的预计算算法[10]，该算法针对多重可加性限制的路径优化问题，基于通过 Bellman-Ford 算法，减小计算复杂度。

4. 路径子空间搜索类

路径子空间搜索是指通过启发式算法，直接限制拓扑图中的某些路径或者路径数量，从而在路径集合中的子空间搜索可行路径的方法。Yuan 和 Liu 设计了路径受限的多重路径受限的分布式算法[9]。Salama 等人对复杂度为 NPC 的延迟受限的最小花费问题（DCLC）设计了分布式启发算法[11]。

5. 花费函数类

为了降低 QoSR 算法的复杂度和提供更好的 QoS 支持，一种可能的方式是通过研究花费函数，从而保证所计算的最小花费路径能够满足 QoS 需求。Korkmaz 等设计了一种在多个可加性受限条件下求解优化路径的通用算法[12]。Ergun 等设计了一条链路基于不同的花费提供不同延迟的启发式算法[13]。Juttner 等基于拉格朗日松弛法（Lagrange Relaxation）设计了 DCLC 的源路由求解算法[14]。

对于目前常用的这几类算法，本文将在第二章详细比对、分析和总结，指出其存在的问题；第三章则提出本课题所使用的多约束 QoSR 算法——遗传-蚁群融合算法，所采用的路由策略是分布式路由。

1.2.2 OSPF 协议研究现状

OSPF 协议是一种典型的单播链路状态（Link-State）路由协议。网络中的路由器不用预先知道网络的每一条路由，通过和邻近的路由器交换链路状态信息，自行计算通向每一个网络的最短路径，这一般发生在一个自治域（AS：Autonomous System）内。在 AS 中，所有的 OSPF 路由器都维护一个相同的描述该 AS 结构的数据库，该数据库通过

链路状态通告（LSA：Link State Advertisement）建立和更新，存放的是自治域中相应链路的状态信息， OSPF 路由器正是通过这个数据库计算出 OSPF 路由表。有关路由表的计算是 OSPF 的核心内容，它是动态生成路由器内核路由表的基础。在 OSPF 路由表条目中，包括有目标地址、目标地址类型、链路路由代价、链路存活时间、链路类型以及下一跳等内容[15,16,17]。作为一种链路状态路由协议，LSA 由运行 OSPF 协议的 AS 内各路由器产生，通过可靠的泛洪（Flooding）机制在所有路由器之间传递。

当前，OSPF 协议使用下面两种方法之一计算链路的路由代价：

⑴OSPF 自动计算使用每个路由接口的代价。OSPF 能自动计算一个接口的耗费，这个算法以每个接口类型支持的带宽为基础。一条路由上所有接口计算值的和（Sum）形成 OSPF 路由决定的基础。基于冗余链路上可获得的带宽，这些值能使 OSPF 计算出最小耗费的路由。

⑵非带宽敏感的缺省值可以用于每一个OSPF接口。如果用户的网络由类似的传输线路组成，那么可以使用缺省值计算路由；但是，在有相当多路由冗余并且使用不同的传输技术的网络中，缺省值将不会选出到任何目的地的最优路由。另外，可以手动地为某个特定接口修改耗费度量，这样使网络管理员在主要仍使用缺省路由耗费的前提下，对网络的流量模式进行合理地规划。

SPF算法是OSPF路由协议的基础，该算法将每一个路由器作为根（Root）来计算其到每一个目的地路由器的距离，每一个路由器根据一个统一的数据库计算出路由域的拓扑结构图，该结构图类似于一棵树，在SPF算法中，被称为最短路径树；最短路径树的树干长度，即OSPF路由器至每一个目的地路由器的距离，称为OSPF的花费（Cost）。不管使用如上的哪种方法计算链路的路由代价，对OSPF节点而言都是不重要的，任何一条路由的代价可以通过把路由上遇到的每个路由器接口代价加起来得到，并在OSPF的最短路径树中记录每一个已知目的地的和代价。

按照 SPF 算法的要求，路由器中路由表依赖于一张能表示整个网络拓扑结构的无向图$G(V,E)$。在图G中，节点V表示路由器，边E表示连接路由器的链路（Link），一般把图G称为 L-S（Link-State，链路状态）图。在信息一致的情况下，所有路由器的 L-S 图都应该是相同的，各路由器的路由表通过 L-S 图计算。L-S 算法包含如下三个步骤：

⑴各路由器主动测试所有与之相邻的路由器的状态，周期性地向邻居路由器发出简短的查询报文，询问邻居路由器当前是否能够访问，如果对方做出反应，说明链路状态

为 UP（激活状态），否则为 DOWN。

⑵各路由器周期性地向所有参与 SPF 的路由器广播其 L-S 信息。

⑶路由器收到 L-S 报文后，利用它刷新网络拓扑图，将相应的连接状态改为 UP 或 DOWN。如果 L-S 发生了变化，路由器立即运用 Dijkstra 前向搜索算法，求出加权无向图中从某给定节点到目的节点的最小花费路由或最佳路由。

基于目前 OSPF 发展应用现状及 RFC2676 有关 OSPF 协议 QoS 扩展的建议，参考文献[18]讨论了扩展 BF 算法在 OSPF-QoSR 中的实现，其思想是首先在链路状态数据库中找出满足 QoS 要求的所有路径，然后利用扩展 BF 算法，在这些路径中计算最短路径；参考文献[19]则在扩展 BF 算法的基础上讨论了利用改进的 Dijkstra 算法实现路由的按需计算。

本课题是在 OSPF 现有工作原理和 RFC2676 的基础上，利用遗传-蚁群融合算法，预期实现 OSPF-QoSR 的预先计算。本文将在第四章详细介绍 OSPF 的工作过程，第五章具体讨论基于融合算法、支持 QoS 的 OSPF 协议扩展的实现方案。

1.3 课题研究内容

1.3.1 研究内容

1. 分析当前具有代表性的几种单播 QoS 算法，比较各种算法的优越性及存在问题，提出将遗传-蚁群融合算法用于解决多约束单播 QoSR 问题。

2. 通过对目前应用范围最为广泛的 OSPF 路由协议工作原理的研究，针对现有 OSPF 协议不支持多约束 QoSR 这一问题，阐明对 OSPF 进行 QoS 扩展的必要性和可行性。

3. 研究 OSPF 协议上的 QoS 扩展实现方法，对现有 LSA 及其发送机制进行扩展改进，以支持业务流的 QoS 请求，并利用融合算法进行 QoSR。

4. 利用网络仿真软件 OPNET Modeler 建模仿真，分析本文算法解决 OSPF-QoSR 问题的可行性、有效性和优越性。

1.3.2 技术难点

1. 融合算法用于解决多约束 QoSR 问题

该算法以基本遗传算法和蚁群算法为基础，克服其各自缺陷，实现了优势互补。首先利用遗传算法生成较丰富的初始信息素，再利用蚁群算法求精确解。已有理论证明融

合算法在求精解效率上优于遗传算法，在时间效率上优于蚁群算法，是求精解效率和时间效率都比较好的一种新的启发式算法[20]。如何利用该算法进行多约束 QoSR，是本课题研究的难点之一。

2. OSPF 协议上 QoS 扩展的实现

在 LSA 和链路状态数据库中添加网络资源信息，即 QoS 相关度量。由于在 LSA 中添加了 QoS 信息，链路状态的改变就要比原始的 IP 网络频繁得多，这就意味着需要对发送 LSA 的机制做一些改动。如何进行 LSA 的扩展和如何改进现有的 LSA 发送机制，也是本课题的难点之一。

3. 算法性能分析

利用网络仿真软件 OPNET 实现 OSPF 协议的多约束单播服务质量路由选择策略的建模和仿真，分析比较使用融合算法进行 OSPF-QoSR 时的网络性能。

1.3.3 创新点

1. 将遗传-蚁群融合算法用于解决多约束 QoSR 问题；
2. 实现了基于融合算法的 OSPF 协议上的 QoS 扩展。

第二章 多约束单播 QoS 路由机制分析

2.1 QoS 路由相关问题

QoSR 本身主要包括两方面的内容[21]：(1)QoSR 协议，用于网络节点之间交互信息和收集网络状态；(2)QoSR 算法，用于基于已知的状态信息计算满足 QoS 限制的可行路径。QoSR 只是网络提供 QoS 服务的一个组件，需要同一些其他技术结合使用。

2.1.1 资源预留

资源预留[22]与 QoS 紧密相连，为给业务流提供 QoS 保证，必须做好如下两项：(1)找到可行路径；(2)沿着可行路径预留资源。前者可由 QoSR 解决，而后者需要资源预留协议（ReSereVation Protocol，RSVP）完成。虽然二者分工明确，但 QoSR 经常以某种方式与资源预留或资源分配的机制结合，以便有效提高网络利用率和减小建连时间[23]。

2.1.2 接纳控制与 QoS 协商

QoS 业务流具有面向连接的特性，因此在业务流进入网络前，通常具有 QoS 连接请求和接纳控制的过程[24]。对于接纳的 QoS 请求，网络应该提供足够的资源保证该业务的 QoS 要求。以下两种情况，网络拒绝接纳一个业务流的 QoS 连接请求：(1)找不到具有足够资源的路径接纳该业务流。造成这种状况的原因是网络没有足够的资源，或者由于算法的制约没有能力找到满足 QoS 限制的路径。(2)考虑到后续 QoS 请求，为达到平衡负载、最大化网络利用率和吞吐量，拒绝不合理的 QoS 请求。

对于不能接纳的 QoS 请求，可以依据网络的当前状况和已经找到的最佳路径，采用 QoS 协商机制在业务和网络服务之间重新协商 QoS 要求。这在多媒体业务中尤为重要，例如网络带宽不足以传输精确图像时，可以通过 QoS 协商降低带宽要求，传输经过进一步压缩的图像。

2.1.3 流量工程

流量工程（Traffic Engineering）研究的主要内容是如何通过平衡负载来降低业务流通过网络时拥塞的概率[25]。如何通过 QoSR 来平衡负载是流量工程的一个重要内容，而通过平衡负载来提高网络的接纳能力也是 QoSR 研究的一个重要组成部分。

2.1.4 MPLS

多协议标签交换（MPLS：Multi-Protocol Label Switching）[26]是一种快速交换的路

由方案，该机制对实现 QoS 很有帮助。MPLS 域由具有 MPLS 交换功能的路由器组成，在域边界为每个分组加上标签，在域内部使用标签交换的方法转发分组。因此 MPLS 可以与 QoSR 结合起来，使用 QoS 在域边界选择满足 QoS 要求的路径，然后通过 MPLS 转发。同时，MPLS 也可以为 QoSR 提供更为精确的网络状态信息，并可以扩展 MPLS 以预留资源。

2.1.5 IntServ

集成服务（IntServ：Integrated Services）[27]是一种能提供语音、视频、实时和传统数据传输等服务，并将这些服务集中于一体的网络服务框架。该框架要求实现：(1)数据传输路径上的每个节点具有控制服务质量的机制；(2)一种在应用业务与网络传输实体之间交换的方法，用于传送应用业务 QoS 要求以及 QoS 控制信息；(3)包括 QoSR 中与此对应的路径选择机制。因此，QoSR 的研究是 IntServ 的一个关键问题。

2.1.6 DiffServ

区分服务（DiffServ：Differentiated Services）[28]被用来解决 Internet 上的 QoS 问题，且具有较高的可扩展性。与 MPLS 类似，具有 DiffServ 功能的路由器组成 DiffServ 域。业务流在域边界汇集成若干类，域内部只区分类而不区分业务流。通过这种聚类的方式，DiffServ 解决了 IntServe 所面临的可扩展性问题，但同时也引入了新的问题。例如，当有大量属于同一类的业务流时，所有的业务使用相同路径，因而可能造成拥塞。如果对这个聚类采用 QoSR，对不同的业务流使用不同的路径或者拒绝某些业务流，则可能避免拥塞和保证特定的服务层协议（SLA：Service-Level Agreement）[12,29]。

2.2 QoSR 算法基础

2.2.1 网络模型与 QoS 度量

网络由交换节点和链路以及主机组成，在研究中往往将其抽象为有权图 $G(V,E)$。一个有向图 G 是由一个非空有限集合 $V(G)$ 和 $V(G)$ 中某些元素的有序对集合 $E(G)$ 构成的二元组，记为 $G=(V(G),E(G))$。其中 $V(G)$ 称为图 G 的顶点集（vertex set）或节点集（node set），元素 $v\in V$ 称为图 G 的一个顶点或节点；$E(G)$ 称为图 G 的弧集（arc set），元素 $e_{ij}\in E$ 记为 $e(v_i,v_j)$ 或 $e_{ij}=v_iv_j$，为 V 中元素的有序对，称为图 G 的一个从 v_i 到 v_j 的

弧。在有向图$G(V,E)$中，令元素$e \in E$具有一组有序数列$(w_1,w_2,...,w_k)$作为e的属性，则称G为有权图，其中$(w_1,w_2,...,w_k)$称为弧e的权。

节点集V可以表示路由器，交换机、集线器等网络节点设备，在研究路由问题时，V中的元素应该是具有路由能力的交换节点；E则为连接V中两个节点的链路，以及具有的k个属性$(w_1,w_2,...,w_k)$，这些属性可以是可用带宽、链路传输延迟、接口队列长度、花费等参数。对于对称网络有$e_{ij}(w_1,w_2,...,w_k)=e_{ji}(w_1,w_2,...,w_k)$，而不对称网络中通常$e_{ij} \neq e_{ji}$。现实中的网络往往是不对称的（如可用带宽），但在研究中为了简便起见，常常使用对称网络模型以减少弧的数量[30]，本文也采用了这种简化方式。

在网络模型中可能不仅链路具有一定的属性，节点也可以具有与链路无关的属性，如 CPU 使用率、缓存使用率、花费等。一种可能的方式是将这些属性与链路属性结合起来考虑，如将链路的传输延迟、节点的排队延迟统一纳入到链路延迟中，将节点队列丢失率和链路丢失率统筹考虑等，这样节点就不再具有单独的属性。本文与通常的研究方式相同，都是基于节点不具有单独属性，并主要考虑链路延时、带宽和花费等约束[31]。

图 2.1 表示了由 7 个结点构成的网络拓扑结构模型，图中的边（链路）用三元组描述：$(delay,bandwidth,\cos t)$，各元素分别表示延时（ms）、可用带宽（$MBps$）和花费。

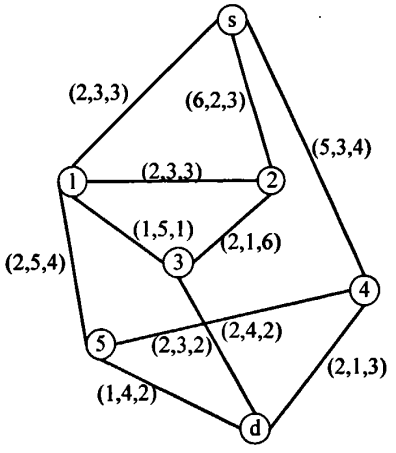

图 2.1 有权图网络模型

在图G中，如果对$i=1,2,...,k-1$有$(v_i,v_{i+1}) \in E$，则$P=(v_1,v_2,...v_k)$为图G的一条从v_1到v_k的路径，也记作$P=\{e_{i,i+1} \mid i=1,2,...,k-1\}$。QoSR 的目标就是选择一条从源节点$v_1$到目标节点$v_k$的可行路径$P$，使之满足业务 QoS 要求，同时最大化网络利用率。如图 2.1

中，从源节点 s 到目标节点 d 的路径有：$(s,1,5,d)$、$(s,2,3,d)$、$(s,4,d)$ 等。

QoS 要求需要通过可测量的 QoS 度量来实现，常用的几种 QoS 度量主要包括可用带宽、端到端延迟、分组丢失率、抖动、花费等，不同的度量具有不同的性质。按照这些性质，QoS 度量可以分为三类：可加性度量、可乘性度量和最小性度量。对于图 G 中的路径 $P=(v_1,v_2,...,v_s)$，若 $e_{i,i+1}\in P$（其中 $i=1,2,...,s-1$），将 $e_{i,i+1}$ 的第 j 个属性记为 $w^j_{i,i+1}$ 或 $w^j(e_{i,i+1})$，整个路径 P 的第 j 个属性记为 w^j_p，则以上三类度量的定义如下[32]：

(1)相加性度量。

若 $w^j_p=\sum_{i=1}^{s-1} w^j_{i,i+1}$，则称路径 P 的第 j 个属性为可加性度量。可加性度量包括延迟、抖动、花费、转发跳数等。例如，一条路径 P 的延迟为从源到目的地的所有链路延迟的总和。如图 2.1 中路径 $(s,1,5,d)$ 的延迟总和为 5，花费为 9。

(2)相乘性度量。

若 $w^j_p=\prod_{i=1}^{s-1} w^j_{i,i+1}$，则称路径 P 的第 j 个属性为相乘度量。例如分组丢失率为可加乘性度量，其中链路丢失率 $0\le w^j_{i,i+1}\le 1$。在求解有关可乘性度量的过程中，可以类比参照可加性度量的有关求解方法[12]。

(3)最小性度量。

若 $w^j_p=\min w^j_{i,i+1}$，则称路径 P 的第 j 个属性为最小度量。例如，一条路径 P 的带宽为从源到目的地所有链路中的瓶颈带宽，所以带宽为最小性度量。如图 2.1 中路径 $(s,1,5,d)$ 的带宽为 3（其中链路 $(s,1)$ 为瓶颈带宽）。对于求解 $w^j_p=\max w^j_{i,i+1}$ 的问题可转化为 $-w^j_p=\min\{-w^j_{i,i+1}\}$ 问题。

2.2.2 多约束优化与 NPC

QoSR 与传统路由相比，根本特点是由用户业务流提出 QoS 要求，网络提供满足这些 QoS 要求的可行路径。这些业务往往要求在多个约束目标下选择优化路径，例如多媒体业务要求网络同时保证一定的带宽、延迟和抖动，并使花费最小。已经证明，这种多约束优化路径的求解无法在多项式时间内完成，即其计算复杂度为 NP 完全（NP-Complete，NPC）或 NP 难（NP-Hard，NPH）[4]。

