第五章 移动卫星 MPLS 网络中的 QoS 路由问题研究

5.1 前言

服务质量（QoS）技术是未来高速卫星通信网的关键技术之一，是实现卫星网络多媒体通信和流量工程的基础[22]。对于卫星网络，QoS 的研究有三个主要的推动力。一是对 QoS 有严格要求的业务的出现，如视频、语音等多媒体业务。二是通过 QoS 研究，有助于提高网络效率，降低网络成本。三是作为地基高速通信系统的重要补充，卫星网需要具有同地面网络相兼容的 QoS 体系，以便可为不同用户提供各种有区别的服务，提高用户的满意度。因此，卫星网络服务质量的研究重点是如何提高网络提供 QoS 保证的能力以实现与地面网络 QoS 体系的兼容，而最终研究的目标是保证用户的 QoS 要求。

一般来说，提高网络的 QoS 的途径有两个，即节点控制和整网或局部网络控制。其中前者是一种较为传统的方法，主要控制业务对单节点共享资源的占用，如缓冲区管理或业务调度等。该方法一般只能在局部取得较好的效果，往往无法顾及网络的整体性能，因此人们开始研究如何通过对路由与信令的控制以达到对业务流或业务连接在网络中传输的直接控制，这种方法可以在保证业务的 QoS 的同时兼顾网络整体资源的优化利用，可以提高网络资源的利用率，这对于卫星网络这种网络资源昂贵的通信系统来说尤为重要，因此，QoS 路由成为解决卫星网络中 QoS 问题的一项关键技术。

QoS 路由问题的主要目标是为接入的业务选择满足其服务质量要求的传输路径，同时保证网络资源的有效利用。对于卫星网络来说，QoS 路由问题需重点考虑以下两方面问题：1）度量参数选择问题，即依据哪些度量参数作为 QoS 寻路标准，在这方面，卫星网络不仅要像地面网络一样的考虑各种与业务有关的参数（如带宽、时延等），还需要考虑卫星网络本身的特点（如切换等）；2）寻路问题，即在寻路标准设定后，如何找到满足业务需求的路径，并保证数据经由选定路径传输到目的节点，它实际上包括两部分内容：路径建立与维护、路径计算与生成[86]，其中前者可以采用扩展的链路状态协议（如 OSPF 等）和面向连接的 RSVP、 CR_LDP 等协议来实现，后者实质上是一个多约束最优路径（MCOP）问题，这是一个 NP 问题[87]，目前在地面成熟网络也没有得到很好的解决，对于卫星网络，由于计算和储存资源的限制，该问题变得更具挑战性，成为卫星路由问题的一个难点。

事实上，基于 QoS 的寻路问题也是 MPLS 技术的核心问题之一。对于 MPLS 技术，其核心思想是先为各业务流（flow）建立 LSP（label switching path），然后在 LSP 上传送业务流。建立 LSP 时，既要考虑业务流的各种 QoS 要求，确保 LSP 能够满足这些要求，也要考虑如何优化利用网络的资源，这即是一个多约束最优路径问题[88]。对于卫星网络，由于卫星节点间存在较大的传输时延，更容易造成通信时延过大等问题，所以研究基于时延受限的最优 QoS 路由对于卫星网络具有尤为重要的意义。

本章重点解决移动卫星 MPLS 网络中最具现实意义的时延受限最优寻路问题，首先针对卫星节点计算资源有限的特点，给出一种快速、有效的时延受限最优路径(DCOP)算法。该算法具有 1）算法可以得到满足约束条件的最优无环解；2）算法计算复杂度小，接近无约束的 Dijkstra 算法；3）采用自适应动态调整技术，不需要人为选择参数，算法适应性好等优点。其次针对卫星时变拓扑的特点，通过将网络切换引入路径费用中，来寻找一条既满足业务时延要求又可尽量降低链路切换对网络性能影响的最优 QoS 路径，并利用前面的 DCOP 算法，给出一种适用于卫星节点的时延受限最小费用路由(SDCMCR)算法。

5.2 现状分析

5.2.1 卫星网络 QoS 路由

由于卫星节点的运动性，网络中频繁出现链路切换，使得针对卫星网络的 QoS 路由技术比地面网络的 QoS 路由技术更为复杂。目前，在卫星 QoS 路由度量参数选择方面，最新的研究表明，只有充分考虑卫星节点的动态特性，才可真正有效的保证卫星用户的 QoS[89]；而对于 QoS 路由计算问题，利用卫星星座特点构建计算方法（如 MLSR 算法[90]）无疑是一种很好的解决方法，但这种情况下得到的算法往往无法推广到别的星座，算法适应性、扩展性差。

在卫星 QoS 路由方面，相关的研究成果可以分为三类。一类是将时变拓扑条件下的路由问题规约到静态拓扑下的路由问题，这类方法在卫星网络路由研究中比较普遍，但研究大多针对普通的路由问题，很少考虑 QoS 路由，这方面的研究主要有：

McMahon[91]给出 GALPEDA 算法将卫星运行周期分为很多个有限的时间区间，并假设在该区间内卫星网络是静态的，利用地面固定网络成熟的 GA 和 LP 算法对网络中优先级高的业务实行带宽预留来保证网络的 QoS，该方法重点考虑了业务流的模型对算

法的影响，并针对不同的业务模型（Poisson 或 Markov）修改算法的参数以达到更好的效果，但算法没有考虑到卫星链路的切换。

Ekici[63,92]给出的 DRA 算法利用圆极轨道 LEO 卫星网络规则的网状拓扑结构，在每个传输节点分别为每一个数据包选择一条具有最短传输距离的路径。该算法可动态的为每一个数据包确定最佳的传输路径，减轻了时变网络拓扑对数据传输的影响，但算法不能很好的应对拥塞问题或卫星节点失效问题，且节点路由效率低，无法适用于视频、话音等实时多媒体业务。

Henderson[62]给出的 GPRP 协议通过将地球表面划分为若干个蜂窝状区域，并规定每个区域都由离蜂窝中心最近的卫星提供服务，来解决星地以及星间链路频繁切换的问题。该方法实现简单，但具有相当的局限性，如无法处理极轨道卫星网络两极链路关闭、轨道对飞等特殊情况，也不能处理多层轨道路由、卫星节点失效等问题。

Gounder[78]将一个卫星网络动态拓扑分为一系列的静态拓扑结构，定义一个卫星网络“快照”为卫星系统某一特定时间的网络拓扑，并规定卫星每增加或断开一个星间链路，则形成一个快照，这样可得到一系列的网络快照的静态拓扑，该方法本身不针对 QoS 路由，但 Gounder 指出可以利用 K 路由来实现网络的 QoS。

类似的方法还有 Mauger[76] 的虚拟节点（Virtual Node）、Chang[77] 的有限状态自动机（FSK）以及 MARKUS WERNER[72]的动态虚拟拓扑路由算法（DVTR）等。

上述方法都没有考虑所得最优路径可存在的时间长短，而该路径可能会很快随着卫星的切换而中断，这些方法无法保证对许多业务（如语音）的 QoS。所以上述算法比较适应简单路由（无连结最短路径路由），对于考虑多种条件的 QoS 路由，效果不明显。

另一类方法是利用卫星运行的规律性，通过对网络拓扑变化的预测来研究网络的 QoS 路由问题，如基于概率的路由（PRP）[73]，该方法为每个业务呼叫尽可能的选取一条在呼叫持续时间内不会发生链路切换的路径；

随后的 Ercetin 在文献[93]中进一步利用 PRP 解决卫星的 QoS 路由问题，并给出了与非预测的路由算法在丢包率和业务阻塞率的比较，得到很好的结果；

文献[89]中通过定义链路的生成时间来选取一条持续时间最长的路径，以保证业务不会因链路切换而中断。

上述这几种算法可降低链路切换对数据传输的影响，减少网络的重路由率，但这些算法一般采用“剪枝”的方法，即强行将不满足切换概率条件的边去掉，以提高所得到的路径的切换概率，这样算法在降低切换中断率的同时也牺牲了网络的传输时延性能，

一些传输时延小的路径因呼叫持续时间不够而被删除，系统的呼叫阻塞率大大上升（尤其是网络负载较高时），网络利用率下降。因此算法虽然可以较好的支持业务的 QoS，但往往很难保证网络资源的有效利用。

