# 基于VNF间性能干扰的服务请求调度策略

郭 胜,史久根,孙 立,谢熠君
(合肥工业大学 计算机与信息学院,安徽 合肥 230601)

**摘 要**:网络功能虚拟化(NFV)技术是当今研究的热点技术之一。目前的虚拟网络功能(VNF)放置方法大都忽视了同位间VNF即处在同一个物理机中的虚拟网络功能,对硬件资源的竞争所形成的干扰问题,会导致网络吞吐量下降。针对此问题,该文建立了以最大化网络吞吐量为目标的混合整数线性规划模型,设计了一个两步调整策略。第一步设计了组合放置算法(CAPA),将资源需求互补的虚拟网络功能进行组合放置;第二步根据不同虚拟网络功能对数据流量的处理特性,设计了流量感知算法(TAA)进行服务请求调度,进一步缓解了在大的数据流量场景下,底层硬件资源竞争更加激烈使得干扰增强的问题。实验结果表明,该文提出的两步调整策略,与忽视干扰因素的一般策略相比提高了网络吞吐量。

**关键词**:网络功能虚拟化;性能干扰;网络吞吐量;组合放置;服务请求调度
**中图分类号**:TP393 **文献标识码**:A **文章编号**:1673-629X(2021)01-0142-07
doi:10.3969/j.issn.1673-629X.2021.01.026

# Service Request Scheduling Strategy Based on VNF Performance Interference

GUO Sheng, SHI Jiu-gen, SUN Li, XIE Yi-jun
(School of Computer and Information, Hefei University of Technology, Hefei 230601, China)

**Abstract**: The network function virtualization (NFV) is one of the hot technologies of today's research. The current virtual network function (VNF) placement method mostly ignores the interference caused by the competition of hardware resources of the VNF between peers in the same physical machine, which leads to the decrease of network throughput. To solve this problem, we establish a mixed integer linear programming model aiming at maximizing network throughput and design a two-step adjustment strategy. In the first step, we design the combination and placement algorithm (CAPA) to combine the VNFs with complementary resource requirements combined placement. In the second step, according to the processing characteristics of different VNFs on data traffic, the traffic aware algorithm (TAA) is designed to perform service request scheduling, which further alleviates the problem of interference enhancement caused by the scenario of large data traffic that the competition of underlying hardware resources is more fierce. The experiment shows that the proposed strategy makes the network performance increase compared with general strategy for ignoring interference factors.

**Key words**: network function virtualization; performance interference; network throughput; combination and placement; service request scheduling

## 0 引 言

网络功能虚拟化(network function virtual, NFV)技术的特点在于实现硬件与软件的分离,通过网络功能的虚拟化,将虚拟网络功能(virtual network function, VNF)运行在通用硬件之上,不需要专用硬件,减少成本支出[1]。虚拟网络功能可以放置在虚拟机(virtual machine, VM)节点提供特定的网络功能[2]。通过VM迁移,VNF可以部署在网络中任意一个商用服务器中,同时这种新的模式也带来了一些挑战。首先,数据包需要经过一组VNF,即服务功能链[3](service function chain, SFC),网络运营商会考虑如何将VNF部署在候选服务器中以减少端到端延时。例如,Zhang等人[4]以最大化平均资源利用率和最小化请求平均响应延时为目标,提出了综合考虑VNF服务链放置及服务请求调度的问题。其次,要提高资源利用率,VNF布局根据流量变化动态调整VNF实例的数量。例如,Eramo等人[5]提出VNF迁移策略,使得网络运营商得知VNF最佳的迁移的时间和地点,以节

**收稿日期**:2020-02-08 **修回日期**:2020-06-11
**基金项目**:国家重大科学仪器设备开发专项(2013YQ030595)
**作者简介**:郭 胜(1993-),男,硕士研究生,研究方向为网络功能虚拟化;史久根,副教授,研究方向为嵌入式系统、计算机网络和无线传感器网络。

约资源。最后,现有的 VNF 实例可能需要合并到另一台服务器以释放和关闭一些服务器来节省资源。现有研究致力于研究应对上述挑战并提供高效的 VNF 放置方法。例如,Li 等人[6]考虑不同 VNF,在同一时隙对硬件计算量的需求不同,故将时变负载相关度最小的 VNF 放置在同一节点,达到计算资源互补,并采用多租户策略共享 VNF,从而减少物理机的使用量来减少资源消耗。

