一种改进的多约束最佳路径算法研究

王 晟,李乐民
(电子科技大学通信与信息工程学院宽带光纤传输与通信系统技术重点实验室,四川成都 610054)

摘 要: 本文描述了MPLS网络中的多约束最佳路径问题,以及该问题的各种变型.分析了现有的解决这些问题的算法存在的各种缺陷,并针对一般性的多约束最佳路径问题的解法,提出了改进的具体措施.通过具体的实例分析和计算机仿真,验证了新算法在性能上的改善,主要的性能指标包括路径计算的成功比率和路径的平均代价等.结合仿真结果讨论了算法中涉及到的关键参数对算法性能的影响.

关键词: 多约束最佳路径;约束选路;多协议标记交换
中图分类号: TP301.6 **文献标识码:** A **文章编号:** 0372-2112 (2004) 04-0529-07

An Enhanced Algorithm for Multiple Constraints Optimal Path Calculation

WANG Sheng, LI Le-min
(*Key Lab of Broadband Optical Fiber Transmission and Communication Networks, UESTC, Chengdu, Sichuan* 610054)

Abstract: One of the key issues in the design of a QoS-oriented service is how to identify a feasible route which satisfies multiple constraints while simultaneously achieving efficient utilization of network resource. The underlying problem can be stated as a Multiple Constraints Optimal Path (MCOP) problem. Recent years, some algorithms have been proposed in literature, but suffered from various disadvantages. They can not meet the practical requirements. In this paper, a formal definition of MCOP problem is stated. A novel enhanced heuristic for solving MCOP problem is proposed, in which elegant path searching techniques are used to achieve higher performance in terms of path selection success ratio and average path cost. Using extensive simulations on random graphs and random assigned link weights, the improvement of the new algorithm proposed is verified, and the impact of some key design parameters on the performance is discussed.

Key words: multiple constraints optimal path; constraint-based routing; multi-protocol label switching

1 引言

目前的因特网是针对"尽力传送"(best effort)业务设计的.随着因特网迅速扩展,各种类型的业务需求不断出现,单一的服务模型已不能满足网络发展的要求.因此,在因特网中提供可保障服务质量(Quality of Service, QoS)的服务类型已成为以因特网为代表的分组网络发展的方向.提供QoS保障的服务除了需要定义各种有效的服务模型(如Diffserv、Inteserv[1]等)外,一个重要的研究方向是如何提供高效率的转发技术.目前,MPLS[2](Multiprotocol Label Switching,多协议标记交换)被公认为是适应未来IP网络发展的转发技术.

MPLS技术的核心思想是为业务流(flow)先建立LSP(Label Switching Path,标记交换路径),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.这就是MCOP(Multiple Constraints Optimal Path,多约束最优路径)问题.该问题可以这样描述:给定网络拓扑结构、资源状态信息,和要求建立路径的源和宿节点;要求找到一条路径,该路径不仅满足各种约束(如跳数、延迟、带宽等),而且是所有满足这些约束的路径(这种满足约束的路径称为可行路径)中"最优"的,优化指标是除各种约束外另一个预先给定的参数,称为主代价.

在MCOP问题中,网络中的每条链路都被赋予一组参数,用于表达各种约束.这些参数可以分为两类,一类是加性的(additive),一类是非加性的(nonadditive).对于加性约束,整条路径的约束属性由该路径经过的各链路相应约束参数之和决定(例如跳数、延迟等);而对于非加性约束,路径的约束值与链路约束参数之间不存在这样的叠加关系.已有研究表明[3],非加性约束往往可以利用预先裁剪的方式处理,因此,在寻路算法中需要考虑的只有加性约束.本文所研究的"多约束最优路径"问题中,约束指的就是加性约束.

当约束数目为1时,MCOP问题退化为RSP(Restricted-Shortest Path,受限的最短路)问题[4].RSP问题仍然要求计算主代价最优的路径,但约束只有一个.另一个与MCOP相关的

收稿日期:2002-08-16;修回日期:2003-09-28
基金项目:国家自然科学基金(No.60002004);教育部科学技术研究重点项目(No.02064)

是 MCP(Multiple Constraints Path,多约束路径)问题[5]. MCP 问题中约束有多个,但只要求计算满足约束的路径,不存在主代价最优的问题.这两个问题都可以看作 MCOP 问题的特例.求解这些问题的方法可以分为两大类.一类是将原问题简化[6,7],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所带来的算法效率问题.另一类方法[8,9]则求助于 k 路由算法[10]:构造一个代价函数,其中同时考虑了主代价和各种约束,通过 k 路由算法计算出前 k 条代价最小的路径,然后从它们中搜索出符合约束且最佳(或近似最佳)的解.不难发现,这两大类方法都有较大缺陷.首先,这些算法只能解决 MCP 或 RSP 问题,无法扩展到 MCOP 问题;另外,算法性能往往与具体参数的选择密切相关,如采用 k 路由算法时,k 值的选择非常关键:若 k 值较大,则难以保证算法效率,若 k 值太小,则可能无法求得最佳解.

