文章编号:1671-4598(2019)07-0236-06 DOI:10.16526/j.cnki.11—4762/tp.2019.07.051 中图分类号:TN711.2 文献标识码:A

基于拓扑感知节点排序的虚拟网络嵌入

马 煜
(陕西中医药大学 信息化建设管理处,陕西 咸阳 712046)

摘要:在共享底层上嵌入多个虚拟网络(VN)是云计算平台和大规模可切片网络测试平台的一个挑战性问题;本文利用马尔可夫随机游走模型,根据网络节点的资源和拓扑属性对其进行排序,这种新的拓扑感知节点排序方法可反映节点的相对重要性;利用节点排序设计了两种 VN 嵌入算法:RW—MaxMatch 和 RW—BFS;仿真实验表明:与现有的嵌入算法相比,拓扑感知节点排序具有较好的资源度量,并且所提出的基于 RW 的算法增加了长期平均收益和接受率。

关键词:网络虚拟化;云计算;虚拟网络嵌入;拓扑感知;随机游走;马尔可夫链

Virtual Network Embedding Based on Topology—aware Node Sorting

Ma Yu
(Division of Informatization Construction and Management, Shaanxi University of Chinese Medicine, Xianyang 712046, China)

Abstract: Embedding multiple virtual networks (VNs) on the shared bottom is a challenge for cloud computing platforms and large—scale sliceable network testing platforms. In this paper, Markov random walk model is used to rank nodes according to their resources and topological attributes. This new topology—aware node ranking method can reflect the relative importance of nodes. Two VN embedding algorithms, RW—MaxMatch and RW—BFS, are designed by using node sorting. The simulation results show that compared with the existing embedding algorithms, the ranking of topology—aware nodes has better resource metrics, and the proposed RW—based algorithm increases the long—term average revenue and acceptance rate.

Keywords: network virtualization; cloud computing; virtual network embedding; topology awareness; random walk; Markov chain

0 引言

共享虚拟化资源可以实现新的计算和网络模式,例如,基于云计算平台[1]和可切片的网络测试平台[2]。云平台或网络基础设施的用户请求共享资源,包括 CPU 容量、存储空间、网络带宽等,而基础设施提供商则尽最大努力为请求提供服务,也就是所谓的虚拟网络(VN)。在这种虚拟化环境中,Vn 的资源分配对用户的计算需求和资源提供者的收益都至关重要。

在多租户网络虚拟化环境中,基础设施提供商(InP)(例如,云提供商)和服务提供商(SP)(如云用户/租户)扮演两个不同的角色,即 InP 管理物理基础设施,而 SP 创建 VN 并提供端到端服务[3]。将 SP 的 VN 请求映射到 InP 的底层网络上,也称为 VN 嵌入。因此,设计启发式方法已经成为 VN 嵌入的主要研究方向[4]。早期的算法通过一个节点的 CPU 容量和带宽来测量节点的资源,而不考虑 VN 和底层网络的拓扑结构。然而,节点的拓扑属性对映射结果的成功和效率具有显着影响。

本文设计了两种新的 VN 嵌入算法:RW—MaxMatch 和 RW—BFS。RW—MaxMatch 算法根据节点的等级将虚拟节点映射到底层节点,然后通过找到具有不可分离路径的最短路径并利用可拆分路径解决多商品流问题,将虚拟链路嵌入到映射节点之间。RW—BFS 算法是基于广度优先搜索的回溯 VN 嵌入算法,其利用节点等级在同一阶段嵌入虚拟节点和链路。

1 网络模型及问题描述

1.1 底层网络

底层网络可以用加权无向图 $G_s=(N_s,L_s,A_s^n,A_s^l)$ 表示,其中,N_s 是底层节点集,L_s 是底层链路集。A_s^n 和 A_s^l 分别表示底层节点和链路的属性。节点的属性包括处理能力、存储和位置。链路的典型属性是其带宽。在本文中,考虑了节点属性的可用 CPU 容量和链路属性的可用带宽。用 P_s 表示底层网络中所有无环路径集合。

图 1 给出了 VN 嵌入的示例,图 1(b)给出了一个底层网络,其中矩形中的数字是节点处可用的 CPU 资源,而链路上的数字表示可用的带宽。

1.2 虚拟网络请求

与底层网络类似,本文使用无向图 $G_v=(N_v,L_v,C_v^n,C_v^l)$ 来表示虚拟网络,其中,N_v 是一组虚拟节点,L_v 是一组虚拟链路。虚拟节点和链路与其容量约束相关,分别由 C_v^n 和 C_v^l 表示。本文还通过 $VNR^{(i)}(G_v,t_a,t_d)$ 表示 VN 请求,其中 t_a 是 VNR 的到达时间,t_d 是 VN 停留在底层网络中的持续时间。当第 i 个 VNR 到达时,底层网络应向 VN 分配