还有一类方法是利用地面网络中成熟的分层或分群管理的方法，来简化卫星路由，以降低卫星网络的动态性对路由的影响，比较典型的有 Lee 的分级卫星路由协议（HSRP）[18,94]和 Akyildiz 的 MLSR 算法[90]等。这些算法通过分级管理，可以减低路由的复杂度，以获得动态的、支持 QoS 的卫星网络路由。但这些算法仅适用于具有相当规模的多层卫星网络，具有很大的局限性。

上述的方法主要是一些解决卫星网络路由问题的策略、协议或框架，而对于基于这些策略或框架的路径计算，往往使用现有成熟最优路径算法（如 Dijkstra 算法、Bellman_Ford 算法等），对于 QoS 路由，由于最优路径计算需要同时涉及多个度量，这是一个 NP 问题，因此人们在这方面做了很多工作[95]，具体情况见下一节。

5.2.2 时延受限最优寻径问题

卫星网络 QoS 路由问题需同时考虑链路长时延和拓扑时变等特性，这是一个多约束最优寻路（MCOP）问题。由于该问题在求解过程中需同时涉及多个度量，导致问题的复杂度为 NPC[87]。为此人们设计了许多近似算法，以获得 MCOP 问题的工程求解。这方面的方法可以分为两大类：一类是将原有问题简化[96,97]，即只求出近似最佳解，目的是缩小搜索范围，以避免大量搜索耗时带来的算法效率问题；另一类方法则求助于 k 路由算法[98,99]，通过构造一个代价函数，该代价函数需同时考虑了主代价和各种约束，并利用 k 路由算法计算出前 k 条代价最小的路径，然后从其中找出符合约束且最佳的解。相关的近似算法主要有：

T-K 算法[100]，首先令 $h(p)=w_1(p)+w_2(p)$，这里 $w_1(p)$和 $w_2(p)$分别为路径 p 的主代价和约束，并以 $h(p)$为代价用 Dijkstra 算法求得最优路，根据约束条件的满足情况，设定一整数 k，再令 $h(p)=w_1(p)+kw_2(p)$，重新用 Dijkstra 算法求得最优路，如解可行则停止，否则调整 k，重新计算，其计算复杂度为 $O(k v \lg_2 v)$。

BG 算法[87]实际是一种 Lagrangian 松弛算法，该算法能够在较短的时间内找到问题的次优解，其计算复杂度为 $O(m(S)n^2)$，$m(S)$为路径的个数，n 为网络节点数。

Chen-Nahrstedt 在[101]中给出一种算法，成功的将多约束问题（MCP）近似的转化为在多项式内可解的问题，该算法通过将 QoS 度量的取值范围 R 或 Z 映射到一个有限

集合 C，以保证多项式可解。算法复杂度为 $O\ (\Pi x_i ve)$。尤其对于单约束最短路径，当约束条件为跳数时，算法多项式可解，其复杂度为 $O\ (v^4)$。

类似的方法还有针对带宽受限的 N-M 路由算法[102]，基于加权线性和的 Jaffe 算法[103]，基于探测的分布式算法[104]等。

这类算法都具有较大的缺陷，首先算法一般不能得到最佳路径，只能得到次优解；其次算法的性能往往与具体的参数密切相关，如有的求解算法[105]需利用 k 路由算法，而 k 的选值将决定算法的效果，若 k 值较大，则难以保证算法的效率，反之，则不能保证找到可行路径；另外，这些算法大多设计复杂，工程实现困难。

解决MCOP的另一个思路是扩展原有的单一量度最优算法（如Dijkstra算法、Bellman_Ford算法），使其能针对多度量求解，如文献[106]给出的多标号算法和EBFA算法[107]等，其中多标号算法是基于Dijkstra算法的扩展，在执行中每个节点须保留若干满足条件的解，导致复杂度大大增加；而EBFA算法是基于Bellman_Ford算法的扩展，与多标号算法一样，该方法的时间和空间复杂度也都是指数级的。上述这些算法可得到满足约束条件的最优解，且具有算法设计简单，不需调整参数，易于实现等优点，但其计算复杂度随网络节点数目的增加呈指数级增长，算法无法在实际中应用。

随后的许多学者对上述算法进行了改进，比较成功的有路径受限算法[108]和HMCOP算法[95,109]，其中[108]通过在EBFA中直接限制可能路径数|Path|的大小从而减小算法的复杂度。即令X为| Path |最大约束数目，在原EBFA算法向Path加入一条可能的最优路径之前，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，能够保证算法复杂度为$O(X^2|N||E|)$。而HMCOP算法则利用约束条件来限制|Path|的大小，来降低算法复杂度，当这种方法复杂度仍然很大时，算法也如[108]一般，对|Path|的大小加以限制[110]，以进一步降低算法的复杂度。这些改进后的算法往往通过限制每个节点的最大路径数X，以增大路由失败概率和降低路由性能为代价（即有时可能得到的解不是最优解），来降低运算复杂度，算法性能不够理想，同时，X的选择决定算法的效率和性能，但它是一个经验参数，受很多因素影响，这将不利于算法的实现和推广。

5.3 一种高效的时延受限最优路径(DCOP)算法

对于卫星网络，一方面，由于卫星资源的昂贵，需要算法找到满足条件的最小费用的解；另一方面，星上计算能力和储存能力的限制需要算法具有很好的运行效率。这些

都给卫星网络 QoS 路由寻径算法提出了更高的要求，而这方面现有的算法一般都是针对地面网络而设计的，在算法的效率和性能方面都很难满足星上要求，因此，本节将在现有算法的基础上，设计一种适合于卫星网络的高效的时延受限最优路径(DCOP)算法，该算法可以得到最优解，并具有很低的计算复杂度。

5.3.1 问题描述以及符号说明

5.3.1.1 符号说明

G（*V*,*E*）：无向图，其中 $V=\{v_1,v_2\ldots v_n\}$是有限节点集，$E\subseteq V\times V$ 是有限边集；

e(i,j)：v_i,v_j 间的一条边；

D(e(i,j))：e(i,j)上的时延值；

C(e(i,j))：e(i,j)上的代价值，可表示费用或其他 QoS 参数；

p (i,j)：v_i,v_j 间的一条路，是路经过的节点集合；

D(*p*(i,j))：路 *p* (i,j)的时延值；

C(*p*(i,j))：路 *p* (i,j)的代价值；

Γ_i：v_i 的所有邻居节点集合；

D_{min} (*s*,*t*)：*s* 到 *t* 的最短时延路的时延值；

C_{min} (*s*,*t*)：*s* 到 *t* 的最小代价路的代价值；

D_{cmin} (*s*,*t*)：*s* 到 *t* 的最小代价路的时延值；

C_{dmin} (*s*,*t*)：*s* 到 *t* 的最短时延路的代价值；

D_r：路径的时延约束值；

Path_Set：算法的数据结构，用于记录算法中的临时路集合；它的操作包括：Remove_ Min_ Path （从 Path_Set 中移出排序参数最小的路），Inset_Path(将一满足条件的路径按代价大小顺序插入 Path_Set 中)；

V_p(y)：算法的数据结构，用于记录已检查过的从源节点到节点 y 的有效路；它的操作包括：Inset_Path_Vp(将已检查过的路插入 Vp(y))，Check_Path_Vp（检查某路 *p* (*s*,y)的时延和代价是否都大于或等于 Vp(y)中任一路的时延值和代价，若否，则返回 FAIL）。

Loop_Det (*p*(*s*,x),y)：检查 y 是否已在路 *p*(*s*,x)中，如在返回 FAIL。

5.3.1.2 问题描述

给定 G（*V*,*E*），且对于任意边 e(i,j)∈*E*，存在非负函数 D(e(i,j))和 C(e(i,j))，则时延

约束最优路径（DCOP）问题可以描述为：在图 G 中，寻找一条从源节点 *s* 到目的节点 *t* 的路径 $p(s,t)$，满足：$D(p(s,t))\leqslant D_r$，且 $min(C(p(s,t)))$。

5.3.2 DCOP 算法设计

5.3.2.1 算法设计思想

DCOP 问题要求在当前网络状态下寻找满足时延要求的最优路径，该问题同时涉及时延和代价两个度量，使得原有的 Dijkstra 算法无法直接使用，这是由于算法在搜索过程中每个节点需针对约束条件保留若干个可能的路径，从而导致问题的复杂度为 NPC。解决的思路是尽量减少算法需要搜索的可行路径，具体来说，在算法搜索过程中，针对任一当前路径，需要根据一定的规则判断该路径是否可行，若可行，则保留，这样被保留的可行路径数目将决定算法的效率。在这方面，多标号算法和 HMCOP 算法都采用仅保留满足约束条件的解来降低算法的搜索空间。