然而现有的 VNF 放置方法大都没有将 VNF 同位间的性能干扰问题考虑在内。虽然虚拟化技术在 VM 之间提供了一定程度的性能隔离[7],但 VNF 在一个共享的硬件基础设施上运行时仍存在明显的性能干扰[8]。当流量监视器与入侵检测系统同时运行时,流量监控器的吞吐量下降了 38.47%[9]。但是,为了最大化资源利用率和降低功耗,网络运营商倾向于将 VNF 实例放在同一物理上的服务器尽可能多,导致资源匮乏竞争和严重的性能干扰。同时,在数据流增大的时候干扰也随之增强。由于数据流的增大,VNF 处理任务增加,所需的资源也随之增加,导致基本硬件资源的竞争更加激烈。

针对上述问题,该文根据不同类型的虚拟网络功能的工作特性,通过在虚拟机中运行工作特性相似的应用以及文献[9]的分析,得出不同的虚拟网络功能对 CPU、Memory、Cache 等资源的依赖程度,设计一个 CAPA 算法解出 VNF 之间的最佳组合,使得整体干扰度最小,并进行放置。其次,根据不同 VNF 对数据流量的处理特性,使用 TAA 算法进行服务请求的调度,从而进一步减小虚拟网络功能的干扰程度。仿真结果表明,该方案缓解了 VNF 间的干扰,提升了网络吞吐量。

## 1　问题描述

该模型需要解决的问题是,根据不同虚拟网络功能间干扰程度不同,对每个 VNF 进行组合,要设法使其双方的干扰度都尽可能小,并对服务链请求进行合理的调度,使得整体网络中的虚拟网络功能受到的干扰最小。

### 1.1　同位间虚拟网络功能干扰

同位间虚拟网络功能干扰是指,同一个物理中 VNF 会产生一定的性能干扰,使得彼此间性能下降。由于 VNF 对各种硬件资源的需求不尽相同,若多个同种资源需求密集型 VNF 放置在同一服务节点,会引起服务节点底层的某一资源过度占用,从而使得服务节点整体处理性能下降。在图 1 中(a)表示当 VNF1 与 VNF2 独立运行在服务器时的处理能力,(b)表示新增的 VNF3 分别放置在先前的服务器中,使得三个 VNF 处理能力都有不同程度的下降。由于各 VNF,对不同的资源需求的比重不尽相同,可以通过资源互补的形式进行合理组合。如图 2 所示,左边部分表示三种 VNF 对不同资源的需求(每个颜色的柱型代表一种资源),右上侧 VNF1+VNF2 表示 VNF1 与 VNF2 进行组合,右下侧 VNF1+VNF3 表示 VNF1 与 VNF3 匹配,故合理的组合可以达到较优的效果。

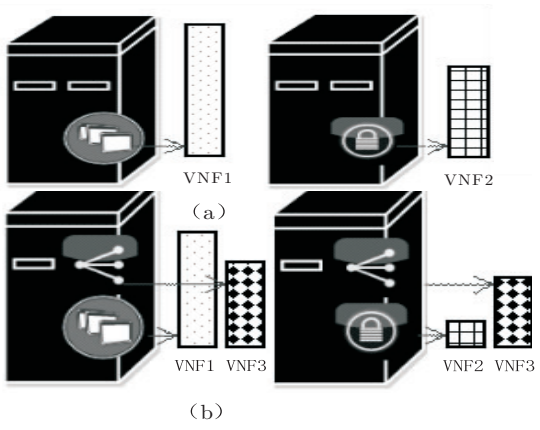


图 1　VNF 间性能干扰示例

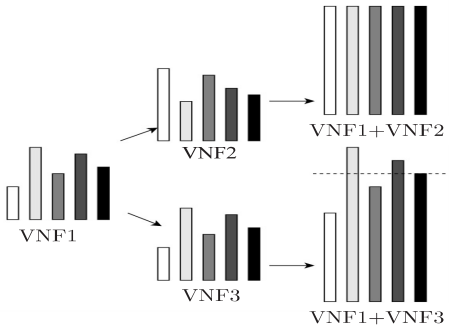


图 2　VNF 匹配方案示例

### 1.2　修改数据流的大小

VNF 会对数据包进行分析,根据特殊的业务需求对数据包添加包头,或者对数据包进行优化压缩,使得数据包的字节数及数据流的大小都被改变[10]。目前 VNF 的总数很多,与路由器的数量相当[11],其中很大一部分的 VNF 有改变数据流大小的性质。如图 3 所示,网络功能 F1 可将数据流修改为原来的两倍,F2 则将数据流修改为原来的一半,所以数据流先经过 F2 时,可减缓 F1 的处理负担,而 VNF 的流经顺序是部分可调的[12-13]。