为了便于探讨什么类型的多约束优化会造成 NPC 的复杂度，QoSR 可以分成四个基本类。⑴链路优化问题：例如，带宽优化路由是寻找具有最大瓶颈链路带宽的可行路径；⑵链路受限问题：例如，带宽受限路由是寻找瓶颈链路带宽大于一定要求的可行路径。这两类基本问题的优化或者限制的目标只与路径的瓶颈有关，而与路径中非瓶颈的链路无关，QoS 参数为最小性度量。⑶路径优化问题：例如，延迟最小路由是在所有可行路径中寻找从源到目的地延迟最小的路径，而判断的标准“延迟”是路径中所有链路延迟的总和，与路径中的每段链路相关；⑷路径受限问题：例如，延迟受限路由要求路径的总延迟满足一定的要求。这两类基本问题判断目标与路径中各个路径相关，QoS 参数为可加性度量。通常各种路由问题可由以上四种基本问题组合而成。

多项式时间可解的问题包括链路受限-链路优化问题、多重链路受限问题、链路受限-路径受限问题、链路优化-路径受限问题。例如带宽受限最小延迟问题是在所有满足带宽要求的路径中，寻找延迟最小的路径，这个问题是链路受限和路径优化问题的组合并且多项式时间可解。

两类 NPC 问题是路径受限-路径优化问题和多重路径受限问题（及其与链路受限、优化问题的组合）。例如，延迟受限的最小花费（Delay-Constraint Least Cost，DCLC）问题是路径受限和路径优化问题的组合，延迟受限并且抖动受限的路由问题是多重路径受限问题，求解这些问题的复杂度都是 NPC 的。对于 QoSR 中的 NPC 问题，通常需要设计启发式算法求解。

2.3 多约束单播 QoSR 算法分析

按照所求解的问题类型和求解方法，常用 QoSR 算法可以分成以下几类：多项式非启发类、伪多项式非启发类、限定 QoS 度量类、路径子空间搜索类和花费函数类。

2.3.1 多项式非启发类

多项式非启发类算法针对的并不是典型的 QoSR 问题，而是多项式可解的问题，但这类问题及其求解思路却是 QoSR 算法的基础。

Wang 和 Crowcroft 使用 Dijkstra 最短路径树算法，实现了带宽延迟受限的源路由求解[3]。首先在网络拓扑图中将带宽不满足要求的链路剪除掉，然后再以延迟为关键字使用最短路径树算法计算。这样求得的路径满足 QoS 要求，并且具有最短延迟。对于分布式方法，他们对最短最宽路径提出了预计算方式。两个节点间的最宽路径是指这两个节

点之间所有的路径中，瓶颈带宽最大的路径[3,33]。最短最宽路径（Shortest-Widest Path）是指在多条最宽路径中延迟最小的路径。每个节点通过链路状态协议维护全局状态，并使用扩展 Bellman-Ford 算法或 Dijkstra 算法为每个可能的连接目的地预先计算出下一跳，这样在连接请求时只需要简单的查找路由表就可以了。这个算法在各个节点网络状态信息一致的情况下保证无回路。

2.3.2 伪多项式非启发类

扩展 Bellman-Ford 算法（EBFA）是一种典型的用于求解多重受限 QoSR 问题的伪多项式非启发类算法[6]，其核心思想是在偏序集中寻找最小元。

定义偏序关系“<”：假设所有的限制都是路径限制，所有的链路度量都是可加性度量，即 $w_P^j=\sum_{i=1}^{k-1} w_{i,i+1}^j$，路径 P 的度量矢量 $\vec{w}_P=(w_P^1,w_P^2,...,w_P^k)$，则 $\vec{w}_P<\vec{w}_Q$ 当且仅当对 $\forall j\in\{1,2,...,k\}$ 有 $w_P^j<w_Q^j$。类似的，可以定义偏序关系“>”和“=”。偏序关系中，并不是任意两个元素 $\vec{w}_P$ 和 $\vec{w}_Q$ 之间，都必然存在“<”、“>”和“=”三个关系中的一种，例如 $\vec{w}_P=(1,2,3)$，而 $\vec{w}_Q=(3,2,1)$，则二者之间不存在其中的任意一种关系。

设 P 是从源节点到目标节点的一条路径，如果不存在从源节点到目标节点的路径 Q 使得 $\vec{w}_Q<\vec{w}_P$，则称 P 为一条最优路径。

从定义可见，两个节点之间的最优路径可能不唯一，造成这个状况的根本原因是偏序关系。对于不同的 QoS 限制或其不同的组合，可能具有不同的最优路径满足要求。对任意 QoS 限制，若存在一条可行路径，则必存在最优路径满足该 QoS 限制。

扩展 Bellman-Ford 算法（EBFA）如表 2.1 所示，每个节点 u 维护一个集合 $PATH(u)$，记录了所有从源节点 s 到 u 的最优路径。主函数（*BELLMAN－FORD*）的前三行是初始化；（4~6）行通过调用 *RELAX* 函数算出从源节点 s 到目的节点 d 的所有最优路径，并保存在 $PATH(d)$ 中；最后检查是否存在满足 QoS 限制的最优路径。*RELAX* 函数第 4 行检查是否存在一条从 s 到 v 的旧路径 q 比经过 u 的新路径好，如果存在则经过 u 的新路径不是最优路径；函数第 6 行检查经过 u 的新路径是否比原来从 s 到 v 的路径好，如果好，则原来的路径不是最优路径，应该从 $PATH(v)$ 中去掉。$PATH(u)$ 中最优路径的数目与

$|N|$和$|E|$成指数关系，因此 EBFA 的时间、空间复杂度都是指数级的。

表 2.1 求解多重受限 QoSR 的扩展 Bellman-Ford 算法

$RELAX(u,v,\vec{w})$	$BELLMAN-FORD(G,\vec{w},\vec{c},s,d)$
(1)For each $w(\vec{p})$ in $PATH(u)$	(1)For $i=0$ to $\|N(G)\|-1$
(2) $flag=1$	(2) $PATH(i)=\phi$
(3)For each $w(\vec{q})$ in $PATH(v)$	(3) $PATH(s)=\{\vec{0}\}$
(4)if $(w(\vec{p})+w(\vec{u},v)\geq w(\vec{q}))$ then	(4)For $i=1$ to $\|N(G)\|-1$
(5) $flag=0$	(5)For each edge $(u,v)\in E(G)$
(6)if $(w(\vec{p})+w(\vec{u},v)<w(\vec{q}))$ then	(6) $RELAX(u,v,\vec{w})$
(7)remove $w(\vec{q})$ from $PATH(v)$	(7)For each $w(\vec{p})$ in $PATH(d)$
(8)if $(flag=1)$ then	(8)if $(w(\vec{p})<\vec{c})$ then return "*yes*"
(9)add $w(\vec{p})+w(\vec{u},v)$ to $PATH(v)$	(9)return "*no*"

2.3.3 限定 QoS 度量

限定 QoS 度量是指将除了一类之外的所有 QoS 度量都限定在一个有限集上。Yuan 和 Liu 在 EBFA 的基础上设计了粒度受限（Limited Granularity）的启发式算法[9]，通过限制$PATH(u)$中最优路径的数目达到减小复杂度的目的。算法对于k限制问题，将$(k-1)$种度量映像到$(k-1)$个有限集中。令$2\leq i\leq k$，$w^i\in(0,c^i)$为路径的$(k-1)$个度量，将w^i映像到X^i个元素$\{a_1^i,a_2^i,...,a_{X_i}^i\}$中，其中$0<a_1^i<a_2^i<...<a_{X_i}^i=c^i$。将度量$w^i(e)$近似为$a_j^i$当且仅当$a_{j-1}^i<w^i(e)\leq w_j^i$，从而有路径$P$的度量$w^i(P)$近似为$a^i(P)$。与 EBFA 类似，每个节点$u$维护一个记录所有从源节点$s$到$u$的最优路径的集合$PATH'=d_n[a^2,a^3,...,a^k]$，其中$a^i\in\{a_1^i,a_2^i,...,a_{X_i}^i\}$。对于多条$d_n[a^2,a^3,...,a^k]$值相同的最优路径，$PATH'$表中只保存其中一条路径。因此每个节点所维护的信息量（该表的大小）$|PATH'|=\prod_{i=2}^{k}X^i$，算法复杂

度为 $O(|N|\cdot|E|\prod_{i=2}^{k}X^i)$。

可以证明，令 n 为常数，$X^2=X^3=...=X^k=nL$ 时，如果对 $2\le i\le k$ 采用 $a_1^i=\frac{c^i}{X^i}$，$a_2^i=\frac{c^i}{X^i}*2$，...，$a_{X^i}^i=c^i$ 等间距近似法，只要存在长度为 L 的路径 P 同时满足：(1) $w^1(P)\le c^1$；(2)对 $2\le i\le k$ 有 $w^i(P)\le c^i-\frac{c_i}{n}$，则粒度受限的启发式算法能够找到一条满足 $\vec{w}(P)\le\vec{c}$ 的路径 P。这说明若每个节点保存的表的大小为 $n^{k-1}|N|^{k-1}=O(|N|^{k-1})$ 时，粒度受限的算法能够找到 $1-\frac{1}{n}$ 近似解的条件是：存在满足 QoS 限制 $(1-\frac{1}{n})\vec{c}$ 的可行路径。

Chen-Nahrstedt 设计了多重路径受限的源路由算法[7]。算法的思想是将 QoS 度量的取值范围 R 或 Z 映射到一个有限集合 C，这样保证多项式可解。以花费 Cost 举例，设 C 为最大花费限制，x 为一个小于 C 的正整数。算法将每个链路的花费映射到集合 $C'=\{1,2,...,x+1\}$ 中，其中属于区间 $[0,C]$ 的花费映射到 $\{1,2,...,x\}$ 中，属于区间 $(C,+\infty)$ 的花费映射为 $(x+1)$；花费 C 映射为 x。可以证明，满足新问题的可行路径在原来的问题中也是一条可行路径，而新问题可以通过扩展 Dijkstra 算法或扩展 Bellman-Ford 算法在多项式时间内求解。算法的性能与 x 密切相关：随着 x 增大，找到可行路径的概率也增大，但同时引入了更大的开销。因此，可以通过选取 x 来调节该算法的性能。

与此类似，对于延迟受限的最小花费问题（DCLC）存在一些伪多项式时间的近似算法[34,35]。对于任意 $\varepsilon>0$，存在一个多项式时间的算法能够找到一条路径，在满足延迟受限的同时，花费不超过最小花费的 $(1+\varepsilon)$ 倍。由于这些算法的复杂度与 ε 有关，因此称为伪多项式算法，如 Lorenz 算法的复杂度为 $O(ne\log e\log\log e+\frac{ne}{\varepsilon})$[35]。Orda 等设计了通过量化花费函数的预计算算法[10]，该算法针对多重可加性限制的路径优化问题，基于通过 Bellman-Ford 算法，将计算复杂度减小到 $O(\frac{1}{\varepsilon}Hn\log C)$，其中 H 为最长路径的跳数，C 为路径花费的上限。

2.3.4 路径子空间搜索

路径子空间搜索是指直接限制拓扑图中的某些路径或者路径数量，从而在路径集合中的子空间搜索可行路径的方法。这种方法在源节点或者中间节点就可以计算出可行路

径，具有较高的启发性。

Yuan 和 Liu 设计了路径受限的多重路径受限的分布式算法[9]。他们研究了 EBFA 复杂度高的主要原因是最优路径的数目$|PATH|$与网络节点数、边数成指数关系，因此可以通过限制$|PATH|$减小复杂度。令X为$|PATH|$最大数目，在原扩展 Bellman-Ford 算法*RELAX*函数第 9 步加入一条最优路径前判断是否满足$|PATH|<X$，若不满足则不加入该最短路径。通过这样简单的限制，保证算法复杂度为$O(X^2|N|\cdot E)$。进而证明只要存在可行路径，当每个节点保存的表项数目$|PATH|$满足$O(|N|^2\cdot\log(|N|))$时，算法能够以极高的概率找到可行路径。

Salama 等人对复杂度为 NPC 的延迟受限的最小花费问题（DCLC）设计了分布式启发算法[11]。与 EBFA 类似，为每个可能的目的网络、每个节点保存一个花费向量和一个延迟向量。为建立 DCLC 可行路径，源节点向目的节点发送一个控制信息，已经部分建立的可行路径的末节点i收到该信息后，将信息继续向目标节点传递。设(i,j)是从i到目标节点的最小花费路径中下一跳的链路，(i,k)为最小延迟路径中下一跳的链路，则传递规则是：只要加上从节点j开始的最小延迟路径仍然满足 QoS 限制，节点i选择(i,j)，否则选择(i,k)。

2.3.5 花费函数

为了降低算法的复杂度并提供更好的 QoS 支持，可以通过研究花费函数，保证所计算的最小花费路径能够满足 QoS 需求。例如，通过将链路花费与利用率结合，能够降低连续阻塞概率，抑制路由抖动[36]。

Korkmaz 等设计了一种在多个可加性受限条件下求解优化路径的通用算法[12]。算法首先构造路径花费函数$g_\lambda(P)=(\frac{w^1(P)}{c^1})^\lambda+...+(\frac{w^k(P)}{c^k})^\lambda$，其中$w^i(P)$为路径$P$的第$i$维属性，$c^i$为一个业务流所对应的 QoS 要求。最小化花费函数所找到的路径，为多重受限 QoS 问题提供了一个连续的近似频谱：从$\lambda=1$的线性近似，到$\lambda=\infty$时的渐进近似。因此原来的多限制 QoSR 问题转化为最小化$\lim_{\lambda\to\infty}g_\lambda(P)$所对应的路径问题。

Ergun 等设计了一条链路基于不同的花费提供不同延迟的启发式算法[13]。该算法的

思想是将传统的 DCLC 问题分解为两个问题：(1)使用传统方法找到一条可行路径；(2)将整个路径的延迟分割成路径中每个链路延迟的限制，从而得到路径的最小花费。

Juttner 等基于拉格朗日松弛法（Lagrange Relaxation）设计了 DCLC 的源路由求解算法[14]。算法的思想是首先构造路径花费函数 $g_\lambda(P)=c(P)+\lambda\cdot d(P)$，其中 c 为原来的花费，$d(P)$ 为路径 P 的延迟，$\lambda\geq 0$。然后以 $g_\lambda(P)$ 为关键字使用 Dijkstra 算法计算最短路径 P，如果 $\lambda=0$ 且同时 P 满足延迟要求，则 P 为原 DCLC 问题的最优解；如果 P 不能满足要求，则可通过增大 λ 而加大对 $g_\lambda(P)$ 的惩罚力度，求得满足延迟要求的路径。令 $L(\lambda)=\min\{g_\lambda(P)\,|\,\forall P\in P(s,t)\}-\lambda D$，其中 D 为延迟要求，则对 $\forall\lambda\geq 0$ 有 $L(\lambda)$ 为原 DCLC 问题的花费下界。算法通过计算特定的 λ 从而最大化 $L(\lambda)$，得到花费下界，并给出一条可行路径。算法的复杂度为 $O(n^2\log^4 n)$。

2.4 算法小结

2.4.1 算法比较及存在问题分析

为了进一步降低复杂性和提高性能，有些 QoSR 算法将以上几种思想结合起来使用。例如，将花费函数和限定 QoS 度量结合，即将花费限定在一个有限整数区间[13]。

以上几类热点启发式算法具有以下问题：多项式非启发类算法只能解决最小性限制的问题，而对一般的 QoSR 问题是不可行的；伪多项式 EBFA 能够精确求解多重受限 QoSR 问题，但由于指数级别的复杂度也是不可取的；限定 QoS 度量的算法复杂度与 QoS 度量粒度关系密切，因此为减小计算开销而极大的限制了应用场合；路径子空间搜索的算法需要较好的启发函数，以增大路由失败概率和降低路由性能为代价，减小了计算开销；花费函数类中，基于特定花费函数的算法难以实现花费的可扩充性，而花费分割则为提供更好的 QoS 支持提供了一种思路。

表 2.2 给出了上文中提到的部分单播 QoSR 算法的比较，其中算法以第一作者命名，状态信息一栏表示计算 QoSR 的节点需要收集和维护的网络状态信息的类别。（注：表中 n 为节点数，e 为边数，X 为最优路径的数目。）

表 2.2 单播服务质量路由算法比较

算法名称	所解决的路由问题	路由策略	时间复杂度	状态信息	启发式类别
Wang	带宽受限	源路由	$O(n\log n+e)$	全局	非启发式
Wang	带宽优化	分布式路由	$O(ne)$	全局	非启发式
EFBA	多重路径受限	分布式路由	伪多项式	全局	非启发式
Yuan	多重路径受限	分布式路由	$O(ne\prod_{i=2}^{k}X^{i})$	全局	限定 QoS 度量
Chen	一般问题	分布式路由	$O(e)$	本地	限定 QoS 度量
Orda	多重路径受限优化	分布式路由	O(多项式/ε)	全局	限定 QoS 度量 ε 优化
Yuan	多重路径受限	分布式路由	$O(X^{2}ne)$	全局	路径子空间搜索
Salama	延迟受限最小花费	分布式路由	$O(n^{3})$	全局	路径子空间搜索
Korkmaz	多重路径受限优化	源路由	多项式	全局	特定 Cost 函数
Ergun	延迟受限优化	源路由	O(多项式/ε)	全局	分割 Cost 函数的 ε 优化
Juttner	延迟受限最小花费	源路由	$O(n^{2}\log^{4}n)$	全局	花费函数拉格朗日松弛

2.4.2 算法有效性分析

一个 QoSR 算法首先要在理论上保证避免回路（或者能够检测排除回路），然后则需要进一步考虑网络实际情况，包括网络模型、路由信息是否陈旧等问题。由于实际情况和理论之间的差别，往往可能对算法的性能甚至正确性造成极大的影响，而实际性能过于低下的算法也就成了无效的算法。