近年来,陆续提出了一些针对 MCOP 问题的算法[11,12].其中[11]只能解决 2 个约束的情况,不能解决一般性的 MCOP 问题.而[12]中提出的 HMCOP(Heuristic for MCOP)算法摆脱了上述限制,希望解决一般性的 MCOP 问题.但是,该算法存在一个关键的问题,即算法中对反向和正向的路径记录都是不完备的(4.2 节将结合实际的例子阐述 HMCOP 的这个问题).综上所述,尽管 MCOP 问题的一些特例(如 RSP、MCP,或 2 个约束的 MCOP 问题)已有不少研究成果,但针对一般的 MCOP 问题的研究目前还处于起步阶段,文献[12]中提出的 HMCOP 算法虽然是针对一般的 MCOP 问题设计的,但存在缺陷.

本文的主要贡献在于,针对 HMCOP 的缺陷提出了改进的途径,得到的新算法称为 EHMCOP(Enhanced HMCOP)算法.仿真结果表明,EHMCOP 算法的性能比 HMCOP 算法有明显改善.本文的后续内容是这样安排的:第二节给出本文用到的一些符号的说明,并给出多约束最佳路径问题的标准描述.第三节简要介绍了本文提出的 EHMCOP 算法的基本思路,给出了整个算法的实现伪码.第四节分析了 EHMCOP 算法的复杂度,通过例子说明了对 HMCOP 算法的两个关键的改进措施,即"可行路径检查"和"正向多路径记录",并给出了仿真实验的方法以及结果.最后的第五节总结全文.

2 符号说明及问题描述

2.1 符号说明

●$G=(V,E)$: 有向图,其中 V 是顶点集合,E 是链路(边)的集合;

●$l(i,j)\in E$, $i,j\in V$:顶点 i 和 j 之间的一条边;

●$m(l(i,j))$, $l(i,j)\in E$:边 $l(i,j)$ 的主代价权重;

●$w_k(l(i,j))$, $l(i,j)\in E, k=1,2,\cdots,K$:边 $l(i,j)$ 的第 k 个约束权重;

●$p(i,j)$, $i,j\in V$:顶点 i 和 j 之间的一条路径,既可以看作所经过的顶点的集合,也可以看作所经过的边的集合;

●$m(p(i,j))$, $i,j\in V$: 路径 $p(i,j)$ 的主代价,是该路径上所有边主代价权重之和,即有

$$m(p(i,j))=\sum_{l(m,n)\in p(i,j)} m(l(m,n));$$

●$w_k(p(i,j))$, $l(i,j)\in E$, $k=1,2,\cdots,K$: 路径 $p(i,j)$ 的第 k 个路径约束参数,是该路径上所有边的第 k 个约束权重之和,即有

$$w_k(p(i,j))=\sum_{l(m,n)\in p(i,j)} w_k(l(m,n)),\quad k=1,2,\cdots,K;$$

●$s_c(i,j)=\sum_{k=1}^{K} w_{p,k}(i,j)$: 路径 $p(i,j)$ 上所有 K 个路径约束参数之和;为叙述方便,文中有时直接使用符号 s_c,不给出路径的端点;

●$Adj(x)$:节点 x 的所有邻接点集合;

●Heap:算法中的数据结构,用于维护计算过程中记录的路径集合;主要的操作包括 Insert_heap(将路径插入 Heap)和 Extract_min(从 Heap 中取出排序参数最小的路径,这里排序参数可以是主代价,也可以是路径约束参数等);

●$L_p(i,j)$:从节点 i 到节点 j 的路径集合,集合大小为 N_r(具体定义参见 3.3 小节);主要的操作为 Insert_path_list,向集合中插入新的路径;

●Normalize($C_k, k\in[1,K]$):对图中所有边的约束权重,按照路径约束值进行归一化处理;其中的 $C_k, k=1,2,\cdots,K$ 是路径的 K 个约束值.

2.2 问题描述

给定有向图 $G=(V,E)$,每个边都包含一个主代价权重:$m(l(i,j)), l(i,j)\in E$,以及 K 个加性约束权重:$w_k(l(i,j))$, $l(i,j)\in E, k=1,2,\cdots,K$.所有这些值都是非负的.给定 K 个路径约束值:C_k, $k=1,2,\cdots,K$.要求找到一条从源节点 s 到宿节点 t 之间的路径 $p(s,t)$,并且满足:

条件 1:$w_k(p(s,t))\leqslant Ck$, $k=1,2,\cdots,K$;

条件 2:$m(p(s,t))$是所有可行路径(满足条件 1 的路径称为可行路径)中最小的.

