

基于节点度分层的路由器级拓扑布局算法

刘金明¹, 万明祥²

(1. 南京邮电大学 计算机学院, 江苏 南京 210003;
2. 南京邮电大学 物联网学院, 江苏 南京 210003)

摘要:在对 Internet 路由器级拓扑的可视化过程中,由于探测结果中节点数量众多和链路复杂,导致布局效果呈现主次不分、边交叉和布局效率低等问题。如何在保证全面展示拓扑中数据和提高布局效率的前提下呈现良好的布局效果是文中的研究重点。针对现有的布局算法都存在布局效果不佳和效率低等问题,提出一种改进的 FR 算法—DHL (Degree Hierarchical Layout) 算法。首先,根据 Internet 路由器级拓扑中节点度分布的幂律性质将节点分为三类;接着对分类后的节点进行分层显示;最后根据层次的不同选取合理的初始温度和迭代次数。实验结果表明,文中算法能有效降低时间复杂度和边的交叉数,并使布局效果体现网络的层次性。

关键词:Internet 路由器级拓扑;DHL 算法;幂律性质;节点度

中图分类号:TP393

文献标识码:A

文章编号:1673-629X(2015)01-0100-07

doi:10.3969/j.issn.1673-629X.2015.01.023

Layout Algorithm of Router-level Topology Based on Hierarchy of Node Degree

LIU Jin-ming¹, WAN Ming-xiang²

(1. College of Computer, Nanjing University of Posts and Telecommunications,
Nanjing 210003, China;

2. College of Internet of Things, Nanjing University of Posts and Telecommunications,
Nanjing 210003, China)

Abstract: In the visualization process of Internet topology at router-level, the effect on the layout appears orderless and the execution time is fairly long due to the large number of nodes and the complexity of links in the measurement results. It focuses on presenting good effect of layout under the premise of ensuring fully to demonstrate the topology data and improve the efficiency of layout. The execution time is fairly long and the layout don't reflect the hierarchy of the network in the existing layout algorithm. In this paper, present an improved FR algorithm called DHL (Degree Hierarchical Layout) algorithm to solve the above problems. First, the nodes are divided into three categories according to power-law distribution of the node degree in Internet topology at router level. Then the nodes in the network are separated into multiple hierarchical layers after classification. Finally, the initial temperature and iterations are selected according to different layers. The experimental results show that DHL algorithm can effectively reduce the execution time and the cross of links, layout is able to reflect the hierarchy of the network.

Key words: Internet topology at router level; DHL algorithm; power-law distribution; node degree

0 引言

随着 Internet 的拓扑结构变得越来越复杂,仅用数据表格或文字的形式来表示网络的拓扑关系,理解起来非常困难。因而,需要使用可视化技术将网络的拓扑结构方便、直观地表示出来。作为网络研究重要的辅助手段,网络拓扑可视化^[1]应以能够准确反映网络

拓扑模型的特征为重要目标,从而发挥其正确引导人们认识和研究网络的作用。Internet 拓扑^[2]按不同粒度划分为路由器级 (Router-Level, RL) 拓扑和自治域级 (Autonomous System, AS) 拓扑。路由器级拓扑中节点代表路由器,边代表路由器之间的 IP 层连接。与自治系统级拓扑相比,路由器级拓扑以较细的粒度来

收稿日期:2014-02-19

修回日期:2014-05-27

网络出版时间:2014-11-17

基金项目:江苏省科技支撑计划 (BE2012849);江苏省研究生科研创新计划项目 (CXZZ12_0483, CXLX12_0481)

作者简介:刘金明 (1987-),男,硕士研究生,研究方向为计算机网络。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20141117.2208.041.html>

描述 Internet 的结构。文中只研究 Internet 路由器级拓扑的可视化,随着网络规模的扩大,网络中出现大量的节点和边,导致布局后出现大量的边交叉以及布局算法效率低下等问题,这将大大影响拓扑的可视化效果。

对路由器级拓扑进行可视化既要满足布局效果主、次分明,具有很好的层次性,布局算法的时间复杂度也要低。目前,对 Internet 拓扑的可视化的研究成果中,大多是关于自治系统级拓扑的,对路由器级拓扑的可视化研究工作很少。但在关于 Internet 路由器级拓扑的研究中,Broido 等人发现节点度分布存在幂律性质^[3-4],即少数路由器节点具有很高的节点度,而大量节点具有较小的节点度,并且节点度分布具有明显的层次性。文中依据这一性质对 FR 算法进行改进,分析设计了一种针对 Internet 路由器级拓扑的可视化布局算法。该算法能降低 FR 算法的时间复杂度,布局效果充分体现路由器级拓扑的层次性。

