流量工程中静态路由算法的研究 [1]

吕 航 孙雨耕 吴 雪

(天津大学电气与自动化工程学院 天津 300072)

摘 要 该文提出了一种应用于流量工程环境中的静态路由算法. 考虑当前的网络资源情况, 分优先级别在网络中计算并配置标记交换路径 (Label Switched Path, LSP) , 当某一优先级有多条 LSP 需要并行配置时, 利用遗传算法搜索最优或较优的配置方案, 使得网络的链路带宽使用率低于管理员定义的某个限定值, 达到合理分布资源的目的. 此外, 提出了一种改进的 Dijkstra 算法计算 LSP 的最短路径.

关键词 流量工程, 路由算法, 标记交换路径

中图号 TN911.3

1 引 言

随着计算机网络应用的发展, 人们对网络所提供的延迟、带宽等能力的要求越来越高. 仅仅通过增加物理设备 (扩充网络) 来满足这种需求是远远不够的, 而且早期 Internet 在选取路由时, 仅仅依据最短路径的方法, 这种路由器工作方式依靠软件来进行选路和数据转发, 速度难以提高, 而且带来下列问题 [1] :

(1) 不同源的最短路径在某些链路上重叠, 导致这些链路拥塞;

(2) 某一条从源到目的地的流量需求超出了最短路径的容量, 网络仍然选择这条路, 而不选择源与目的地间的一条比较长的、资源没得到充分使用的路径。

实际上产生拥塞的原因不能单纯归结为带宽的不足, 信息流在整个网络中的不平衡分布也是产生通信热点和瓶颈的重要原因之一, 所以, 20 世纪 90 年代后期, 流量工程的研究成为网络界的研究热点。流量工程指现有运行的 IP 网络的性能评价和性能优化, 一般包括用来测量、建模、描述、控制 Internet 业务量的技术应用和科学原理, 还包括应用这些技术达到一定的性能指标 [2] 。流量工程的主要功能是把业务量映射到实际的物理拓扑上, 实施流量工程的目的是使 ISP 能够在现有的网络资源条件下, 对其路由域内的流量分布进行精确控制, 从而提高资源利用率, 使网络运行更有效, 为用户提供更好的服务 [2] 。

MPLS(Multi-Protocol Label Switching) 把标记交换结构和网络层路由结合在一起, 其基本思想是在 MPLS 域的入口处给数据包附上一个短的固定长度的标记 (基于转发等同类 (FEC — Forwarding Equivalence Classes) 的概念) , 在通过 MPLS 域时, 根据附在数据包上的标记来做转发决定 (不再依赖于数据包头的内容了) , 一条 LSP 就是数据流量通过 MPLS 域的一条标记交换路径, MPLS 负责将 IP 包按照一条预先确定的路径进行转发 [3] 。

路由算法分为静态路由算法和动态路由算法. 动态路由算法在考虑资源约束条件下一次仅计算一条 LSP , LSP 计算顺序的不同决定了 LSP 的物理路径也不同, 即先配置的 LSP 有更多的资源可以利用, 后配置的 LSP 可利用的资源相对较少, LSP 的配置顺序改变, LSP 的物理路径也会改变, 资源使用情况也会改变. 静态算法同时考虑每一条链路的资源约束和所有 LSP 的需求, 它执行全局的计算, 比较每次计算的结果, 选择最好的方案作为全局配置的结果, 静态算法的输出优化了整个网络资源的使用, 而且可以以任意顺序建立 LSP 路径 [4] 。

目前流量工程中静态路由算法的文章不是很多,使用最多的、最简单的是肖曦鹏的算法 [1] ,其主要思想是一次一条地计算 LSP 的路径, 首先从高优先级的 LSP 开始, 对同一优先级的 LSP , 先计算带宽需求最大的 LSP 的路径, 然后是需求次大的, 直到结束. 该算法简单、计算速度快, 但优化的结果精确度不是很高。

[1] 2002-09-24 收到, 2003-01-13 改回
教育部博士学科点基金资助项目 (2000005634)