3 EHMCOP 算法描述

3.1 总体框架

EHMCOP 算法的主体模块主要包括以下三个步骤:

第 1 步 调用反向计算模块,计算图中每个节点到目的节点 t 的 s_c 最小的路径(图 1 中主体模块伪码的第 2 行).s_c 表示的是路径各约束参数之和(参见 2.1 小节).每条路径都被记录下来以备正向计算模块使用;

第 2 步 判断从源点 s 到目的点 t 的路径 $p(s,t)$是否超过约束,如图 1 主体模块伪码的第 3~5 行所示.注意,由于所有的约束参数都归一化了,所以当 K 个约束参数之和大于 K 时,必然有约束超过了限制.而一旦超过限制,则可以确认不存在满足约束的路径.因为所有其它的路径约束参数之和都比 $p(s,t)$大.若没有超过限制,执行第 3 步;

第 3 步 调用正向计算模块,计算从源点 s 到目的点 t 的最佳路径(图 1 主体模块伪码的第 6 行).

反向计算模块采用类似于 Dijkstra 算法[13]的方法实现,邻接点检查时累积的是链路约束参数(图 1(b)第 8 行),更新判决的依据是约束参数之和 s_c(第 9 行).反向计算模块的作用有两个,一是判断满足约束的路径是否存在;另一个作用是为每个节点 i 记录一条到目的点 t 的 s_c 最小的路径 $p_r(i,t)$,称为节点 i 的反向路径.正向计算过程中会根据这些反向路

径来判断是否存在从 s 到 t 的可行路径.

正向计算模块同样是仿照 Dijkstra 算法构造的.假定当前最佳路径为 $p(s,x)$,对于 x 的邻接点 y,构造路径 $T(s,y)$(图 1 正向计算模块 10 ~ 12 行),同时累积约束参数与主代价.然后检查 $T(s,y)$与反向计算得到的路径 $p_r(y,t)$约束参数之和,若约束不超限(图 1 正向计算模块 22 ~ 27 行),说明至少存在一条经 $T(s,y)$到达目的点 t 的总的可行路径(我们称路径 $T(s,y)$是可行的),故将 $T(s,y)$插入多路径集合中(关于多路径集合的具体描述参见 3.3 节).若约束超限,也不能认为 $T(s,y)$不可行,还需要进行进一步的可行路径检查(图 1 正向计算模块的 14 行,关于可行路径检查模块的具体实现参见 3.2 节),当可行路径检查模块发现存在其它可行路径时,仍将 $T(s,y)$插入多路径集合(图 1(d)15 ~ 20 行).

```
Enhanced_HMCOP(G(V,E),s,t,C_k,k∈[1,K])
1  BEGIN
2  Reverse_Calculation(G(V,E),t,C_k,k∈[1,K])
3  IF Σ_{k=1}^{K} w_k(p(s,t)) > K THEN
4  RETURN FAIL
5  ENDIF
6  Forward_Calculation(G(V,E),s,C_k,k∈[1,K]
7  IF w_k(p(s,t))≤1 ∀_k∈[1,K] THEN
8  RETURN p(s,t)
9  ENDIF
10  RETURN FAIL
11  END
```

图 1(a)　主体模块伪码

```
Reverse_Calculation (G(V,E),t,C_k,k∈[1,K])
1  BEGIN
2  Normalize (C_k,k∈[1,K])
3  Initialize(Heap)
4  WHILE(Heap≠Ø) DO
     //从 Heap 中取出 s_c 最小的路径
5  p_r(x,t) = Extract_min(Heap)
6  FOR y∈Adj(x) DO
7  T(y,t) = l(y,x) + p_r(x,t)
8  w_k(T(y,t)) = w_k(l(y,x)) + w_k(p_r(x,t)),k = 1,2,…,K
9  IF s_c(T(y,t)) < s_c(p_r(y,t)) THEN
10  p_r(y,t) = T(y,t);
     //按 s_c 的大小插入排序
11  Insert_heap(Heap,p_r(y,t))
12  ENDFOR
13  ENDFOR
14  ENDWHILE
15  END
```

图 1(b)　反向计算模块伪码

```
Further_check(T(s,y))
1  BEGIN
2  G'(V,E) = G(V,E) - T(x,y)
3  C' = C_k - w_k(T(s,y)),k = 1,2,…,K
4  Reverse_Calculation G'(V,E),y,t,C'_k,k∈[1,K]
5  IF w_k(p(y,t))≤1 ∀_k = 1,2,…,K THEN
6  RETURN SUCCEED
7  ENDIF
8  ELSE
9  RETURN FAIL
10  ENDELSE
11  END
```