1 相关研究

从网络拓扑可视化的角度来看,对 Internet 的拓扑可视化布局方法大致可分为两大类:物理布局和逻辑布局。

物理布局^[5]是指将地理位置信息与布局坐标信息相结合,管理员通过拓扑图能直观查看网络节点的位置信息。最具代表性的就是地理位置映射算法,严格说来,这种方法并不属于可视化布局的研究范围,它只是简单地将节点的地理位置信息解析为与之对应的布局坐标。

逻辑布局^[6]是指布局过程只基于节点间的连接关系,而不是基于地理位置的布局方法。该方法让研究者在布局的过程中,集中关注网络拓扑自身的内在特性,按需为节点分配布局空间,而无需考虑它们在实际地理位置上的冗余和交错,在布局方面有着物理布局所不具备的优势。逻辑布局是拓扑可视化领域研究的重点,在这一分支产生了许多算法:树型拓扑布局算法、射线布局算法、层次型布局算法、网格型布局算法、启发式拓扑布局算法和力导向拓扑布局算法。文献[7]提出了一种树型拓扑布局算法,首先将复杂的图转换为树,再以树型结构进行布局。该算法效率高,布局速度快;采用该算法在一定程度上可以体现网络拓扑的层次性。算法在节点数目较多时,可能造成拓扑图中节点重叠和边交叉。文献[8]提出了一种射线布局算法,在得到图的生成树后,该算法将其根节点置于中心位置,按照树的层次结构,依次将节点布局在以根节点为圆心的同心圆上。但是当节点数目过多时,该算法将导致设备在圆上分布不均;圆心上的设备超过一定的数目就会造成大量的节点重叠。文献[9]提出

了一种层次型布局算法,该算法比较注重对图内部层次结构关系的处理,它按照一定的标准将图进行分层处理,先为每一个节点确定一个坐标,然后以产生尽可能少的边交叉为目标对节点的坐标进行优化和最终排列。该算法在处理节点数目较少的网络时有较好的表现,但是当节点数较多时,该算法在处理时就会占用很大的作图区域,或者导致显示结果中节点过于密集。文献[10]中提出了一种网格型布局算法,根据节点数量和层次关系,将平面切分成很多正方形区域,将节点按照一定的规律放在正方形中,一个正方形中只能放一个节点,再将节点之间的连线连起来。它很好地解决了布局图空间分配和利用问题。但该算法使得在一个大型的网络拓扑图中,父子节点之间的连接关系不够直观,对于认识和研究网络拓扑结构造成了一定的影响。文献[11]中提出了一种启发式拓扑布局算法。该算法采用分治的思想,将要处理的网络通过一定的规则或条件分割成若干个较小的区域或集群,然后将这些分割好的区域当成一个整体对待,通过一定的计算和排列将这些不同的区域在图内进行排序和放置,采用该算法的关键问题就是要解决节点的聚类问题。力导向拓扑布局算法^[12-15]的基本思想是把每个节点想象成一个质子,质子之间存在吸引力和排斥力。在布局的过程中,通过对由质子和边组成的物理系统的调整,最终使该物理系统达到一个力学平衡的状态,从而达到对整个拓扑图的布局。该类算法所布局的图满足:均匀的边长、均匀的顶点分布和呈现效果的对称性好等优点;该类算法是基于物体的物理类比,算法的行为比较容易预测和理解,最重要的是算法具有很强的理论基础,引进了经典力学的相关理论。这些优点是上述布局算法所不具备的,所以该类算法成为最新的研究重点。Eades 在文献[12]中提出了 Spring Embedded 算法,这是最早提出的力导向拓扑布局算法。该算法基本思想是模拟力学平衡原理,提出了弹力模型这一概念,将图中的节点模拟为钢环,连接则模拟为弹簧;布局过程则是不断调整钢环的位置,使钢环和弹簧所组成的物理系统达到力学平衡状态。该算法对于小规模的数据比较实用,在对比比较大的数据集布局时发现该算法运行时间太长,而且布局效果不理想,所以后续有很多学者对该算法进行了改进。Kamada 和 Kawai 在文献[13]中基于 Eades 的思想提出了 KK (Kamada-Kawai) 算法。该算法的基本思想是假想图中任意两个节点之间皆存在一条弹簧,且弹簧的自然长度与这两个节点间的最短路径长度成正比。算法以求导数的方法计算系统能量,认为当能量达到最小时布局效果最好。虽然算法的执行时间减少了,但是算法的模型复杂,实现起来比较困难。文献[14]中,Da-