本文提出了一种更优化的静态路由算法，分优先级别地配置 LSP 路径，但在配置同一优先级的多条 LSP 路径时，使用并行算法，同时配置该级别的所有 LSP，以达到全局最优或较优的目的。此外，在计算和配置过程中，始终保证链路的带宽使用率不超出某个限定值，从而合理地分配网络资源，尽量使网络负载均衡，防止部分链路过度使用而其它链路却未得到充分使用。计算多约束的 LSP 的路径时，使用一种改进的 Dijkstra 算法。

本文第 2 节介绍算法思想、模块的划分以及各模块的功能及实现方法；第 3 节介绍算法仿真结果和仿真结果的分析；第 4 节为结束语。

2 算法思想及其实现

由于算法的目标是为了避免部分链路过载、部分链路却未得到充分使用，所以，如果限定每一条链路的带宽使用率，使它们都不超出某个最大值的限制，则可以保证不会出现过载的链路，减小拥塞的概率，因此，计算和配置 LSP 路径的难点在于：(1) 满足各 LSP 的约束；(2) 符合链路最大带宽使用率的限制；(3) 全局优化的目的。

其中，LSP 的约束有：

(1) 带宽需求 (Band-width)；

(2) 资源类属性，即链路颜色约束 (Resource class affinity attributes)；

(3) 管理性指定路由 (Administratively specified explicit paths)；

(4) 路由跳数限制 (Hop-limit)；

(5) 资源满足优先级别 (Priority attribute)；

(6) 资源抢占属性 (Preemption attribute)。

除了上面 6 种约束外，计算 LSP 的物理路径时，还要考虑链路的费用 Metric。

定理[5] 基于约束路由中同时使两种或两种以上相互独立的约束达到最优的路由问题是 NP 完全问题。

根据定理，使所有约束均达最优是不可能的，只能根据需要选择一个约束作为优化的目标，其它的作为限制条件。本文在计算 LSP 路径时，提出一种改进的 Dijkstra 算法，处理 LSP 的多约束，把跳数、残留带宽率和费用作为量度来优化，并利用多量度的主次关系，计算 LSP 的路径，详细的过程在单条路模块中介绍。本静态路由算法控制几个模块之间的相互操作，在不同的网络状态下，调用不同的模块，从而协调各模块的关系，达到优化网络资源的目的。主要思想是依据各 LSP 的优先级别，分优先级别配置 LSP，属于同一优先级别的 LSP 尽量使用并行算法同时配置，以达到全局优化的目的，并保证带宽使用率不超过最大限制，配置同一优先级别的算法流程如图 1 所示。

从图 1 中看出，算法中涉及到多个功能模块，其中单条路算法模块主要是根据当前网络信息，计算多约束条件下的一条 LSP 路径；K 条路模块主要是在当前的网络资源下，为某一个 LSP 计算满足多约束的 K 条路径，且 K 条路经过的链路都不完全重叠，随后的穷举法模块和遗传算法模块的配置方案使用的 LSP 路径就是从 K 条路径中选择；穷举法模块使用穷举法为条数较少的 LSPs 快速地找出一个最优的符合最大链路使用率限制的配置方案；遗传算法模块利用遗传算法的全局寻优的特性，为条数较多的 LSPs 迅速找出最优的或较优的符合最大链路使用率限制的配置方案；一条条计算模块的目的是在并行计算不能找到配置方案时利用肖曦鹏的算法方案 [1] 配置某级别的 LSP，保证本算法的结果至少不比肖曦鹏的算法差。

2.1 各模块功能和实现方法

在本节将具体介绍各个模块的功能和实现的方法，这些模块配合使用将可以实现系统的功能。

2.1.1 *单条路算法模块* 本模块在当前网络状态下，仅计算一条 LSP 的路径，而且通过分阶段来处理各种约束，并运用改进的 Dijkstra 算法计算 LSP 的路径。经典的 Dijkstra 算法中量度是唯一的，或者是费用最小路，或者是距离最短路等等，当出现费用相等或距离相等的多条备选路时，没有别的量度来衡量，只是随机选择其中的一条作为最短路。而我们的单条路算法中，使用多量度来衡量，从定理得知，多量度的算法是 NP 完全问题，所以不可能使多量度