1. 路由回路问题

在 Salama 等人所提出的算法[11]中，由于控制信息可能不断的在最小延迟路径和最小花费路径之间变化，这样可能造成回路，这就属于算法本身理论上的缺陷。可采用检测回退的方法来解决这个问题：当控制信息对同一节点访问两次时，说明算法产生了回路，这时回退路径选择过程，直到找到一个原来在其后使用最小花费路径的节点，该节点将原来沿最小花费路径传递改为沿最小延迟路径传递。可以证明，在所有节点网络信息一致的前提下，通过这种机制能够保证消除回路。

如果在上述算法中，不断地出现回路并不断地采用回退过程消除回路，则会严重影响算法的性能。Sun 和 Landgendorfer 针对这个问题，提出了使用避免回路的方法取代检测回路并回退的方法，改善了 Salama 算法最差情况下的性能[37]。改进算法首先沿着最小延迟路径，直到从某一个节点开始，加上其最小花费路径的延迟满足业务的 QoS 限制。

这样就避免了回路的产生。

Sobrinho 通过代数方法详细研究了分布式 QoS 中路由回路和最短路径问题[38]。他定义了关于链路度量的二元操作和一个抽象的全序关系，以及在这个关系中路径度量的保序性。其中保序性是指，对任意两条路径 P、Q，如果给这两条路径同时追加上路径 L 时，在序关系中 P 和 Q 的关系与 $(P+L)$ 和 $(Q+L)$ 的序关系相同（也就是说追加过程不改变原来路径度量的序关系）。Sobrinho 还证明，路径中 QoS 限制的度量函数的保序性，是能够成功使用通过 Dijkstra 算法计算最短路径的充要条件；否则计算过程中可能产生回路。

2. 陈旧信息的影响

在研究 QoSR 问题时，通常假设每个节点已知本地状态信息，并通过链路状态协议或距离向量协议的交互获得（聚集的）全局状态。但是，在传播本地状态的过程中，有不可忽略的网络传输延迟和协议交互延迟而且网络状态是不断更新的，这样就造成了网络状态信息在时间上的不精确性，即陈旧性。

很多 QoSR 算法是基于节点已知网络当前状态信息的，然而在实际中可能由于节点信息的陈旧性而导致算法性能降低甚至失效。Shaikh 等人详细研究了陈旧性对算法的影响[36]，研究中使用的算法是：(1)首先剪枝，即在拓扑图中剪除带宽不满足要求的链路（可选）；(2)计算最短路径（跳数最少）；(3)在最短路径中找最小花费路径。

这样，算法找到的路径是最小花费的最短可行路径或者最小花费的最短路径（如果不使用第一步）。通过大量模拟实验，说明信息的陈旧性小（即节点保存的信息较接近网络的真实状态）时，应该使用剪枝的机制；而陈旧性大时不应使用。

2.5 本章小结

本章首先介绍了 QoS 路由的相关问题，然后详细分析了当前常用的几类多约束单播 QoSR 算法，通过比较说明了这几类算法存在和需要进一步研究解决的问题。

第三章 遗传-蚁群融合算法

3.1 遗传算法及其特点分析

3.1.1 遗传算法概述

遗传算法（Genetic Algorithm, GA）是模拟自然界中生物遗传机制的随机搜索算法，由美国 Michigan 大学的 John Holland 教授等人于 1975 年首先提出[39]。它以达尔文的生物进化论——“适者生存、优胜劣汰”和孟德尔的遗传变异理论——“生物遗传进化主要在染色体上，子代是父代遗传基因在染色体上的有序排列”为基础，仿真生物在染色体层面的各种遗传优化作用，通过选择、交叉、变异等优化过程设计的人工寻优方法。

遗传算法是具有“生成+检测”（generate-and-test）迭代过程的搜索算法，标准遗传算法的基本处理流程如图 3.1 所示：

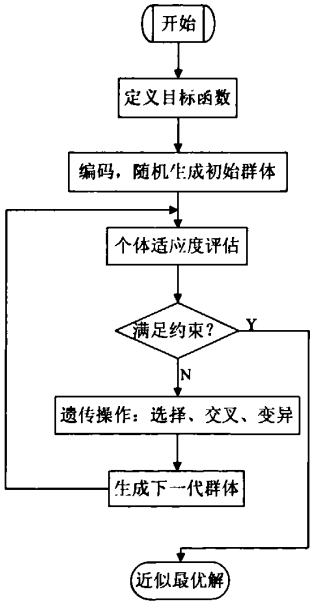

图 3.1 遗传算法的基本流程

由图 3.1 可见，遗传算法是一种群体型操作，该操作以群体中的所有个体为对象。选择（Selection）、交叉（Crossover）和变异（Mutation）是遗传算法的三个主要操作算子，它们构成了所谓的遗传操作（Genetic Operation），使遗传算法具有了其它传统方法所没有的特性。遗传算法中包含了如下五个基本要素：

⑴参数编码：GA 不能直接处理解空间的解数据，在进行搜索之前必须通过编码将

它们表示成遗传空间的基因型串结构数据，这些串结构数据的不同组合便构成了不同的点。

⑵初始群体的设定：由于 GA 的群体型操作需要，必须为遗传操作准备一个由若干初始解组成的初始群体。随机产生 N 个初始串结构数据，每个串结构数据称为一个个体，N 个个体就构成了一个群体，GA 以这 N 个串结构数据作为初始点开始迭代。

⑶适应度函数的设计：适应度（Fitness）函数表明个体或解的优劣性，并作为以后遗传操作的依据。不同的问题，适应度函数的定义方式也不同。

⑷遗传操作设计：选择操作的目的是为了从当前群体中选出优良的个体，使它们有机会作为父代为下一代繁殖子孙，GA 通过选择过程体现这一思想，进行选择的原则是适应性强的个体为下一代贡献一个或多个后代的概率大，选择实现了达尔文“适者生存”的原则；交换操作是遗传算法中最主要的遗传操作，通过交换操作可以得到新一代个体，新个体组合了其父辈个体的特性，交换体现了信息交换的思想；变异操作首先在群体中随机选择一个个体，对于选中的个体以一定的概率随机地改变串结构数据中某个串的值，同生物界一样，GA 中变异发生的概率很低，通常取值在 0.001~0.01 之间，变异为新个体的产生提供了机会。

⑸控制参数设定：主要是指群体大小和使用遗传操作的概率等的设定。

以上五个要素构成了遗传算法的核心内容。

3.1.2 遗传算法特点分析

遗传算法是在解集合的一个子集内进行搜索最优解或近似最优解的搜索算法，它可在解的质量和求解效率上达到一种较好的平衡，对求解 NP 难问题比较有效。作为一种随机优化与搜索方法，遗传算法具有如下优点：

⑴GA 以决策变量的编码作为运算对象，搜索过程不直接作用在变量上，使得遗传算法可直接对结构对象（如：集合、序列、矩阵、树、图等）进行操作。

⑵GA 是一种概率搜索算法，非确定性规则，灵活性强。虽然这种概率特性也会使群体中产生一些适应度不高的个体，但随着进化过程的进行，新的群体总会更多地产生出许多优良个体。

⑶GA 对搜索空间没有任何特殊要求（如连通性、凸性等），只利用适应度作为搜索信息，对问题的依赖性较小，具有高度的非线性，适应范围更广。

⑷GA 同时对包含多个个体的群体进行搜索，具有一种隐含的并行性，使得它非常适合于大规模并行分布式处理。

⑸GA 是收敛的，文献[40]中定理 2.7 指出，具有变异概率 $Pm \in (0,1)$，交叉概率 $Pc \in [0,1]$，同时采用比例选择法且在选择前保留当前最优解的遗传算法可收敛到全局最优解。

作为一种较新的算法，遗传算法在实际应用中还有许多地方有待进一步研究和改进，主要集中在以下几个方面：⑴对于系统中的反馈信息利用不够，搜索速度比较慢[41]；⑵更擅长全局搜索而局部搜索能力不足，当求解到一定范围时往往做大量无为的冗余迭代，求精确解效率低。研究表明，遗传算法可以用极快的速度达到最优解的 90%左右，但要达到真正的最优解则要花费很长的时间。

3.2 蚁群算法及其特点分析

3.2.1 蚁群算法概述

蚁群算法(Ant Colony Optimization, ACO)，又称蚂蚁算法，是一种用来在图中寻找优化路径的概率型技术，属于随机搜索算法，由意大利学者 Marco Dorigo 等在 1991 年最早提出[42]，其灵感来源于蚂蚁在寻找食物过程中发现路径的行为：蚂蚁总能找到巢穴与食物源之间的最短路径。研究发现，蚂蚁这种群体协作功能是通过一种遗留在其来往路径上的叫做信息素（Pheromone）的挥发性化学物质来进行通信和协调的，整个蚁群正是通过信息素相互协作，形成正反馈，从而使多个路径上的蚂蚁都逐渐聚集到最短的那条路径上。所谓正反馈现象是指蚂蚁不断在经过的路径上释放信息素，而信息素的浓度会随着经过该路径的蚂蚁数量增大，蚂蚁在回巢或觅食时会选择信息素浓度较大的路径，这样就会有更多的蚂蚁选择此路径，也就是说某一路径上经过的蚂蚁越多，则后来者选择该路径的概率就越大。在蚂蚁觅食行为的启发下，学者们在计算机上模拟真实蚂蚁的群体行为，并把该思想用于解决众多复杂的实际应用问题，从而产生了蚁群算法。

计算机学者首先把待解决问题转换成相应的构建图，然后让人工蚁群在构建图中仿照真实蚂蚁的行为：人工蚂蚁在构建图中游走，并根据某些规则释放一定量的人工信息素，随着信息素在构建图中某些成分上的不断积累，人工蚂蚁可以探测人工信息素的浓度并以此为依据构造出问题的解。这种方法具有分布性、并行性、全局寻优、无须依赖

具体问题的数学特性等特点，能够在较短的时间内发现问题的近似最优解，迅速成为最成功的启发式算法之一，具有广泛的应用领域。例如：在网络路由处理中，网络流量分布不断变化，网络链路或节点会随机地失效或重新加入，蚁群算法的自身催化与正向反馈机制正好符合了这类问题的求解特点；蚁群觅食行为所呈现出的并行与分布特性还使得算法特别适合于并行化处理；在聚类、网页文档分类及主题图显示、智能机器人控制等方面，蚁群算法也得到了成功的应用。

3.2.2 蚁群算法特点分析

众多研究已经证明蚁群算法具有很强的发现较好解的能力，因为该算法不仅利用了正反馈原理，在一定程度上可以加快进化过程，而且是一种本质并行的自组织算法，不同个体之间不断进行信息交流和传递，从而能够相互协作，有利于发现较好解。作为一种求解组合最优化问题的新型通用启发式算法，其优点主要如下：

⑴分布式计算：蚁群算法是一个分布式的多智能体系，具有本质的并行性，它在问题空间的多点同时独立地进行解搜索，易于并行实现，也增加了算法的可靠性[43]。

⑵自组织：组织指令来自于系统内部的组织行为称为自组织，即在没有外界作用下系统从无序到有序的进化过程。蚁群算法体现了这一过程，在算法初期，单只蚂蚁无序地寻找解，经过一段时间的算法演化，蚂蚁越来越趋向于寻找到接近最优解的一些解，这大大增强了算法的鲁棒性[44]。

⑶正反馈：反馈代表信息输出对输入的反作用，正反馈是指以现在的行为去加强未来的行为。蚂蚁选路释放的信息素的堆积是一个正反馈的过程，在较优解经过的路径上留下更多的信息素，更多的信息素又吸引了更多的蚂蚁。这个正反馈过程使得路径上的初始值不断地扩大，引导整个系统向着最优解的方向进化[44]。

类比于任何一种随机搜索算法，蚁群算法本身也不可避免的存在某些缺点，具体表现如下：

⑴搜索时间较长，收敛速度慢。

⑵易陷于局部最优解：由于正反馈机制太强，搜索进行到一定程度后，所有个体所发现的解完全一致，不能对解空间进一步搜索，算法过早收敛于非全局最优解，不利于发现更好的解，即出现停滞现象（Stagnation Behavior）。

⑶对于本文所研究的服务质量路由，蚁群算法并不能直接用于 QoSR 算法，理由主

要有以下两点[45]：①QoSR 算法中，每一条链路都有 2～3 种参数约束，而蚁群算法本身只有 1 种参数约束，很显然不能直接将蚁群算法用于 QoS 路由选择；②蚁群算法采用多只蚂蚁分头寻找，按概率转移，最后再相遇。若将其直接用于网络路由算法中，会造成计算复杂，管理困难。

3.3 基于遗传算法和蚁群算法的融合算法

3.3.1 算法思想

综上，遗传算法具有快速随机的全局搜索能力，但对于系统中大量反馈信息的利用不够，更擅长全局搜索而局部搜索能力不足，求精确解效率低；蚁群算法通过信息素的累积和更新收敛于最优路径上，具有分布式并行全局搜索能力，但初期信息素匮乏，求解速度慢，且由于强大的正反馈机制易陷于局部最优解。遗传-蚁群融合算法将遗传算法与蚁群算法相融合，克服各自缺陷，优势互补。利用遗传算法生成信息素分布，利用蚁群算法求精确解[20]。求精解效率优于遗传算法，时间效率优于蚁群算法，是求精解效率和时间效率都比较好的一种新的启发式算法。

融合算法首先应用标准遗传算法，结合所解决问题，采用十进制实数编码，随机生成初始群体，结合目标函数定义适应度函数，通过选择、交叉、变异这三种基本遗传操作，成生下一代群体，循环迭代得到问题的优化解。然后应用蚁群算法，利用遗传算法已求得的优化解进行信息素初始化，而后计算转移概率，更新信息素，从而循环迭代得到问题的最优解。以遗传算法所得优化解作为蚁群算法的初始信息素，正是对“融合”的体现，也是本算法的核心所在。本文将遗传-蚁群融合算法应用于扩展 OSPF 协议的多约束单播 QoS 路由选择问题的求解中，仿真测试证明该路由选择算法是有效可行的。算法具体流程见图 3.2：

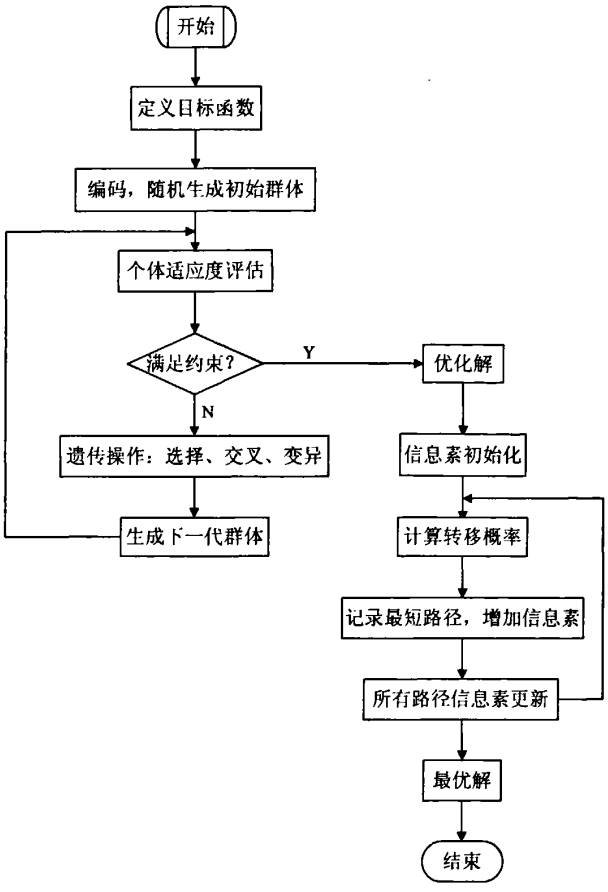

图 3.2 遗传-蚁群算法流程图

3.3.2 算法描述

1. 优化解求解

利用标准遗传算法，进行迭代处理，得到目标问题的较优解。

Step1-1 编码：根据网络拓扑图，以随机深度优先搜索算法得到从源节点到目的节点的路径，该路径上的节点序列即为染色体编码，一个节点序列就是一条染色体，即一条路径对应着一个个体。该编码可用单链表或一维数组实现。

Step1-2 初始群体的生成：由于遗传算法群体型操作的需要，必须为遗传操作准备一个由若干个初始解组成的初始群体。初始群体的每个个体都是通过随机方法产生的。本算法以随机深度优先搜索，搜索到 N 条路径，以此作为初始群体，N 为群体规模。

需要指出的是，对于 QoS 请求中的带宽约束条件，只需根据网络拓扑即可作出判断，过滤掉拓扑图中所有不满足带宽约束的链路。经过滤后的图可能不是连通图，如果源节点和目的节点不在同一连通子图中，路由选择是无意义的。因此，本算法假定经预处理

后的图是连通的，并且所有的链路都已经满足带宽约束。

Step1-3 适应度函数的确定：适应度函数是评价个体优越性的标准。该适应度函数的建立必须满足以下条件：①被选路由的总体代价应最小；②满足各项 QoS 约束。

本算法的适应度函数 $f(p(s,d))$ 由目标函数和惩罚函数组成，其定义分别如下[46]：

$$f_c = \frac{a}{\cos t(p(s,d))}$$

$$f_d = \Phi_d(delay(p(s,d)) - Dp)$$

$$f_{dj} = \Phi_{dj}(delay\_jitter(p(s,d)) - DJp)$$

$$\Phi_d(Z_d) = \begin{cases} 1, Z \le 0 \\ r_d, Z > 0 \end{cases}$$

$$\Phi_{dj}(Z_{dj}) = \begin{cases} 1, Z \le 0 \\ r_{dj}, Z > 0 \end{cases}$$

则 $f(p(s,d)) = f_c(Af_d + Bf_{dj})$

其中，a 是正实系数，f_c 体现路径费用对个体优良程度的影响，考虑到费用越大，个体性能越差，适应度越小，故取费用函数的倒数来保证这一点；$\Phi_d(Z_d)$ 是延时度量的惩罚函数，当个体满足延时约束（$delay(p(s,d)) \le Dp$）时，其值为 1，否则等于 r_d（$0 < r_d < 1$）；$\Phi_{dj}(Z_{dj})$ 是延时抖动度量的惩罚函数，当个体满足延时抖动约束（$delay\_jitter(p(s,d)) \le DJp$）时，其值为 1，否则等于 r_{dj}（$0 < r_{dj} < 1$）。r_d 和 r_{dj} 值的大小决定惩罚程度，通常需要通过试验确定；正实数 A、B 分别为 f_d 和 f_{dj} 的加权系数，表示延时和延时抖动在适应度函数中所占的比重，取值根据具体应用而定。