vidson 和 Harel 在弹力模型的基础上提出了 DH (Davidson-Harel) 算法。其基本思想是根据美观约束构造约束函数,再使用模拟退火算法求解约束函数。虽然布局效果更美观,但是因为加入了模拟退火算法,增加了约束条件,所以耗时更多。文献[15]中 Fruchterman 和 Reingold 改进了弹力模型,提出了 FR (Fruchterman-Reingold) 算法。该算法的基本思想是将节点模拟成原子颗粒,它们之间存在着相互的吸引力和排斥力。FR 算法在计算节点之间的作用力时进行了简化,在所有的节点之间计算排斥力,而吸引力只在相互连接的节点之间进行计算。另外,FR 算法引入温度变量,使系统迅速冷却,相对于 Spring Embedded 算法、KK 算法和 DH 算法,FR 算法的效率大为提高。但是 FR 算法的执行时间仍比较长,尤其当节点数目超过 500 个时,效率明显下降。

通过对 Spring Embedded 算法、KK 算法、DH 算法和 FR 算法的综合比较与评估,文中对 FR 算法进行改进,提高算法的效率、减小边的交叉数目,使最后的布局效果充分体现 Internet 路由器级网络的层次性。

2 FR 算法

FR 算法对弹力模型进行了改进,引进了粒子物理理论,将图中的节点看作是微观世界的粒子,该算法简单易行。该算法将搜索最优化布局的过程看作是由弹簧连接的粒子的动态相互作用的过程。该算法不容易找到真正的力平衡状态,所以其终止条件通常被设定为一定的迭代次数,每次迭代由三部分组成:

(1) 对节点进行初始布局,一般随意放置;

(2) 通过公式(1)计算各个节点受到的吸引力,通过公式(2)计算各个节点受到的排斥力;

$$F_a(|D(u,v)|) = |D(u,v)|^2/k \quad (1)$$

$$F_r(|D(u,v)|) = -k^2/|D(u,v)| \quad (2)$$

其中, $k = C\sqrt{W * H / V_n}$, C 为常量, W 为布局的宽度, H 为布局的高度, V_n 为节点的数目; $D(u,v) = u.pos - v.pos$, $u.pos$ 和 $v.pos$ 表示节点 u 和 v 的位置。

(3) 通过公式(3)计算由排斥力所产生的位移,若节点间有边相连,通过公式(4)和(5)计算由吸引力所产生的位移。

$$v.disp = v.disp + (D(u,v)/|D(u,v)| * F_r(|D(u,v)|) \quad (3)$$

$$e.v.disp = e.v.disp + (D(u,v)/|D(u,v)| * F_a(|D(u,v)|) \quad (4)$$

$$e.u.disp = e.u.disp + (D(u,v)/|D(u,v)| * F_a(|D(u,v)|) \quad (5)$$

其中, $v.disp$ 为节点 v 在排斥力作用下产生的位

移; $e.v.disp$ 和 $e.u.disp$ 为节点 u 和 v 在吸引力作用下产生的位移。

根据该算法构造的部分 Internet 路由器级拓扑图如图 1 和图 2 所示。



图 1 节点数为 100 的效果图

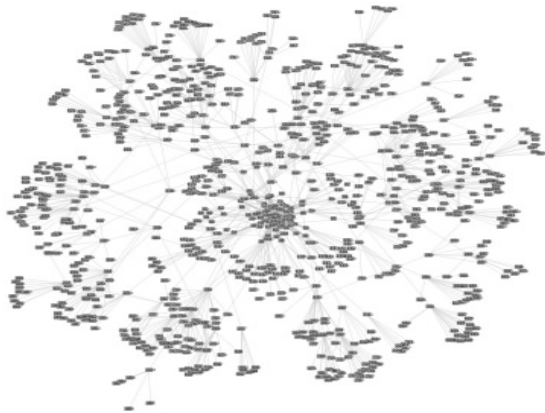


图 2 节点数为 1 000 的效果图

从图 1 可以看出,当节点数较少时,FR 算法能够较清晰地展现 Internet 路由器级拓扑的结构关系,但是没有表现出拓扑结构的层次性质。从图 2 的布局效果图中可以看出,图中存在不同程度的交叉连接或“麻团”现象,也不能清晰地展现 Internet 的结构关系和层次性,图中并不能直观地观察 Internet 路由器级拓扑结构。

3 DHL 算法

当节点数较多时,FR 算法的布局效果主次不分明,不能充分展现 Internet 路由器级拓扑的层次性。所以,文中利用节点度分布的幂律性质对 FR 算法进行改进,提出 DHL (Degree Hierarchical Layout) 算法。该算法力求降低布局过程中的时间复杂度和减少边的交叉数,并使布局效果能充分体现网络的层次性。