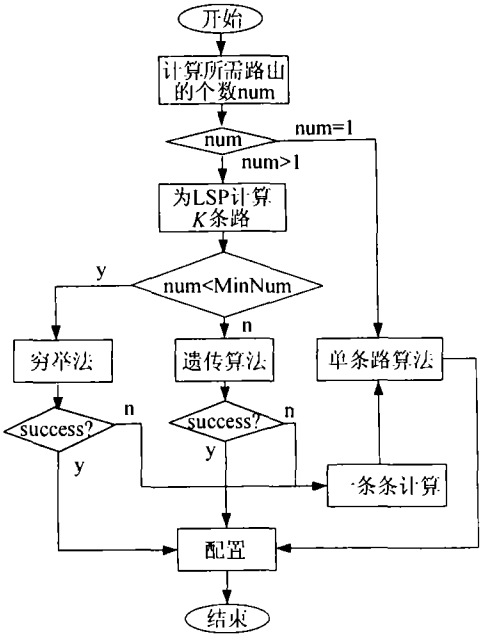

图 1　配置 LSP 的控制规则流程图

同时最优，所以我们运用主次目标量度的方法。当出现多条备选路，且这些备选路径主目标量度相等时，可以辅助以第 2 目标或第 3 目标量度来选择路径。

首先分析 LSP 的多个约束：判断对应着 LSP 优先级的链路上的可预留带宽是否满足 LSP 带宽需求，如果不满足，先从拓扑中剪除这些链路；不满足颜色要求的链路也先从拓扑中剪除，在剩余拓扑中运行改进的 Dijkstra 算法，会自动满足带宽需求和颜色要求；指定路由可以按分段来处理，比如从源点 s 到目的点 t，中间顺序指定了必须经过的节点 a 和 b，我们可以分为 3 段来处理，首先计算 s 到 a 的路径，然后计算 a 到 b 的路径，最后计算 b 到 t 的路径，这 3 段路径综合一起就是需要计算经过指定点 a 和 b 的 LSP 路径，当然计算过程中必须有机制来防止环路的产生；由于按优先级的先后顺序配置 LSP 路径，本算法不予考虑资源抢占属性。

从上面分析，对于 LSP 的多约束，除了跳数之外，其它的都可以在计算一段路径前得到满足，只要考虑以跳数为量度的最短路算法即可得到 LSP 路径，但我们的目标是为了负载均衡，偏重于链路的使用率，选择路径时，尽量选择残留带宽率大的链路 (当然该链路有足够的带宽允许 LSP 通过)，而且路径费用应该越小越好，跳数越短越好，因此，改进的 Dijkstra 算法中，确定需要优化的目标量度是残留带宽率 Rbr，费用 Metric 和跳数 Hop，由于同时优化这些量度是 NP 完全问题，所以，根据本算法的需要，确定优化的目标量度的主次顺序为 {Rbr, metric, Hop}，前面的最先得到优化。如果偏重于别的方面，改变量度的主次顺序即可。

在改进的 Dijkstra 算法中，每个节点包含如下状态信息：

```
struct state
  {
    int ulPredecessor ;    // 已着色的前向节点
    int ulRbr[4]           // 残留带宽率
    int ulMetric           // 路径费用
    int ulHop              // 跳数
    enum {permanent tentative} Label // 是否着色的标记
```

};

链路的残留带宽率 Rbr 定义为 Rbr=(AB−RB)/TB 。其中 AB 为链路的可利用带宽； RB 为 LSP 需求的带宽； TB 为链路总的带宽。

改进的 Dijkstra 算法点着色过程如下：

(1) 当源节点或某一节点着色后，更新所有未着色点的属性值 (按照上述结构)；

(2) 判断所有未着色点中哪一个的残留带宽率最大并选择该点着色，然后循环其它未着色点，转 (1)；否则，转 (3)；

(3) 如果有多个点的残留带宽率相同，则选择费用最小的点着色，循环其它未着色点，转 (1)，否则，转 (4)；

(4) 如果多个点的最小费用相同，则选择跳数最小的点着色，然后循环其它未着色点，转 (1) 否则，转 (5)；

(6) 如果多个节点的跳数也相同，则随机选择一点着色，循环其它未着色点，转 (1)。

上述过程中，如果某个属性值 (如跳数) 大于规定的最大限制 (如最大跳数限制)，程序立刻停止并退出。通过改进的 Dijkstra 算法，使得 LSP 路径尽量经过带宽使用率较低的链路，减少拥塞的可能。