Step1-4 选择操作：选择操作的目的是为了从当前群体中选出优良的个体，使他们有机会作为父代为下一代繁殖子孙。判断个体优良与否的准则就是各自的适应度值，个体适应度越高，被选择的机会就越多。

本算法采用和适应度值成比例的概率方法来进行选择操作。具体地说，就是首先计算群体中所有个体适应度的总和（$\sum f$），再计算每个个体的适应度所占的比例

（$f_i/\sum f$），并以此作为相应的选择概率（Ps），即

$$Ps_i = \frac{f(p_i)}{\sum_{i=1}^{N} f(p_i)}$$

Step1-5 交叉操作：交叉操作是遗传算法中最主要的遗传操作，正是由于交叉操作得到了新一代个体。简单的交叉可分两步进行，首先对配对库中的个体进行随机配对；其次，在配对个体中随机设定交叉处，配对个体彼此交换部分信息。

在本算法中，交叉操作共分以下几个步骤进行：

①从群体中随机选择两个个体 p_i 和 p_j（即以随机深度优先搜索算法得到两条路径），找出两条路径中除源节点和目标节点之外所有相同的节点，构成相同节点集 Ω（$\Omega \subseteq V$）。若 $\Omega = \phi$，则不进行交叉操作，转 Step1-6；否则，转②；

②随机选择相同节点集 Ω 中的两个节点 a、b（注：a、b 在 p_i、p_j 中出现的先后次序应相同，否则不交叉），交换 p_i、p_j 中 a、b 之间的链路；

③如果交换链路后所得的新 p_i、p_j 中有重复的节点，则将相同节点间的链路删除，以确保交叉后的路径中无循环路由。

Step1-6 变异操作：变异操作也是随机进行的，目的在于保持群体的多样性，避免求解过程陷入局部最优。变异算子用一个很小的概率（变异概率 Pm）随机地改变染色体串上的基因，其效果是增加了群体的多样性、扩大了搜索空间。依已设定的变异概率 Pm 随机选择一个个体 p_i，产生一个随机概率 Pi。若 $Pi \le Pm$，则对 p_i 实行变异。具体操作步骤如下：

①随机选择 p_i 中的两个节点 c、d，以 c 为源节点 d 为目的节点，根据随机深度优先搜索算法搜索网络拓扑图，得到从 c 到 d 的新的随机路径；

②将得到的由 c 到 d 的新路径插入到原路径 p_i 中；

③如果插入后所得的 p_i 中有重复节点，则删除相同节点间的链路，确保变异后路径中无循环路由。

Step1-7 判断：执行选择、交叉和变异三项遗传操作后，生成下一代群体。转 Step1-3，根据适应度函数计算新生群体的个体适应度，适应度值小的个体将会被淘汰，适应度值大的个体将获得更多的生存繁殖机会，从而转 Step1-4 继续遗传操作，直至得到问题的优化解，转 Step2-1，继续蚁群算法部分。

2. 最优解求解

以第一阶段所得优化解初始化蚁群算法的信息素，继而更新信息素，递归迭代得到问题的最优解。第一阶段算法结束后得 N'（$N' \le N$）条优化路径，V'（$V' \subseteq V$）为优化解（即 N' 条优化路径）中所有节点构成的集合，V' 中有 n 个元素。此时，将 m 只蚂蚁置于这 n 个节点，开始算法的第二阶段。

Step2-1 信息素初值的设置：$\Gamma_{ij}(0) = \Gamma_C + \Gamma_G$。

设 $\Gamma_{ij}(t)$ 为 t 时刻从节点 i 到节点 j 的路径上积累的信息素强度，则 $\Gamma_{ij}(0)$ 表示初始时刻由 i 到 j 的路径上的信息素值，即信息素的初值；Γ_C 是根据具体求解问题规模给定的一个信息素常数，Γ_G 是第一阶段由遗传算法所得优化解转换的信息素值。在此，进行了两种算法的融合衔接，以遗传算法所得的优化解初始化蚁群算法的信息素值，很好的克服了基本蚁群算法初期信息素匮乏的缺陷。

Step2-2 转移概率的确定：蚂蚁 k（$k = 1,2,...,m$）在运动过程中，根据各条路径上信息素的强度决定转移方向，t 时刻位于某一节点 i 的蚂蚁 k 一次只能选择一个目标节点 j。t 时刻位于节点 i 的蚂蚁 k 选择节点 j 为目标节点的转移概率为 $P_{ij}^k(t)$，定义[47]

$$P_{ij}^k(t) = \begin{cases} \dfrac{[\Gamma_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{g \in V_k} [\Gamma_{ig}(t)]^\alpha [\eta_{ig}]^\beta}, j \in V_k \\ 0, j \notin V_k \end{cases}$$

式中，η_{ij} 表示由节点 i 转移到节点 j 的启发信息，即蚂蚁从节点 i 转移到节点 j 的期望程度，该启发信息由要解决的问题给出，本 QoSR 算法中取 $\eta_{ij} = \dfrac{1}{\cos t(i,j)}$；$V_k$ 表示蚂蚁 k 下一步允许选择的节点集合；α 为信息启发式因子，表示轨迹的相对重要性，反映蚂蚁在运动过程中所积累的信息素在指导蚁群搜索中的相对重要程度，其值越大，该蚂

蚁越倾向于选择其他蚂蚁经过的路径，蚂蚁之间的协作性越强；β为期望启发式因子，表示能见度的相对重要性，反映启发式信息在指导蚁群搜索过程中的相对重要程度，其值越大，则该状态转移概率越接近于贪心规则。

Step2-3 信息素更新模型：采用蚂蚁圈模型进行信息素更新，即一圈中只有最短路径的蚂蚁才进行信息素的修改增加。而后更新所有路径的信息素，转 Step2-2，重新计算转移概率，循环迭代得到最优解。

所有路径轨迹的更新方程[47]均采用：$\Gamma_{i,j}(t+1)=\rho\cdot\Gamma_{ij}(t)+\sum\Delta\Gamma_{ij}^{k}(t)$。

其中，$\Delta\Gamma_{ij}^{k}(t)$为蚂蚁k在路径(i,j)上留下的单位长度轨迹信息素数量，设Z_k为第k只蚂蚁在本次循环中所走的路径长度，则$\Delta\Gamma_{ij}^{k}(t)=\frac{Q}{Z_k}$（$Q$是一个常数）[48]；$\rho$（$0\le\rho<1$）表示轨迹的持久性。

3.3.3 算法实例

将本算法用于图 2.1 所示网络模型的服务质量路由计算。

对任一链路e考虑三个属性：$(delay,bandwidth,\cos t)$，分别表示延时（ms）、可用带宽（$MBps$）和花费，并且假定网络中有连接的两个节点之间可以相互进行数据传递。

首先，对$\forall e\in E$，定义以下三种度量：延时函数$delay(e)$、带宽函数$bandwidth(e)$和花费函数$\cos t(e)$：

$$delay(p(s,d))=\sum_{e\in p(s,d)}delay(e) \tag{3.1}$$

$$bandwidth(p(s,d))=\min\{bandwidth(e),e\in p(s,d)\} \tag{3.2}$$

$$\cos t(p(s,d))=\sum_{e\in p(s,d)}\cos t(e) \tag{3.3}$$

其中，$s\in V$为路由起点（即源节点），$d\in\{V-\{s\}\}$为路由终点（即目的节点），$p(s,d)$表示从源节点s到目的节点d的路由路径。公式（3.1）表示路径的延时等于该路径上所有链路的延时之和；公式(3.2)表示路径的带宽等于路径上链路带宽的最小值；公式(3.3)表示路径的费用等于路径上所有链路的费用之和。

其次，利用探测分组探测相邻路由器的延时、带宽和花费，接收邻居路由器的路由广播，将探测到的路由信息和接收到的路由信息记录，得到整个网络的链路状态信息，即图 2.1 所示。该网络拓扑包含 7 个节点，11 条链路，s 有连接建立请求，为源节点，目的节点为 d。QoS 约束条件为：最大延时 $Dp=9ms$，所需带宽 $Bp=2MBps$。

定义 $U \subseteq E \times E$ 是网络中单播情况下（源节点，目的节点）的集合。单播 QoS 路由选择即为在有向图 $G=(V,E)$ 中，为特定的 $u \in U$（$u=(s,d)$）找出花费最小，同时满足单播服务所要求的 QoS 请求的路径。如果算法能找到满足下列约束条件的路径，则路由请求 u 将被接受：

①延时约束：$delay(p(s,d)) \leq Dp$；

②带宽约束：$bandwidth(p(s,d)) \geq Bp$；

③花费约束：在所有满足上述两个条件的路径中，$cost(p(s,d))$ 最小。

第三，根据所需带宽过滤网络拓扑，将图 2.1 中不满足带宽要求的链路（2—3，4—d）删除（见图 3.3）。过滤后的网络拓扑模型为：

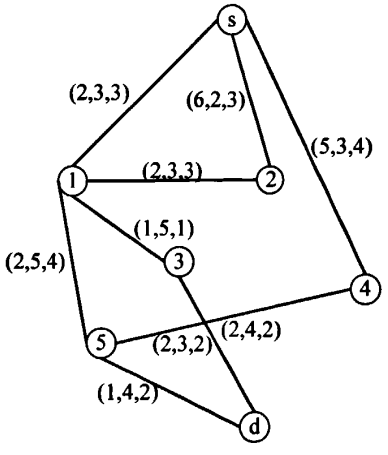

图 3.3　过滤后的网络拓扑及其参数

根据算法描述，对图 3.3 的网络拓扑模型进行计算。遗传-蚁群融合算法中遗传算法迭代次数固定为 10 代，变异概率 $Pm=0.01$；蚂蚁算法中各路径信息素初值 Γ_C 设为 60[49]，遗传算法求解结果转换的信息素值是经过路径加 2[20]，信息素更新 $\rho=0.8$，$Q=1000$。

在满足上述所有条件下，找到的符合约束条件的路径及相关参数值如表3.1所示：

表3.1 满足QoS约束的路径

候选路径	最大延时	最小带宽	花费
s—1—5—d	5	3	9
s—1—3—d	5	3	6
s—4—5—d	8	3	8

通过仿真计算，可以得到全局最优解，QoS最优路径为$s-1-3-d$，QoS度量值为$(5ms,3MBps,6)$，结果见图3.4（用粗线箭头表示）：

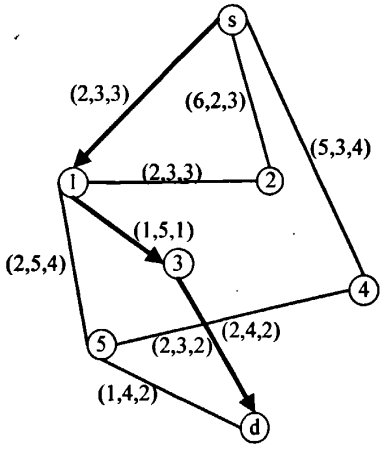

图3.4 遗传-蚁群融合算法所得最佳路由

若使用基本遗传算法，得QoS路径为$s-1-5-d$，QoS度量值为$(5ms,3MBps,9)$。显然，基本遗传算法所求解的精确度不如遗传-蚁群融合算法，两种算法所求得QoS路径及其相关参数值的比较见表3.2：

表 3.2 基本遗传算法与融合算法所得路径及其参数值比较

算法	所得路径	最大延时	最小带宽	花费
基本遗传算法	s—1—5—d	5	3	9
融合算法	s—1—3—d	5	3	6

3.4 本章小结

遗传-蚁群融合算法继承了遗传算法和蚁群算法的优点，实现了优势互补，即先利用遗传算法生成较丰富的初始信息素分布，再利用蚁群算法求精确解，在优化性能和时

间性能上都取得了很好的效果。其求精解效率优于遗传算法，时间效率优于蚁群算法，是用于解决多约束条件下路由选择的一种比较好的算法，既能满足多约束条件路由选择的需要，又能降低路由计算的复杂度，提高路由器的路由选择效率。

第四章 开放式最短路径优先（OSPF）协议

4.1 OSPF 网络拓扑结构

OSPF 对网络中的 AS 进行了进一步的划分，把一个网络或一系列相邻的网络分为编号区域（Area），一个区域的拓扑结构对于自治系统的其余部分是不可见的。这种信息的隐藏可以带来路由信息量的显著降低。同时，域内的路由只由域本身的拓扑结构决定，使其不受域外错误信息的影响。OSPF 还定义了一个特殊的域，称之为主干（Backbone），其编号为 0，所有的区域都与主干相连，主干负责向所有的非主干区域分发路由信息。主干在逻辑上必须是连续的，与其它域一样，在主干之外其拓扑结构是不可见的。图 4.1 表示了在分级的路由结构中，AS、区域主干和区域之间的关系。

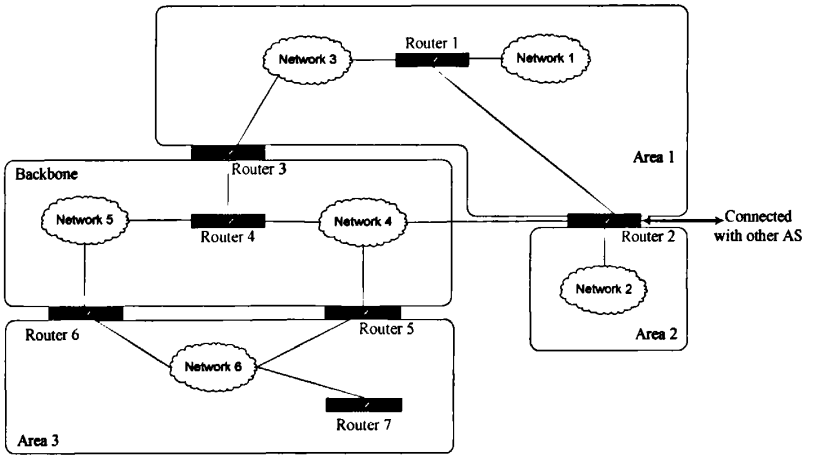

图 4.1 OSPF 中 AS、主干和区域间的关系

OSPF 属于网络层，其功能实现是建立在 IP 能够接收并向相邻站点发送分组的基础之上。所有 OSPF 分组都从一个公共报头开始，如图 4.2 所示：

版本号	类型	分组长度
路由器 ID		
域 ID		
校验和		认证类型
认证		

图 4.2 OSPF 分组报头格式

(1)版本号：用 8bit 标识所采用的 OSPF 路由协议的版本，设置为 2，表明当前版本

是 OSPFv2。

⑵类型：接下来的 8bit 定义了 OSPF 数据包类型。OSPF 数据包共有五种类型，分别用数字 1~5 标识。如图 4.3 所示：

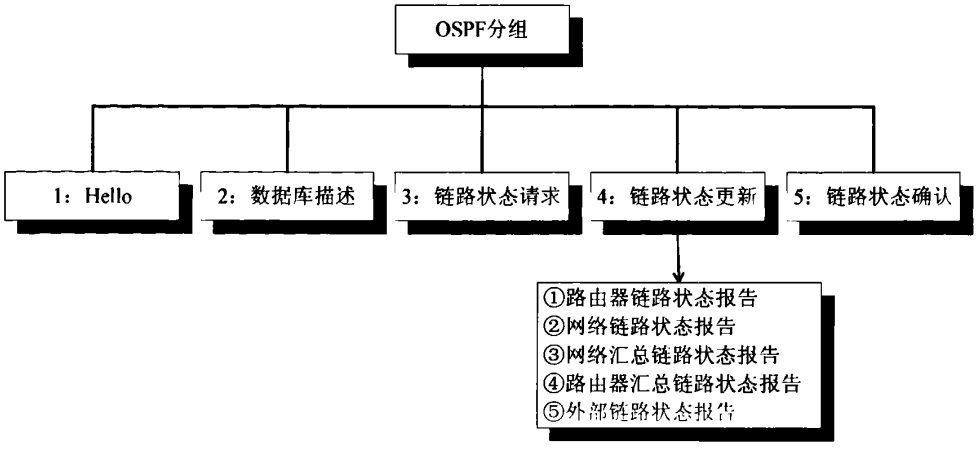

图 4.3 OSPF 分组的类型

⑶分组长度：该 16bit 定义了包括首部在内的整个数据报的总长度。

⑷路由器 ID：用于描述数据包的源路由器地址，以发送该分组的路由器 IP 地址来表示，占 32bit。

⑸域 ID：32bit 的区域标识，用于区分 OSPF 数据包所属的区域号，数值 0 为主干域保留。

⑹校验和：用于对整个分组进行差错检测，标记数据包在传递过程中有无误码。每个 OSPF 头包含一个 16bit 的校验和域，用于检查在传输过程中对报文造成的破坏。发送方对每个消息运行数学计算，然后把结果存储在这个域中；接收方对接收到的报文运行相同的算法并把结果与存储在校验和域中的结果进行比较。如果报文无损到达，两个结果应一样；如果结果不同，说明 OSPF 报文在传输过程中被破坏，接收方会简单地把受损报文丢弃。

⑺认证类型：16bit 的认证类型域标识 OSPF 认证的方式。OSPF 可以通过认证 OSPF 信息的发送者来防止导致假路由信息这样的攻击。OSPF 只定义了两种认证方式：0：不认证；1：简单的口令认证。

⑻认证：头中剩下的 32bit 携带的是认证数据，接收方利用此信息来确定信息的发送者。OSPF 允许网络管理员使用各种级别的认证：从无认证，到简单认证，到最强大的 MD（Message Digest：消息摘要）认证。基本结构中包含 OSPF 节点所需的用于决定报文是否应接收并作进一步处理，还是应丢弃的所有信息。在传输过程中受损的（校验