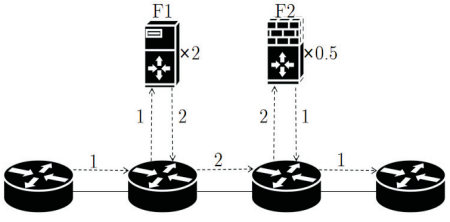


图 3　VNF 修改数据流示例

数据流的减小和增大直接影响 VNF 工作时对资

源的竞争程度。因此合理的调度方案可以进一步减小干扰。

## 2 系统模型

图 $G(E,V)$ 表示交换机网络模型,节点 $v \in V$ 都有一个物理机,设定网络中的物理机是统一的,每个物理机有资源 $R(u)(r_1,r_2,\cdots,r_n)$ , $r_i$ 表示各项分资源。$(u,v) \in E$ 表示连接节点 $u$ 与 $v$ 的链路传播时延为 $\mathrm{De}(u,v)$ 。网络为具有全局拓扑视图的 SDN 网络[14],设定网络中的交换机,连接高容量光链路,因此每条链路的带宽容量不做约束[15]。

定义(干扰度):假设两个向量 $\boldsymbol{a}(x_1,x_2,\cdots,x_n)$ 和 $\boldsymbol{b}(y_1,y_2,\cdots,y_n)$ 之间的夹角为 $\theta$ , $\boldsymbol{a}$、$\boldsymbol{b}$ 向量的长度分别是 $\|\boldsymbol{a}\|$ 和 $\|\boldsymbol{b}\|$ , $\theta$ 所对的边的边长为 $\|\boldsymbol{a}-\boldsymbol{b}\|$ 。根据余弦定理:

$$\|\boldsymbol{a}-\boldsymbol{b}\|^2 = \|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2 - 2\|\boldsymbol{a}\|\|\boldsymbol{b}\|\cos\theta \frac{x-\mu}{\sigma} \tag{1}$$

$$\boldsymbol{a}\cdot\boldsymbol{b} = \|\boldsymbol{a}\| \cdot \|\boldsymbol{b}\|\cos\theta \tag{2}$$

根据上述公式可以得到:

$$\cos XY = \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{i=1}^{n} Y_i^2}} \tag{3}$$

$\cos\theta$ 就是余弦相似度,两个向量之间的夹角越小,其夹角余弦越大(越相似)。因此可用余弦相似度来度量两个 VNF 对资源的需求相似度。向量 $a(x_1,x_2,\cdots,x_n)$ 对应 VNF 对各种资源的占有率 $X(X_1,X_2,\cdots,X_n)$ ,由于不同 VNF 对每种资源的依赖程度不一,故定义 $\lambda(\lambda_1,\lambda_2,\cdots,\lambda_n)$ , $\lambda_i$ 表示 VNF 对某一种资源依赖程度的权值。VNF 间的干扰度定义如下:

$$\cos XY = \frac{\sum_{i=1}^{n} \lambda X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{i=1}^{n} Y_i^2}} \tag{4}$$

其物理概念为同一服务节点中的一个 VNF 的资源的占有率与另一个 VNF 对资源的实际需求率的一个契合程度。网络吞吐量与网络中 VNF 间的干扰程度呈反比例关系,定义反比例系数 $\beta$ ,吞吐量 $P(m)$ 与干扰度存在下述关系:

$$P(m) = \beta\cos XY \tag{5}$$

决策变量的定义:用 $F$ 表示所有流的集合,用 $Z$ 表示网络中 VNF 个数。任意一条流 $f \in F$ 由三元组 $(\mathrm{src}_f,\mathrm{dst}_f,M_f)$ 表示其属性,其中 $\mathrm{src}_f$ 表示流 $f$ 的源节点, $\mathrm{dst}_f$ 表示流 $f$ 的目的主机, $M_f$ 表示流 $f$ 需要经过的 VNF 集合。$\mathrm{ratio}(m)$ 表示 $\mathrm{VNF}_m$ 的流改变因子。用符号 → 来定义 VNF 的依赖关系,若 $m \to n$ 则 $n$ 依赖于 $m$ ,即业务流需先经过 $m$ 再经过 $n$ 。

首先定义 VNF 处理顺序的传递性,例如 $m \to n$ , $n \to k$ ,那么 $m \to k$ 。用二进制变量 $d(m,n)$ 来描述这种关系:

$$d(m,n) = \begin{cases} 1, m \to n \\ 0, \text{other} \end{cases} \tag{6}$$

$x_f^{u,v}$ 表示流 $f$ 经过的路径:

$$x_f^{u,v} = \begin{cases} 1, f \text{ through } (u,v) \\ 0, \text{ other} \end{cases} \tag{7}$$

用二进制变量 $y_f^{u,v}$ 表示流服务请求中的一个 VNF,是否实例化到 $m$ 中。

$$y_m^u = \begin{cases} 1, m \text{ place in } u, m \in M_f \\ 0, \text{other} \end{cases} \tag{8}$$