2.1.2 K 条路模块 本模块在当前网络状态下，计算满足约束的 K 条路。这 K 条路不是一次全部计算出来，而是一次计算一条，直到计算出 K 条路来，算法思想为：

(1) 运行单条路算法模块；

(2) 如果能得到一条最短路，则判断路径集合 S(设 S 以队列形式组织) 中是否包含该路径，如果不包含则将该路径加入路径集合 S 尾端；

(3) 从 S 头端取一条路径 $p1$，判断 $p1$ 是否有继承链路 (继承链路定义为，要得到该最短路径必须从初始网络中剪除的链路)，如果有，则剪除；

(4) 然后从 $p1$ 的源节点出发依次剪除 $p1$ 的各条链路，任何时候 $p1$ 上只能有一条链路被剪除，即剪除下一条链路时必须恢复 $p1$ 原来被剪除的链路；每剪除 $p1$ 的一条链路，运行一次单条路算法模块，从而得到一条新的路径 $p2$，如果 S 不包含 $p2$，则记录 $p2$ 的继承链路 (等于 $p1$ 的继承链路加上刚刚剪除的链路)，并将 $p2$ 加入 S 尾端；

(5) 判断 S 中的路径数目是否等于 K，如果等于 K，则已经得到所需的 K 条路径，停止；否则重复 (4) 和 (5)，直到 S 中的路径数目等于 K，或者无法得到新的路径为止。

如果算法终止后 S 中只有 $X(X<K)$ 条，则从 S 中选择路径量度最短的 $K-X$ 条路径进行复制，从而使得 S 中的路径数目为 K。该模块的目标函数与单条路算法模块相同，均为优化的主次顺序为 {Rbr, metric, Hop} 的混合目标量度。

2.1.3 穷举法模块 对于 m(m 小于某个预定值) 条 LSP，每个 LSP 有 K 条路径，则在并行计算时，穷举次数为 K^m，所以对于 m 较小的情况，使用穷举法能快速收敛，找出最优的组合方案，使得每个 LSP 选择属于自己的 K 条路中的某一条，以任意顺序配置这 m 条 LSP 的路径，配置结果都一样，且链路使用率最小。穷举法模块实现很简单，列举所有可能的组合方案，选择能使得最大的链路使用率最小化的方案输出。假设有 M 种方案，网络中有 N 条链路，则穷举的目标函数为：

$$\min_{j\in M}(\max_{i\in N}\text{Link}[i].\text{fLinkRate}) \tag{1}$$

其中 Link[i] 的带宽使用率 fLinkRate 定义为

$$\text{fLinkRate}=[\text{ulUsedBandWidth}]/[\text{ulMaxReservableBandWidth}] \tag{2}$$

其中： fLinkRate：链路的使用率； ulUsedBandWidth：链路的已使用的带宽，定义为所有经过该链路的 LSP 的带宽需求之和； ulMaxReservableBandWidth：链路的最大可预留带宽 (固定值，程序运行中不变)。

比较所有的方案的优劣时，比较每个方案中链路带宽使用率最大的那个值，选择值最小的那个方案输出。

2.1.4 遗传算法模块　对于 LSP 的数目 m(m 大于某个预定值)，同时配置这些 LSP 时，如果仍用穷举法，运算时间非常长，搜索空间太大，利用遗传算法的特性合理编码，用遗传操作迅速找出满足要求的配置方案。