和指出这一点）及没有通过认证的报文会被丢弃。

4.2　OSPF 工作原理

OSPF 路由协议的工作流程如图 4.4 所示：

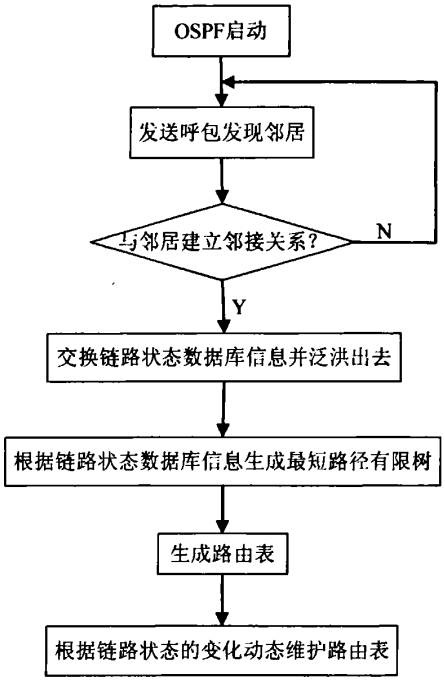

图 4.4　OSPF 工作流程图

4.2.1 建立邻接关系

建立邻接关系的目的是交互路由信息，并不是所有的邻居路由器之间都建立邻接关系。建立邻接关系包括两个主要步骤：Hello 协议和交互链路状态数据库信息。

使用 Hello 协议有两个目的：⑴寻找并发现邻居路由器；⑵在广播型与非广播型网路上选举指定路由器及其备份。

路由器每隔一个“Hello 间隔”发送一次 Hello 分组，内容包括链路上指定路由器的地址（如果还没有指定路由器就设为 0）以及备份指定路由器的地址（没有也设为 0）。邻居列表说明该路由器发现的与之相邻的路由器。Hello 协议的分组格式如图 4.5 所示。当路由器收到一个 Hello 分组时，如果发送这个分组的路由器还没有出现在本地路由器的邻居列表中（通过检查路由器 ID 是否匹配实现），那么路由器就可以认为发现了一个新的邻居，在把这个路由器加入邻居列表之后，路由器开始试图与对方建立双向的邻接

关系。

图 4.5 Hello 分组格式

如果分组可以在两个路由器之间的链路上双向流动，那么这两个路由器就可以开始交互路由信息了，通过查看对方路由器的邻居列表，很容易检测出双向连通性。如果本地路由器的 ID 未出现在对方路由器的邻居列表中，就意味着它还尚未接收到本地发送的 Hello 分组，那么就宣布这个连接为单向连接（1-way），不能用来路由；如果本地路由器的 ID 出现在对方路由器的邻居列表中，就建立了一个双向连接。而如果在路由器死亡间隔时间内都没有接收到来自该邻居路由器的 Hello 分组，就认为这个邻居已经死亡，将其从邻居列表中删除。

在建立双向连接之后，路由器首先进入一个“等待状态”，并且就这样保持一段与路由器死亡时间间隔相等的“等待间隔”。在此间隔内，路由器继续传送 Hello 分组，而不参与选举指定路由器和备份指定路由器的过程，只是将指定路由器和备份指定路由器的标识域置为 0，并监听进来的 Hello 分组，进而采用下面的措施为选举进行初始化准备：对于每个邻居，路由器记录其优先级及连接状态（单向或双向），并记录此邻居是否建议其自身作为指定路由器或其备份。只有达到双向状态的邻居才能认为可以参与选举。选举按如下过程进行：

⑴如果一个或多个邻居建议他们自身作为备份指定路由器，那么拥有最高优先级的将被选为指定路由器。在出现具有最高优先级的路由器有多个的情况时，拥有最大 ID 的路由器将被选中。

⑵如果没有邻居建议其自身作为备份，拥有最高优先级的邻居将被选中，或者，在具有最高优先级的路由器有多个的情况下，选择 ID 最大的一个。

(3)如果一个或多个邻居建议他们自身作为指定路由器，将选择拥有最高优先级的一个作为指定路由器。在出现优先级相等的情况下，拥有最大 ID 的路由器将被选中。

(4)如果没有邻居建议其自身作为指定路由器，备份将升级为指定路由器。由于指定路由器和其备份不能相同，必须重新执行步骤(1)和(2)。

此选举过程保证在可能的情况下，一定要能够选举出指定路由器，同时也必须保证，在指定路由器出现故障时，备份指定路由器能够顺利成为指定路由器。过程的步骤(1)和(3)可以确保：如果路由器已经就指定路由器及其备份达成一致意见，那么这一信息就会保持稳定。这也是使用一个“等待周期”的原因，一个刚刚启动或刚刚恢复的路由器不会立刻将自身升级为备份或指定状态，并在“等待周期”结束之前已经获知网络中其他所有路由器的状态。

交互链路状态数据库的目的是使双方的数据库“同步”。在点到点的链路上建立了双向连接之后，在网络链路中路由器与指定路由器之间，都由交互过程来完成数据库的初始同步，而接下来的维护数据库的工作则由“泛洪”过程来完成。OSPF 的交互过程是非对称性的，这个过程的第一步是从路由器中选择“主（master)”和“从（slave)”，当这个角色的选择达成一致之后，两个路由器就交换它们的数据库描述信息，即“数据库描述报文（DDP：Data Description Packet)”。数据库描述报文并不包含完整的数据库信息，它只给出了概要，即数据库中每一行的标题。其格式如图 4.6 所示：

<table>
<tr><td colspan="7">OSPF 分组报头，类型=2（Data Description）</td></tr>
<tr><td>全 0</td><td>全 0</td><td>选项</td><td>全 0</td><td>I</td><td>M</td><td>M/S</td></tr>
<tr><td colspan="7">DD 序列号</td></tr>
<tr><td colspan="7">链路状态首部（20 字节）</td></tr>
<tr><td colspan="7">邻居路由器</td></tr>
</table>

图 4.6　数据库描述分组格式

想要启动交换过程的路由器首先发送一个空的 DDP。其中 I、M 和 M/S 位都设为 1，DD 序列号被设置成一个随机值（一般使用当前的时钟)。然后，另一个路由器发送一个“确认”分组表示同意在这一交换期间扮演“从属路由器”的角色。该“确认”分组装载有相同的序列号，I 位设置为 1，M/S 位设置为 0（从属路由器)。在没有得到确认的

情况下，初始分组将每隔“重传间隔”重复发送一次。解决冲突的办法可利用一种简单的约束消除（tie-breaking）算法：如果一个路由器正在等待一个确认时却收到一个请求，那么它就将发送地址与其自身的地址进行比较，如果发送地址大于其自身的地址，它就接受从属路由器的角色并确认这个分组；否则，它就忽略所接收的分组，因为他自身的分组稍后将会被其它路由器所确认。

一旦主从角色分好，就可以开始进行非对称的交换。主方路由器在一系列 DDP 中发送其数据库里记录的描述，I 位被设置成 0，M/S 被设置成 1，M 位也设置为 1（最后一个分组除外）。对这些分组进行序列编号，一次发送一个。每个分组发送完毕以后，从属路由器将发送一个确认分组，即一个 DDP，其中含有相同的序列号，但 M/S 位被设置成 0，确认分组带有从属路由器数据库里记录的描述。如果确认在重传间隔内未被收到，主路由器将重复发送它的分组；如果从属路由器收到一个与前面分组序列号相同的分组，它就重复前面的确认。主路由器的重传机制保证了主路由器与从路由器的分组都能被正确接收。

当主路由器传送它最后一个记录描述时，把 M 位设置为 0。如果从路由器仍然有记录要传送，它就继续发送 M 位置为 1 的确认。主路由器将继续发送 M 位被置 0 的空描述分组并接收确认，直到最后接收到一个 M 位被置为 0 的确认。至此，交换结束。

在交换期间，主路由器与从路由器将处理它们在分组以及其确认中发现的链路状态记录描述。首先核实是否有一个与他们数据库中的“类型”、“广播路由器”或“链路状态 ID”相同的记录，并利用比较算法来检测这个记录是否有一个更大或相等的序列号。如果路由器需要这个链路状态记录，它们就将这个记录描述放入一个“请求记录”的列表之中。在交换过程中和交换结束之后，它们通过链路状态请求分组（LS Request：Link State Request）来请求获得这些记录。链路状态请求分组如图 4.7 所示：

OSPF 分组报头，类型=3（Link State Request）
链路状态类型
链路状态 ID
广播路由器
……

图 4.7　链路状态请求分组

该分组中包含一组链路状态记录的标识符，每个链路状态记录都可以由链路状态类

型、链路状态 ID 和广播路由器惟一确定，这三个数据用来作为这个记录的标识符。当收到这类请求时，路由器将使用与扩散新的记录相同的过程来发送一组链路状态更新。每当接收到链路状态更新时，这个记录描述就从请求记录列表中删除。

4.2.2 链路状态数据库

在相同域的 OSPF 路由器都力图通过上述过程，形成一个“同步”的数据库，这个数据库由“链路状态报告”（LSA：Link State Advertisement）组成，这些记录代表网络的拓扑结构并用来计算最短路径。链路状态报告的类型有五种（图 4.2）：路由器、网络、网络汇总、路由器汇总和自治系统外部链路。各种类型的链路状态报告都有相同的“首部”，如图 4.8 所示；链路状态类型与链路状态 ID 的关系如图 4.9 所示。

链路状态生存时间	选项	链路状态类型
链路状态 ID		
广播路由器		
链路状态序列号		
链路状态校验和	链路状态记录长度	

图 4.8　链路状态报告首部

类型值	链路状态类型	链路状态 ID
1	路由器链路	产生该 LSA 的源路由器的 IP 地址
2	网络链路	指定路由器的 IP 地址
3	网络汇总链路	目的网络的 IP 地址
4	路由器汇总链路	所描述的自治系统边界路由器的 IP 地址
5	外部链路	目的网络的 IP 地址

图 4.9　链路状态类型与 ID

1. 路由器链路状态报告：通知一个路由器的所有链路，其分组格式如图 4.10。第一个字节中有 3 位，被称为 V、E 和 B。V 只有在这个路由器是一个虚拟链路的端点且路由器所连接的网络是这个虚拟链路的中转网络时才设置为 1；E 和 B 表示路由器是否是一个区域边界路由器和自治系统边界路由器。每个链路都有链路 ID、链路数据和链路类型，类型可以取 3 种数值：(1)如果此链路是通往另一个路由器的点到点链路，则链路 ID 就是这个路由器的邻居路由器的 ID，而且链路数据是这个路由器的接口 IP 地址。(2)如果此链路连接到中转网络，则链路 ID 就是这个指定路由器接口的 IP 地址，而且链路

数据是这个路由器接口的 IP 地址。(3)如果此链路连接到一个末梢网络，则链路 ID 就是这个 IP 子网的编号，而且链路数据是对应的网络或子网的掩码。

0	V	E	B	0	链路数
链路 ID					
链路数据					
链路类型			TOS 号		TOS0 的度量
TOS=x			0		距离
……					

图 4.10　路由器链路状态报告

2．网络链路状态报告：宣告连接到一个网路上的链路，其分组格式如图 4.11 所示。内容包括 32 位的网络或子网掩码，连接的路由器这个重复的字段定义了所有与指定路由器建立邻接关系的路由器 ID。邻居路由器的数量可以从记录的长度推算出来。

网络掩码
连接的路由器（Attached Router）
……

图 4.11　网络链路状态报告

3．汇总链路报告：网络汇总链路与路由器汇总链路是由区域边界路由器产生和发布的，用来报告到达域内的网络和区域边界路由器的路径，每个目的站点都有一个单独的链路状态报告。网络汇总链路报告与路由器汇总链路报告具有相同的分组格式，如图 4.12 所示：

网络掩码		
TOS=x	0	距离
……		

图 4.12　汇总链路状态报告

4．外部链路报告：由自治系统边界路由器发布，用来生成到达自治系统外部的网络的路由，由外部网关协议（EGP）获取，其分组格式如图 4.13 所示。如果设置了 E 位，则表明该 TOS 的距离值与内部距离值不可比，并且应该看作“大于任何内部路由器”；当 E 设为 0 时，距离值可以加到内部路径的开销中去，以计算到达目的站点的路径的开销。“发送地址”（Forwarding Address）指明发往指定目的站点应该使用的目的地

址；“外部路由标签”是一个 32 位域，由边界路由器用来交换有关路由信息。

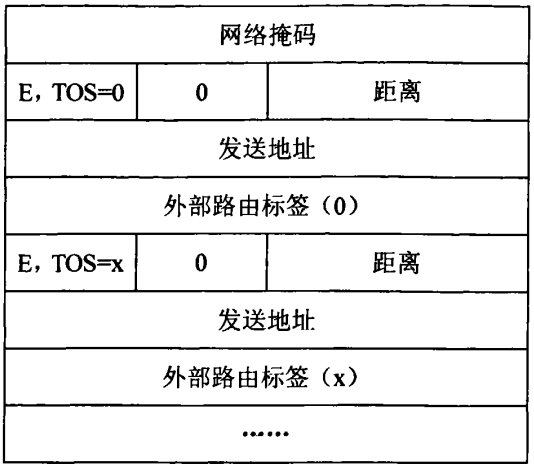

图 4.13　外部链路报告

链路状态报告分别描述了域内、域外和自治系统外部的链路状态，在获取了相应的链路状态报告之后，就可以根据这些报告的内容，按照链路状态算法计算出相应的路由表项了。链路状态报告的产生是一个与网络、邻居路由器的状态密切相关同时又是并行的过程。一个路由器为它所连接的每一个域产生一个路由器链路状态报告，该报告描述了这个路由器与这个域的所有链路的状态集合。这个链路状态报告将通过泛洪过程被扩散到整个域内，但不会超出这个域的范围。

4.2.3 泛洪过程

当一条链路发生状态变化时，与该链路相对应的路由器将发布新版本的链路状态报告，各种不同类型的链路状态报告被装载在链路更新分组中，如图 4.14 所示：

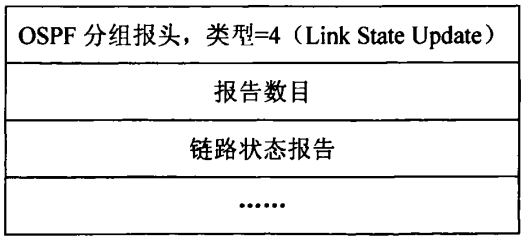

图 4.14　链路状态更新分组

跟在 OSPF 报头后面的是报告数目，用 32bit 定义了一个分组可以通知多少条链路的状态，接着是链路状态报告本身。事实上，只有链路状态更新分组中才有完整的链路状态报告，因此路由信息最终都要通过链路状态更新分组来传递。

当收到一个新的链路状态报告时，这个报告将被安排在其他所有接口进行传送。无论以怎样的方式，接收者都会向发送这一更新分组的路由器确认这个报告。为保证扩散

过程的可靠性，该路由器会以一定间隔重传它的更新信息，直到收到确认为止。链路状态确认分组具有公共的 OSPF 首部和普通的链路状态报告首部，其格式如图 4.15 所示：

OSPF 分组报头，类型=4（Link State Acknowledgement）
链路状态报告首部
……

图 4.15 链路状态确认分组

4.3 链路状态路由算法

L-S（Link-State，链路状态）算法又叫 SPF 算法，是 OSPF 路由协议的基础，SPF 算法即 Dijkstra 算法。按照算法的要求，路由器中路由表依赖于一张能表示整个网络拓扑结构的无向图$G(V,E)$。在图G中，节点V表示路由器，边E表示连接路由器的链路，一般把图G称为 L-S 图。在信息一致的情况下，所有路由器的 L-S 图都应该是相同的，各路由器的路由表通过 L-S 图计算。在 Dijkstra 算法中作如下定义：

N：网络中所有节点的集合；S：源节点；M：已由算法归并的节点的集合；len（i，j）：节点 i 和 j 之间链路的权值，当 i 与 j 间无连接时 len（i，j）=∞；C（n）：算法求得的当前从 S 到 n 的最小花费路由的花费。算法具体描述如下[50]：

```
Begin
    M={S};
    ∀节点n∈（N-S），置C（n）= len（S，n）;
    S到n的最短路径=S到n路径；//S与n直接相连时
    do while M≠N
        ∃节点w∈（N-M）且C（w）最小；
        M=M∪{w};
        ∀节点n∈（N-M），置C（n）=Min（C（n），C（w）+len（w，n））;
        if （C（w）+len（w，n））<C（n）
            从S到n的最短路径=从S到w的最短路径加上从w到n的路径；
        end if
    end do
End
```

算法中的每一次循环都得到了当前从源节点到各目的节点的路径及耗费，而在 M 中是已确定了最短路径的目的节点的集合（即不需在新的循环中修改的目的节点）。这样，直到所有的目的节点到源节点的最短路径都确定了，算法即结束。在这个算法中使用了“最佳原理”：如果从节点 A 到节点 B 的最佳路由经过节点 C，那么在该路由上，从 C 到 B 的那一段也同时是从 C 到 B 的最佳路由；从 A 到 C 的那一段也同时是 A 到 C 的最佳路由。因为，若从 C 到 B 还存在一条更由路径 L，那么 L-A 的长度小于 CBA 的长度，与 CBA 是最优路径矛盾，同理可知 A 到 C 那一段也是 A 到 C 的最佳路由。

该算法的主要优点在于：⑴L-S 信息向全网广播，各路由器都使用相同的原始数据进行路径计算，保证了各路由器 L-S 图的一致性；⑵各路由器在本地进行路由计算，路由信息不会反过来对原路由器发生作用，使路由的收敛性得到了保证；⑶广播的路由信息只与该路由器相连的链路数目和状态有关。