DHL 算法主要有两个关键特色:

(1) 按照路由器级节点度分布的幂律性质,对节点进行分类,之后分层显示;

(2) 根据层次的不同选取合理的初始温度和迭代

次数。

3.1 层次划分

在布局过程中,FR算法将输入的节点作为一个整体进行布局,因此当节点数目较多时,布局时间长。而文中根据节点度将大的数据集分解成相对较小的数据集,使这些小的数据集中的节点拥有相似的节点度,然后将这些节点分配在同一层,再运用DHL算法进行布局。

Internet路由器级拓扑可以看成是一个无向图 $G=(V,E)$,其中 $V=\{v_1, v_2, \dots, v_m\}$ 为节点集, $E=\{e_1, e_2, \dots, e_n\}$ 为边集,并定义 $d_i(1 \leq i \leq m)$ 为节点 $v_i(1 \leq i \leq m)$ 的节点度, $L_j(1 \leq j \leq k)$ 为不同的层次。根据节点度的不同将节点分成 k 个节点子集,并将这 k 个节点子集分别布局在 $L_j(1 \leq j \leq k)$ 中。其中, L_1 层放置节点度最高的节点子集, L_k 层放置节点度最低的子集,从 L_1 到 L_k 层,节点子集的节点度逐步下降。

文中的DHL算法将这些节点分为三类:核心节点($d_i \geq x_2 + 1$)、中间节点($x_1 \leq d_i \leq x_2$)和树形节点($1 \leq d_i < x_1$)。核心节点是指节点度大于 $x_2 + 1$ 的节点;中间节点是指拓扑图中除了核心节点和树形节点之外的节点;树形节点是指节点度大小位于1和 x_1 之间的节点,通过以下计算得到 x_1 和 x_2 的值。

由于Internet路由器级拓扑的节点度分布满足幂律性质^[3-4],可以用公式(6)表示:

$$P(d) = C(d)^{-r} \quad (6)$$

其中, d 为节点度, $P(d)$ 为节点度为 d 的节点出现的频率; r 和 C 为常量。为了方便计算,用 x 代替公式(6)中的 d ,得到公式(7):

$$p(x) = C(x)^{-r} \quad (7)$$

对公式(7)两边积分可得:

$$1 = \int_{x_{\min}}^{\infty} P(x) dx = C \int_{x_{\min}}^{\infty} x^{-r} dx = \frac{C}{1-r} [x^{-r+1}]_{x_{\min}}^{\infty} \quad (8)$$

通过公式(8)可得:

$$C = (r-1)x_{\min}^{r-1} \quad (9)$$

在Internet路由器级的拓扑中,最小的节点度为1,由文献[2]可得 $r=2.1$,将两个参数带入公式(9)中可得 $C=1.1$ 。

在文献[4]中,作者通过对大量数据的分析得到路由器级拓扑中将近55%的节点位于树中,并且将近30%的节点的节点度为1。有近15%的节点是节点度较大的节点,这些节点位于网络的核心位置,结合公式(7)得到公式(10):

$$0.25 = \sum_{x=2}^{x_1} P(x) = 1.1 \sum_{x=2}^{x_1} x^{-2.1} \quad (10)$$

可得 $x_1 \approx 2$,同样,通过公式(11)可得到 $x_2 \approx 30$ 。

$$0.30 = \sum_{x=3}^{x_2} P(x) = 1.1 \sum_{x=3}^{x_2} x^{-2.1} \quad (11)$$

通过以上分析计算得到,DHL算法按照节点度的大小分为三层, L_1 层中放置核心节点, L_2 层放置中间节点, L_3 层放置树形节点。在布局的过程中,首先布局 L_1 层中的节点,当该层布局完成后,锁定该层中的节点。当节点锁定后,虽然参与吸引力和排斥力的计算,但是不产生位移;当这些步骤完成后再布局 L_2 层中的节点,同样当布局完成后,锁定该层的节点;之后在布局下一层中的节点,直到整体布局完成。DHL算法是可扩展的,中间节点可继续细化拆解为多个层次,但文中只采用三层。

3.2 层次内参数的选取

3.2.1 初始温度的选择

DHL算法同样借鉴模拟退火算法的思想,引入了温度参数。温度参数同时代表一次迭代过程中节点所能移动的最大距离。当节点受力相同时,温度越高,移动的距离越远;温度越低,移动的距离越近。

