知网个人查重服务报告单(全文标明引文)

报告编号:BC20231213203202634171256 检测时间:2023-12-13 20:32:02

篇名：基于Dinic算法的网络路由优化

作者：孙潇桐
所在单位：西北大学

检测类型：课程作业(本科)
比对截止日期：2023-12-13

检测结果

去除本人文献复制比：0% 去除引用文献复制比：0% 总文字复制比：0%
单篇最大文字复制比：0% ()

重复字符数：[0] 单篇最大重复字符数：[0] 总字符数：[6289]

（注释：无问题部分 文字复制部分 引用部分）

1. 基于Dinic算法的网络路由优化 总字符数：6289

相似文献列表

去除本人文献复制比：0%(0) 去除引用文献复制比：0%(0) 文字复制比：0%(0)

原文内容

成绩	

成绩
计算机网络期中大作业
题目：基于Dinic算法的网络路由优化
学生姓名孙潇桐
学号 2021117405
指导教师郝星星
院系软件学院
专业软件工程
年级 2021
教务处制
二〇二三年十二月

摘要
本论文旨在通过引入最大流算法，实现对网络连接质量的综合优化，特别关注大带宽和低延迟两个关键指标。相较于传统的最短路径算法，我们考虑结果而非过程，使用最大流算法找到实际网络中节点间的最大流量，从而指导数据包的转发策略。通过Dinic算法的增广，成功求解了最大流问题，并在实际网络建模中取得显著的性能提升。论文最终通过实现的结果展示，验证了算法在网络资源最大化利用和负载均衡方面的有效性。

关键词： 最大流算法，大带宽，Dinic算法

Abstract
This paper aims to comprehensively optimize network connection quality by introducing the maximum flow algorithm, with a specific focus on two key metrics: high bandwidth and low latency. In contrast to traditional shortest path algorithms, our approach prioritizes results over processes, utilizing the maximum flow algorithm to identify the maximum flow between nodes in actual networks, guiding packet forwarding strategies. Through the augmentation of the Dinic algorithm, we successfully solved the maximum flow problem and achieved significant performance improvements in practical network modeling. The results presented in this paper

validate the effectiveness of the algorithm in maximizing network resource utilization and load balancing.
Keywords：Maximum Flow Algorithm, High Bandwidth, Dinic Algorithm
目录

1 绪论
1.1 课题研究的背景和意义
在当今数字化时代，网络通信扮演着关键的角色，影响着我们的日常生活和商业活动。随着网络规模的增大和网络应用的多样化，网络路由的优化变得尤为重要。网络路由不仅仅关乎数据的快速传输，还涉及到网络的性能、可靠性、安全性等多方面的因素。

随着网络的不断发展和应用场景的变化，对网络路由的要求也在不断提高。现在的网路需求可以分为两个大类----延迟和带宽，而Dinic算法[1]解决的最大流问题正好就能用于搜寻最大带宽的路径。因此，有必要对Dinic算法进行深入研究，以满足现代网络通信的需求。

通过优化Dinic算法，我们可以提高网络的传输效率，缩短数据传输时间，从而提升整个网络的性能。而且随着网络规模的扩大，传统的路由算法可能面临扩展性的问题。通过改进Dinic算法，我们可以使其更好地适应大型网络环境，提高网络的可扩展性。

在现在这个万物互联的时代，网络安全前所未有的重要，优化的Dinic算法可以对网络中的流量进行更有效的管理，提高对网络攻击的识别和应对能力，从而增强网络的安全性。通过对Dinic算法的深入研究，我们可以获得对网络流优化更全面的理解，为未来网络算法的设计和研究奠定基础，推动网络技术的不断创新。

总体而言，研究优化Dinic算法在网络路由中的应用对于提高网络性能、可扩展性和安全性，以及支持新兴网络应用具有深远的意义。这一研究将有助于推动网络技术的进步，为构建高效、安全、可靠的网络基础设施提供重要支持。
1.2 论文内容和创新点
本论文旨在通过对网络连接质量的综合考量，引入最大流算法来实现对网络中大带宽和低延迟两个指标的优化。与传统的最短路径算法不同，我们关注的是结果而非过程，考虑到短路径未必能提供最大带宽或最低延迟，可能受线路干扰或交换机性能限制的影响。通过最大流算法，可以找到从源节点到目标节点的最大流量发包方式，从而指导终端向哪个节点发送数据包以最大化带宽，同时通过计算延迟的倒数来实现最小延迟。
2 最大流模型
2.1 最大流算法
我希望将网络的连接质量分为两个指标，一个是大带宽，另一个是低延迟。为了达到这一点，我们就需要使用最大流算法。通过最大流算法，我们就能找到从当前节点到目标节点的最大流量的发包方式。与最短路径算法不同，我这里考虑的仅仅是结果而不是过程。