收稿日期:2019-02-14; **修回日期**:2019-02-26。

作者简介:马 煜(1989-),男,河北保定人,助理工程师,硕士,主要从事计算机网络安全,算法研究方向的研究。

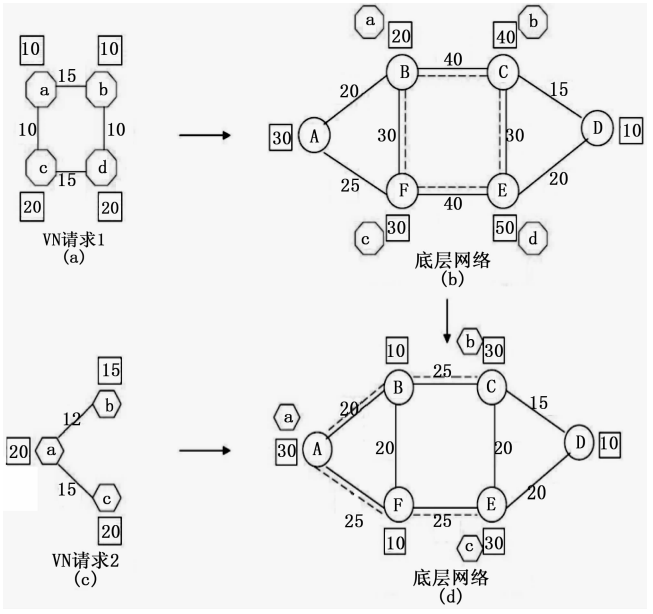

图 1　VN 嵌入的示例

资源以满足虚拟节点和链路的要求。如果没有足够的底层资源可用，则应拒绝或推迟 VNR。当 VN 离开时，释放分配的底层资源。在使用 VN 时，所有节点可能都有不同的角色，例如，目录或文件服务器等。为了简单起见，本文忽略了这些依赖关系。图 1（a）和图 1（c）表示具有节点和链路要求的两个 VN 请求。

1.3　VN 嵌入问题描述

VN 嵌入问题由从 G_v 到 G_s 子集的映射 $M:G_v(N_v,L_v)\rightarrow G_s(N'_s,P'_s)$ 定义，其中，$N'_s\subset N_s$，$P'_s\subset P_s$。映射可以分解为两个映射步骤：（1）节点映射将虚拟节点放置到满足节点资源约束的不同底层节点；（2）链路映射将虚拟链路分配给满足链路资源要求的底层上的无环路径。

图 1（a）和图 1（b）给出了 VNR 1 的 VN 嵌入方案。图 1（c）和图 1（d）给出了 VNR2 的另一个 VN 嵌入解决方案，其中还显示了剩余资源。注意，不同 VNR 的虚拟节点可以映射到同一个底层节点上。

1.4　嵌入目标

VN 嵌入的主要目标是在 VN 请求到达和离开时，将 VN 映射到底层网络以有效利用底层网络资源。与文献［5］类似，在 t 时接受 VNR 的收益可通过以下公式计算：

$$R(G_v,t)=\sum_{d_v\in N_v}CPU(d_v)+\sum_{l_v\in L_v}BW(L_v)\qquad(1)$$

其中：$CPU(d_v)$ 和 $BW(L_v)$ 分别表示 CPU 的虚拟节点 d_v 和链路 L_v 的带宽要求。

在 t 时接受 VNR 的成本定义为分配给该 VN 的底层总资源的总和：

$$C(G_v,t)=\sum_{d_v\in N_v}CPU(d_v)+\sum_{l_v\in L_v}\sum_{l_s\in L_s}BW(f^{l_v}_{l_s},l_v)\qquad(2)$$

其中：如果底层链路 l_s 为虚拟链路 l_v 分配带宽资源，则 $f^{l_v}_{l_s}\in\{0,1\}$ 和 $f^{l_v}_{l_s}=1$，否则 $f^{l_v}_{l_s}=0$。$BW(f^{l_v}_{l_s},l_v)$ 是从 l_s 分配给 l_v 的带宽。

从 InPs 的角度来看，一种高效且有效的在线 VN 嵌入算法可以最大限度地提高 InPs 的收益，并从长远来看提高底层网络的利用率，长期平均收益为：

$$AR=\lim_{T\rightarrow\infty}\frac{\sum_{t=0}^{T}R(G_v,t)}{T}\qquad(3)$$

底层网络的 VNR 接受率可表示为：

$$RAR=\lim_{T\rightarrow\infty}\frac{\sum_{t=0}^{T}VNR_s}{\sum_{t=0}^{T}VNR}\qquad(4)$$