图 1(c)　可行路径检查模块伪码

```
Forward_Calculation (G(V,E),s,C_k,k∈[1,K]
1  BEGIN
2  Normalize (C_k,k∈[1,K]
3  Initialize(Heap)
4  WHILE(Heap≠Ø) DO
     //从 Heap 中取出主代价最小的路径
5  p_{r,n}(s,x)) = Extract_min(Heap)
     //若当前标记点为终点 t,则终止循环
6  IF x = t THEN
7  BREAK
8  ENDIF
9  FOR y∈Adj(x) DO
10  T(s,y) = l(x,y) + p_{r,n}(s,x)
11  m(T(s,y)) = m(l(x,y)) + m(p_{r,n}(s,x))
12  w_k(T(s,y)) = w_k(l(x,y)) + w_k(p_{r,n}(s,x)),k = 1,2,…,K
13  IF max_{k=1}^{K} {w_k(T(s,y)) + w_k(p_r(y,t))} > 1 THEN
14  Further_check (T(s,y))
15  IF Further_check SUCCEED THEN
16  Insert_path_list (L_p(s,y)),T(s,y)
17  IF Insert_path_list SUCCEED THEN
18  Insert_heap(Heap,T(s,y)
19  ENDIF
20  ENDIF
21  ENDIF
22  ELSE
23  Insert_path_list (L_p(s,y),T(s,y))
24  IF Insert_path_list SUCCEED THEN
25  Insert_heap(Heap,T(s,y))
26  ENDIF
27  ENDELSE
28  ENDFOR
29  ENDWHILE
30  END
```

图 1(d)　正向计算模块伪码

在提出的 EHMCOP 算法中，多路径集合及可行路径检查这两个机制的引入是对 HMCOP 算法最主要的改进措施. 4.2 节中还将结合实例具体阐述这些改进措施的原因和效果.

3.2 可行路径检查

反向路径 $p_r(y,t)$ 计算的只是 s_c 最小的路径，不能完备地确定从 y 到 t 的各个最小约束值. 例如 $s_c = w_1 + w_2 = 10$，单个约束 w_1 和 w_2 的取值可以有多种可能，既可能取 2、8，也可能取 5、5. 正向路径计算时，$T(s,y)$ 与 $p_r(y,t)$ 的约束累积超限，只能说明 $T(s,y)$ 与 $p_r(y,t)$ 级联不是可行路径，并不代表经 $T(s,y)$ 到目的点 t 不存在可行路径. 因此，发现累积约束超限时，需要进一步检查可行性.

具体的步骤参见图 1 可行路径检查模块的伪码描述：首先，从原图 $G(V,E)$ 中删除正向路径 $T(s,y)$ 的节点和链路（这主要是为了避免路由成环），得到新的图 $G'(V,E)$. 然后从总约束 $C_k, k = 1,2,\cdots,K$ 中扣除正向计算累积的约束值，得到新的约束 $C'_k, k = 1,2,\cdots,K$. 最后调用反向计算模块在新图 $G'(V,E)$ 中按照新的路径约束值 $C'_k, k = 1,2,\cdots,K$ 查找从节点 y 到目的点 t 是否存在满足约束要求的路径.

3.3 正向多路径记录

图 1 所示的正向计算模块中，$L_p(s,y)$ 代表的是一个从 s 到 y 的正向路径集合，路径编号从 1 到 N_r，其中 N_r 为允许记录的最大路径数目：

$$L_p(s,y) = \left\{ p_{r,n}(s,y) \;\middle|\; \begin{array}{l} 1 \leqslant n \leqslant N_r \\ \forall\, n, m(p_{r,n}(s,y)) < m(p_{r,n+1}(s,y)) \\ \forall\, n, \exists\, k, w_k(p_{r,n}(s,y)) < \min_{i=1}^{n-1}\{w_k(p_{r,i}(s,y))\} \end{array} \right\}$$

集合中路径的排序（下标编号）方法为：

●编号小的路径主代价小，即严格的升序；

●编号大的路径虽然主代价大，但至少有一个约束参数比排在前面的路径都小.

进行插入操作（Insert_path_list）时，将严格按照上述原则查找插入位置，若查找不成功，则返回插入失败. 查找不成功有两种可能，一是没有满足上述原则的插入位置，一是插入位置超过了最大路径数目 N_r. 若插入成功，但总的路径数目超过 N_r，则删除多余的路径.

$L_p(s,y)$ 中维护的路径要么主代价较小，要么主代价较大，但约束参数较小. 在后续的路径检查中，总是先挑选主代价较小的路径，若不能构成完整的可行路径，则选择主代价次之的. 按这种方式一直进行下去，找到的第一条满足约束的可行路径必然是所有可行路径中主代价最小的.

显然，N_r 的大小与算法的性能密切相关. 定性地看，N_r 越大，则记录的路径越多，搜索的余地越大，找到最佳解的可能性也就越大；另一方面，N_r 越大，所需的记录空间越大，并且搜索的耗时越多. 4.4 节中还将结合仿真结果进一步讨论 N_r 的选择对算法性能的影响.