换句话说，就是短的路径并不总是带宽最大或者延迟最低的，路径可能因为一些线路干扰或者是交换机性能的限制而反而没有其他更长的路径优秀。而且最大流算法寻找的是多个路径流量的最大值，最后可以得到每条路径的容量。得到的容量就可以指导终端应该向哪个节点发包，向那个节点发送百分之多少的包来最大化带宽。延迟的话应该是越低越好，于是我们就在计算延迟的时候计算延迟的倒数就能达到我们的目的。
2.2 结合实际建模
于是在计算的时候，我们将当前节点设为源点s，将目的节点设为汇点t。终端与终端之间都测量他们之间链路的最大带宽w，将w设为这条链路的容量（Capacity）c，并将这条链路抽象成一条有向边。节点u和节点v之间的容量记为，实际的流量记为。对每条边来说，都有经过的流量小于容量，形式化的说就是。

我们的目的是求出从源点s到汇点t最大流量，我们希望指定最大的流f，也就是最大化流量，其中V就是网络中的所有节点。
3 Dinic算法的引入
3.1 Ford－Fulkerson 增广
Dinic算法是一种基于Ford－Fulkerson 增广的算法，该方法运用贪心的思想，通过寻找增广路来更新并求解最大流，在给定网络G及其上的流f，有如下定义。

对于边我们将其容量与流量之差称为剩余容量，即。将网络G中所有结点和剩余容量大于0的边构成的子图称为残量网络，即，其中。

我们将上一条从源点s到汇点t的路径称为增广路。对于一条增广路，我们给每一条边都加上等量的流量，以令整个网络的流量增加，这一过程被称为增广，最大流的求解可以被视为若干次增广分别得到的流的叠加。

此外，在 Ford－Fulkerson 增广的过程中，对于每条边，我们都新建一条反向边。我们约定，这一性质可以通过在每次增广时引入退流操作来保证，即增加时应当减少同等的量，这样的目的是为了引入允许反悔的贪心策略。

3.2 使用Dinic算法实现Ford－Fulkerson增广

考虑在增广前先对做BFS分层，即根据节点u到源点s的距离把结点分成若干层。令经过u的流量只能流向下一层的节点v，即删除u向层数标号相等或更小的节点的出边，我们称剩下的部分为层次图。形式化地，我们称是的层次图，其中。

如果我们在层次图上找到一个最大的增广流，使得仅在上是上是不可能找出更大的增广流的，则我们称是的阻塞流，此处的增广流可能是多条增广流合并得到的，不一定是单条增广流。

接下来是Dinic算法的基本流程：

1. 在上BFS得到的层次图
2. 在上DFS得到阻塞流
3. 将并到原先的流f中，即
4. 重复以上的操作，直到BFS没法找到拥有新的增广流的层次图

3.3 时间复杂度的计算

我们可以将Dinic算法分为两个部分，一个部分是BFS分层，第二个部分是DFS计算阻塞流。每次BFS后必然执行一次DFS，所以我们可以分别计算BFS和DFS的部分，再将他们乘起来。

由于我的能力有限，我查询到算法的增广轮数，也就是BFS次数是的。单轮增广中 DFS 求阻塞流的时间复杂度是，再将他们乘起来就是总的复杂度。

4 Dinic算法的实现

4.1 Dinic算法的改进

单纯的Dinic算法只能得到最大流的流量，但是如果要指导路由器的转发策略，我们必须对算法进行改进，以使得其可以输出具体的路径。

首先，先执行一次完整的Dinic算法，按照我3.2节描述的方式。然后再执行一次BFS操作，将每条边的容量减去他们经过的流量，得到他们的剩余流量。再通过一次DFS，通过计算每个点的剩余容量是否能继续进行，来借用DFS的性质得到多条路径，我们再在路径的行驶过程中进行压栈和弹栈，当检测到可以到达汇点t的时候停止，将栈中的节点输出，便是一条路径。