其中：VNR_s 是底层网络成功接受 VN 请求的数量。

本文还考虑了长期收益成本比，进而量化底层网络的资源利用效率：

$$R/C=\lim_{T\rightarrow\infty}\frac{\sum_{t=0}^{T}R(G_v,t)}{\sum_{t=0}^{T}C(G_v,t)}\qquad(5)$$

如果 VN 嵌入解决方案的长期平均收益大致相同，则 VN 接受率和 R/C 比率越高越好。

2　拓扑感知节点排序

VN 嵌入涉及到节点映射和链路映射。节点映射需要借助具有足够 CPU 资源的底层节点来实现，而链路映射需要在任意两个选定底层节点上以及路径上都有足够的链路资源来实现。文献［6］在将节点映射和链路映射分为两个不同执行阶段：节点映射阶段完成节点选择，链路映射阶段完成链路分配和路径选择。本文在节点映射阶段采用了不同的拓扑组合方法来提高链路映射的成功率和效率。通过对比具有相同 CPU 资源的节点 $n1$ 和节点 $n2$，如图 2 所示。

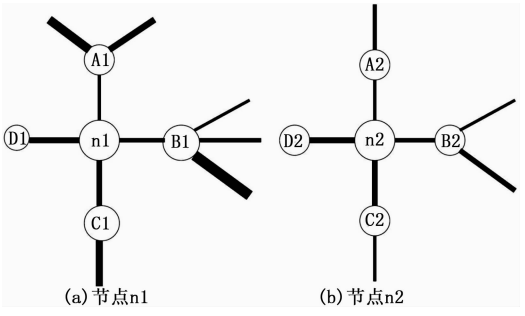

图 2　拓扑感知节点示例

图 2 中，较大的节点表示具有更多 CPU 资源的节点，较宽的线表示具有更多资源的节点映射。节点 $n1$ 比节点 $n2$ 的具有更大的概率实现链路映射，这是由于节点 $n1$ 的邻居节点拥有更多的资源。

本文定义了节点等级的概念来度量节点的资源可用性。即给定的节点 u 的等级由其 CPU 功率和其输出链路的总带宽决定。它还受到从 u 到达的节点等级的影响。使用其 CPU 和输出链路的总带宽进行第一个建模。通过将可到达的节点分为两个组来进行第 2 个建模，即从 u 到输出链路的节点和通过多跳从 u 到输出链路的节点。在本文中，定义了

一个跳跃概率来模拟一个节点通过多跳从 u 到达的可能性，并定义一个正向概率来模拟来自 u 的正向链路中相邻节点的影响。令：

$$H(u) = CPU(u) \sum_{l \in L(u)} BW(l) \tag{6}$$

其中：在底层网络上，$L(u)$ 是 u 的所有输出链路的集合，$CPU(u)$ 是 u 的剩余 CPU 资源，$BW(l)$ 是链路 l 的空闲带宽资源，在虚拟节点上，$CPU(u)$ 和 $BW(l)$ 分别是节点 u 的容量约束。节点 u 的初始 NodeRank 值为：

$$NR^{(0)}(u) = \frac{H(u)}{\sum_{v \in V} H(u)} \tag{7}$$

假设 $u, v \in V$ 是两个不同的节点。令：

$$p^J_{uv} = \frac{H(v)}{\sum_{\omega \in V} H(\omega)} \tag{8}$$

$$p^F_{uv} = \frac{H(v)}{\sum_{\omega \in nbr_1(u)} H(\omega)} \tag{9}$$

其中：p^J_{uv} 表示从节点 u 到节点 v 上着陆的跳跃概率，$nbr_1(u) = \{v \mid (u,v) \in E\}$，$p^F_{uv}$ 表示从节点 u 到节点 v 的正向概率，其中 $(u,v) \in E$。显然，

$$\sum_{v \in V} p^J_{uv} = 1, \sum_{v \in nbr_1(u)} p^F_{uv} = 1 \tag{10}$$

概率 p^J_{uv} 和 p^F_{uv} 可以分别被视为从节点 u 和节点 u 相邻节点可到达的任何节点对节点 u 的资源投票。非相邻节点的投票意味着 u 和 v 之间应该存在多跳路径，因此，两个节点的拓扑信息也可以嵌入到概率中。