4 性能分析与比较

4.1 复杂度分析

算法主要由反向计算和正向计算两个模块组成. 反向计算模块是典型的 Dijkstra 类型算法，复杂度为 $O(|V|^2)$[13]. 正向计算模块中，针对每个邻接点的检查共需进行 $|E|$ 次，最坏情况下每次检查都需调用可行路径检查模块，故邻接点检查复杂度为 $O(|E||V|^2)$；正向计算模块中另一个主要的耗时操作是从 Heap 中提取最小主代价路径，由于采用多路径记录，该操作共进行 $N_r|V|$ 次，每次耗时 $O(N_r|V|)$，总耗时 $O(N_r^2|V|^2)$. 因此，整个正向计算模块复杂度为 $O(\max(N_r^2,|E|)|V|^2)$. 整个 EHMCOP 算法复杂度为反向和正向计算模块之和，为 $O(|V|^2 + \max(N_r^2,|E|)|V|^2)$. 若采用二进制堆（binary heap）或 Fibonacci 堆（Fibonacci heap）等高效查找数据结构来实现 Heap，则复杂度会降低一些[13].

4.2 两点改进

文献[12]中的 HMCOP 算法存在两个问题. 即反向计算和正向计算时记录的不完备性. 下面分别用例子来说明这种不完备性的原因，并阐述本文提出的 EHMCOP 算法是如何克服这两个问题的.

图 2 中，每条边上记录了三个值，依次分别是主代价，约束 1 和约束 2. 假定要求约束 1 和约束 2 都不能大于 1，求从 s 到 t 的主代价最小的路径.

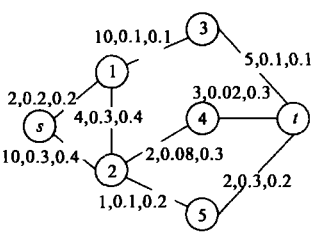


图 2(a) 例图 1

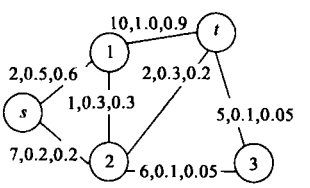


图 2(b) 例图 2

容易验证，图 2(a) 中最佳路径是 $P_1 = (s,1,2,5,t)$，P_1 的主代价为 9，约束分别为 0.9 和 1；但采用 HMCOP 算法得到的却是 $P_2 = (s,2,5,t)$，主代价为 13，约束分别为 0.7 和 0.8. 下面来分析原因. 对于节点 2 来说，反向计算后记录的是路径 $P_3 = (2,4,t)$，主代价为 5，约束和为 0.7，两个约束分别为 0.1 和 0.6. 正向计算时，首先标记节点 1，因为该点到 s 主代价最小且存在可行路径. 从节点 1 出发更新节点 2 时，就会出现问题了：路径 $P_4 = (s,1,2)$ 与 $P3$ 的各约束累积会导致约束 2 大于 1($0.2 + 0.4 + 0.6 = 1.2$)，也就是说，路径 $(s,1,2)$ 与反向计算的结果 P_3 级联是不可行的路径，所以该点不能更新，而原来的路径 $P_5 = (s,2)$ 与 P_3 级联是可行路径，故最终给出的路径是 P_2. 问题的关键在于，反向计算所记录的结果是不完备的，路径 P_4 与 P_3 级联不可行，并不代表从源点经路径 P_4 到目的点不存在可行路径，事实上，存在路径 $P_6 = (2,5,t)$，与 P_4 级联是可行路径，但在 HMCOP 算法中，P_6 的约束和大于 P_3，反向计算时被忽略了.

在本文提出的 EHMCOP 算法中，节点 2 更新过程中，当发现 P_4 与 P_3 级联是不可行路径时，还需要进一步进行检查，即调用 Further_check 模块判断是否存在从节点 2 出发到目的点 t 的路径，使得该路径与 P_4 的级联为可行路径. 从而有效补偿了反向记录的不完备性.

HMCOP 算法的另一个问题是正向计算时只记录一条路

径.以图2(*b*)为例.定义 $P_1=(s,1,2)$, $P_2=(s,2)$, $P_3=(2,t)$, $P_4=(2,3,t)$.当正向计算模块检查节点2时,若只记录一条正向路径,则 P_1 被记录(主代价为3),而 P_2 被丢弃(主代价为7).在后续路径(即从节点2出发到目的点 t 的路径)的查找中,只能选择 P_4,因为只有这条路径与 P_1 级联后形成的完整路径才是满足约束的.但是,从图中不难看出,路径 P_2 与 P_3 级联形成的完整路径不仅是满足约束的,而且主代价也较小.这条路径之所以被忽略,是因为 P_2 虽然主代价较高,但与后续路径级联后的总路径反而较短,在不能确定后续路径的检查结果时过早丢弃 P_2,将导致正向路径记录不完备,从而得不到整体最佳的结果.

本文提出的EHMCOP算法中,由于在每个节点都记录了多条正向路径,后续路径的查找过程中多了一些选择,从而可以保证整体最佳路径不会因正向路径记录的不完备而被忽略.