其中多标号算法的判断规则是：设当前路径为 p_{sx}，则判断 p_{sx} 是否满足约束条件。该规则条件过于宽松，算法需要保留的可行路径仍很多，不能有效减少算法的搜索空间（见图 5-2）。HMCOP 算法在这方面作了很大的改进，其规则为：设当前路径为 p_{sx}，若经过路 p_{sx} 到目的节点 *t* 基于时延最小的路径为 p_{st}，则判断 p_{st} 是否满足约束条件来决定是否保留当前路径（见图 5-2）。该规则的理论依据如下：

命题 5-1：对于图 G 中的任意节点 *s*、*t* 和 m，设路 $p(s,m)$的时延为 x，*t* 到 m 的最小时延为 y，若 $x+y>D_r$，则经过路 $p(s,m)$的所有路 $p(s,t)$都不满足条件 $D(p(s,t))\leqslant D_r$。

证明：过节点 m 的路 $p(s,t)$可以表示为路 $p(s,m)$和路 $p(m,t)$的组合，相应的 $D(p(s,t))=D(p(s,m))+D(p(m,t))$。又由定理“如果网络中每个圈的长度为非负，则网络中每条路径的长度不小于相应的最短路径长度；且每条最短路径的子路径也是最短路径。”[111]，由命题可知 $D(p(s,m))=x$ 而 $p(m,t)$的最小时延为 y，即对于所有的路，有 $D(p(m,t))\geqslant y$ 成立，$D(p(s,t))\geqslant x+y>D_r$。命题得证。

事实上，多标号算法的判断规则也可看作是命题一的一种特殊情况，此时命题成立条件为 $x+y>D_r+y$。由此，上述两种算法可行路径区间可直观的表示为图 5-1 中的区间(abgh)和区间(aijh)。这里用坐标$(d+d`,c+c`)$表示当前路径 U(*s*,m)，其中，*d* 为路径时延，*c* 为路径代价，而 *t* 到 m 的最小时延为 *d*`、最小时延为 *c*`。

由图 5-1 可看出，HMCOP 算法的可行路径区间(abgh)明显小于多标号算法(aijh)，

算法所需要搜索的节点和边的数目（即搜索空间）也将显著减少（见图 5-2），所以算法复杂度明显比多标号算法小，但算法的效率与 D_r 和 D_{min} 相对位置关系很大，当 D_r 逼近 D_{min} 时，可行路径区间很小，算法效率很高，随着 D_r 与 D_{min} 差距拉大，算法效率下降很快（参见第 5.3.3 节实验 3），所以该算法设计者提出此时还须利用限制每个节点保留的最大路径数来进一步降低算法复杂度，但这样会牺牲算法结果的准确性和实用性。针对这种问题，本文给出的 DCOP 算法将通过进一步限制可行路径区间大小来提高算法效率，并在 HMCOP 算法等的基础上，作了如下两方面的改进：

1) 在解空间内找到一个路 P，使其 $D \leqslant D_r$，C 尽量小。令 C 值为 C_m，并以 C_m 为约束条件来限制可行解区间的大小。

该改进的判断规则为：设当前路径为 p_{sx}，若经过路 p_{sx} 到目的节点 t 基于代价 C 最小的路径为 p_{st}，则判断 p_{st} 是否满足约束条件 C_m 来决定是否保留当前路径。具体分析如下。

命题 5-2：对于图 G 中的节点 s、t 和 m，存在一条代价为 C_m 路 $P_m(s,t)$，设路 $p(s,m)$的代价为 x，t 到 m 的最小代价为 y，若 $x+y>C_m$，则经过路 $p(s,m)$的所有路 $p(s,t)$都不会比路 $P_m(s,t)$优。

证明可参见命题 5-2。略。

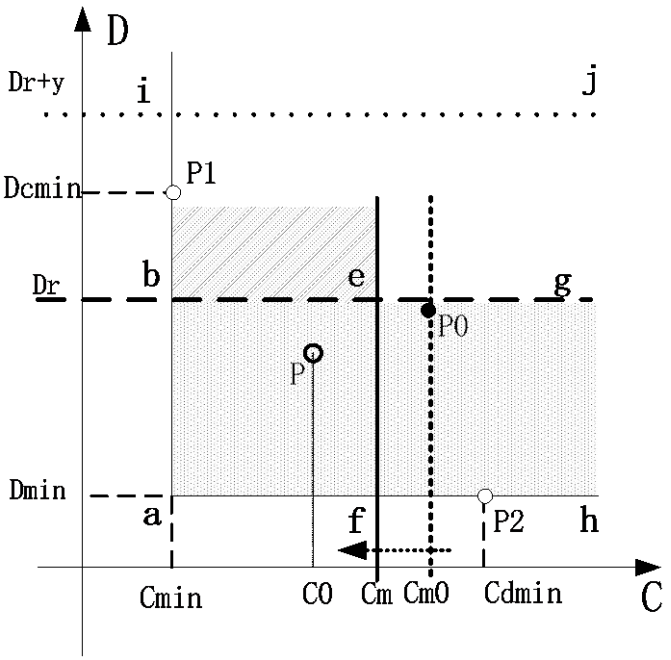

图 5-1　三种算法可行解区间比较

由图 5-1 可看出，改进后算法的可行路径区间（abef）将大大减少，算法所需要搜

索的节点和边的数目以及每个节点需保留的可行路径数目也将大大减少（见图 5-2），算法效率进一步提高。事实上，对于 DCOP 问题，下面的命题是成立的。

命题 5-3：设网络 G 中 *s* 到 *t* 的 DCOP 问题的解为路径 P，对于任意当前路径 *u*，若 $u \subseteq P$，则 *u* 必定位于区间 abef 内（如图 5-1 所示）。

证明：设当前节点为 m，则 $m \in P$，若 m 到 *t* 的最小时延为 x、最小代价为 y，由命题 5-1、命题 5-2，有如下关系成立：$x+D(u) \leqslant D(P) \leqslant D_r$，$y+C(u) \leqslant C(P) \leqslant Cm$。证毕。

命题 5-3 可以保证算法的正确性，即 DCOP 问题的全局最优解必然位于区间 abef 内。

2）在算法运行过程中，采用动态调整 C_m 的方法，使可行解区间进一步优化。如图 5-1 所示，算法首先得到 C_{m0}，在算法运行中，C_m 不断动态调整，逐渐向最终解 P 靠近。这种动态调整使算法具有很好的自适应性，可以克服诸如 HMCOP 算法效率受 D_r 和 D_{min} 相对位置关系影响等缺点，算法适应性大大加强（参见第 5.3.3 节实验 3）。