对于任何节点 $v \in V$，令：

$$NR^{(t+1)}(v) = \sum_{u \in V} p^J_{uv} \cdot p^J_u \cdot NR^{(t)}(u) + \sum_{u \in V} p^F_{uv} \cdot p^F_u \cdot NR^{(t)}(u) \tag{11}$$

其中：$p^J_u + p^F_u = 1$，$p^J_u \geqslant 0$，$p^F_u \geqslant 0$，$t = 0,1,\ldots$。p^J_u 和 p^F_u 是偏差因子。

对于具有 $V = \{v_1, v_2, \ldots, v_n\}$ 的 n 个节点的网络，令 $NR^{(t)}_i = NR^{(t)}(v_i)$，并且利用 $NR^{(t)} = (NR^{(t)}_1, NR^{(t)}_2, \ldots, NR^{(t)}_n)^T$ 表示在迭代 t 处节点等级的向量，其中 $t = 0,1,\ldots$。可得：

$$NR^{(t+1)} = T \cdot NR^{(t)} \tag{12}$$

其中：T 是由马尔可夫链定义的一步转移矩阵：

$$T = \begin{pmatrix} p^J_{11} & p^J_{12} & \cdots & p^J_{1n} \\ p^J_{21} & p^J_{22} & \cdots & p^J_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ p^J_{n1} & p^J_{n2} & \cdots & p^J_{nn} \end{pmatrix} \cdot \begin{pmatrix} p^J_1 & 0 & \cdots & 0 \\ 0 & p^J_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & p^J_n \end{pmatrix} + \begin{pmatrix} 0 & p^F_{12} & \cdots & p^F_{1n} \\ p^F_{21} & 0 & \cdots & p^F_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ p^F_{n1} & p^F_{n2} & \cdots & 0 \end{pmatrix} \cdot \begin{pmatrix} p^F_1 & 0 & \cdots & 0 \\ 0 & p^F_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & p^F_n \end{pmatrix} \tag{13}$$

由于 T 是最大特征值等于 1 的随机矩阵，因此，T 是稳定的。这保证了上述递推关系收敛到 $NR^{(*)} = (NR^{(*)}_1, NR^{(*)}_2, \ldots, NR^{(*)}_n)^T$，即稳态分布 [7]。这可以使用由算法 1 给出的经典迭代方案 [8] 来计算。

算法 1：NodeRank 计算方法 1：给定一个正值 ε，$i \leftarrow 0$

```
repeat
NR^(i+1) ← T · NR^(i)
δ ← ‖ NR^(i+1) − NR^(i) ‖
i ++
until δ < ε
```

3 NodeRank 映射：RW－MaxMatch 和 RW－BFS 算法

利用拓扑感知的节点资源等级 $NR^{(*)}$，本文设计了两种新的 VN 嵌入算法，即 RW－MaxMatch 和 RW－BFS，它们都将底层网络和虚拟网络作为输入。

3.1 RW－MaxMatch

3.1.1 描述和讨论

RW－MaxMatch 是一个两阶段 VN 嵌入算法。在第一阶段中，它首先计算每个虚拟节点和底层节点的 NodeRank 值。然后，它根据虚拟节点的 NodeRank 值以非递增的顺序对虚拟节点进行排序，并在底层节点上执行相同的操作。设 $m = \mid N_v \mid$ 和 $n = \mid N_s \mid$。则排序的虚拟节点为：$N^1_v, N^2_v, \ldots, N^m_v$，其中 $NR_1 \geqslant NR_2 \geqslant \cdots \geqslant NR_m$；已排序的底层节点为：$N^1_s, N^2_s, \ldots, N^m_s$，其中 $NR_1 \geqslant NR_2 \geqslant \cdots \geqslant NR_n$。

RW－MaxMatch 将 N^1_v 映射到 N^1_s，前提是 N^1_s 的 CPU 容量和链路容量能够满足节点 N^1_v 的要求。然后它以类似的方式将 N^1_v 映射到 N^1_s，其中 $i = 2,\ldots,m$。为了方便起见，本文将此映射称为 L2S2 映射（它代表“大到大和小到小”的映射），该阶段在算法 2 中执行。

算法 2：RW－MaxMatch 节点映射阶段

```
使用算法 1 计算 G_s 和 G_v 中所有节点的节点值，给定值为 ε
按照非递增顺序的 NodeRank 值对 G_s 中的节点进行排序
按照非递增顺序的 NodeRank 值对 G_v 中的节点进行排序
使用 L2S2 映射过程将虚拟节点映射到底层节点
if 满足节点资源约束 then
return NODE_MAPPING_SUCCESS
else
return NODE_MAPPING_FAILED
end if
```