4.3 仿真方法

4.4节中的每个结果由多次实验结果平均得到,并给出置信度为95%的置信区间;每次实验测试十个不同的拓扑,每个拓扑采用Transit-Stub模型随机生成[14],每个拓扑中包含100个节点;对于每个拓扑,都随机产生十次不同的链路权重,其中每条链路的主代价在范围[1,100]内随机选择,约束权重有两个(即 $K=2$),分别在范围[1,50]和[1,100]内选择;对于每次链路权重配置,都随机产生2000次不同的请求,一次请求包括源宿节点和二个路径约束值.对于每次请求,源宿节点对是在所有的节点中随机选择的,而路径约束值 C_1 和 C_2 是这样确定的:首先计算从 s 到 t 的 w_1 最小路径 p_1,和 w_2 最小路径 p_2;然后在范围[$b_1^* w_1(p_2)$, $b_2^* w_1(p_2)$]内随机选择一个值作为 C_1,在范围[$b_1^* w_2(p_1)$, $b_2^* w_2(p_1)$]内选择 C_2 值,其中 b_1 和 b_2 是固定值.

每次实验测试的请求数目为NE=200000,即每次实验需要调用NE次EHMCOP和HMCOP进行计算.根据表1,显然有:NO+BF+SE+SH+BS=NE.

为了比较EHMCOP和HMCOP两个算法的复杂度,实验中统计了两个算法的耗时比TR.每次调用EHMCOP和HMCOP算法的前后都记录计算机时间,累积时间差并对累积次数求平均,分别得到两个算法的平均耗时.由于计算机时间的统计在不同的运行环境下有一定误差,所以采用耗时比,即求两个平均耗时之比值.注意到无可行路径时,两个算法都只执行了第1、2步,而且反向计算模块耗时是固定的(两个算法没有差别),因此平均耗时的计算中没有累积这些结果(只累积NE-NO次).

测试算法性能主要考虑两个指标[12]:成功比例SR和平均主代价AC.SR越大则算法性能越好.为便于比较,计算SR时不考虑无可行路径的情况,因为此时所有的算法都无法得到结果,反映不出差别.故EHMCOP算法的SR计算公式为:(BS+SE)/(NE-NO),而HMCOP算法的SR为:(BS+SH)/(NE-NO).平均主代价AC通过累积路径主代价,然后求平均得到.为便于比较,只累积两个算法都成功的那些结果,即累积BS次.AC越小,意味着算法计算出的路径主代价越接近最佳值.

表1　仿真结果中用到的符号说明

NE	每次实验测试的请求数目
NO	无可行路径的次数(算法在第2步终止的次数)
BF	两个算法都失败的次数(算法执行完第3步后失败的次数)
SE	EHMCOP成功,而HMCOP不成功的次数
SH	HMCOP成功,而EHMCOP不成功的次数
TR	EHMCOP算法平均耗时与HMCOP算法平均耗时之比
BS	两个算法都成功的次数
ME	两个算法都成功,但EHMCOP计算的路径主代价较小的次数
MH	两个算法都成功,但HMCOP计算的路径主代价较小的次数
SR	算法的成功比例
AC	计算出的路径的平均主代价

4.4 仿真结果

本节给出两组仿真实验的结果.第一组实验研究了 N_r 的大小对算法性能的影响.仿真中 b_1 和 b_2 取值分别为0.9和1.1.表2给出了 N_r 在不同取值下的仿真结果,图3给出了AC和SR的比较结果.

首先注意到,随着 N_r 的加大,SH减小了;当 $N_r\geqslant 8$ 时,SH等于0.这说明若 N_r 取值大于等于8,则凡是HMCOP算法能计算出来的路径,EHMCOP算法都能计算出来.而且约有6000余次路径请求,EHMCOP算法可以计算出满足要求的最佳路径,而HMCOP算法不能;这个值约占总实验次数NE的3%,若考虑到约有100000多次根本没有可行路径的情况,这个比例还要更高一些.事实上,从图3(*a*)中可以看出,$N_r\geqslant 8$ 时,与HMCOP算法相比,EHMCOP算法的成功比例高了近10%.

第二,随着 N_r 的加大,MH减小了;当 $N_r\geqslant 7$ 时,MH等于0.这意味着,$N_r\geqslant 7$ 时,EHMCOP算法计算出的路径主代价至少不大于HMCOP算法的计算结果.而且约有近2000(ME)次EHMCOP算法的计算结果优于HMCOP算法.从图3(*b*)中也可以看出,EHMCOP算法的平均主代价比HMCOP算法小.

第三,从计算耗时比例TR的实验结果可以看出,EHMCOP算法的耗时基本上维持在HMCOP算法的9~10倍之间,并没有无限制增长.从4.1节的复杂度分析可见,算法耗时取决于 N_r^2 与 $|E|$(图中链路数目)的较大值,N_r^2 约为100左右,而测试用的100节点的图中,链路数目远大于这个值.所以算法耗时主要取决于链路数目.