用二进制编码 $X_mX_{m-1}\cdots X_1$ 表示一个个体，其中 X_i 表示四位二进制码，代表选中第 $i(1\le i\le m)$ 条 LSP 的 K 条候选路径中的某一条 (当 K 值不大于 16 时，用 4 位二进制码足以表示)，对于 m 条 LSP 进行配置，每个个体的二进制码长度位 $4m$ 位，一个 $4m$ 位的二进制码表示 m 条 LSP 的配置方案，X_i 取值范围为 $1,\cdots,K$，这样，可以将 X_i 做如下定义：

$$X_i=\begin{cases}1, & \text{选中 LSP}i\text{ 的 }K\text{ 条候选路径中的第 1 条路径;}\\ 2, & \text{选中 LSP}i\text{ 的 }K\text{ 条候选路径中的第 2 条路径;}\\ \vdots & \\ K, & \text{选中 LSP}i\text{ 的 }K\text{ 条候选路径中的第 }K\text{ 条路径;}\end{cases}$$

例如，如果要建立 3 条 LSP，每条 LSP 有 16 条候选路径，我们的编码就采用 12 位二进制码 $X_3X_2X_1$ 来表示，$X_i\in\{1,2,\cdots,16\}$ $i=1,2,3$，例如二进制码 001100100100 这个字符串表示 LSP1 选取第 4 条备选路径，LSP2 选取第 2 条备选取路径，LSP3 选取第 3 条备选路径。采用这种编码方式，编码和解码都非常简单，容易实现，而且直观。

为了实现方便，选择最简单的遗传操作，复制算子把当前群体中的个体按与适应度成比例的概率复制到下一代中，适应度最好的个体复制到下一代的机会最大，此复制过程用赌盘选择来实现；交叉算子有多种，简单的单点交叉作用过程是：首先从群体中随机选择两个需要杂交的个体，然后随机选择杂交的位置，使这两个个体交换从交叉点到末尾的位段；变异算子以某个概率随机改变染色体串上的某些位，对于二进制串就是相应的位从 0 变 1，或从 1 变 0[6]。对于不同的遗传算子，优化的结果是不同的，在本算法的遗传算法模块中，交叉概率选择为 0.8，变异概率选择为 0.1，算法的仿真结果在第 3 节给出。

本模块的目标函数是最小化链路最大带宽使用率，所以应根据此目标函数来建立适应度函数，定义适应度函数为 (假设网络中有 N 条链路)：

$$\text{fitness}=\begin{cases}1.0-\max\limits_{i\in N}\text{Link}[i].\text{fLinkRate}, & \max\limits_{i\in N}\text{Link}[i].\text{fLinkRate}<1.0\\ 0, & \text{其它}\end{cases}$$

其中，链路 i 的 fLinkRate 定义与 (2) 式一样。适应度越大，链路的最大带宽使用率越小。

确定了编码机制、遗传算子和适应度函数后，该模块功能就可以实现了。

2.1.5 一条条计算模块　该模块使用肖曦鹏算法的思想[1]，对某级别需计算的多条 LSP，首先按带宽需求大小确定 LSP 计算的先后次序，需求大的 LSP 先计算并使用资源，对排序后的 LSP 依次调用单条路算法模块计算路径，算法实现步骤如下：

(1) 按带宽需求大小排序；

(2) 调用单条路算法模块计算某个 LSP 的路径；

(3) 使用资源；

(4) 判断是否符合最大使用率限制，如果是，记录该路径，否则，该路径结果置为空。

(5) 重复 (2)-(4) 直至所有的 LSP 都被执行。

通过上层系统的控制，以上算法模块可以协调运行，达到合理分配网络资源的目的。随后叙述上层系统的控制规则。

2.2 算法的控制规则

在配置 LSP 的物理路径时，由于限定链路的带宽使用率，所以部分 LSP 的路径所经过的链路可能不满足使用率限制，则可认为此 LSP 没有满足要求的结果，即算法运行过程中允许部