在第二阶段中，RW－MaxMatch 将虚拟链路映射到底层链路（参见算法 3）。链路嵌入阶段与文献 [9] 类似。即通过对 k 值进行二进制搜索来搜索 k－最短路径，找到最恰当的最短路径，直到在底层网络上找到满足虚拟链路带宽要求的路径为止。如果底层网络支持路径分割，本文将使用多商品流算法[10]将虚拟链路嵌入到映射虚拟节点之间的底层链路。

算法 3：RW－MaxMatch 链路映射阶段

1：if 路径不可分割 then
2：使用 k－最短路径算法映射虚拟链路
3：else
4：使用多商品流算法映射虚拟链路
5：end if
6：if 链路资源约束满足 then
7：return LINK_MAPPING_SUCCESS
8：else
9：return LINK_MAPPING_FAILED
10：end if

3.1.2　时间复杂性分析

迭代方案（即算法 1）已被证明可以产生任何所需的精度 ε，其迭代次数与 $\max\{1,-\log\varepsilon\}$ 成比例[10]。k－最短路径[11] 链路映射算法和多商品流问题 [10] 都可以用多项式时间求解。因此，RW－MaxMatch 是 $|G_s|$，$|G_v|$ 和 $\max\{1,-\log\varepsilon\}$ 的多项式时间算法。

3.2　RW－BFS

3.2.1　描述和讨论

算法 4：RW－BFS

1：给定值 ε 使用算法 1 计算 G_s 和 G_v 中所有节点的 NodeRank 值
2：构造 G_v 的广度优先搜索树
3：在广度优先树的每个级别中根据其 NodeRank 值以非递增顺序对 G_v 中的所有节点进行排序
4：backtrack_count＝0
5：选择一个正整数
6：for 每个节点 N_v^i 都属于 G_v do
7：构建 N_v^i 的候选底层节点列表
8：if $Match(N_v^i) == 1$ then
9：$Match(N_v^{i+1})$
10：else
11：if backtrack_count＜＝Δ then
12：$Match(N_v^{i-1})$
13：backtrack_count＋＋
14：else
15：return BFS_FAILED
16：end if
17：end if
18：end for
19：return BFS_SUCCESS

RW－BFS 的主要框架由算法 4 给出，并且匹配函数的细节由算法 5 给出。与 RW－MaxMatch 不同，RW－BFS 是基于广度优先搜索（bfs）的一阶段回溯 VN 嵌入算法，它在同一阶段嵌入虚拟节点和链路。该算法的目的是在不考虑链路映射阶段的影响情况下，先对所有虚拟节点进行两级嵌入，从而导致不必要的占用底层网络的带宽资源。例如，当虚拟网络中的虚拟节点彼此非常接近（例如，只有一个跳跃）时，底层网络中两个映射虚拟节点的距离可能相当长。因此，两个节点之间的虚拟链路必须映射到长路径，从而导致带宽资源的浪费。

算法 5：匹配函数的细节

1：if N_v^i 是根 then
2：将 N_v^i 映射到具有最大 NodeRank 的底层节点上
3：return MATCH SUCCESS
4：end if
5：$k=1$
6：while $k < Max\_Hop$ do
7：for 从 N_v^i 的候选底层节点列表中的映射父节点满足 k 跳约束的每个 N_v^i do
8：if 对 (N_v^i, N_s^j) 满足所有容量限制 then
9：return MATCH_SUCCESS
10：end if
11：end for
12：k＋＋
13：end while
14：return MATCH FAILED

为了解决这个问题，RW－BFS 首先计算了 G_s 和 G_v 中节点的 NodeRank 值。然后构造了 G_v 的广度优先搜索树，其中根节点是具有最大 NodeRank 值的虚拟节点。在搜索树的每个级别上，节点按其 NodeRank 值的非递增顺序排序。对于 G_v 中的每个虚拟节点 N_v^i，本文构建了候选底层节点列表，该列表由可用 CPU 容量和可用带宽资源之和至少与虚拟节点相同的节点组成。列表中的底层节点也按其非增加顺序的 NodeRank 值进行排序。在最后一步中，将 G_v 中的每个虚拟节点嵌入到满足 CPU 资源需求的底层节点，并使用使用最短路径算法进行广度优先搜索，将直接连接到虚拟节点的虚拟链路映射到满足带宽和连接性约束的底层路径上。如果虚拟节点是 G_v 的根节点，则将其嵌入具有最大 NodeRank 值的底层节点。如果在候选底层节点列表中没有合适的 NIV 映射，则将回溯到先前的虚拟节点 N_v^{i-1}，将其重新映射到另一个底层节点，并继续映射 N_v^i。匹配函数（算法 5）中显示的常量值 Max _ Hop 是距离 N_v^i 的映射父节点的最大距离。跳跃约束的使用是为了减少 BFS 的搜索空间。它还可以避免将虚拟节点放置在离已经映射的节点太远的位置，以节省底层链路资源。