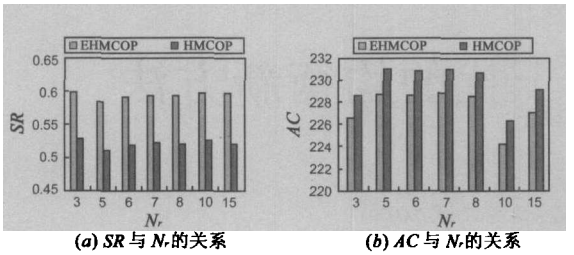
第二组实验测试了EHMCOP算法在不同路径约束下的性能.由于路径约束 C_1 和 C_2 分别在[$b_1^* w_1(p_2)$, $b_2^* w_1(p_2)$]和[$b_1^* w_2(p_1)$, $b_2^* w_2(p_1)$]内随机选择,所以 b_1 和 b_2 的大小决定了约束的"苛刻程度",b_1 和 b_2 的值越小,说明满足相应约束的路径越难找到.表3给出了 b_1 取值分别为0.7、0.8、0.9、1.0、1.1,$b_2=b_1+0.1$ 时的仿真结果,图4给出了SR和AC的比较结果.这组实验中 N_r 取值为10.

表 2 测试 N_r 大 b_1 小对算法性能影响的仿真实验结果

N_r	NO	BF	SE	SH	TR	BS	
						ME	MH
3	115295 (112603,117986)	33878 (32324,35432)	6058 (5389,6728)	296 (248,343)	9.64 (9.28,10.0)	44473(42762,46184)	
						1932 (1714,2149)	46 (29,64)
5	116095 (113978,118212)	34854 (34412,35295)	6128 (6073,6183)	8 (6,10)	9.27 (8.7,9.83)	42915(41218,44612)	
						1878 (1613,2142)	4 (0,8)
6	115575 (110814,120335)	34468 (32859,36076)	6252 (5457,7046)	1 (0,2)	9.3 (9.23,9.38)	43704(40502,46905)	
						1846 (1642,2050)	1 (0,2)
7	117047 (115949,118144)	33613 (32504,34721)	6107 (5756,6459)	2 (0,4)	9.24 (8.93,9.55)	43231(41778,44684)	
						1921 (1626,2217)	0 (0,0)
8	116365 (113174,119556)	34025 (32155,35895)	6134 (6078,6191)	0 (0,0)	9.56 (9.25,9.86)	43476(41111,45842)	
						1940 (1848,2033)	0 (0,0)
10	117728 (115811,119645)	32984 (31654,34313)	6050 (5669,6431)	0 (0,0)	9.37 (9.1,9.64)	43238(42748,43729)	
						1882 (1599,2165)	0 (0,0)
15	116520 (114354,118686)	33704 (33127,34281)	6316 (5976,6656)	0 (0,0)	9.64 (9.19,10.1)	43460(41915,45005)	
						1792 (1570,2014)	0 (0,0)

表 3 测试路径约束大小对算法性能影响的仿真实验结果

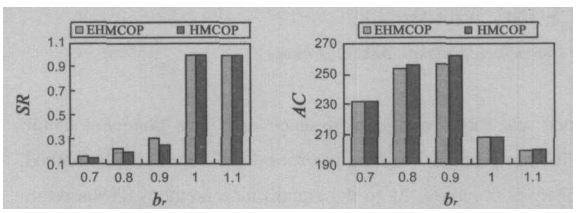
b_1	NO	BF	SE	SH	TR	BS	
						ME	MH
0.7	187636 (187229,188044)	10335 (10010,10659)	250 (155,344)	0 (0,0)	6.07 (5.19,6.96)	1779(1530,2028)	
						26 (0,52)	0 (0,0)
0.8	168010 (166185,169835)	25005 (23578,26432)	822 (616,1028)	1 (0,2)	7.4 (6.5,8.31)	6162(5435,6889)	
						261 (214,308)	0 (0,0)
0.9	140400 (139538,141262)	40925 (39443,42407)	3731 (3399,4063)	0 (0,0)	9.86 (9.58,10.15)	14944(13933,15955)	
						1497 (1185,1809)	0 (0,0)
1.0	38 (0,76)	224 (73,375)	111 (26,196)	0 (0,0)	7.13 (6.83,7.43)	199627(199400,199854)	
						2259 (2064,2453)	0 (0,0)
1.1	0 (0,0)	0 (0,0)	2 (0,4)	0 (0,0)	8.08 (7.91,8.25)	199998(199994,200002)	
						2315 (2197,2433)	0 (0,0)



(a) SR 与 N_r 的关系　　(b) AC 与 N_r 的关系

图 3　第一组实验的 SR 与 AC 结果

不难看出,EHMCOP 算法在各种约束情况下都比 HMCOP 性能好,无论成功的次数,还是路径的平均主代价都优于 HMCOP 算法.在不同的约束条件下,性能改善的程度是不同的.从图 4(*a*)中可见,在约束非常严格(如 $b_1 = 0.7$)和约束非常宽松(如 $b_1 = 1.1$)的情况下,成功比例几乎没有区别.约束严格时,两个算法都很难找到满足条件的路径,从 N0 的大小可见,$b_1 = 0.7$ 时,90% 以上的路径请求都无法满足;反之,约束宽松时,两个算法都可以容易地找到满足约束地路径;因此,在这两个极端,EHMCOP 算法的性能改善程度有限.