分层完成后,由于每一层的节点拥有不同的特征,DHL算法给每一层分配不同的初始温度。最低层拥有数量较少的核心节点,这些节点是网络的核心部分,对整个布局效果影响最大,这些高度连接的节点决定了图的最终结构,其他层都是以该层为基础演化而成。所以,该层节点彼此之间的间距应该足够大,防止最后的布局效果出现“麻团”现象。因此,给该层分配最大的初始温度,定义为 InitTemp_1 ,温度和布局区域大小成正比,通过公式(12)获得。

$$\text{InitTemp}_1 = W/10 \quad (12)$$

其中, W 为布局的宽度。

与此相反,对于高层中的节点,虽然节点数量比较多,但是对整体布局影响比较小。所以应该使这些节点尽快达到能量最低的状态,以节省布局所用的时间,提高效率,也避免这些节点在最后的位置周围出现“摆动”现象。因此,对于高层次中的节点,文中将采用相对低的初始温度。该温度由上一层的初始温度和该层中的节点数占总节点数的比例决定,通过公式(13)得到。

$$\text{InitTemp}_{i+1} = \text{InitTemp}_i * |V|_{i+1} / |V|, i=1 \text{ 或 } 2 \quad (13)$$

其中, $|V|_{i+1}$ 为 $i+1$ 层的节点数; $|V|$ 为节点总数。

3.2.2 迭代次数的选取

迭代次数如果设定太小,则不能使系统的能量达到一个比较小的状态,因而无法得到良好的布局效果;如果设定太大,系统或许早已达到了比较好的平衡状

态,这将增加算法的执行时间。为了使布局时间达到最小,应该限制每层中的节点移动时间,使节点尽快达到能量最小的状态。FR 算法实现中整个布局过程中使用完全相同的一个迭代次数,这往往造成不必要的时间浪费。

通过 3.2.1 小节中给定初始温度为每一层次中的节点位移划定了一个范围,层次越高,节点运动的范围越小。由于高层次中的节点不需要移动太大的距离,所以这些层次上所需的迭代次数也不必太大。

给最低层赋予较大的迭代次数,这样既能为后续的计算提供更精确的范围,而且由于最低层时节点的个数是最少的,因此即使迭代次数较高,运行的时间也比较少。此后随着布局图层次的不展开,迭代次数的大小也逐渐减小。当到达较高层时,虽然此时的节点数比较多,但是迭代次数已经降低了,因此所需的计算时间也相应地减少了许多。并且由于前面层次的计算已经将节点的大致范围确定下来,本层次所需完成的只是对节点的位置进行比较小范围的调整而已,因此这样既不会影响最后的布局效果,还能提高算法的执行效率。

对于 FR 算法的实现,在可视化工具 JUNG 和 Prefuse 中,作者做了大量实验,发现迭代次数取 700 时,系统的布局效果和算法执行时间都较理想。DHL 算法实现中仍沿用 700 作为 L_1 层的迭代次数,即 $\text{Ite}_1 = 700$ 。其余层次的迭代次数通过公式(14)计算得到:

$$\text{Ite}_{i+1} = \text{Ite}_i * |V|_i / |V|_{i+1}, i = 1 \text{ 或 } 2 \quad (14)$$

其中, $|V|_i$ 为第 i 层时的节点数。

3.3 算法的实现

DHL 算法中首先根据节点度对节点进行分类,接着将节点分配到不同的层次,最后进行布局。算法的伪代码如下:

算法:基于节点度分层的路由器级拓扑布局算法。

输入:图结构;

输出:图节点的布局位置。

```

1. for  $m = 1$  to  $|V|$  do
2.  $d \leftarrow$  the degree of nodes;
3. map nodes to different levels ;
4. Partition nodes into layers  $L_1, L_2, L_3$ ;
5. for each  $L_j (1 \leq j \leq 3)$  do
6.  $\text{area} \leftarrow W * H$ ;
7. initialize  $G_j = (V_j, E_j)$ ;
8.  $k = C \sqrt{\frac{W * H}{|V|_j}}$ ;
9. function  $F_r(|D(u, v)|) = -k^2 / |D(u, v)|$ ;
10. for  $i = 1$  to iterations do
11. for each  $v \in V_j$  do
12.  $v.\text{dis} = 0$ ;
```

```

13. for  $u \in V_j$  do
14. if  $(u \neq v)$  then
15.  $D \leftarrow v.\text{pos} - u.\text{pos}$ ;
16.  $v.\text{disp} \leftarrow v.\text{disp} + (\frac{D}{|D|}) * F_r(D)$ ;
17. function  $F_a(|D(u, v)|) = |D(u, v)|^2 / k$ ;
18. for each  $e \in E_j$  do
19.  $D \leftarrow e.v.\text{pos} - e.u.\text{pos}$ ;
20.  $e.v.\text{disp} \leftarrow e.v.\text{disp} - (\frac{D}{|D|}) * F_a(D)$ ;
21.  $e.u.\text{pos} \leftarrow e.u.\text{pos} - (\frac{D}{|D|}) * F_a(D)$ ;
22. for each  $v \in V_j$  do
23.  $v.\text{pos} \leftarrow v.\text{pos} + (\frac{v.\text{disp}}{|v.\text{disp}|}) * \min(v.\text{disp}, T_j)$ ;
24.  $v.\text{pos}_x \leftarrow \min(\frac{W}{2}, \max(\frac{-W}{2}, v.\text{pos}_x))$ ;
25.  $v.\text{pos}_y \leftarrow \min(\frac{H}{2}, \max(\frac{-H}{2}, v.\text{pos}_y))$ ;
26.  $T_j \leftarrow \text{cool}(T_j)$ ;
```