3.2.2　时间复杂性分析

由于回溯算法具有指数时间复杂性，为了避免指数爆炸，本文引入了一个上限 θ 来限制节点重新映射操作的次数（在算法 4 步骤 11 中）。

4　实验分析

在本节中，首先描述了性能评估环境，然后介绍了本文的主要评估结果。实验主要研究了两种算法与现有算法的性能比较以及拓扑感知节点排序的优点。

4.1　评估设置

本文实现了两个模拟器来评估算法的性能。RW－MaxMatch 模拟器是在文献 [12] 中设计的 VN 嵌入模拟器的改进版本，RW－BFS 模拟器是从零开始实现。两个模拟

器都可以在文献［13］中公开使用。

底层网络拓扑结构配置为具有 100 个节点，约 500 个链路，对应于中型 ISP。本文使用 GT－ITM 工具生成底层网络。底层节点、链路的 CPU 和带宽资源是实数，且均匀分布在 50 到 100 之间。假设 VNR 的到达遵循泊松过程，平均到达率为每 100 个时间单位 5 个 VN，并且每个 VNR 具有指数分布的寿命，平均为 500 个时间单位。在每个 VNR 中，虚拟节点的数量由 2 到 20 之间的均匀分布决定。平均 VN 连接固定为 50％，即 n 个节点 VN 平均具有 $n(n-1)/4$ 个链路，与文献［14］相似。虚拟节点和链路的 CPU 和带宽要求是实数，均匀分布在 0 到 50 之间。本文对每个模拟运行大约 50000 个时间单位，这相当于在一个模拟实例中平均得到 2500 个 VNR。

本文的模拟实验评估了表 1 中列出的 8 种算法。对这 8 种算法运行 10 个不同的实例，并记录十次运行的算术平均值作为最终结果。在本文中，支持位置约束并不是研究的重点，因此没有将本文的算法与文献［15］中提出的考虑位置约束的算法进行比较。本文还考虑将本文的算法与文献［16］中提出的基于子图同构的虚拟网络嵌入算法进行比较。然而，在文献［16］中的算法仅限于由有向图建模的底层网络和虚拟网络，因此不适用于公平比较，因此我们将其排除在结果之外。

表 1 算法描述比较

符号	算法描述
RW－MM－SP	使用 RW 模型计算 NodeRank 值。两阶段映射：将具有较大 NodeRank 值的虚拟节点映射到具有较大 NodeRank 值的底层节点，如果底层不支持路径分割，则使用最短路径嵌入虚拟链路。MM 表示 MaxMatch。
CB－MM－SP	只需使用公式(6)计算节点资源等级。CB－MM－SP 的其他步骤与 RW－MM－SP 相同。CB 表示 CPU 乘以带宽。
RW－BFS	使用 RW 模型计算 NodeRank 值。一阶段映射：基于广度优先搜索在同一阶段嵌入虚拟节点和链路。
CB－BFS	只需使用公式(6)计算节点资源等级。CB－BFS 的其他步骤与 RW－BFS 相同。
RW－MM－MCF	与 RW－MM－SP 类似，具有路径分割。
CB－MM－MCF	与 CB－MM－SP 类似，具有路径分割。
BL－SP	文献[17]中提出的基线算法。
BL－MCF	使用 MCF 嵌入虚拟链路的基线算法。

利用方程（10）给出的 p_v^I 和 p_v^F 的偏差因子来平衡局部和全局节点资源。实验表明，将 p_v^I 设置为 0.15，将 p_v^F 设置为 0.85，与 PageRank 中使用的值相同，可以获得最佳的结果。迭代方案中参数 ε 的值设置为 0.0001。由于底层网络的直径为 3，本文将匹配函数（算法 5）中的 Max _ Hop 值设置为 3。为了在不消耗大量执行时间的情况下获得更好的映射质量，上限 Δ（限制了 RW－BFS 中的重新映射步骤）设置为 $3n$，其中 n 是 VNR 中的虚拟节点数。

4.2 评估结果

本文的评估结果量化了所提出的两种算法（如图 3（a）、图 3（b）和图 3（c）所示）的效率，并揭示了使用 RWmodel 计算节点资源等级的优点（如图 4（a）、图 4（b）和图 4（c）所示）。为了进行评估，使用了几个性能指标，包括方程式（3）定义的长期平均收益、方程式（4）定义的 VNR 接受率、方程式（5）定义的长期 R/C 比率以及这些算法的运行时间。本文将模拟得出的主要观察结果总结如下。