分 LSP 没有计算结果，不一定所有的 LSP 都必须配置于网络中，如果管理员设定的最大链路带宽使用率较小，肯定存在部分 LSP 不能配置。

本算法的控制规则包括：(1) 首先按优先级对所有 LSP 进行分类，优先级相同的属于同一类；(2) 先计算高优先级的 LSP 的路径并配置于网络中，高优先级的 LSP 使用资源后，再在网络剩余资源中配置低优先级的 LSP，依次进行，直到结束；(3) 在配置某一个优先级的 LSP 时，不失一般性考虑，根据此优先级需要配置的 LSP 条数，做出不同的选择，如图 1 所示，当

(1) 优先级只有一条需要配置，那么调用单条路算法模块，计算出满足约束条件的物理路径，如果在网络中配置此物理路径并使用资源时，满足最大链路使用率的限制，则配置该 LSP 于网络中，否则此 LSP 的计算结果不能被选取，不使用资源；

(2) 该优先级有多条需要配置，但是 LSP 数目不大于某一个预定值 (比如 3 条)，则调用穷举法模块并行配置所有的 LSP。首先调用 K 条路算法模块为该组的每一条 LSP 计算出满足约束的 K 条路径来，然后利用穷举法，找出最好的组合，如果按照此种组合方案配置能够满足最大链路使用率限制，则在网络中按此方案配置并实际使用资源；否则，调用一条条计算模块，一次一条的调用单条路算法模块为这些 LSP 计算路径。

(3) 该优先级有多条需要配置，且 LSP 数目大于预定值，则调用遗传算法模块同时配置该优先级别的 LSP。首先为该组的每一条 LSP 计算满足约束的 K 条路径，然后为该组 LSP 的配置方案进行编码，利用遗传操作，寻找满足使用率要求的最优的或较优的配置结果。如果在规定的次数内找到最优或较优的配置方案，则按该方案配置并实际使用资源；否则，调用一条条计算模块，一次一条地调用单条路算法模块为这些 LSP 计算路径。

以上的控制规则和模块的划分可以应用于实际系统中，此种方案的静态路由算法通过在同一优先级内尽量运行并行计算，使 LSP 不全都选择最短路，尽量避开拥塞点和瓶颈点，计算的路径以任何次序配置，结果都相同，可以避免先配置的优先使用资源的弊端，解决现有网络中存在的部分问题。

3 算法仿真

根据上面提出的控制规则和模块的功能划分编程实现本静态路由算法，算法需要的输入信息，如拓扑信息、链路资源信息和 LSP 需求信息，都是通过文件传送给算法，参照文献 [7] 制定链路的数据结构和 LSP 的数据结构。

程序在执行过程中始终保证，链路的高优先级的可预留带宽比低优先级的可预留带宽大 (0 级别最高)，而且都必须小于最大可预留带宽，即

$$\text{ulUnreservedBandWidth}[i] \leq \text{ulMaxReservedBandWidth}\ (0 \leq i \leq 7)$$

由于在本算法执行过程中，需要通过遗传算法来选择优化的配置方案，所以如果使用比较小的拓扑，那么需要配置的 LSP 数目较少，调用遗传算法模块会浪费时间和资源，因此，为了仿真遗传算法的功能和体现静态算法的优势，我们建立一个比较大的拓扑图来仿真算法的结果。

仿真建立一 50 个节点，127 条链路的网络，网络中有 98 条 LSP 需要计算和配置，并分为 8 个优先级别，这些信息都通过文件存储并传送给算法，拓扑图形太过复杂，这里不给出，需要者可以与作者联系。为了仿真本静态算法的功能和体现本算法的优势，在此拓扑上，依次运行传统的最短路算法和本静态路由算法，并根据各自计算的结果配置 LSP 的路径，使用资源，在两种情况下都分析链路的带宽使用情况，从而比较算法的优劣。图 2 表示根据传统的最短路径算法计算并配置 LSP 路径后，链路的带宽使用情况，图 3，4，5 表示在不同的链路最大带宽使用率 fMaxLinkRate 限制下运行本静态路由算法，计算并配置 LSP 路径后，链路的带宽使用情况。

图 2 表示用传统的最短路径算法计算并配置 98 条 LSP 后链路的带宽使用情况，对链路的带宽使用率没有限制，只要不超过链路的最大可使用带宽即可，此算法能够全部配置所有的 98 条路径，配置完成后链路带宽使用情况如图 2 所示。