算法第 1~4 步中按照节点度的大小将节点分为三类:核心节点、中间节点和树形节点。并将节点分配到相应的层中。第 9~21 步中包括三个主要工作:(1)计算每次迭代过程中排斥力在节点之间产生的位移;(2)计算每次迭代有关节点之间的吸引力产生的位移;(3)计算吸引力和排斥力在每次迭代中合力产生的位移。第 22~25 步计算节点在布局图中的位置。第 26 步对温度进行处理。

3.4 算法的复杂度分析

Spring Embedded 算法的时间复杂度为 $O(|V|^3)$ (其中的 $|V|$ 表示节点数目)。KK 算法的时间复杂度虽然有所降低,但是仍然接近 $O(|V|^3)$,且需要 $O(|V|^2)$ 的空间用来存储边的长度。DH 算法中由于采用了模拟退火算法,该算法的执行时间太长,导致 DH 算法的时间复杂度为 $O(|V|^2 |E|)$ (其中 $|V|$ 表示节点数目, $|E|$ 表示边数目)。FR 算法一次迭代的时间复杂度 $O(|V|^2 + |E|)$,整个算法的时间复杂度为 $O(n_{\text{ite}}(|V|^2 + |E|))$,其中 n_{ite} 表示算法的总迭代数。

由于 FR 算法一次迭代的时间复杂度为 $O(|V|^2 + |E|)$,那么 DHL 算法在 $L_j (1 \leq j \leq k)$ 层中每次迭代的时间复杂度为 $O(|V_j|^2 + |E_j|)$,其中 $|V_j|$ 表示 j 层中的节点数目, $|E_j|$ 表示 j 层中的边数。那么 DHL 算法中每一层总的算法时间复杂度为 $O(\text{Ite}_j(|V_j|^2 + |E_j|))$,其中 Ite_j 为 j 层的总迭代数。DHL 算法的总时间复杂度为三层的时间复杂度之和,表示为 $O(\sum_{j=1}^3 \text{Ite}_j(|V_j|^2 + |E_j|))$ 。

4 实验及结果分析

实验开发语言是 Java, 开发环境是 jdk1.7 + Eclipse4.3, 实验的机器配置为: CPU Intel Core2 双核 T4300 @ 2.10 GHz; 内存 2 GB; 硬盘 250 GB; 操作系统为 Windows XP。

结合图布局的评价标准和 Internet 路由器级拓扑的自身特点, 文中通过边的交叉数、算法的执行时间对 Spring Embedded 算法、FR 算法和 DHL 算法进行实验评估, 最后给出各算法的布局效果。对于路由器级拓扑数据的获取主要有 2 种方式:

(1) 通过 Traceroute 的探测方式, 该方法获得的拓扑图具有节点数较多、平均节点度较大的特点;

(2) 互联网数据分析合作协会 (Cooperative Association for Internet Data Analysis, CAIDA) 提供的路由器级拓扑数据, 这些数据标注了路由器之间的连接路径; 也包含完整的按时间间隔获取的路由器级拓扑数据。

文中实验数据采用第二种方式获得, 共进行了六组实验, 实验数据如表 1 所示。

表 1 Internet 路由器级拓扑数据

	边数目	节点数目
第一组数据集	1 078	500
第二组数据集	1 368	700
第三组数据集	2 091	1 000
第四组数据集	3 574	2 000
第五组数据集	4 762	3 000
第六组数据集	7 848	4 000

4.1 执行时间

算法的执行时间是指布局图从其初始状态到收敛状态所消耗的时间, 它是衡量力导向算法效率的重要标准。对表 1 中的数据, 每个算法各执行十次, 取其平均值作为实验的最终结果, 实验结果如图 3 所示。