4.2 改进后算法的实现

先进行一次Dinic算法，下面是伪代码：

```
function BFS():
Set array depth to all 0
Set array depth to 1 at the index s(starting point)
queue ← new Queue
push s in to queue
while que is not empty:
front ← first node of queue
remove the first node of queue
for each edge i connect to node front:
let node j ← node next to i
if the flow in i > 0 and depth[Node] equals 0:
depth[j] ← depth[front] + 1
push j to the back of queue
if sink point is reachable return true
else return false
function DFS(now, maxflow):
if now is the sink point t:
return maxflow
sum ← 0
for each edge i next to node now:
let node j ← the node
if flow of edge i > 0 and depth[Node] == depth[now] + 1:
f ← DFS(j, min(maxflow, the flow of edge i))
the flow of edge i minus f
the flow of reverse edge of i plus f
sum ← sum += f
maxflow minus f
if maxflow is 0 then break this loop
if sum is 0 remove node now
return sum
```

接下来使用一次BFS计算每个边的剩余容量：

```
function pre(s):
queue ← new Queue
```

```
push start point s to the back of queue
mark node s as visited
while queue is not empty:
front ← front of queue
remove first node from queue
for each edge i connect to node front:
to ← the other side of edge i
if origin value of edge i is not 0:
origin value minus the flow of edge i
if node to is visited before than continue
mark node to as visited
push to the back of queue
```

然后通过DFS根据之前计算的流量，得到路径。

```
function DFS(now, t, value, path):
if value is 0 than return
mark now as visited
append node now to the back of the array path
if now is the sink point t:
mark t as not visited
log the path in the array path
remove the last node from array path
return
rest_value ← value
for each edge i connected to the node now:
to ← the other side to edge i
if to is visited or res_value ≤ 0 than continue
now_value ← min(res_value, origin value of edge i)
DFS(to, t, now_value, path)
res_value ← res_value - now_value
remove the last element of array path
mark node now as not visited
return
```

4.3 结果展示

我用C++实现了我上面提到的代码，假设有如下的拓扑图：

假设源点是4，汇点是3。通过我的程序的计算，可以得到：

可以看到，从源点4到汇点3，最大流是50，由3条路径组成。通过这样的计算，我们就能指导路由器对数据包的转发可以分三条路径进行，实现通路之间的负载均衡，并最大化网络资源的使用率。

总结与展望

本论文通过引入最大流算法，以优化网络连接质量为目标，实现了对大带宽和低延迟的综合考量。在最大流算法的基础上，利用Dinic算法进行增广，得到了网络中节点之间的最大流量，并成功将其应用于指导数据包的转发策略，以实现对网络资源的最大化利用和通路之间的负载均衡。

通过论文的研究，我们深入探讨了最大流算法在网络优化中的应用。相较于传统的最短路径算法，我们的方法更加注重结果，通过最大流算法找到了实际网络中具有最大带宽和最小延迟的路径。在实际的建模中，我们将网络节点抽象成有向图的形式，成功地应用了Dinic算法来求解最大流问题。该方法不仅考虑了带宽的最大化，还通过计算延迟的倒数实现了对延迟的优化，从而在网络性能上取得了显著的改进。

尽管本论文在网络优化方面取得了一定的成果，仍有一些方向值得进一步深入研究和探索。首先，可以考虑引入更复杂的网络拓扑结构，以更全面地模拟实际网络环境。其次，可以进一步优化算法的实现，提高计算效率，并探讨在大规模网络中的应用可行性。此外，考虑到网络动态变化的情况，未来的研究可以集中在自适应算法的设计上，以适应实时变化的网络条件。

总体而言，本研究为网络性能优化提供了一个有益的思路，未来的研究将继续致力于推动网络优化算法的发展，以更好地适应不断演变的网络环境。

参考文献

[1] DINITZ Y. Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation[J]. Soviet Math. Dokl., 1970, 11: 1277-1280.

5. 复制比按照“四舍五入”规则,保留1位小数;若您的文献经查重检测，复制比结果为0，表示未发现重复内容，或可能存在的个别重复内容较少不足以作为判断依据
6. 红色文字表示文字复制部分;绿色文字表示引用部分(包括系统自动识别为引用的部分);棕灰色文字表示系统依据作者姓名识别的本人其他文献部分
7. 系统依据您选择的检测类型(或检测方式)、比对截止日期(或发表日期)等生成本报告
8. 知网个人查重唯一官方网站:https://cx.cnki.net