图 3 表示设定链路的最大带宽使用率 fMaxLinkRate=0.6 时的配置结果，在链路最大带宽使用率 0.6 的限制下，调用本静态路由算法，不能配置全部 98 条 LSP 的路径，只可以配置 98

条 LSP 中的 90 条，并保证配置这 90 条 LSP 后，链路的最大带宽使用率不超过 0.6，剩余的 8 条 LSP 在当前的带宽使用约束的限制下不能找出满足 LSP 需求的路径。

图 4 表示设定链路的最大带宽使用率 fMaxLinkRate=0.8 时的配置结果，在链路最大带宽使用率 0.8 的限制下，调用本静态路由算法，不能配置全部 98 条 LSP 的路径，只可以配置 98 条 LSP 中的 97 条，并保证配置这 97 条 LSP 后，链路的最大带宽使用率不超过 0.8，剩余的 1 条 LSP 在当前的带宽使用约束的限制下不能找出满足 LSP 需求的路径。

图 5 表示设定链路的最大带宽使用率 fMaxLinkRate=0.9 时的配置结果，在链路最大带宽使用率 0.9 的限制下，调用本静态路由算法，能够配置全部 98 条 LSP 的路径，并保证配置这 98 条 LSP 后，链路的最大带宽使用率不超过 0.9。

从仿真结果可以看出，在图 2 中，使用传统的最短路算法时，有几条链路带宽使用率过高，有些甚至接近于 1.0，这些链路负载过高，很有可能引起拥塞；在我们的静态算法中，虽然限制链路最大使用率为 0.6 (图 3) 和 0.8(图 4) 时，分别有 8 条和 1 条 LSP 不能找到满足约束的路径，但是配置其余的 90 条和 97 条 LSP 的路径时，能保证网络的带宽使用率低于限定的值，且各条链路的使用率相对比较平均，而且与传统的最短路算法 (图 2) 相比，不存在使用率过高的链路，网络负载平衡了许多，大大降低了拥塞的概率，极大的改善了网络的性能；虽然限制链路最大使用率为 0.9 时，存在一条链路使用率接近于 0.9，但是与图 2 比较，还是改善了许多，负载情况也相对平均一些。所以，本静态路径算法可以平衡网络负载，合理的分配网络资源，尽量避免 LSP 在部分关键链路上的重叠，减少网络拥塞的概率，使得网络资源得到充分利用，改善网络的性能。

对于本拓扑数据来说，限制链路的最大带宽使用率 0.8 时，效果最好，虽然有 1 条 LSP 无法找到路径，但整个网络负载平衡情况更好些。在实际使用的时候，对于不同的拓扑，需要选择合适的链路最大使用率，才可以使得网络得到更好的优化。