对于多标号算法、HMCOP 算法和本算法，由于可行解空间与算法需要搜索的节点和边的数目（即搜索空间）是相互对应的，下面给出这种相互对应的示意图（图 5-2）。

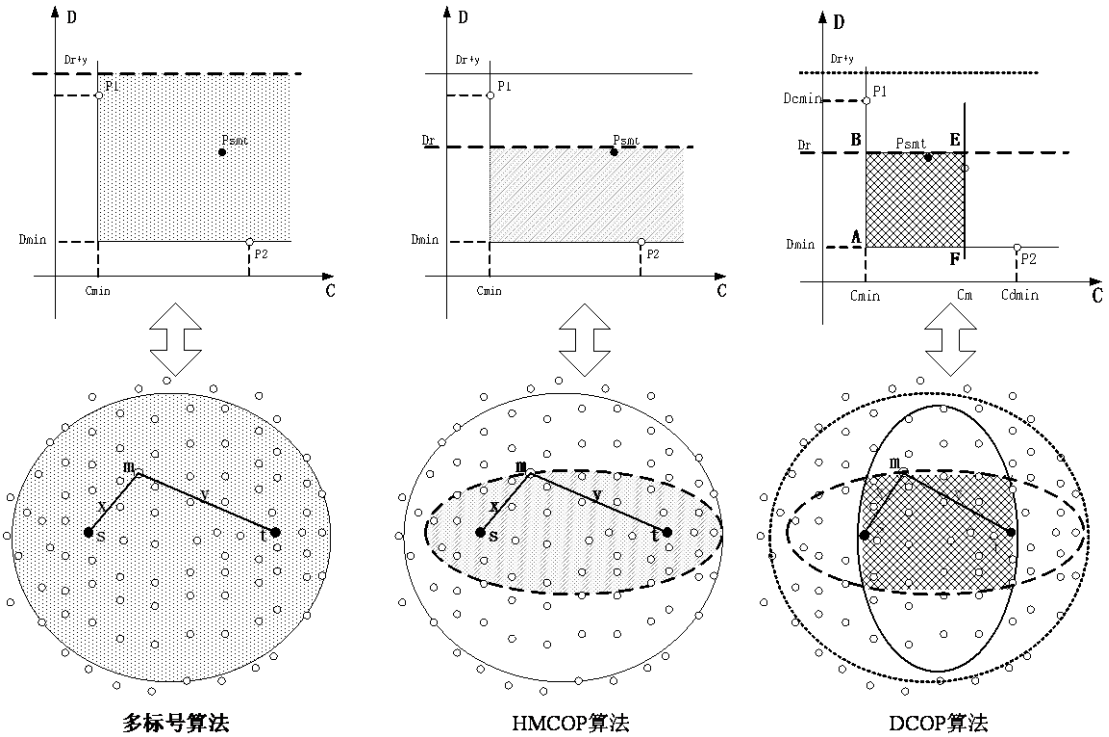

图 5-2　三种算法可行解区间以及搜索空间比较示意图

由图 5-2 可知，本算法需要搜索的节点和边的数目将明显小于前两种算法，算法复杂度将进一步降低。

5.3.2.2 C_m 取值分析

由上节分析可知 C_m 的大小将直接影响算法的效率，若令 C_0 为某 DCOP 问题的最优路径对应的代价值，则 C_m 的值越接近 C_0，算法效率越高，所以寻找合适的 C_m 值将是一个关键问题，本算法在这方面采用首先取得一个可行值，再逐步动态调整的其大小，以获得最优值的设计思路，具体来说：

1) 对于某一个 DCOP 问题，若问题有解，则一定有 $D_{min} \leqslant D_r$ 存在，所以可令 $C_m = C_{dmin}$（见图 5-1）。

2) 若 $D_{cmin} > D_r$，则可以通过修改最小代价路 p_1（见图 5-1）来获取比 C_{dmin} 更优的 C_{m0}。

3) 在算法计算过程中，对每一个可行解 $p(s,y)$，计算其代价值与 $C_{dmin}(y,t)$之和，并动态的调整 C_m 的值，使其逐渐向 C_0 收敛（见图 5-1）。

5.3.2.3 算法描述

DCOP 算法的主体模块包括以下三个步骤：

第 1 步：分别求 G 中任一节点 y 到目的节点 *t* 基于代价和时延的最优路；判断 $D_{min}(s,t)$是否大于约束条件，若大于则算法停止(图 5-3(a)3 行)；判断 $D_{cmin}(s,t)$ 是否满足约束条件，若满足则算法停止(图 5-3(a)5 行)；记录 $D_{min}(y,t)$、$C_{min}(y,t)$和 $C_{dmin}(y,t)$，以备主约束计算用。

第 2 步：调用 C_m 求解模块，求得算法初始的代价限定条件 C_m。

第 3 步：调用主约束计算模块，计算从源节点 *s* 到目的节点 *t* 的满足约束条件的最优路径。

Cm 求解模块采用修改最小代价路的方法获得，模块从路的目的节点 *t* 开始，依次检查每一个节点。具体实现为：对于已检查的节点 x，首先获得 x 的下一个节点 y(图 5-3(b)5 行)；然后判断路径 $p(s,y)$的时延值与 $D_{min}(y,t)$之和是否满足约束条件，若满足则判断其代价值与 $C_{dmin}(y,t)$之和是否小于 C_m，如小于，则替代之(图 5-3 (b)6-8 行)。

主约束计算模块采用类似于 Dijkstra 算法的方法来进行最优路求解。该模块在计算中需维护一个临时路集合 Path_Set，将当前满足条件的路（可行解）加入该集合，其具体实现为：

假设当前的最优路为 $p(s,x)$，对于 x 的邻接节点 y，首先检查 y 是否已在路 $p(s,x)$

中，若在则表示路径 $p(s,y)$存在环路，该路无效(图 5-3 (c)7 行)；计算路径 $p(s,y)$的时延值与 $D_{min}(y,t)$之和，若约束超限，则路 $p(s,y)$不是可行解 (图 5-3 (c)8 行)；计算路径 $p(s,y)$的代价值与 $C_{min}(y,t)$之和，若约束超限，则路 $p(s,y)$不是可行解 (图 5-3 (c)9 行)；动态调整 C_m，对于可行解 $p(s,y)$，计算其代价值与 $C_{dmin}(y,t)$之和，如小于 C_m，则替代之(图 5-3 (c)10-12 行)；最后，将路 $p(s,y)$的时延和代价值与节点 y 已记录的时延和代价值相比，若不都大于或等于则将该路加入 Path_Set 中，并将路 $p(s,y)$加入 $V_p(y)$中(图 5-3 (c)13-16 行)。

模块确定当前的最优路径的原则是：临时路径集合中代价最小的路径，如果几条都具有同样的最小代价，则判断这几条路的传输时延与该路最外一点 y 到 t 的最小时延值之和，取最小的作为当前最优的路径(图 5-3 (c) 14 行)。

模块停止的条件为：找到符合条件的解或当前 Path_Set 为空(图 5-3 (c) 3 行和 5 行)。

```
DCOP（G(V,A),D,C, s,t, Dr)
1 BEGIN
2 Dijkstra (G(V,A),D, t)
3 IF Dmin(s,t)> Dr THEN RETURN FAIL
4 Dijkstra (G(V,A),C, t)
5 IF Dcmin (s,t)≤ Dr THEN RETURN p(s,t)
6 Cm = Get_Cm(p(s,t))
7 Main_Calculation(G2(V,A),D,C, s,t, Dr)
8 RETURN p(s,t)
9 END
```

图5-3(a) 主体模块伪码

```
Get_Cm(p(s,t))
1 BEGIN
2 Cm = Cdmin(s,t)
3 y = t
4 WHILE(y≠s) DO
5 y = NEXT_NOTE(p(s,t),y)
6 IF D(p(s,y))+Dmin (y,t)≤Dr AND C(p(s,y))+Cdmin (y,t)< Cm
7 Cm = C(p(s,y))+Cdmin (y,t)
8 ENDIF
9 ENDDO
10 RETURN Cm
11 END
```

图 5-3（b） Cm 求解模块伪码

```
Main_Calculation(G(V,A),D,C, s,t, Dr)
1 BEGIN
2 Initialize(Path_Set, Vp(i)i∈V)
3 WHILE(Path_Set≠F）DO
4 p(s,x)= Remove_ Min_ Path (Path_Set)
5 IF(x = t) THEN RETURN p(s,x)
6 FOR y∈Γx DO
7 IF Loop_Det (p(s,x),y) FAIL THEN CONTINUE
8 IF D(e(x,y))+ D(p(s,x))+ Dmin(y,t)> Dr THEN CONTINUE
9 IF C(e(x,y))+ C(p(s,x))+ Cmin(y,t)> Cm THEN CONTINUE
10 IF C(e(x,y))+ C(p(s,x))+ Cdmin(y,t)< Cm THEN
11 Cm = C(e(x,y))+ C(p(s,x))+ Cdmin(y,t)
12 ENDIF
13 IF Check_Path_Vp(Vp(y)，p(s,y)) FAIL THEN
14 Inset_Path(Path_Set, p(s,y))
15 Inset_Path_Vp(Vp(y)，p(s,y))
16 ENDIF
17 ENDFOR
18 ENDDO
19 END
```

图 5-3（c） 主约束计算模块伪码

5.3.2.4 DCOP 算法性能分析

算法的计算复杂性主要体现在第一步和第三步，其中第一步需要用两次 Dijkstra 算法，其计算复杂度为 $O(|V|^2)$。第三步主约束计算模块的计算复杂度与算法的搜索空间有关，假设该空间的节点数目为 n，节点的最大邻居数目为 E_m，每个节点保留的最大路径数目为 L_m，则主约束计算模块最多进行$(n-2)L_m$ 次主循环(图 5-3 (c) 3 行)。每次主循