(c) SR 与 b_r 的关系　　(d) AC 与 b_r 的关系

图 4　第二组实验的 SR 与 AC 结果

图 4 中可以看出,b_1 越大,ME 也越大.这说明约束越宽松,EHMCOP 算法计算出主代价较小的路径越容易.这是由于约束越宽松,正向计算时查找主代价较小的路径受到的限制越小,"可行路径检查"的效果也越明显.但是,图 4(*b*)表明,约束宽松时,尽管与 HMCOP 算法相比,EHMCOP 算法得到主代价小的路径次数多,平均主代价 AC 的大小却没有明显改善.这意味着主代价的改进程度并不大.

5　结论

针对多个加性约束的 MCOP 问题,我们改进了现有的 HMCOP 算法,分别提出了"多路径记录"和"可行路径检查"两个具体措施,以克服 HMCOP 算法中存在的正向和反向路径记录不完备的问题,所提的算法称为 EHMCOP 算法.给出了算法的伪码描述,分析了算法的复杂度,并结合两个具体的实例分析了 EHMCOP 算法是如何解决 HMCOP 算法所存在的问题的.最后通过仿真实验,详细比较了本文所提的算法与 HMCOP 算法.结果表明,在成功比例、平均路径主代价等关键性能指标上,新算法都比 HMCOP 算法好.

参考文献:

[1] X Xiao, L M Ni. Internet QoS: A big picture[J]. IEEE Network, 1999, 13(2): 8 - 18.

[2] T M Chen, T H Oh. Reliable services in MPLS[J]. IEEE Communication Magazine, 1999, 37(12): 58 - 62.

[3] Z Wang, J Crowcroft. Bandwidth-delay based routing algorithms[A]. Proc of IEEE global Telecommunications Conference 1995, GLOBECOM'95[C]. Singapore: IEEE, 1995. 2129 - 2133.

[4] R K Ahuja, T L Magnanti, J B Orlin. Network Flows: Theory, Algorithms, and Applications[M]. Prentice Hall, Inc, 1993.

[5] J M Jaffe. Algorithms for finding paths with multiple constraints[J]. Networks, 1984, 14: 95 - 116.

[6] A Orda. Routing with end-to-end QoS guarantees in broadband networks[J]. IEEE/ACM Trans on Networking, 1999, 7(3): 365 - 374.

[7] F Ergun, R Sinha, L Zhang. QoS routing with performance-dependent costs[A]. Proc. of the INFOCOM 2000 Conference[C]. Tel AVIV, ISrel, IEEE: 2000, (1): 137 - 146.

[8] H F Salama, D S Reeves, Y Viniotis. A distributed algorithm for delay-constrained unicast routing[A]. Proc. of the INFOCOM 97 Conference [C]. Kobe, Japan: IEEE, 1997, 4(1): 84 - 91.

[9] J Zhou. A new distributed routing algorithm for supporting delay-sensitive applications[A]. Proc of (ICCT'98)[C]. Beijing, Chian: IEEE, 22-24 Oct. 1998: S37-06(1 - 7).

[10] D Eppstein. Finding the k shortest paths[A]. Proc. of the 35th Annual Symposium on Foundations of Computer Science[C]. Santa Fe, New Mexico, USA: IEEE, 1994. 154 - 165.

[11] T Korkmaz, M Krunz, S Tragoudas. An efficient algorithm for finding a path subject to two additive constraints[A]. Proc of the International Conference on Measurement and Modeling of Computer Systems (ACM SIGMETRICS'00 Conference)[C]. Santa Clara, California, International Conference on Communication Technology, USA: ACM, 2000, 1: 318 - 327.

[12] T Korkmaz, M Krunz. Multi-Constrained Optimal Path Selection[A]. Proc of the IEEE INFOCOM 2001. Conference Proc[C]. Anchorage, Alaska, USA: 2001. 834 - 843.

[13] T H Cormen, C E Leiserson, R L Rivest. Introduction to Algorithms [M]. The MIT press and McGraw-Hill: sixteenth edition, 1996.

[14] K I Calvert, M B Doar, E W Zegura. Modeling internet topology[J]. IEEE Communications Magazine, 1997, 35(6): 160 - 163.

作者简介:

王　晟　男,1971 年生于四川成都,博士,副教授,目前主要研究方向为通信网与宽带通信技术.

李乐民　男,1932 年生于浙江吴兴,教授,博士生导师、工程院院士,主要研究方向为通信网与宽带通信技术.