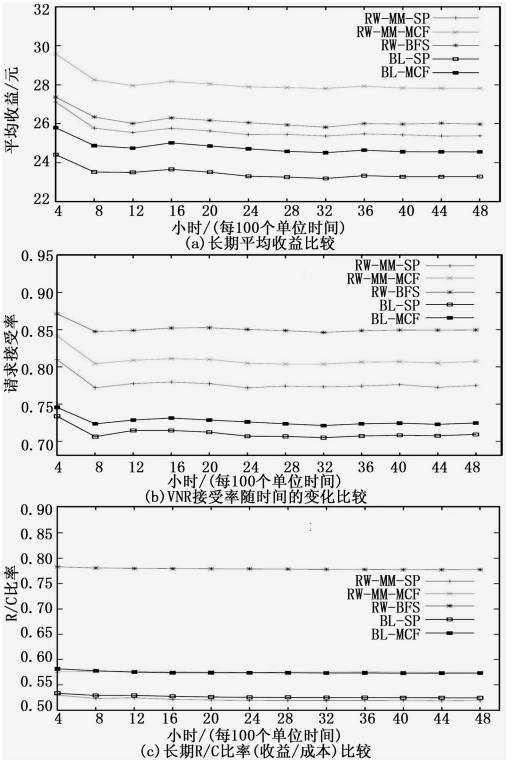

图 3 基于 RW 和现有算法的比较

4.2.1 接受率和平均收益的提高

图 3（a）和图 3（b）表明，无论底层网络的路径可分割性如何，所提出的 RW－MM－SP 和 RW－BFS 都可以产生比基线算法更高的收益和接受率。这两个图表明，拓扑感知的节点排序在 VN 嵌入过程中起着重要作用。虽然 RW－BFS 产生了最高的接受率，但该算法的长期平均收益低于 RW－MM－MCF，这是因为 RW－BFS 倾向于接受较小的 VNR 并拒绝许多较大的 VNR，从而使底层网络上的剩余资源具有较大的碎片化。

4.2.2 RW－BFS 产生最高的长期 R/C 比率

由于 RW－BFS 使用宽度优先搜索来映射虚拟节点，并且避免将虚拟链路映射到可能导致更多资源消耗的长底层路径上。图 3（a）表明，在不可分割路径的情况下，RW－BFS 产生的收益最高，这与其长期高 R/C 比率密切相关（如图 3（c）所示）。较低的 VN 嵌入成本为未来的 VNR 节省了更多的空间。

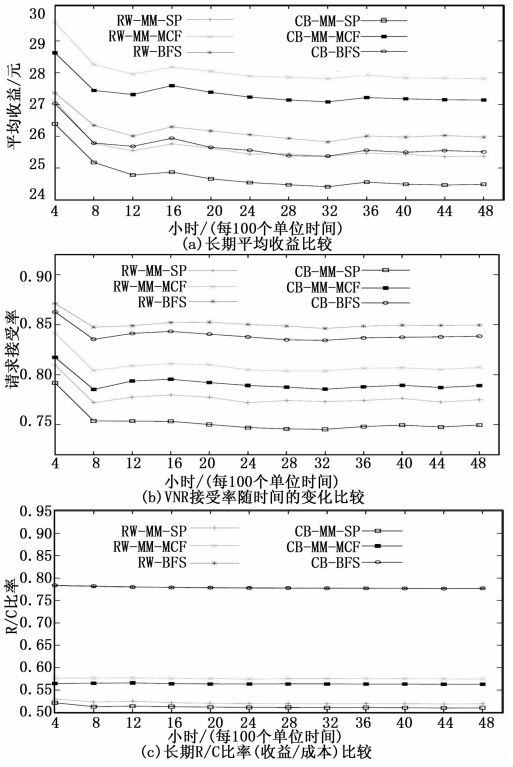

图 4　基于 RW 和基于 CB 的算法之间的比较

4.2.3　NodeRank 具有更好的节点资源度量

与简单地使用等式（6）作为节点资源的度量相比，计算 NodeRank 可得到更高的收益和接受率。对于两阶段 VN 嵌入算法，图 4（a）和图 4（b）表明，RWMM－SP 和 RW－MM－MCF 比 CB－MM－SP 和 CB－MM－MCF 提供更大的收益和更好的接受率。NodeRank 还有助于提高单级 VN 嵌入算法的收益和接受率。使用 NodeRank，资源排名不仅由节点本身决定，而且还受其邻居节点的影响。具有较高 NodeRank 值的节点往往会有一组 NodeRank 值越高的良好邻居节点，这可以增加满足 VN 请求约束的概率，从而有利于 VN 嵌入过程。然而，如图 4（c）所示，NodeRank 对提高 R/C 比率没有显著贡献。