环需要最多 E_m 次副循环(图 5-3 (c) 6 行)，每次副循环中最耗时的是图 5-3 (c)的第 14 行，需要最多(n-2)L_m 次比较，所以该模块的计算复杂度为 $O(E_m n^2 L_m^2)$，整个算法的计算复杂度[1]为 $O(|V|^2+ E_m n^2 L_m^2)$。

由上面的分析可知，主约束计算模块的计算复杂度主要与 n 和 L_m 有关，由于算法在保证最优解的同时有效的降低需要搜索的空间，此时的 n 和 L_m 一般很小，算法复杂度接近 $O(|V|^2)$。表 1 中给出了在节点数$|V|$为 100-1000 时不同规模的网络，本算法和 HMCOP 算法的主约束计算模块在最坏情况下所需要的最多比较次数，取得该数据的测试条件参见下一节的实验 2，最终结果为得到的 10000 个数据中的最大一个。

表 5-1　两种算法主约束计算模块所需最多比较次数表（万次）

$\|V\|$ /个	100	200	300	400	500	600	700	800	900	1000
HMCOP	7.5	34.7	72.9	153	233	358	432	636	831	1040
SDCOP	0.9	3.4	5.6	10.5	13.2	14.5	17.1	20.4	21.9	23

表 5-1 可看出在最坏的情况下，本算法主约束计算模块所需要的最多比较次数约等于节点数的平方，整个算法的计算复杂性约为 $O(|V|^2)$，与经典的 Dijkstra 算法计算复杂性大体相当。而同样条件下的 HMCOP 算法计算复杂性就要远远大于 $O(|V|^2)$，本算法在计算复杂度方面比 HMCOP 算法至少改进了一个数量级，详细的仿真结果见下一节。

5.3.3 仿真分析

本节给出了 DCOP 在算法运行效率和性能方面的仿真测试，并对比同类型的已有算法 HMCOP。其设计如下：每组实验测试十个不同的拓扑，拓扑采用 Transit-Stub 模型随机生成[112]，每个网络中的链路代价在范围[0,1]内随机选取，时延在范围[0,20]内随机选取。实验针对每个网络都产生 1000 次不同的请求，每次请求包括源宿节点和时延约束值，源宿节点是在所有节点中随机选取，而时延约束值则在[D_{min}, D_{cmin}]内随机取得，这样设计是考虑到在 $D_{min} > D_r$ 和 $D_{cmin} \leqslant D_r$ 时，两种算法都只执行了第一步，在耗时和性能方面没有区别。每组实验针对不同的测量内容可得到 10000 个数据，最终的结果取这些数据的均值。仿真测试分为如下三组实验。

5.3.3.1 算法运行效率和性能比较

本组实验给出网络节点规模（节点数为 500）相同时，两种算法在时间、空间复杂

[1] 注：若采用 4.5 节的最短路径优化算法，则计算复杂度还会小一些。

度和算法性能三方面的对比结果。由于 HMCOP 算法在运行中每个节点保留的最大路径数 Vpr 将直接影响算法的效率和性能，实验分别给出 Vpr=1,…20 时的结果。

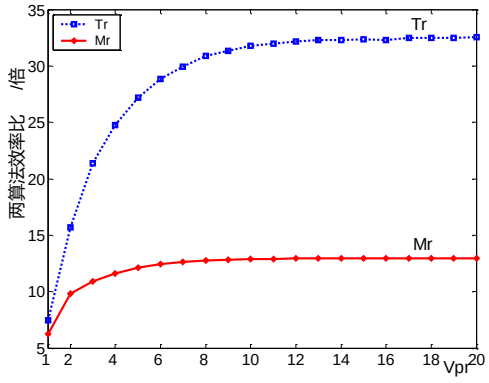

图 5-4（a） Tr、Mr 实验结果

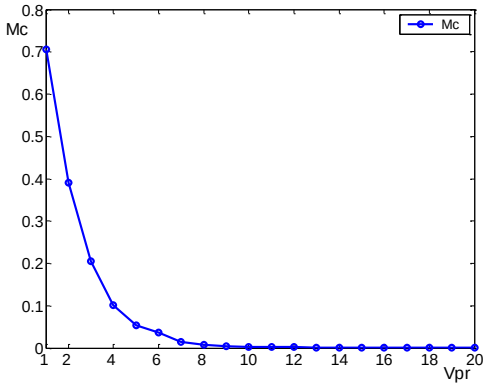

图 5-4（b） Mc 实验结果

图 5-4 中 Tr 表示算法的耗时比，Tr=（HMCOP 主约束计算模块耗时）/（DCOP 主约束计算模块耗时），主要考虑算法主约束计算模块的耗时情况，其大小等于每次调用相应模块时，调用前后的计算机时间差。Mr 表示两种算法所需的储存空间比，主要考虑算法在计算过程中所需要的临时队列的大小，其计算表达式为：Mr=（HMCOP 临时队列大小）/（DCOP Path_Set 队列大小 $+V_p$ 队列大小）。Mc 表示两种算法寻路性能比，比较两种算法得到的路径的代价，代价越小的得到的路径越优，其计算表达式为：Mc=（发生 DCOP 得到的代价小于 HMCOP 得到的代价的情况的次数）/10000。

由图 5-4 可得当 Vpr=1 时，HMCOP 算法的效率接近 DCOP 算法（Tr= 7.5），但此时 DCOP 算法有 70.5%的解优于 HMCOP 算法，随着 Vpr 的变大，存在两种相反的变化趋势：Tr 迅速上升（图 a）而 Mc 迅速下降（图 b），在 Vpr=10 时 Tr 在 32 附近趋于平稳（此时两种算法找到的最优解基本相同，Mc→0），而 Vpr=8 后 Mc 趋于零。结果表明 DCOP 算法在计算复杂度方面有明显改善，同时在空间复杂度方面也得到了很好改善（Mr=10 左右）。

实验中也记录了 DCOP 得到的路的代价值大于 HMCOP 的情况，其结果为零，说明 DCOP 算法在最优解求解性能方面也存在相当优势。

5.3.3.2 算法对网络规模变化的适应性比较

本组实验给出网络规模对两种算法影响情况的对比分析。实验以节点数为 100 时每种算法的主约束计算模块耗时和所需要的临时队列大小为基准，记为 Bt 和 Bm。分别给出节点数为 100-1000 时每种算法的主约束计算模块耗时相对 Bt 的大小，即 Ts =（主

约束计算模块耗时）/Bt，Ms = (临时队列大小)/ Bm。实验中 HMCOP 的 Vpr 取值为 20。

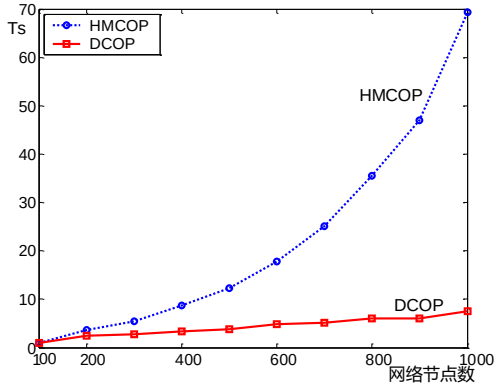

图 5-5（a） Ts 实验结果

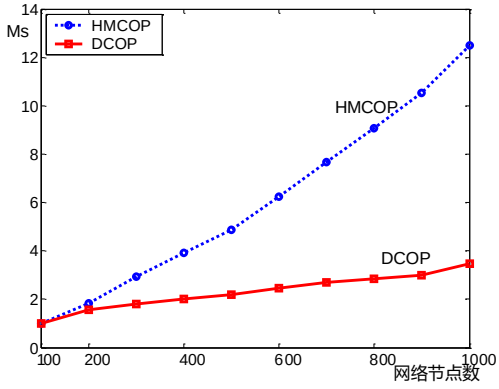

图 5-5（b） Ms 实验结果

由图 5-5 可得，随着网络规模的变大，两种算法的 Ts、Ms 都呈上升趋势。其中 HMCOP 算法的 Ts、Ms 变化剧烈，且呈发散状态，而 DCOP 算法的则变化缓慢，且趋于平稳。由此可见随着网络节点数目的增加，HMCOP 算法运行效率将迅速下降，其算法对网络规模的适应性远不如 DCOP 算法。