但是，由于 Dijkstra 算法本身的局限性使得 OSPF 协议不支持多约束 QoS 路由。因此，面对网络上越来越多的多媒体类应用，现有 OSPF 路由协议已逐渐不能完全满足业务流的 QoS 请求，对 OSPF 协议完善改进使其支持多约束 QoSR，是很多学者着力研究和解决的问题。在了解 QoS 路由机制和 OSPF 协议的工作原理之后，本文将在不改变 OSPF 现有工作流程的基础上，实现基于融合算法的多约束 OSPF-QoSR。

4.4　本章小结

本章详细分析了 OSPF 协议的工作原理及路由算法，阐明对 OSPF 进行 QoS 扩展的必要性，指明本课题的核心任务。

第五章 融合算法 OSPF-QoSR 的实现

5.1 OSPF 协议上的 QoS 扩展

5.1.1 扩展前提

在 OSPF 协议上进行 QoS 扩展的目标是提高 OSPF 对 QoS 业务流的处理能力，同时最低限度地减少对协议本身的修改[51]。假设 QoS 路由计算是控制在一个自治域范围内的，并且要求这个自治域中的所有路由器都使用支持 QoS 功能的 OSPF 协议。在这个基础上，OSPF 协议上的 QoS 扩展有如下四点思路：

⑴路由预先计算。在数据报到来之前计算出到达目的节点的所有有效路径。通过提前计算，产生一个区别于 OSPF 路由表的独立的 QoS 路由表（详见 5.1.2），从而尽可能减少路由器的计算负载，这在大型网络路由中尤其适用。

与预先计算相对应的是按需计算，即根据数据报流量需要计算最佳路由，若无 QoS 请求，则保持正常路由计算。按需计算要求网络结构相对稳定，且仅适应于单个 OSPF 域[52]。本文使用了应用范围更为广泛的预先计算。

⑵采用分布式路由策略。QoS 路由表中只需保存当前网络状态相关信息和下一跳节点，不需要考虑整个路由路径，这样可以避免对路由表的数据结构进行太大的改动。

⑶选择适合于业务流 QoS 请求的路径不仅仅依赖于业务流本身的特性，还需要网络资源的可利用信息。对算法而言，应该能够统计所有已分配的网络资源。为了适应不同 QoS 的要求，需要改进 LSA 和链路状态数据库以包含网络资源信息，即 QoS 度量（如链路延时、可用带宽、花费等）。

链路延时：和时间相关的度量变量，其变化信息需要通过扩展的 LSA 扩散出去；

可用带宽：和速率相关的度量变量，其变化信息需要通过扩展的 LSA 扩散出去；

跳数：描述网络费用的变量，其变化信息直接反映在路径选择算法中，不需要扩展的 LSA。

⑷将网络资源信息考虑在内，用遗传-蚁群融合算法进行 QoSR。

根据网络拓扑结构图和每条边的 QoS 度量，利用融合算法进行路径计算。预先计算出所有可能的 QoS 路径，生成 QoS 路由表，对于一个特定的业务流，查询 QoS 路由表，从路由表中获取转发信息。

5.1.2 QoS 路由表

QoS 路由表与原始的 IP 路由表不同：IP 路由表是一个包含目的网络地址、网络掩码、下一跳路由器地址等字段的队列，每个目的地址或目的子网对应一条路由表项；而 QoS 路由表是一个矩阵，每一行对应一个目的地址或目的子网，列对应 QoS 路径的跳数。

QoS 路由表可以表示成一个 $K \times H$ 的矩阵，K 是网络拓扑结构图中所有目的节点的数目，设 N 为一个 OSPF AS 中所有路由器的数目，则 $K = N - 1$；H 是一条路径中所允许的最大可能跳数，其取值可以是网络的直径。每个路由器都维护这样一个 QoS 路由表，并根据该路由表进行选路，从而得到“最优路径”。

矩阵 QoS 路由表的表项 (k,h) 包含两个域 $(k = 1,2,...,K; h = 1,2,...,H)$：

(1) $bw(s,d)$：源节点 s 到目的节点 d 最多 h 跳的路径上的最大可用带宽；

(2) *neighbor*：该域表示的是从源节点到目的节点的可用带宽为 *bw* 的路径上，邻接源节点的下一跳节点。在整个 OSPF 域中，每个路由器节点都有一个唯一的编号（ID）用以标识自己，在 *neighbor* 域中，存放的就是下一跳路由器节点的编号。

需要说明的是，QoS 路由表项中并没有包含链路延时这一度量，只是考虑了带宽请求和跳数。这是因为 A. Orda 和 A. Sprintson 证明，当一个业务流有多个 QoS 请求时，可以把它们转换为带宽要求[10]，Guerin 等则曾提出用数学等式把延时约束映射为带宽约束[53]。此外，也有试验数据表明，在带宽满足需要的情况下网络延迟一般都是比较小的，可忽略不计[54]。

5.2 实现过程

5.2.1 改进的 OSPF 报文

1. OSPF Option（选择）域扩展

OSPF 分组的 Hello 报文，数据库描述报文等 5 种报文中都包含有 Options（选择）域字段，标识了 OSPF 路由器所能提供的各种可选服务并用以描述其广播路由的能力。如图 5.1 所示[55]：

*	*	DC	EA	N/P	MC	E	T

图 5.1 OSPF 分组的 Option 域

T-bit 表示 TOS 路由能力，为向后兼容先前的 OSPF 版本。扩展后的 T-bit 用来表示路由器的 QoS 能力，并重新命名为 Q-bit。在 Hello 分组中，该比特表示路由器是否支持 QoS 路由；在路由器 LSA 或概括链路 LSA 中设置该比特，说明数据包中含有 QoS 域；在网络 LSA 中，该比特表明通告中描述的网络是否支持 QoS。对于那些不能识别 Q-bit 的路由器，会忽视这个参数，不至于产生负面的效应；如果路由器能够识别 Q-bit，当收到路由器 LSA 或概括链路 LSA 时，可以从中分析得到 QoS 参数并进行路由计算。

2. OSPF TOS（Type of Service，服务类型）域扩展

OSPF-QoSR 的“QoS 编码”在语义上兼容并扩展了 OSPF 的“TOS 编码”，除了保留 4 位长的 OSPF TOS 定义，QoS 资源编码还启用了各 LSA 中 TOS 域的第 5 位，即有 32 种组合。图 5.2 对比了扩展前后 LSA 的 TOS 域：

图 5.2 LSA 的原 TOS 域与扩展后的 TOS 域

此外，考虑到 OSPF 路由器并不能理解 OSPF-QoSR TOS 域的所有位，OSPF-QoSR 目前只定义了带宽和延时度量。这样，即使 OSPF 路由器不考虑最高有效位，带宽和延时仍可以分别被映射为原 TOS 定义中的“最大吞吐量”和“最小延时”，达到了对现有 OSPF 协议及其实现影响最小的目的[56]。图 5.3 表示了 QoS 编码与 TOS 编码的对应关系：

OSPF encoding	TOS values	OSPF encoding	QoS encoding values
0	0000 正常	32	10000
2	0001 最小花费	34	10001
4	0010 最高可靠性	36	10010
6	0011	38	10011
8	0100 最大吞吐量	40	10100 带宽
10	0101	42	10101
12	0110	44	10110
14	0111	46	10111
16	1000 最小延时	48	11000 延时
18	1001	50	11001
20	1010	52	11010
22	1011	54	11011
24	1100	56	11100
26	1101	58	11101
28	1110	60	11110
30	1111	62	11111

图 5.3 OSPF TOS 编码与 QoS 编码

3. OSPF Metric（度量）域扩展

OSPF 包的 Metric 域仅提供了 16 位来编码度量值，如果采用可用资源的线性表示，链路支持带宽到 $GByte/s$ 是不可能的（因为线性编码可提供的最大带宽为 $2^{16}-1=65535\approx 64KByte/s$，远远小于 $GByte/s$）。为了这个解决问题，通常采用指数编码，这就需要选择合适的隐含基数及尾数和指数的位数。

⑴带宽编码规则：采用二进制 0-1 编码，用 16 位表示可用带宽，其中 3 个最高有效位是指数，剩下的 13 位做尾数，假设基数为 8，那么可用带宽的最大值为 $(2^{13}-1)\times 8^{x} Byte/s$（$x$ 即为 3 个高位表示的指数值，$x\in\{0,1,2,...,6,7\}$），指数值不同，编码所表示的带宽最大值亦不同。

假设一条链路的可用带宽为 $8Gbit/s$，因为 $8Gbit/s=8\times 1024^3 bit/s=1024^3 Byte/s$，而 $1024^3=4096\times 8^6$，按上述编码规则，$x=6$，则高 3 位编码为 110；又因为 $4096=2^{12}$，故后 13 位编码为 0100000000000。所以，该 OSPF 包 Metric 域的 16 位带宽编码为：1100100000000000。

⑵延时编码规则：网络延时以微秒（μs）为单位，其编码规则类似于带宽的指数编码规则。同样以高 3 位作为指数位数，后 13 位为尾数，假设基数为 4，那么延时最大值为 $(2^{13}-1)\times 4^{x}\mu s$。

4. 链路状态更新

在 LSA 中添加 QoS 信息，链路状态的改变要比原始 IP 网络频繁得多，这意味着需要对 LSA 的发送机制做一些改动。

假定所有路由器都能够获取当前链路的状态信息（可用带宽、链路时延等），并能够将这些信息通过 OSPF 的扩展机制扩散到网络中去，如何扩散和何时扩散将是很重要的问题。理想状况下，我们希望每一个路由器都能拥有网络上所有链路最新的可用带宽信息，这样才能保证其选择路径时做出最正确的判断。然而，无论把状态更新的频率设置多快，也不可能达到理想的状态。过于频繁的状态更新会严重浪费网络可用带宽，而且由于网络收敛问题，信息总是会有延迟的[54]。

一种操作是周期性地发送更新消息（Period based Updating，PB），发送周期可以根据网络和路由器的负载决定，设置一个固定的超时定时器。这种方法的主要缺点是一条

链路带宽的变化在一个周期内不能正确反映在网络中，尤其是当超时定时器设置较大时，可能导致错误的路由选择。

为了解决这个问题，可以采用一种综合性的更新机制——触发更新操作，在尽可能最少的网络负载下满足 QoS 对链路状态信息的要求。理想的情况下，路由器应该能够获知网络中所有链路的可利用带宽的当前值，这样的话会带来非常多的更新消息。通常在更新频率与网络信息的可信度之间采用折中的方法，当链路状态信息发生显著变化时才触发更新消息。所谓的“显著”可以是绝对值或者是相对值。绝对值是将链路状态分成几个等级，超过边界值即认为链路状态发生了变化：如果一个路由器发送过一条声明带宽为 B 的 LSA，设边界值为 $B/2$ 和 $2B$，那么当前可用带宽 $Bc < B/2$ 或者 $Bc > 2B$ 时，会再次发送 LSA；相对值是在链路状态变化超过了预先设定的阈值 T（百分比）时，触发更新：Bc 表示当前可用带宽值，Bn 表示最新通告的可用带宽值，当 $|Bn - Bc| / Bn > T$（$Bn > 0$）时触发更新，当 $Bn = 0$ 时，总是触发更新。采用触发更新时，周期性更新也可以同时采用，用以限制发送更新的间隔时间[51]。

5.2.2 基于融合算法的 OSPF-QoSR

基于遗传-蚁群融合算法的 OSPF-QoSR，即在网络拓扑中寻找一条跳数最少（花费最小），并且在满足带宽要求的前提下提供尽可能大的可用带宽的最优路径。讨论算法之前，假设：

⑴进行路由选择之前，已知到达目的节点 d 的业务流 QoS 带宽请求 Bp；

⑵预处理：根据网络拓扑图中各链路属性做出判断，过滤掉图中所有不满足带宽约束的链路（即删除所有 $bandwidth(p(s,d)) < Bp$ 的边）；

⑶过滤后的图可能不是连通图，如果源节点和目的节点不在同一连通子图中，路由选择是无意义的。所以，本算法还需假定经预处理后的图是连通的，并且所有的链路都已经满足带宽约束（$bandwidth(p(s,d)) \geq Bp$）。

1. 算法思想

以源节点 s 为例调用该算法，计算到达所有目的节点的本地 QoS 路由表，查找最优路径：

Step1 初始化 QoS 路由表。将表中所有项的 bw 域设为 0，$neighbor$ 域设为 Null。

Step2 $h=1$，即 QoS 路由表矩阵的第一列（一跳路径）。bw 域设为从源节点 s 出发

与 s 直接相连的边上的可用带宽值，*neighbor* 域设为路由器 s 的邻居路由器 ID。与 s 非直接相连的目的节点的相关表项依然为空，表示一跳无法到达。

Step3 $h=i(i=2,3,...,H-1)$ 时，利用遗传-蚁群融合算法计算 QoSR（详见 3.3）。

①从源节点 s 出发，随机深度优先搜索到每一目的节点 d 跳数为 h 的可能路径，构成初始群体；

②根据适应度函数，评估初始群体中每一个体（即随机深度优先搜索所得可能路径）的适应度，适应度值小的个体会被淘汰，对适应度值大的个体转③；

③进行遗传操作，生成下一代群体，转②，重新评估个体适应度；

④循环迭代得到问题的优化解，以优化解初始化蚁群算法的信息素；

⑤计算转移概率，更新信息素，递归迭代得最优解。

修改 QoS 路由表项的第 h 列：$bw(s,d)=\max\{bw(h),bw(h-1)\}$，其中 $bw(h)$ 表示从源节点 s 到目的节点 d 跳数为 h 的路径上的可用带宽，第 h 列的 bw 域即为从 s 到 d，h 跳和 $h-1$ 跳路径上可用带宽的较大值；*neighbor* 域为所选路径上与 s 相邻的下一跳路由器 ID。

Step4 $h=i+1$ 时，重复 Step3，直到一次计算所得路径均为空时结束，这样每个路由器都得到一个所有目的节点的最多 H 跳的 QoS 路由表。一个给定目的的业务流向路由器提出带宽请求，转 Step5 查询路由表，获得转发信息。

Step5 路由表查询。假定有一个业务流请求，目的为 d，带宽请求为 B。查询算法如下：在 QoS 路由表矩阵中查询目的为 d 的行，如果不存在，本算法结束，转而查询 OSPF 路由表，进行正常的数据报处理与转发；否则，从该行的第一项开始逐项查询，如果 $bw(s,d)<B$，即现有网络状况不能满足请求，则阻塞该业务流；如果 $bw(s,d)\geq B$，则返回该路径（跳数为 h，下一跳为 *neighbor*）。若存在多条等价路径，可在路径选择中采用负载均衡的方法[51]。

2. 算法伪代码描述

```
struct tab_entry   //QoS 路由表项
{
  double bw;
  int neighbor;
};
```

```
struct tab_entry RT[K-1][H-1];  //QoS 路由表矩阵
OSPF-QoSR(G,V,E,s,H)
{
  for(int k=0;k<=K-1;k++)  //初始化 QoS 路由表
    for(int h=0;h<=H-1;h++)
    {
      RT[k][h].bw=0;
      RT[k][h]. neighbor=Null;
    }
  for(源节点 s 的所有直接相邻节点 s')  //修改 QoS 路由表矩阵第一列
  {
    RT[s'][0].bw=B(s,s');
    RT[s'][0]. neighbor=ID(s');
  }
  for(int h=1;h<=H-1;h++)  //循环选路生成 QoS 路由表
  {
    for(int k=0;k<=K-1;k++)
    {
      RT[k][h].bw=RT[k][h-1].bw;
      RT[k][h].neighbor=RT[k][h-1].neighbor;  //逐行复制第（h-1）列表项到第 h 列
       {
          调用 GA-AA 算法，搜索所有从 s 到 k、跳数为（h+1）的路径，循环迭代得当前网
          络状况下的最优解;
          if(list[h+1]==Null)  //若从 s 到 k 跳数为（h+1）的路径不存在，清空表项
          {
            RT[k][h].bw=0;
            RT[k][h].neighbor=Null;
          }
          else  //得出从 s 到 k 跳数为（h+1）的最优路径，修改路由表项
            if(RT[k][h-1].bw<B(s,k))
            {
              RT[k][h].bw= B(s,k);
              RT[k][h].neighbor=ID(s—>nextnode);
            }
            else
            {
               RT[k][h].bw=0;
              RT[k][h].neighbor=Null;
            }
      }
    }
    for(int k=0;k<=K-1;k++)  //若第 H-1 次循环计算所得路径均为空时，跳出循环
      if(list[k]==Null)
        h=H;
```

```
    }
    Find(RT,s,d,B)  //路由表查询函数
    {
      for(int k=0;k<=K-1;k++)
      if(d==k+1)
        for(int h=0;h<=H-1;h++)
          if(RT[d][h].bw>=B)
            return RT[d][h].neighbor;
          else
            break;
      else
       break;
    }
}
```

3. 算法实例

同样以图 2.1 所示网络模型为例，用基于遗传-蚁群融合算法的 OSPF-QoSR 计算从源节点 s 到各目的节点的最优路径。

Step1 初始化 QoS 路由表项，*bw* 域设为 0，*neighbor* 域设为 Null。

Step2 $h=1$：计算从 s 到网络中其余各节点最大跳数为 1 的最优路径。只能得出与 s 直接相邻的节点 1、2、4 的最优路径，分别为 $s-1$、$s-2$、$s-3$，填入 QoS 路由表中的表项分别为（3，1）、（2，2）、（3，4），其余各项不变。

Step3 $h=2$：计算从 s 到网络中其余各节点最大跳数为 2 的最优路径。首先复制第一列表项信息到第二列，然后逐行修改路由表项。对目的节点 1 而言，最大跳数为 2 的路径只有一条，为 $s-2-1$，此时 $B(s,1)=2$，小于从 s 到节点 1 一跳时的可用带宽 3，故最优路径应为 Step1 计算出的路径，路由表项置为（0，Null）；对目的节点 2 而言，最大跳数为 2 的路径为 $s-1-2$，$B(s,2)=3$，大于一跳时的可用带宽 2，故修改路由表项为（3，1）；对目的节点 3 而言，最大跳数为 2 的路径为 $s-1-3$，$B(s,3)=3$，修改表项（0，Null）为（3，1）；依据算法，目的节点 4、5、d 对应的表项应分别为（0，Null）、（3，1）、（0，Null）。