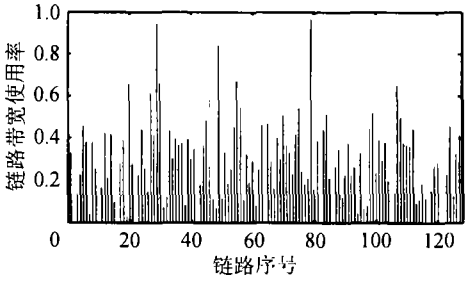

图 2　传统的最短路算法的配置结果

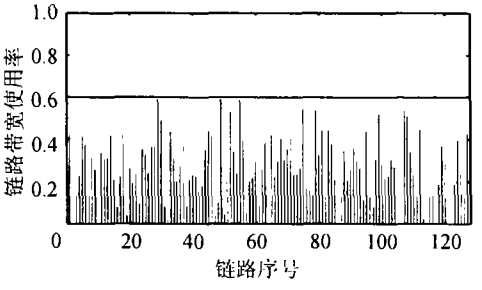

图 3　fMaxLinkRate=0.6 时的配置结果

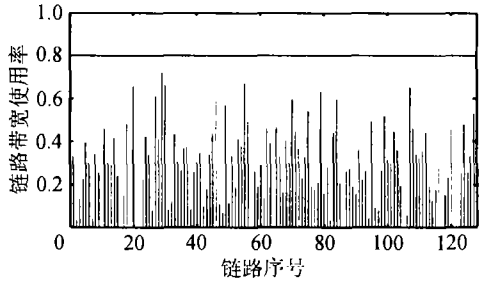

图 4　fMaxLinkRate=0.8 时的配置结果

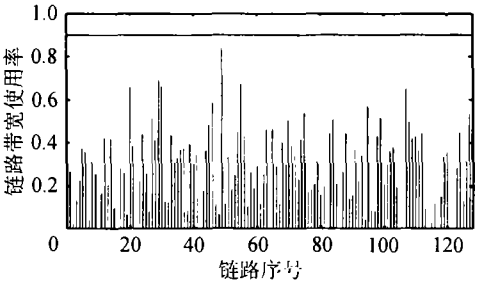

图 5　fMaxLinkRate=0.9 时的配置结果

4 结束语

本文介绍了一种静态路由算法的控制规则以及该算法的模块功能划分和实现，通过与传统

的最短路算法比较可以看出，使用本静态路由算法可以避免部分链路过度使用，而部分链路却未得到充分使用的弊端，且能合理地分配网络资源，改善网络的性能。本文是与华为公司的合作项目的总结，该项目已经通过了华为公司的验收并得到一致好评。本算法控制流程简单有效，可以根据管理员的需要，计算并配置全部 LSP 的路径，也可以稍作调整，在现有网络中调整部分 LSP 的路径，同样可以优化网络的性能，使用非常灵活、方便。

参 考 文 献

[1] Xipeng Xiao, A. Hannan, B. Bailey, M. Ni Lionel, Traffic engineering with MPLS in the Internet, March, 2000, http://www.cse.msu.edu/'xiaoxipe.

[2] D. O. Awduche, Angela Chiu, Xipeng Xiao, A framework for Internet traffic engineering, Internet draft, draft-ietf-tewg-framework-00.txt, Jan, 2000.

[3] O. Awduche, J. Malcolm, J. Agogbua, O'Dell, Requirements for traffic engineering over MPLS, IETF, INTERNET-DRAFT, MPLS Working Group, draft-ietf-mpls-traffic-eng-01.txt, June, 1999.

[4] Juniper networks, Traffic engineering for the new public network, Jan 25, 1999, http://www.cse.msu.edu/'xiaoxipe.

[5] Zheng Wang, J. Crowcroft, Bandwidth delay based routing algorithms, IEEE GlobeCom 1995, Singapore, Nov 1995.

[6] 刘勇，康立山，陈毓屏，非数值并行算法—遗传算法，北京科学出版社， 1995, 1, 6–11.

[7] C. Srinivasan, A. Viswanathan, D. Nadeau, MPLS traffic engineering management information base using SMIv2, draft-ietf-mpls-te-mib-05.txt, November 21, 2000.

RESEARCH ON STATIC ROUTING ALGORITHM WITH TRAFFIC ENGINEERING

Lü Hang Sun Yugeng Wu Xue

(*School of Electrical and Automation Eng., Tianjin University, Tianjin* 300072, *China*)

Abstract A static routing algorithm which applied in traffic engineering environment is put forward. This algorithm calculates and configures the path of LSP by taking the current network resource into account. When needing to configure more than one LSPs in a priority, Genetic Algorithm(GA) is used to find the optimal or sub-optimal configuration for all this LSPs and make the maximal link bandwidth usage lower than the value defined by the administrator. This algorithm can rationally distribute the network resource. Besides, an improved Dijkstra algorithm is put forward to calculate the shortest path for LSP.

Key words Traffic engineering, Routing algorithm, Label Switched Path(LSP)

吕 航： 男， 1976 年生，博士生，研究方向为通信网 (含计算机网络) 的性能优化、流量工程与网络规划.

孙雨耕： 男， 1940 年生，教授，博士生导师，研究方向为图论与系统优化，电路理论，可靠性通信网系统优化设计，通信网中的流量控制工程与基于无线公网的无线数据通信技术.