5.3.3.3 算法效率与约束条件“严格”程度的关系

HMCOP 算法的运行效率与 D_r 和 D_{min} 相对位置有关，称为算法约束条件“严格”程度，D_r 越逼近 D_{min} 则约束条件越“严格”。本组实验给出不同“严格”程度对两种算法影响情况的对比分析。实验将 “严格”程度分为 10 级，其取值 $C_i=D_{min}+(D_{cmin}-D_{min})\times(2\times i-1)/20,(i=1,\ldots10)$。以 i=1 时 DCOP 的主约束计算模块耗时和所需要的临时队列大小为基准，记为 Bt 和 Bm，分别给 i=1,…10 时每种算法的主约束计算模块耗时相对 Bt 的大小，即 Th=（主约束计算模块耗时）/Bt，Mh=(临时队列大小)/ Bm。此时 HMCOP 中的 Vpr 取值为 10，网络节点数为 500。

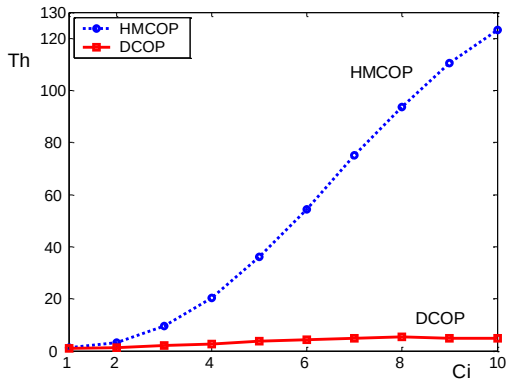

图 5-6（a） Th 实验结果

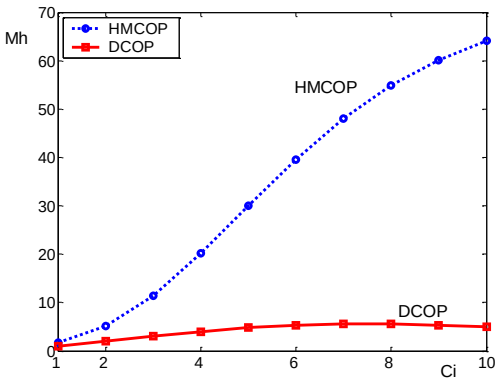

图 5-6（b） Mh 实验结果

由图 5-6 可得在约束条件非常“严格”（i=1）时，HMCOP 的效率接近 DCOP（Th=

1.2、Mh=1.7），但随着约束条件的“放宽”，HMCOP 算法的效率迅速下降，而由于 DCOP 算法的自适应性，算法对约束条件的“严格”程度不敏感。实际应用中约束条件是与业务相关的，这使得此时的 HMCOP 算法无法针对各种业务都取得很好的效果，算法实用性不如 DCOP 算法。

5.3.4 结论

基于时延约束的最优路径求解算法在目前网络通信中具有十分重要的应用，一直以来都没能得到很好的解决。本章采用自适应参数设计，有效的减小搜索空间，给出一种 DCOP 算法，仿真表明该算法不仅在取得最优解、计算复杂度等关键指标上优于 HMCOP 算法，且具有适用于更大的网络规模、更复杂的业务类型、容易实现以及得到的最优路无环等优点。该算法已经用 C++实现，可应用于各种通信网络，尤其适用于网络资源有限、节点计算能力弱、链路延迟较大的卫星网络和无线移动网络。

DCOP 算法不仅可有效解决单约束最优路径问题，经过简单修改，该算法也可扩展到求解多约束最优路问题上。

5.4 卫星 MPLS 网络中的 QoS 最优寻径问题

卫星网络具有链路长时延、拓扑时变以及节点计算资源有限等特点，因此，针对卫星 MPLS 网络的 QoS 最优寻径算法须同时考虑这些特点对 QoS 路由性能的影响。5.3 节重点针对网络资源有限、节点计算能力弱、链路延迟较大等特性，给出了相应的解决算法——DCOP 算法。但对于卫星网络还需要考虑拓扑时变特性，以保证建立的 QoS 最优路具有相当的时效性和稳定性，这正是本节工作的重点。

卫星 MPLS 网络 QoS 路由问题需同时考虑链路时延长和拓扑时变等特性，而拓扑的时变性可最终影响网络路径费用的大小，这是因为链路切换会给网络带来巨大的额外开销（如重路由带来的路由和信令开销等）。基于这种思路，本节通过将网络切换引入路径费用中，来寻找一条既满足业务时延要求又可尽量降低链路切换对网络性能影响的最优 QoS 路径，并在 DCOP 算法的基础上，给出一种适用于卫星节点的时延受限最小费用路由(SDCMCR)算法来解决在卫星链路时延长的情况下寻找满足时延限制条件且受切换影响最小的路径的问题，该算法可兼顾网络业务中断率和业务阻塞率等性能，且计算复杂度低，可适应星上设备处理能力的要求。

5.4.1 费用模型

本算法所使用的网络模型见定义 3-1。这里利用定义 3-3 中的路的生存时间来描述两节点间可持续通信的时间，记为 $TL_{ij}(t)$，且有如下关系存在：对于网络 G，若在周期 T 内节点 i 与 j 一直相连（如轨内星间链路），则 $d_{ij}(t)$为一连续函数，$TL_{ij}(t)=\infty$；若节点 i 与 j 会发生切换（如轨间星间链路），$d_{ij}(t)$为以切换时间为分段点的分段函数，设分段点为 $t_1, t_2\ldots$，若 $t\in[t_1, t_2]$且 $d_{ij}(t)\neq\infty$，则 $TL_{ij}(t)= t_2- t$。

t 时刻，若 η 为任意边 e(i,j)上的某一业务流，且该业务还需要持续的时间为 x，如 $x>TL_{ij}(t)$，则该业务将会因网络的切换而中断，将这种中断的概率定义为业务 η 的中断概率，记为 $P_\eta(t)$，其大小由 t 时刻该业务的持续时间分布模型 ξ 和 $TL_{ij}(t)$决定，可表示为：$P_\eta(t)=f(\xi, TL_{ij}(t))$，这里，ξ 由不同的业务决定，如 Poisson 模型（语音通信）等。

为了保证链路中断后业务的 QoS，需要采取重路由或链路带宽预留等方法，这都会给网络带来巨大的额外开销。因此可以将这种链路切换对业务的 QoS 以及网络的影响折合为该链路针对业务 η 的额外费用。不过这种影响不仅与 $P_\eta(t)$有关，还与当前网络的负荷情况以及业务对路径可靠性的敏感程度等因素有关。这是因为当网络的负荷率比较高的时候，链路切换时无论是重新申请带宽还是提前预留带宽都比较困难，切换后不仅业务的 QoS 很难保证，还会对此时的网络流量分布带来巨大影响。

根据上述分析，则考虑切换影响后 e(i,j)针对业务 η 的费用模型 $c_\eta(t)$可以表示为式（5.1），该模型基于如下五个假设：

1）e(i,j)因切换而带来的额外费用 $c_\eta(t)$是以网络的负荷率、$P_\eta(t)$以及业务对路径可靠性的敏感程度为变量的函数，其中，$P_\eta(t)$是影响 $c_\eta(t)$大小的主要因素；

2）$c_\eta(t)$应满足如下边界条件：当 $P_\eta(t)=0$ 时，$c_\eta(t)$等于 e(i,j)上业务 η 的正常费用（即未考虑切换时的费用 $c(t)$）；当 $P_\eta(t)\to1$，$c_\eta(t)\to\infty$；

3）$c_\eta(t)$ 的大小与网络目前的负荷情况有关，当网络的负荷率比较高的时候，链路切换时无论是重新申请带宽还是提前预留带宽都比较困难，这时切换对业务 QoS 和网络性能的影响相对较大，即此时 $c_\eta(t)$较大；反之，$c_\eta(t)$较小；

4）$c_\eta(t)$ 的大小还与具体的业务 η 有关，对于一些对切换不“敏感”（即受切换影响小）的业务，如数据业务，$c_\eta(t)$值应该较小，否则，$c_\eta(t)$值较大，这里再引入一个参数 γ 来描述不同业务对 $c_\eta(t)$的影响，γ 的具体取值参见 5.4.4.3 节；