Step4 同理，可计算出 $h=3$、$h=4$ 时从 s 到各目的节点的最优路径。当 $h=5$ 时，从 s 到各目的节点跳数、为 5 的最优路径均不存在，计算结束。由此得出 s 的 QoS 路由表，如表 5.1 所示：

表 5.1 源节点 s 的 QoS 路由表

K \ H	1	2	3	4	5
1	(3，1)	(0，Null)	(0，Null)	(0，Null)	(0，Null)
2	(2，2)	(3，1)	(0，Null)	(0，Null)	(0，Null)
3	(0，Null)	(3，1)	(0，Null)	(0，Null)	(0，Null)
4	(3，4)	(0，Null)	(0，Null)	(0，Null)	(0，Null)
5	(0，Null)	(3，1)	(0，Null)	(0，Null)	(0，Null)
d	(0，Null)	(0，Null)	(3，1)	(0，Null)	(0，Null)

Step5 假定有一个业务流请求，目的为 d，带宽请求为 $B=2MBps$，查询 QoS 路由表，求由源节点 s 到 d 的最优路径：首先按行查询，找出编号为 d 的行；然后按列查询，找出 $bw(s,d)\geq B$ 的项 $RT[d][3]=(3,1)$，即找到满足该条件的最优路径的可用带宽为 3，下一跳节点为 1，继续查询节点 1 的 QoS 路由表，依此法类推，即可得到满足该业务流请求的最优路径。

5.3 本章小结

本章详细阐述了在 OSFP 协议上进行 QoS 扩展的具体实现方案，包括 QoS 路由表的概念、OSPF 报文中相关域的扩展及添加 QoS 信息后的 LSA 发送更新机制的改变策略等，并运用第三章讲述的遗传-蚁群融合算法进行 OSPF-QoSR。

第六章 OPNET 仿真环境介绍及仿真结果分析

6.1 OPNET 概述

6.1.1 OPNET 简介

OPNET Modeler 是当前业界领先的网络技术开发环境，以其无与伦比的灵活性运用于设计和研究通信网络、设备和协议。Modeler 为开发人员提供了建模、仿真及分析的集成环境，大大减轻了编程和数据分析的工作量，加速了研发过程，现已被广泛应用于大型网络的开发。

Modeler 所能应用的领域包括端到端网络结构设计（End to End Network Architecture Design）、系统级仿真（System Level Simulation for Network Devices）、新协议开发和优化（Protocol Development and Optimization）、网络应用优化及调配分析（Network Application Optimization and Deployment Analysis）等。Modeler 采用分层建模方式（Hierarchical Network Modeling）：从协议间关系看，节点模块建模完全符合 OSI 标准，从上到下依次是应用层→TCP 层→IP 层→IP 封装层→ARP 层→MAC 层→物理层；从网络层次关系看，提供了三层建模机制，最底层为进程（Process）模型，以状态机来描述协议，其次为节点（Node）模型，由相应的协议模型构成，反映设备特性，最上层为网络模型。三层模型和实际的协议、设备、网络完全对应，全面反映了网络的相关特性。Modeler 是面向对象建模的（Object-Oriented Modeling），每一类节点开始都采用相同的节点模型，再针对不同的对象，设置特定的参数。在 Modeler 中各种协议的代码都是完全公开的（Total Openness），代码的注释也非常清楚，使得用户更容易理解协议的内部运作。采用混合建模机制，把基于包的分析方法和基于统计的数学建模方法结合起来，既可以得到非常细节的模拟结果，也大大提高了仿真效率。在物件拼盘中，Modeler 包含了详尽的模型库，包括：路由器、交换机、服务器、客户机、ATM 设备、DSL 设备、ISDN 设备等，随着 OPNET 版本的提高模型库也不断增加。

6.1.2 OPNET 工作流程

图 6.1 显示了使用 Modeler 进行仿真的流程，给出了从建立模型、运行仿真到收集最后结果的一般工作流程。

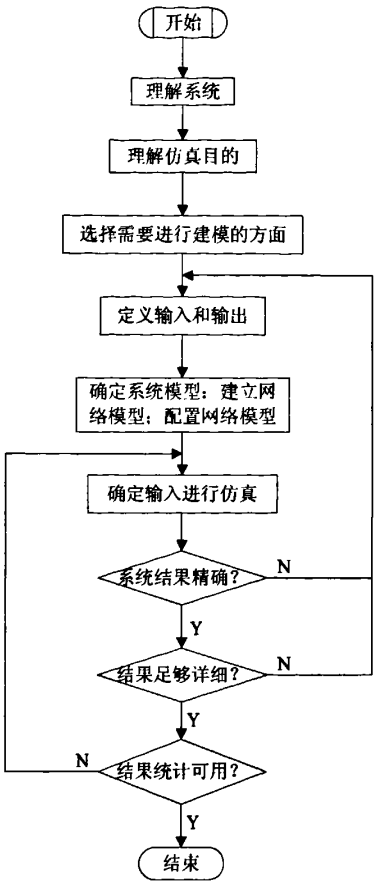

图 6.1　工作流程

⑴理解系统：精确地理解要模拟的系统对象是整个建模的第一个环节，使用者对系统理解的精确性直接影响到所建模型的精确性。

⑵理解仿真目的：仿真结果能帮助使用者解决什么问题。

⑶选择需要进行建模的方面：即从前面的问题中得到建模的目标。

⑷定义输入和输出：输入可能是固定的（如网络拓扑结构），也可能是变量（如业务产生源的业务生产率）。研究一个系统的时候，一般是保持一些变量不变，然后在一定范围内变化一两个变量，接着就是确定输出（如端到端的时延、吞吐量等）以及显示这些输出的最好方法（图形、表、动画等）。

⑸确定系统模型：建立网络模型，配置网络模型。

⑹确定输入，进行仿真。

⑺系统结果是否精确：结果的容错性和精确性都需要进行验证。一般来说，需要对输出做出一些预测，然后对预测的结果和实际的仿真效果进行比较。

⑻结果是否足够详细：根据需要，适当增大输出范围或将输入限制在一个较小的范

围内。

⑼结果是否统计可用：应使得仿真达到一个稳定的状态，如果一个模型不能够达到稳定状态，也就说明仿真系统本身不是很稳定。

综上，使用 Modeler 进行仿真可以大体分成以下 6 个步骤：配置网络拓扑（Topology），配置业务（Traffic），收集结果统计量（Statistics），运行仿真（Simulation），调试模块再次仿真（Re-simulation），最后发布结果和拓扑报告（Report）。

6.2 OPNET 网络建模

6.2.1 建模过程

建模中最基本的概念是“等同性”。建模的过程实际上是将实际的系统映射到仿真环境中，仿真环境对实际系统的逼近程度直接影响到仿真结果的有效性。但是由于建模本身的复杂性，仿真系统只能从某些方面去模拟实际系统的行为。因此，“等同性”并不是指仿真系统和实际系统完全的等同，而是指在某些方面、某些层次反应实际的系统。使用 OPNET Modeler 进行建模的过程如图 6.2 所示：

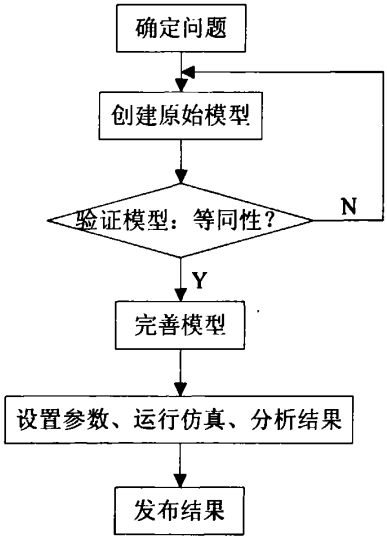

图 6.2 建模过程

⑴确定模型需要解决的问题：收集和消化网络工程设计文档，主要包括网络拓扑结构、网络协议和标准、网络设备、网络链路、网络应用及业务量特征等。

⑵创建原始模型：首先，建立网元模型，对于基本模型库中已有的网络设备，根据网络设备的接口配置对现有模型进行修改，优化网络设备模型；对于基本模型库中没有的网络设备，需要自己开发新设备模型。其次，建立网络模型，就是在网元模型建立好

之后，依据仿真网络，建立起网元模型之间的有机连接，从而将整个仿真网络系统映射为 OPNET 网络模型。最后，建立网络流量模型。

⑶验证模型：以获得一定的“等同性”。对于出现的不等同性，找出原因，如果不影响系统性能，可以忽略过去；如果影响到系统的性能，则需要进行修改。如果无法获得模型与实际系统的一定“等同性”，则返回第 2 步，重新创建模型。

⑷完善模型：对模型做出修改，以解答未来的问题。

⑸设置仿真参数及条件，运行仿真，查看并分析结果。

⑹发布仿真结果。

6.2.2 仿真过程

完成网络中所有的模型设计之后，就可以运行仿真来研究系统的性能和行为。一般来说，仿真的执行和信息收集需要完成以下三个步骤：

⑴选择仿真过程中要收集的网络性能统计参数，按统计参数的收集范围可将其分为全局统计量（Global Statistics）和对象统计量（Object Statistics）；按统计量的类型可分为统计数据（Statistic Data）和动画数据（Animation）。统计数据是一对数据值的集合，其中一项是自变量，另一项是因变量，通常情况下，自变量是仿真时间，因变量是收集结果的统计量；动画数据记载了网络模型在仿真过程中的动态行为，如数据包在网络中的流动过程等。

⑵仿真序列设计，选择合适的仿真参数，在不影响仿真结果可用性的前提下尽可能提高仿真计算效率。仿真参数包括仿真时间（Duration），统计量收集的数据个数，统计量原始数据的处理模式等。

⑶执行仿真、分析结果。比较仿真结果与实验或测量结果，验证模型和仿真方法的正确性，如有需要，改变网络模型或仿真参数，重复仿真过程。

6.3 仿真实验及结果分析

6.3.1 实验环境

本实验是在 Microsoft Windows XP SP2 的操作系统上使用 OPNET 8.0 作为仿真软件进行算法模拟，计算机系统硬件配置如下：

CPU：Mobile Dual Core Intel Core Duo，1866MHz（14×133）；

内存：1014 MB（DDR2-667 DDR2 SDRAM）；

硬盘：WDC WD1200BEVS-75RST0（111GB，IDE）；

网络适配器：Broadcom 440×10/100 Integrated Controller。

在程序中选择 OPNET Modeler 8.0，运行 OPNET Modeler，OPNET Modeler 8.0 的主界面如图 6.3 所示。

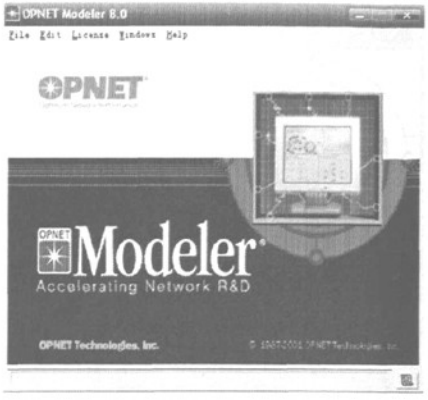

图 6.3　OPNET Modeler 8.0 的主界面

各个模型的建立分别在 OPNET 提供的相应编辑器中完成：网络拓扑结构使用网络编辑器构造；网络中的各种通信设备模型，如路由器、节点、链路等在节点编辑器中完成；路由过程中，各个路由器和节点的状态在进程编辑器中进行，路由算法的代码编写和编译在进程编辑器的内部实现，各种控制包使用包格式编辑器生成。

6.3.2 构造网络模型并配置网络参数

利用 OPNET 建立一个 OSPF 网络模型，如图 6.4 所示，每一个节点代表一个固定子网，子网内部结构如图 6.5。子网内部局域网模型为 10BaseT_LAN，子网之间通过其内部的路由器互连，所选路由器模型为 ethernet4_slip8_gtwy；作为数据流发送端的客户机与接收端的服务器分别位于名为 Boston 和 Atlanta 的子网内部，均为 ppp_wkstn_adv；链路模型为 LAN_Mod_PPP_DS0（64Kbps）和 10BaseT。

图 6.4 中节点 Application Configuration 用于配置应用程序，Profile Configuration 用于配置应用规格，QoS Attribution Configuration 用于配置 QoS 参数，Application Configuration 和 Profile Configuration 共同为 Client 配置应用，为 Server 配置服务，也即为整个网络配置流量。

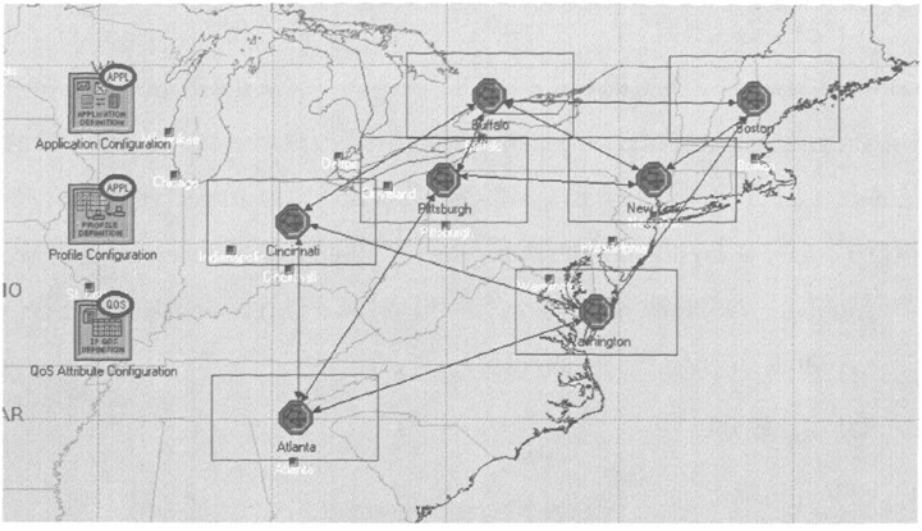

图 6.4　仿真 OSPF 网络模型

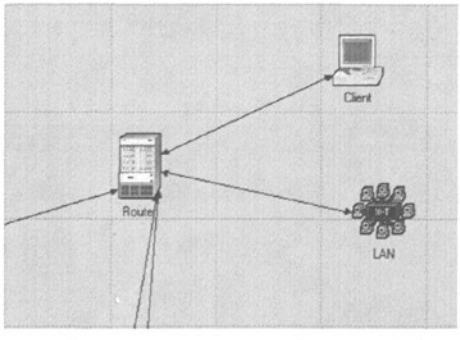

图 6.5　Boston 子网内部结构

本仿真实验中定义的 Application 为 OSPF 网络中具有 QoS 要求的 Video Conferencing，支持此应用的各项相关配置以 Video Conferencing 的 Application Configuration 为例，如图 6.6 所示，其余在此不赘述。至此，OSPF 网络模型初步建立。

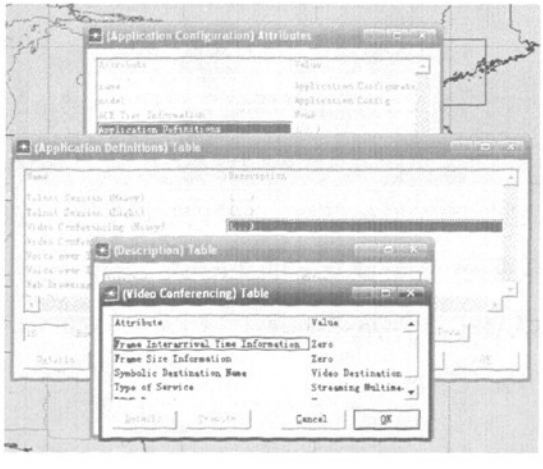

图 6.6　Video Conferencing 应用配置

本实验是在同一网络模型下分别实现基于遗传-蚁群融合算法（GAAA）和扩展的 Bellman-Ford 算法（EBFA）的 OSPF-QoSR，并对两种算法在整个网络的开销及数据包端到端的延迟等方面进行比较。为此，还需要进一步改进模型，设置一些重要的统计变量：在网络节点的过程模型中，设置路由器的数据传送率为 10^5packet/second；在数据流收发模型中，定义 Delay、Bandwidth、Loss_Rate 和 Routing Table Interval。复制 Scenarios，在两个完全相同的 OSPF 网络模型中分别运用两种算法进行路由，编译通过各个过程模块（Process Model），搜集统计参数，运行仿真，比较结果。

6.3.3 仿真结果比较分析

1. OSPF 开销统计

对整个网络收集全局统计量（Global Statistics），选择 OSPF→Traffic Sent（bits/sec），运行仿真。仿真结束后，在工作空间点击右键选择 Compare Results，比较两种算法对整个 OSPF 网络开销的影响，如图 6.7 所示：

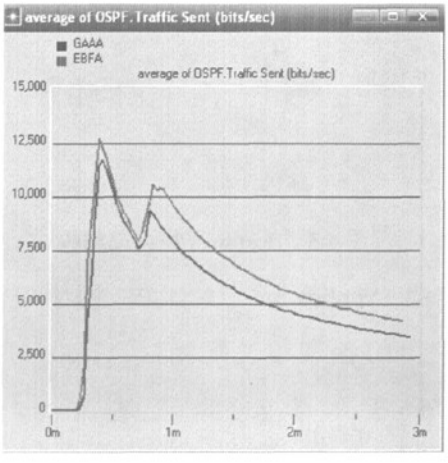

图 6.7　两种算法 OSPF 开销统计

图中蓝色曲线表示使用 GAAA 路由算法时整个 OSPF 网络的开销，红色曲线表示使用 EBFA 算法时的网络开销。从该仿真结果图可知：⑴基于 GAAA 的 OSPF-QoSR 是收敛的。仿真在 0~50s 之间，整个网络中路由交换的数据量比较大且变化较快；仿真进行到大约 50s 以后时，数据通信量逐渐减小并趋于稳定，这是因为此时 OSPF 网络达到收敛状态。⑵基于 GAAA 的 OSPF-QoSR 比基于 EBFA 的 OSPF-QoSR 更早达到收敛状态（前者在大约 50s 时开始收敛，后者在大约 1min 时才开始收敛）并在网络收敛后具有更小的开销。