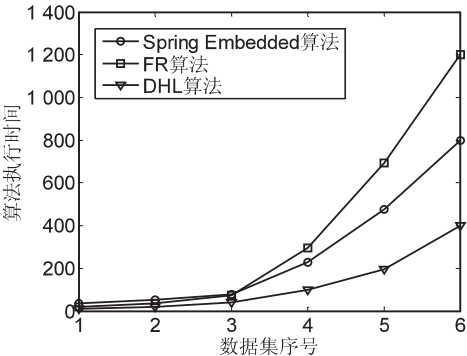


图 3 各布局算法所用时间对比图

根据图中实验所得数据可以看出 FR 算法较之 Spring Embedded 算法, 更不适用于路由器级拓扑的布局。当节点数小于 1 000 时, FR 算法的布局时间比 Spring Embedded 算法执行时间略短, 是 DHL 算法的 2

倍。当节点数目大于 1 000 时, FR 算法执行时间均比 Spring Embedded 算法长, 且上升的幅度比后者更大, 是 DHL 算法执行时间的 3 倍。当节点数目超过 2 000 时, Spring Embedded 算法和 FR 算法的执行时间均大幅度上升。较之这两种布局算法, 文中的 DHL 算法执行时间的曲线较平滑, 呈缓慢上升趋势。

4.2 边的交叉数分析

边的相交数 $cross(G)$ 是指图 G 中两两相交的边的个数。文中对表 1 中的六组数据集分别进行 Spring Embedded 算法、FR 算法和 DHL 算法三种布局, 并计算出各自的 $cross(G)$, 实验结果如图 4 所示。

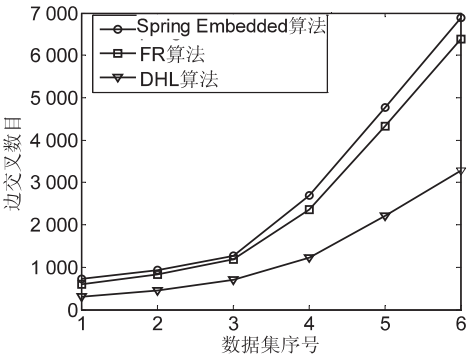


图 4 各布局算法的边交叉数

从实验结果中可以看出, Spring Embedded 算法和 FR 算法的边交叉数基本一致, 随着节点个数的增加, 尤其当节点数超过 1 000 时, 两种布局算法的边交叉数都大幅度上升。另外, 随着节点个数的增加, 三种算法的布局效果图的质量都有不同程度的下降, 这是因为随着边数的增加, 在布局区域面积不变的情况下, 二维空间中边的交叉不可避免。但是 DHL 算法中边交叉数的上升幅度较平缓, 边的交叉数也只是 FR 算法的 50%。

4.3 布局效果

取节点数为 1 000, 边数为 2 091 作为实验数据, 采用三种布局算法对这组数据进行布局, 生成拓扑图, 对比三种算法的布局效果。图 5 中, (a)、(b) 和 (c) 显示的是 Spring Embedded 算法、FR 算法和 DHL 算法所布局的图。

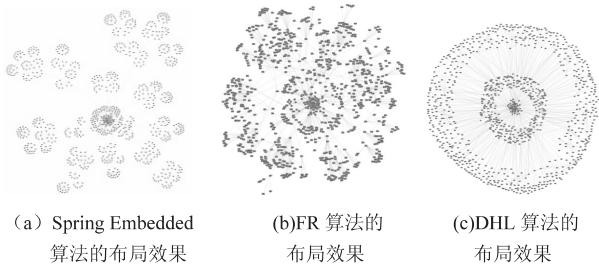


图 5 三种算法的布局效果图

从图 5 中可以明显看出, Spring Embedded 算法和 FR 算法均没能体现出路由器级拓扑的层次性。而

DHL 算法可以明显地看出网络的三个层次,核心节点位于网络的中心,中间节点围绕在核心节点的周围,树形节点位于整个布局图的边缘。

5 结束语

文中分析了 FR 算法,并针对算法执行时间长和所布局的图不能反映出网络的层次性等问题,提出一种改进的 FR 算法—DHL 算法。针对 Internet 路由器级拓扑节点度分布的幂律性质对网络进行分层显示,并根据层次的不同选取合理的初始温度和迭代次数,实验结果表明 DHL 算法有效地降低了算法的时间复杂度和边的交叉数,且布局效果体现了网络的层次性。但是,DHL 算法仍然存在部分交叉的边,如何减小边的交叉数目将是下一步研究的重点。

参考文献:

[1] 吴 鹏,李思昆. 适于社会网络结构分析与可视化的布局算法[J]. 软件学报,2011,22(10):2467-2475.