5）$P_\eta(t)$等对业务 QoS 的影响不是呈线性关系，当 $P_\eta(t)$ 或网络负荷率越接近 1，这

种影响将越明显，假设这种关系为指数关系。

$$c_\eta(t)=\frac{e^{(\gamma\times\alpha\times P_\eta(t))}}{1-P_\eta(t)}\times c(t) \qquad (5\text{-}1)$$

其中 $c(t)$为 e(i,j)上业务 η 的正常费用（未考虑切换时），当 $P_\eta(t)=0$ 时，$c_\eta(t)=c(t)$；γ 为业务因子，表示业务对路径可靠性的敏感程度，γ>0；α 为网络负荷率，α∈[0,1]。

为了便于研究和描述，本节假设：给定时刻 t、G（$V,E,D(t)$）和业务 η。则与时间 t 有关的函数都可用一个常数来表示（如 $P_\eta(t)$可表示为 P_η）。

5.4.2 SDCMCR 问题描述

若 G 中任意边 e(i,j)∈E，存在非负的 d_{ij} 、P_η 和 c_η，则卫星时变网络中的时延受限最小费用路由(SDCMCR)问题可描述为：在图 G 中，寻找一条从源节点 s 到目的节点 t 的路径 U(s,t)，满足：D(U(s,t))≤D_r，且 min(C(U(s,t)))。

这里 D(U(s,t))为路 U(s,t)的时延值，其值等于该路各边 d 之和，D(U(s,t))= $\sum_{e\in U(s,t)} d$；C(U(s,t)) 为路 U(s,t)的费用值，其值等于该路各边 c_η 之和，C(U(s,t))= $\sum_{e\in U(s,t)} c_\eta$ 。

5.4.3 SDCMCR 算法设计

SDCMCR 算法的主体模块包括以下三个步骤：

第 1 步：算法初始化。首先根据相应的业务 η 和业务到达时间 t，计算出卫星网络 G 各边的时延 d、中断概率 P_η 和费用 c_η。其次，分别求 G 中任一节点 y 到目的节点 t 基于费用和时延的最优路；判断 D_{min} (s,t)是否大于约束条件，若大于则算法停止；判断 D_{cmin} (s,t)是否满足约束条件，若满足则算法停止；记录 D_{min} (y,t)、C_{min} (y,t)和 C_{dmin}(y,t)，以备主约束计算用。

第 2 步：调用 DCOP 算法的 C_m 求解模块（见图 5-3（b）），求得算法初始的费用限定条件 C_m。

第 3 步：调用 DCOP 算法的主约束计算模块（见图 5-3（c）），计算从源节点 s 到目的节点 t 的最优路径。

5.4.4 仿真分析

本节给出了 SDCMCR 在算法运行效率和性能方面的仿真测试，并对比多种同类型

的已有算法。其设计如下：（1）实验所用的卫星网络轨道参数选用美国的 Teledesic 移动卫星系统，共 288 颗卫星均匀分布在 12 个轨道面，轨高 1375km，倾角 84.7°，每颗星周围有 8 颗邻居，为十字架状分布；（2）业务 η 为简单流，其平均业务持续时间 μ=3 分钟，则 G 中边 e(i,j)的中断概率可表示为：$P_\eta = e^{-(TL_{IJ}/\mu)}$；（3）网络节点的排队过程为最简单流损失制模式，业务报平均服务时间为 1ms，则两节点间的时延可近似表示为：(x/*c*+1/(1-α))ms，其中 x 为节点间距离，*c* 为光速，α 为网络负荷率；（4）针对该卫星网每次实验都随机产生 10000 次不同的请求，每次请求包括：请求发生的时间 *t*、源宿节点对、时延约束值和 *c*(*t*)，其中源宿节点是在所有节点中随机选取，而 *t* 在[0,T]中随机选取，*c*(*t*)在[5,10]中随机产生。每组实验针对不同的测量内容可得到 10000 个数据，最终的结果取这些数据的均值。仿真测试分为如下三组。

5.4.4.1 算法运行效率和性能比较

本组实验给出了本算法和 HMCOP 算法在 Teledesic 移动卫星系统网络模型下时间、空间复杂度和算法性能三方面的对比结果。考虑到在 $D_{min} > D_r$ 和 $D_{cmin} \leqslant D_r$ 时，两种算法都只执行了第一步，在耗时和性能方面没有区别，实验中的时延约束值在[D_{min}, D_{cmin}]内随机取得。由于 HMCOP 在运行中每个节点保留的最大路径数 Vpr 将直接影响算法的效率和性能，实验分别给出 Vpr=1,…20 时的结果。

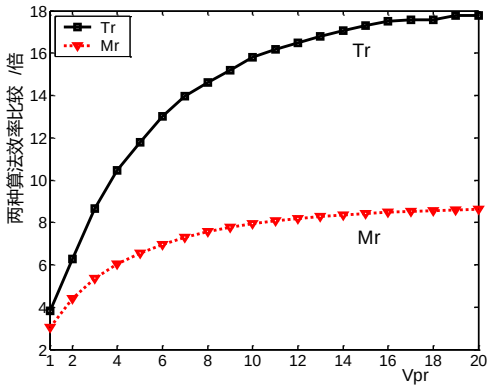

图 5-7（a） Tr、Mr 实验结果

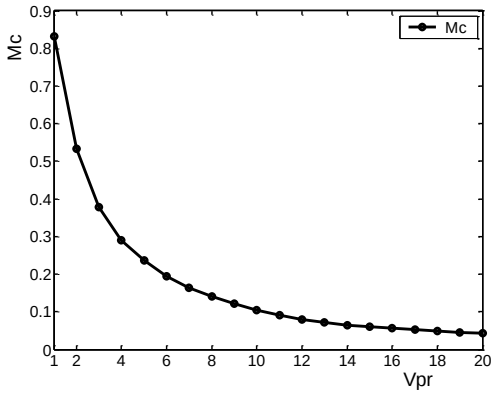

图 5-7（b） Mc 实验结果

图 5-7 中 Tr 表示算法的耗时比，Tr=（HMCOP 主约束计算模块耗时）/（SDCMCR 主约束计算模块耗时），主要考虑算法主约束计算模块的耗时情况，其大小等于每次调用相应模块时，调用前后的计算机时间差。Mr 表示两种算法所需的储存空间比，主要考虑算法在计算过程中所需要的临时队列的大小，且有：Mr=（HMCOP 临时队列大小）/（SDCMCR Path_Set 队列大小 $+V_p$ 队列大小）。Mc 表示两种算法寻路性能比，比较两

种算法得到的路径的费用，费用越小的得到的路径越优，且有：Mc=（发生 SDCMCR 得到的费用小于 HMCOP 得到的费用的情况的次数）/10000。

由图 5-7 可得当 Vpr=1 时，HMCOP 的效率接近 SDCMCR（Tr=3.8），但此时 SDCMCR 有 83%的解优于 HMCOP，随着 Vpr 的变大，存在两种相反的变化趋势：Tr 迅速上升（图 5-7（a））而 Mc 迅速下降（图 5-7（b）），在 Vpr=12 时 Tr 在 17 附近趋于平稳，而 Mc 也趋于零。结果表明 SDCMCR 在算法计算复杂度方面有明显改善，同时在空间复杂度方面也得到了很好改善（Mr=8 左右），所以，SDCMCR 比 HMCOP 更能适应星上设备处理能力有限的实际情况，可以用于星上在线 QoS 路由计算。

5.4.4.2 业务阻塞率以及中断率比较

本组实验给出了本算法和 PRP 算法、未考虑链路切换的路由算法（如 GALPEDA 算法）在不同网络负荷率下，业务因切换而被中断的概率以及业务请求被阻塞情况的比较结果。实验中的时延约束值 D_r=100ms，γ=5，PRP 算法的路径中断概率门限值为 0.1。