4.2.4　基于 RW 的算法运行时间与其他算法相当

图 5 给出了同一 PC 上运行的这些 VN 嵌入算法的平均执行时间。基于 RW 的算法与基于 CB 的算法消耗时间几乎相同，即使基于 RM 的算法需要计算每个 VNR 的底层网络和虚拟网络中的节点等级。对于具有 100 个节点和 500 个链路的底层网络，计算节点等级的迭代过程在大约 7 次迭代中收敛到合理的容差，即算法迭代计算合理。

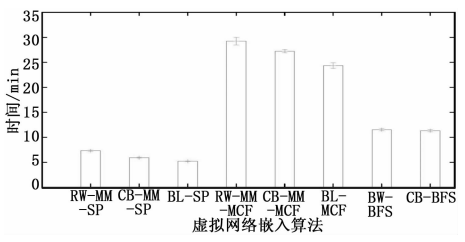

图 5　嵌入算法在 95%置信区间的执行时间比较

5　结论

本文基于马尔可夫随机游动计算了节点资源的拓扑感知等级。利用节点的资源等级作为资源度量，提出了 RW－MaxMatch 和 RWBFS 两种新的算法。仿真结果表明，本文的算法在长期平均收益和 VNR 接受率方面优于其他的方法。同时还证明了应用 RW 模型计算节点资源等级以获得更高的接受率和更高的收益的优点。

参考文献：

[1] 谢　盈. 云计算数据中心安全防护技术研究 [J]. 西南民族大学学报（自然科学版），2018，44（6）：616－620.

[2] 李　红，孙三山，周朝荣，等. 5G 切片化网络中基于吞吐量最大化的用户切换机制 [J]. 重庆邮电大学学报（自然科学版），2018，30（6）：739－745.

[3] 朱天雄，贾华宇，李灯熬，等. MOCVD 生长 AlGaInAs/InP 多量子阱激光器 [J]. 光通信研究，2018（4）：31－34＋60.

[4] 凌　杰，黄　刚. 基于 Docker 的 Hadoop 集群网络性能分析 [J]. 信息技术，2018（2）：15－18.

[5] Yang Q，Li W，de Souza J N，et al. Resilient virtual communication networks using multi－commodity flow based local optimal mapping [J]. Journal of Network and Computer Applications，2018，110：43－51.

[6] 刘新波，王布宏，杨智显，等. 一种基于拓扑势的虚拟网络映射算法 [J]. 电子与信息学报，2018，40（7）：1684－1690.

[7] 吕定辉. 稳态网络抗攻击频率准确预测方法仿真 [J]. 计算机仿真，2018，35（11）：396－400.

[8] Wang Q，Ren J，Wang Y，et al. CDA：A Clustering Degree Based Influential Spreader Identification Algorithm in Weighted Complex Network [J]. IEEE Access，2018，6：19550－19559.

[9] 于　雷. 一种基于协作博弈的虚拟网络嵌入策略 [J]. 信息与控制，2016，45（4）：449－455.

[10] 徐　薇，吴钰炜，陈彩华. 基于交替方向乘子法的大规模线性多商品流问题求解算法 [J]. 计算数学，2018，40（4）：436－449.

[11] 刘正平，钟　诚，张雄宝，等. 基于启发信息并行求解无环 K 最短路径 [J]. 小型微型计算机系统，2017，38（7）：1506－1511.

[12] Zhang P，Wu S，Wang M，et al. Topology based reliable virtual network embedding from a QoE perspective [J]. China Communications，2018，15（10）：38－50.

[13] Cao H，Hu H，Qu Z，et al. Heuristic Solutions of Virtual Network Embedding：A Survey [J]. China Communications，2018，15（3）：186－219.

[14] 李海舟. 基于云计算的智慧校园虚拟网络节点定位研究 [J]. 计算机与数字工程，2018，46（12）：2544－2551.

[15] 谢　枫，孟相如，赵志远，等. 基于邻接节点与拓扑结构感知的虚拟网络映射算法 [J]. 计算机工程，2018，44（9）：107－112.

[16] 邹　裕，覃中平. 混合网络的资源分配与虚拟机部署优化算法 [J]. 控制工程，2018，25（3）：509－515.

[17] Mallik A，Banerjee M，Woodroofe M. Baseline zone estimation in two dimensions with replicated measurements under a convexity constraint [J]. Statistica Sinica，2018，28：1481－1502.