[2] Vazquez A,Pastor-Satorras R,Vespignani A. Internet topology at the router and autonomous system level[DB/OL]. 2002. arXiv:cond-mat/0206084.

[3] Faloutsos C,Kang U. Managing and mining large graphs: patterns and algorithms[C]//Proceedings of the 2012 ACM SIGMOD international conference on management of data. [s. l.]:ACM,2012:585-588.

[4] Broido A,Claffy K. Internet topology:connectivity of IP graphs[C]//Proc of international symposium on the convergence of IT and communications. [s. l.]:International Society for Optics and Photonics,2001:172-187.

[5] Mútray P,Haga P,Laki S,et al. On the network geography of the Internet[C]//Proc of IEEE. Shanghai:IEEE,2011:126-

130.

[6] Cheswick W,Burch H. Mapping the Internet[J]. IEEE Computer,1999,32(4):97-102.

[7] Motta E,Mulholland P,Peroni S, et al. A novel approach to visualizing and navigating on topology[C]//Proc of ISWC. Berlin:Springer,2011:470-486.

[8] Aupetit S,Puret A. Visual tools to select a layout for an adapted living area[C]//Proc of AAATE. [s. l.]:[s. n.],2009:345-355.

[9] Hochstein S. View from the top:hierarchies and reverse hierarchies in the visual system[J]. Neuron,2012,36(5):791-804.

[10] Inoue K,Shimozono S,Yoshida H,et al. Application of approximate pattern matching in two dimensional spaces to grid layout for biochemical network maps[J]. PLoS ONE,2012,7(6):e37739.

[11] Bhatele A,Kale L V. Heuristic-based techniques for mapping irregular communication graphs to mesh topologies[C]//Proc of 2011 IEEE 13th international conference on high performance computing and communications. Banff:IEEE,2011:765-771.

[12] Eades P. A heuristic for graph drawing[J]. Congressus Numerantium,1984,42:149-160.

[13] Kamada T,Kawai S. An algorithm for drawing general undirected graphs[J]. Information Processing Letters,1989,31(1):7-15.

[14] Davidson R,Harel D. Drawing graphs nicely using simulated annealing[J]. ACM Transactions on Graphics,1996,15(4):301-331.

[15] Fruchterman T M J,Reingold M. Graph drawing by force-directed placement[J]. Software - practice and Experience,1991,21(11):1129-1164.

《计算机技术与发展》投稿须知

1. 新投稿可登陆 <http://www.xactad.org> 在线投稿系统。投稿前请作者自审一遍,论文要求主题突出、用语规范、层次清楚、结构严谨、文字精练、文理通顺、逻辑性强。

2. 论文题目不应超过 20 个汉字。

3. 作者姓名及作者所在单位部门、城市、邮政编码(多位作者不在同一单位应分别开列)。

4. 摘要须包含目的、方法、结果、结论 4 方面的内容,300 字以上。

5. 关键词 4~8 个为宜。

2~5 项的内容必须具备中、英文对照。

6. 作者简介:姓名、出生年、性别、学位、研究方

向;

导师简介:姓名、职称、研究方向。

7. 作者在投稿时须注明是否是中国计算机学会(CCF)会员(高级会员、普通会员、学生会会员)。若是会员,请注明会员号(凡第一作者为 CCF 会员/高级会员/学生会会员者,将享受 85 折的版面费优惠。)

8. 投稿时请写明详细通信地址、邮政编码、联系电话、Email 信箱等各项必备内容。审稿周期为 30 天左右,若录用,将发送电子录用通知单,请经常登陆系统查看稿件所处状态。稿件刊登后赠送样刊 2 本。

基于节点度分层的路由器级拓扑布局算法

作者：[刘金明](#)，[万明祥](#)，[LIU Jin-ming](#)，[WAN Ming-xiang](#)

作者单位：[刘金明, LIU Jin-ming\(南京邮电大学 计算机学院, 江苏 南京, 210003\)](#)，[万明祥, WAN Ming-xiang\(南京邮电大学 物联网学院, 江苏 南京, 210003\)](#)

刊名：[计算机技术与发展](#)

英文刊名：[Computer Technology and Development](#)

年，卷(期)：2015(1)

引用本文格式：[刘金明](#). [万明祥](#). [LIU Jin-ming](#). [WAN Ming-xiang](#) [基于节点度分层的路由器级拓扑布局算法](#)[期刊论文]-[计算机技术与发展](#) 2015(1)