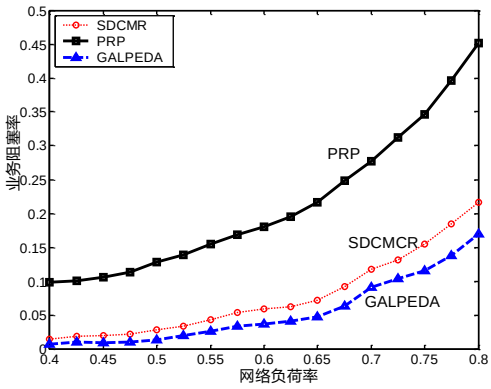

图 5-8（a） 三种算法业务阻塞率比较结果

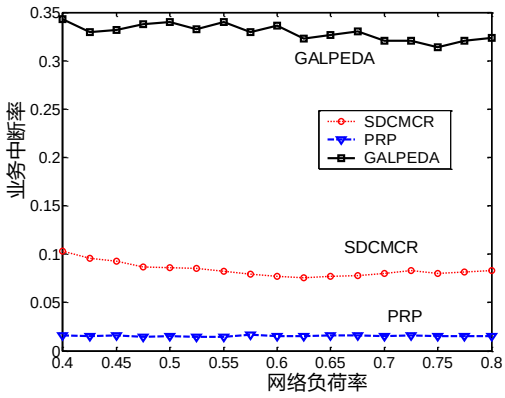

图 5-8（b） 三种算法业务中断率比较结果

由图 5-8（a）可得随着网络负荷率的增加，三种算法业务请求的阻塞概率都呈上升趋势，不过很明显，SDCMCR 在这方面的性能要明显好于 PRP，与 GALPEDA 大体相当。这是因为 PRP 在计算时会剪切掉中断概率高的链路，使得业务过于集中在某些链路而造成较高的阻塞概率，同时这也会带来网络流量分布不均，网络整体利用率低等缺点，而 SDCMCR 算法则可克服这些缺点，且该算法中业务因网络切换而中断的概率要远远小于没有考虑未来拓扑切换情况的 GALPEDA（图 5-8（b）），因而可有效降低切换对业务 QoS 性能的影响。结果表明 SDCMCR 算法能很好的兼顾网络业务阻塞率和业务中断率两方面的性能，尤其适合于切换频繁的移动卫星网络。

图 5-8（b）还可看出随着负荷率 α 的增加，业务中断率呈下降趋势，这也有利于算

法在较高网络负荷时的使用。

5.4.4.3 γ 值的选取对业务阻塞率和中断率的影响

γ 表示业务对路径可靠性的敏感程度，它与具体的业务有关。本组实验给出了 γ 值与业务阻塞率和中断率的关系，由此可得出不同的业务 γ 值可能的取值。实验中的时延约束值 D_r=100ms，α=0.5。

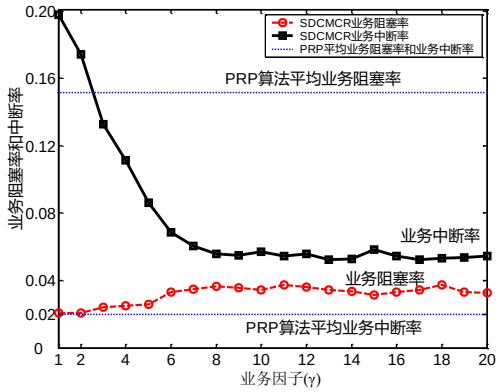

图 5-9　γ 值与业务中断率和阻塞率的关系

由图 5-9 可得随着 γ 值的增加，算法中业务因网络切换而中断的概率将逐渐下降，当 γ=9 后，中断率逐渐稳定在 0.04 左右，接近 PRP 的中断率，与此同时，业务请求的阻塞概率也略有上升，但仍远远好于 PRP。实验结果表明，对于某些业务（如数据业务），网络可以通过灵活的路由策略来降低切换对业务的影响，此时算法的 γ 可取较小的值（如 2 或 3），以牺牲路径的稳定性来获取其他方面的利益（如阻塞率、网络流量更均衡等）；而对于某些业务（如语音业务），链路切换需要网络采用信道预留等高代价方式来平滑，此时算法的 γ 可取较大的值（如 6 或 7），以尽量降低业务的中断概率。

5.4.4.4 算法业务中断概率分析

算法的业务中断概率与该业务需要服务的时间是密切相关的，一般来说，业务需要服务的时间越长，在该服务时间内发生中断的可能性将越大。本组实验给出了算法的业务中断概率与业务平均持续时间 μ 的关系，实验中的时延约束值 D_r=110ms，α=0.5，γ = 10。

如图 5-10 所示，随着 μ 的增加，网络中业务中断概率也将增加，对于 μ 比较小的业务（如语音 μ≤5*min*），本算法可以较好的保证该业务的 QoS，业务因切换而中断的概率一般都为 6%~8%左右，对于业务平均持续时间 μ 小于 10 分钟的业务，算法都基本上可以保证业务因切换而中断的概率不会超过 10%；而对于 μ 比较大的业务（如视频），

这时网络中业务因为切换而中断的概率将比较大（大于 20%，有的甚至可达 50%），因此，还需要采用路径保护等方法来进一步保证业务传输的 QoS，这方面的研究内容，将在下一章讲述。

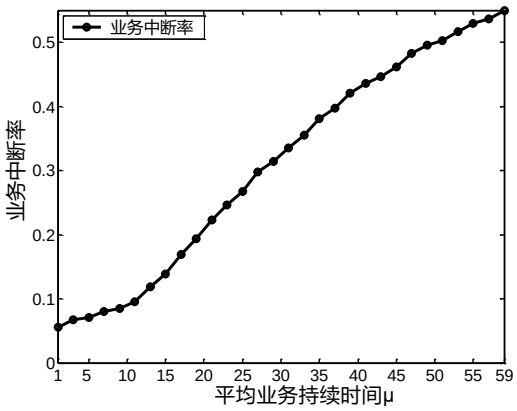

图 5-10　业务中断率与 μ 的关系

5.4.5 结论

本节给出的 SDCMCR 算法充分考虑了卫星 MPLS 网络大时延、拓扑时变以及星上处理能力有限等特点，重点解决在这种特殊环境下寻找满足时延限制要求的最优路径问题。算法首先通过引入业务的中断概率来预测卫星网络的动态性，并将链路切换对业务的 QoS 以及网络的影响折合为该业务在链路上的额外费用，给出相应的费用模型，最后通过寻找满足时延限制要求的最小费用路径来有效降低链路切换对业务的 QoS 以及网络流量分布的影响。该算法采用自适应参数设计、有效减小搜索空间的方法来有效降低算法的复杂度，以适应算法对星上计算和储存能力的要求。仿真表明算法不仅在计算复杂度方面优于 HMCOP 算法，且能很好的兼顾网络业务阻塞率和业务中断率两方面的性能，一方面算法中业务因切换而中断的概率大大下降，另一方面在业务请求被阻塞的概率和网络流量均衡性等方面本算法也优于同类型的 PRP 算法。因此，与同类型的算法相比，更具有实用性。

5.5 本章小结

QoS 路由技术在宽带卫星通信网络中具有十分重要的意义，一直以来都未能得到很好的解决，尤其是对于 QoS 路由技术中的难点——QoS 路由寻径计算问题。由于卫星网络的大时延、拓扑时变以及星上处理能力有限等特点，使得该问题的解决变得更具挑战性。

针对上述难点，本章重点解决卫星 MPLS 网络 QoS 路由寻径问题中最有实际意义的时延受限最优寻径问题。首先给出一种计算复杂度低且不会牺牲寻径性能的时延受限最优路径(DCOP)算法，并在此基础上，针对卫星拓扑时变的特点，给出一种适用于卫星节点的时延受限最小费用路由(SDCMCR)算法。该算法具有以下五方面有点：1）可得到满足约束条件的最优无环解，且计算复杂度小，接近无约束的 Dijkstra 算法；2）以时延为约束条件，尤其适合于链路时延大的网络，如卫星网络；3）可有效降低链路切换对 QoS 以及网络性能的影响；4）在保证业务 QoS 的同时，可兼顾网络资源的有效利用，可以更好的保证卫星网络宝贵的网络资源不被浪费；5）采用自适应动态调整技术，算法适应性、扩展性好。与此同时，由于算法采用通用的卫星网络模型（见第三章），使得该算法可适用于多种卫星网络（无论星座构型如何，只要具有长时延、拓扑时变等本文所给出特点，算法都可适用）。这种通用性设计，对卫星网络 QoS 路由的进一步研究和建设，无疑都是有利的。

上述算法已用 C++实现，仿真表明算法不仅在计算复杂度等方面优于成熟的 HMCOP 等算法，而且在性能方面也优于目前卫星网络常用的 QoS 路由算法，如 GALPEDA 算法和 PRP 算法等。