2. 数据包端到端的延时

对整个网络收集全局统计量（Global Statistics），选择 Video Conferencing→Packet End-to-End Delay（sec），运行仿真。仿真结束后，在工作空间点击右键选择 Compare Results，比较两种算法在支持视频会议业务时数据包端到端的延时，如图 6.8 所示：

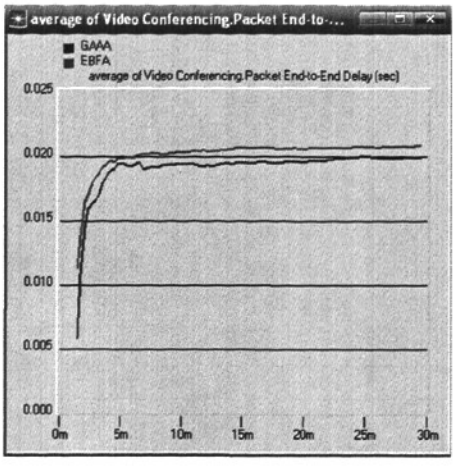

图 6.8　两种算法数据包端到端的延时比较

图中蓝色曲线表示使用 GAAA 路由算法时 Video Conferencing 数据包的端到端延时，红色曲线表示使用 EBFA 算法时数据包端到端的延时。从该仿真结果可以看到，当系统达到稳定状态之后，使用 GAAA 进行支持 Video Conferencing 应用的 OSPF-QoSR 时，数据包端到端的延时是小于使用 EBFA 时数据包端到端延时的（前者约为 19ms，后者约为 21ms）。

综上，基于融合算法的 OSPF-QoSR 在网络开销和数据包端到端的延时上均比扩展 BF 算法有了一定的提高，这是因为融合算法不再进行单一的“最短路径选择”，而是动态地进行“最优路径选择”，即数据包总是沿着最优路径传送，使得 OSPF 网络能够更快地达到收敛状态并拥有相对较小开销，且使数据包端到端的延时有了一定程度的减小。在当今网络业务流对 QoS 提出更高要求的情况下，本算法为多约束 OSPF-QoSR 提供了新的思路。

6.4　本章小结

本章简单介绍了网络仿真软件 OPNET Modeler 的工作流程，着重讲述了其建模和仿真的过程，使用该软件进行仿真实验，验证了本文所提出的基于遗传-蚁群融合算法的 OSPF-QoSR 机制的可行性，并与 RFC2676 所建议的基于扩展 BF 算法的路由机制进

行仿真结果对比和分析，证明了本文路由算法在网络开销、数据包端到端的延时等方面的优越性。

结论与展望

随着计算机硬件及Internet网上各种多媒体和实时业务的发展，对路由协议进行QoS功能扩展成为网络发展的一个迫切需要。OSPF 协议作为当前应用最为广泛的内部网关协议，IETF 对其各种版本进行不断修正和改进，其应用范围也将继续扩大。本文在RFC2676 定义的 OSPF 协议扩展方案的基础上，实现了基于遗传-蚁群融合算法的OSPF-QoSR。

1. 本文通过对 OSPF 协议的 QoS 扩展研究，得出以下结论：

⑴采用遗传-蚁群融合算法进行多约束 QoSR，该算法克服了两种算法的各自缺陷，通过对二者的“融合”——即以遗传算法所得优化解初始化蚁群算法的初始信息素值，循环迭代，求得多约束 QoSR 问题的最优解。

⑵对 OSPF 协议的工作原理及其路由算法深入研究，说明作为一种典型的链路状态协议，OSPF 基于 Dijkstra 算法，而该算法要求以某一固定的链路状态信息来计算最短路径，这就使得当前的 OSPF 协议不支持多约束 QoSR 机制，本文的任务就是实现多约束 OSPF-QoSR。

⑶充分考虑到对现有 OSPF 协议的兼容，提出了 OSPF-QoSR 的具体实施方案，其基本思路是在对当前 OSPF 协议报文格式和工作机制做最小改动的前提下，最大程度地支持多约束 QoSR，实现基于遗传-蚁群融合算法的 OSPF-QoSR。

⑷利用网络仿真软件 OPNET 构造支持 QoS 的 OSPF 网络，模拟仿真实现本文基于融合算法的多约束 OSPF-QoSR 机制，并将其在网络开销、数据包端到端延时等性能上与 RFC2676 所推荐的扩展 Bellman-Ford 算法进行比较，仿真结果表明本文算法是可行的、有一定优越性的，为今后大型 OSPF 网络中多约束 QoSR 机制的研究提供了新的思路。

2. 对未来工作的展望

本文研究和探讨了基于遗传-蚁群融合算法的 OSPF-QoSR 机制，对于 OSPF 协议的 QoS 扩展和融合算法的实现提出了一些观点和方案。由于时间和其它因素影响，本文设计中还有很多不成熟和值得进一步研究的地方。

⑴本文算法所采用的路由策略是分布式路由，即每个节点只需收集和维护一定的网络状态信息，不必知道完整的可行路径，各个节点独立计算可行路径，但是可能由于信息的不一致性会造成回路问题，这就需要额外的回路检测算法，而本文是在假设不存在

回路问题的理想前提下讨论遗传-蚁群融合算法的。

⑵根据计算可行路径的时刻，QoSR 分为预先计算和按需计算两种，各有利弊。本文选取了应用范围比较广泛的路由预先计算，弥补了按需计算中按请求触发路径计算开销大的不足，但由于 QoS 业务的多样性，路由表为了包含每个可能的 QoS 度量，其规模会比较庞大。如何在这两种计算方式中寻找一个扬长避短的平衡点，是下一步工作值得研究的问题。

⑶在 LSA 中添加 QoS 信息后，必然要对 LSA 的发送机制做一些改动，本文建议采用触发更新与周期更新相结合的方式，如何限制发送更新的间隔时间、何时触发更新消息，是与实际网络状态密切相关的，在这个问题上还有待于根据实际情况进行更深入的探讨。

⑷由于时间有限，融合算法在实现上尚有一些不尽人意的地方，例如没有检测排除回路，忽略了信息陈旧性对算法性能的影响等。这些不足将有待于进一步修改和弥补。

⑸OPNET Modeler 作为当前业界领先的网络技术环境，其功能相当强大。本文实验仅仅利用 OPNET 实现了本文算法在一个特定 OSPF 网络中进行 QoS 路由的仿真模拟，只考虑了诸多网络性能中比较有代表性的两个，即网络开销和数据包端到端延时。改进网络模型、选择更佳更多的网络性能参数、大量比较不同算法的性能以证明本文算法在更多方面的优越性是下一步研究工作的重点。

参考文献

[1] E. Crawley, R. Nair, B. Rajagopalan, H. Sandick. A Framework for QoS-based Routing in the Internet[S]. RFC2386, Aug. 1998

[2] Quality of Service-Glossary of Terms[EB/OL]. May 1999, http://www.qosforum.com/ white-papers/qos-glossary-v4.pdf

[3] Z. Wang and J. Crowcroft. QoS Routing for Supporting Multimedia Applications[J]. IEEE Journal on Selected Areas in Communications, Vol. 14, No.7, September 1996, pp: 1228~1234

[4] M. S. Garey, D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness[A], W. H. Freeman, New York, 1979

[5] J. M. Jaffe. Algorithms for Finding Paths with Multiple Constrains[J]. Networks, vol.14, pp: 95~116, 1984

[6] R. Widyono. The Design and Evaluation of Routing Algorithms for Real-time Channels [A]. TR-94~024, International Computer Science Institute[C], UC Berkeley, 1994

[7] S. Chen and K. Nahrstedt. On Finding Multi-Constrained Paths[C]. In Proceeding of IEEE ICC, 1998

[8] X. Yuan. On the Extended Bellman-Ford Algorithm to Solve Two-Constrained Quality of Service Routing Problems[C]. In Proceedings of the English International Conference on Computer Communications and Networks, October 1999

[9] X. Yuan, X. Liu. Heuristic Algorithms for Multi-Path Routing Combined with Resource Reservation[C]. In Proceedings of IEEE INFOCOM, 2001

[10] Orda and A. Sprintson, QoS Routing: the Pre-computation Prspective[C]. In Proceedings of IEEE INFOCOM, 2000

[11] H. F. Salama, D. S. Reeves and Y. Viniotis. A Distributed Algorithm for Delay -Constrained Unicast Routing[C]. In Proceedings of IEEE INFOCOM, April 1997

[12] T. Kormaz, M. Krunz. Multi-Constrained Optimal Path Selection[C]. In Proceedings of IEEE INFOCOM, 2001

[13] Funda Ergun, Rakesh Sinha, Lisa Zhang. QoS Routing with Performance-Dependent Costs[C]. In Proceedings of IEEE INFOCOM, March 2000

[14] Juttner, B. Szviatovski, I. Mecs et al. Lagrange Relaxation Based Method for the QoS Routing Problem[C]. In Proceedings of IEEE INFOCOM, 2001

[15] Bernard Fortz, Mikkel Thorup. Optimizing OSPF/IS-IS Weights in a Changing World[J]. IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, 2002, 20(4): 756-767

[16] David Watson, Farnam Jahanian, Craig Labovitz. Experiences with Monitoring OSPF on a Regional Service Provider Network[C]. Proceedings of the 23rd International Conference on Distributed Computing Systems, 2003

[17] 牟春燕. OSPF 路由协议及其实现算法[J]. 杭州电子工业学院学报, 2003, 23(1): 80～84

[18] 张静, 冉晓旻, 胡捍英. 一种基于 OSPF 扩展的预计算 QoS 路由算法研究[J]. 计算机科学, 2006, 33(7): 47~51

[19] 华晏, 钱松荣. 支持 QoS 路由机制的 OSPF 扩展的研究[J]. 计算机工程与设计 2006, 27(3): 415~417

[20] 丁建立, 陈增强, 袁著祉. 遗传算法与蚂蚁算法的融合[J]. 计算机研究与发展, 2003, 40(9): 1351～1356

[21] 徐恪, 吴建平, 徐明伟. 高等计算机网络——体系结构、协议机制、算法设计与路由器技术[M]. 北京: 机械工业出版社, 2003: 231

[22] Zhang, S. Deering, D. Estrin, S. Shenker and D. Zappala. RSVP: A New Resource ReSerVation Protocol[S]. IEEE Network, September 1993

[23] Cindon, R. Rom, and Y. Shavitt. Multi-Path Routing Combined with Resource Reservation[C]. In Proceedings of IEEE INFOCOM, April 1997

[24] Awerbuch, Y. Azar, S. Plotkin and O. Waarts. Throughput-Competitive On-line Routing[C]. In Proceedings of 34th Annual Symposium on Foundations of Computer Science, 1993

[25] IETF Internet Traffic Engineering working group(TEWG)[EB/OL], http://www.ietf.org/html.charters/tewg-charter.html

[26] Rosen, A. Viswanathan, R. Callon. Multiprotocol Label Switching Architecture[S]. RFC 3031, January 2001

[27] IETF Integrated Services (intserv) working group[EB/OL],

http://www.ietf.ogr/html.charters/intserv-charter.html

[28] IETF Differentiated Services (intserv) working group[EB/OL], http://www.ietf.ogr/html.charters/diffserv-charter.html

[29] X. Xiao and L. M. Ni. Internet QoS: A Big Picture[J]. IEEE Network, Vol.13, No.2, March/April 1999

[30] S. Chen and K. Nahrstedt. An Overview of Quality-of-Service Routing for Next-generation High-speed Networks: Problems and Solutions[J]. IEEE Network, Vol. 12, No.6, November 1998, pp: 64~79

[31] 徐恪, 吴建平, 徐明伟. 高等计算机网络——体系结构、协议机制、算法设计与路由器技术[M]. 北京: 机械工业出版社, 2003: 215～216

[32] B. Wang, J. C. Hou. Multicast Routing and Its QoS Extension：Problems, Algorithms, and Protocols[J]. IEEE Network, Vol: 14, No.1, Jan/Feb 2000, pp: 22~36

[33] R. Guerin and A. Orda. Networks with Advance Reservations: The Routing Perspective [C]. In Proceedings of IEEE INFOCOM, 2000

[34] Raz, Y. Shavitt, Danny Raz and Yuval Shavitt. Optional Partition of QoS requirements with Discrete Cost Functions[C]. In Proceedings of IEEE INFOCOM, 2000

[35] H. Lorenz, Ariel Orda, Danny Raz and Yuval Shavitt. Efficient QoS Partition and Routing of Unicast and Multicast[C]. In Proceedings of IEEE International Workshop on QoS, 2000

[36] Shaikh, J. Rexford, K. G. Shin. Evaluating the Impact of Stale Link State on Quality-of-Service Routing IEEE/ACM Transactions on Networking[J], Vol.9, No.2, April 2001, pp: 162~176

[37] Q. Sun and H. Landgendorfer. A New Distributed Routing Algorithm with End-to-End Delay Guarantee[A]. Unpublished paper, 1997

[38] L. Sobrinho. Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet[C]. In Proceedings of IEEE INFOCOM, 2001

[39] Holland Jh. Adaptation in Nature and Artificial Systems[D]. The University of Michigan Press, 1975, MIT Press, 1992

[40] 陈国良, 王煦法, 庄镇泉等. 遗传算法及其应用(第一版)[M]. 北京: 人民邮电出版社, 1996

[41] 李敏强, 徐博艺, 寇纪淞. 遗传算法与神经网络的结合[J]. 系统工程理论与实践, 1999, 19(2): 65~69

[42] Colorni A, Dorigo M, Maniezzo V. Distributed Optimization by Ant Colonies[A]. Proceedings of the 1st European Conference on Artificial Life, 1991, 134~142

[43] 吴斌, 史忠植. 一种基于蚁群算法 TSP 问题分段求解算法[J]. 计算机学报, 2001, 24(12): 1328～1333

[44] 段海滨. 蚁群算法原理及其应用[M]. 北京: 科学出版社, 2005

[45] 陈骏坚, 李腊元. 用新型蚂蚁算法求解 QoSR 问题[J]. 武汉理工大学学报（交通科学与工程版）, 2005, 29(3): 342～345

[46] 黄晓雯, 贺细平, 唐贤英. 基于遗传算法的 QoS 路由选择与仿真[J]. 计算机仿真, 2003, 20(6): 43～45

[47] Marco Dorigo, Eric Bonabeau, Theraulaz Guy. Ant algorithms and stigmergy[J]. Future Generation Computer System, 2000, 16(8): 851~871

[48]吴庆洪, 张纪会, 徐心和. 具有变异特征的蚁群算法[J]. 计算机研究与发展, 1999, 36(10): 1240~1245

[49] Marcus Randall, Andrew Lewis. A parallel implementation of ant colony optimization[J]. Journal of Parallel and Distributed Computing, 2002, 62(9): 1421~1432

[50] 徐恪, 吴建平, 徐明伟. 高等计算机网络——体系结构、协议机制、算法设计与路由器技术[M]. 北京: 机械工业出版社, 2003: 180～181

[51] 关国利. 基于 QoS 的 OSPF 的预先计算算法[J]. 郑州大学学报(理学版), 2005, (12): 46~48

[52] 樊明, 马跃, 蒋砚军. 一种支持 QoS 的 OSPF 扩展算法（按需计算）[J]. 北京邮电大学学报, 2001, (2):65~68

[53] R. A. Guerin, A. Orda, Williams. QoS Routing Mechanisms and OSPF Extensions[C]. In Proceedings of IEEE GLOBECOM, 1997

[54] 赵世鑫, 潘雪增, 平玲娣. OSPF 路由协议上的服务质量扩展[J]. 计算机应用, 2003, 23(9): 31~34

[55] Moy J. OSPF Version 2[S]. STD 54, RFC 2328，April 1998

[56] D. Williams, S. Kamat, R. Guerin etc. QoS Routing Mechanisms and OSPF Extensions[S] (RFC2676), The Internet Society, 1999.8

攻读学位期间取得的研究成果

[1] 王卫亚，王凤琳. 多约束条件下路由选择算法研究[J]. 计算机应用，2007，27(10)：2395-2397

[2] 段一飞，林关成，王凤琳. 基于 DHT 的 P2P 网络资源定位模型研究[J]. 宝鸡文理学院学报(自然科学版)，2007，27(1): 60-63，88

[3] 龚颖，段一飞，王凤琳. PCI EXPRESS 的发展和应用趋势[J]. 科技信息，2007，242(30)：82-84

致谢

随着论文的完成，我即将告别三年的研究生生活，走向新的工作岗位。由衷地感谢学校和学院为我们提供了良好的学习和科研环境，使我能专心完成硕士期间的学业。

能顺利完成这段学习研究，首先得益于我的导师王卫亚副教授，在此衷心感谢王老师。在科研和日常生活中，王老师给了我无微不至的指导、关怀和帮助。他的悉心指导和言传身教，不仅使我在理论和实践上得到了锻炼和提高，更重要的是让我学到了许多研究问题的方法。王老师严谨求实的治学态度、广博的学识和深刻敏锐的思维让我深深体会到了真正的学者风范。从王老师那里我不只学到了科学研究的方法，更学到了实事求是的处事态度，在以后的工作学习中我将会受益匪浅。

感谢参考文献中致力于 OSPF 协议及 QoS 路由机制研究的学者们，他们的研究给了我很大的启发和思路。

感谢评阅本论文的专家、教授在百忙之中提出了宝贵意见，使得本论文得以完善。

另外要感谢的是我的同门龚颖、李伟兵、韩玉慧和陈冬冬在我论文撰写过程中对我的支持和帮助。在论文的研究学习阶段，他们对我的论文提出了许多有益的建议和意见，对我的论文最终完成给予了很大的帮助。

同时，还要感谢我身边的每一位老师和同学，感谢他们在这三年里对我各方面无私的帮助和关怀，与他们共同学习和生活的日子让我终生难忘。

最后，向导师王卫亚副教授以及给予我关心和帮助的老师和同学们，致以诚挚的谢意！