流 $f$ 在网络中经过的是一条多跳路径,用 $i$ 表示跳数,若共有 $N$ 跳,那么 $1 \leqslant i \leqslant N$ 。用 $f(u,i)$ 表示第 $i$ 跳所在的位置:

$$f(u,i) = \begin{cases} 1, \text{the } i \text{ jump through } n \\ 0, \text{other} \end{cases} \tag{9}$$

同一节点允许部署一条流的多个 VNF 实体,对于流 $f$ 来说有以下情况:

$$\exists u, i \neq j: f(u,i) = 1 \& f(u,j) = 1 \tag{10}$$

约束条件: $t_{\mathrm{in}}(f,u)$ 和 $t_{\mathrm{out}}(f,u)$ 分别表示流 $f$ 进入节点 $u$ 和流出节点 $u$ 的流量数据率。对于流 $f$ 来说,如果它在节点 $u$ 经过了 VNF 的处理,那么 $t_{\mathrm{in}}(f,u)$ 和 $t_{\mathrm{out}}(f,u)$ 满足如下关系:

$$t_{\mathrm{out}}(f,u) = \frac{2-y_m^u}{2} t_{\mathrm{in}}(f,u) \prod_{m \in M_f} \mathrm{ratio}(m) \times (y_m^u + 1), \forall u \in V \tag{11}$$

对于网络中的物理节点来说,所有服务请求的 VNF 集合实例化到该点上所需的资源总和必须不超过该物理节点 $u$ 所能提供的节点容量, $R(u)$ 与式(4)中的 $X_i$ 以( $Y_i$ )对应,约束如下:

$$\sum_{f \in F} \sum_{m \in M_f} y_m^u \leqslant R(u), \forall u \in V \tag{12}$$

构成一条路径的所有中间链路 $\mathrm{De}(u,v)$ 之和要小于所设定的时延容限 $M\mathrm{De}$ 。约束如下:

$$\sum_{u \in V} x_f^{u,u+1} \times \mathrm{De}(u,u+1) < M\mathrm{De} \tag{13}$$

如果 $\mathrm{VNF}_n$ 依赖于 $\mathrm{VNF}_m$ ,那么流 $f$ 将先经过 $\mathrm{VNF}_m$ ,即有以下约束:

$$\left[\sum_{i \in [1,j]} y_v^u \times f(u,j) \times i - \sum_{i \in [1,k]} y_m^v \times f(v,k) \times i\right] \times d(m,n) > 0 \tag{14}$$

用下式表示链路中的流守恒:

$$t_{\mathrm{out}}(f,u) = t_{\mathrm{in}}(f,u+1) \tag{15}$$

最后文中目标是最大化网络吞吐量, $T$ 表示参数因子, $T$ 与 $t_{\mathrm{in}}(f,u)$ 呈反比。

$$\max \sum_{f\in F}\sum_{i\in[1,N]} t_{\text{in}}(f,i)\times T+\sum_{f\in F}\sum_{m\in M_f} P(m) \qquad (16)$$

## 3　算法分析

该文所要完成的目标,在式(11)~式(15)的约束下求解是一个 NP-难问题。因为在所给定的节点容量约束下,不考虑资源种类和各 VNF 对不同资源的需求权重时,问题将转换成装箱问题,装箱问题是常见的 NP-难问题[16]。完成该目标需要两个算法实现,首先通过自主设计的基于贪心的 CAPA 算法得到 VNF 间的最佳组合,并进行放置。然后通过 TAA 算法进行流服务请求调度,使得数据流经过各个节点的累加值最小,进一步减小 VNF 间性能干扰。所以通过该文提出的两步调整策略可完成目标。

### 3.1　算法描述

算法 1:MIMP 算法。

输入:虚拟网络功能集合 $L$,网络拓扑 $G(E,V)$,拓扑中心节点 $c$;

输出:组合放置集合 $L'$。

1. 构建优先矩阵 $\boldsymbol{\psi}$
2. 镜像复制
3. 延时接受算法匹配
4. 重复项删除
5. 干扰度求和排序
6. if 原 $\sum P(m)$ >重组 $\sum P(m)$ do
7. 拆分重组 Return to step 5
8. else 构建组合集 $\xi$
9. Dijkstra 算法选点
10. 组合集映射到节点集
11. 得到组合放置集合 $L'$
12. 结束

算法 1:

(1)输入数据处理:首先输入所有 VNF 资源需求集合 $L$,网络拓扑 $G(E,V)$ 和给定的拓扑中心节点 $c$,通过式(4)进行干扰度计算,生成 VNF 间的干扰度矩阵 $\boldsymbol{W}$。然后将矩阵的每一行元素按照干扰程度大小进行升序排列得到优先列表矩阵 $\boldsymbol{\psi}$,越靠前则最优。复制列表矩阵,一份做原优先列表矩阵,另一份为镜像 $\boldsymbol{\psi}'$。镜像处理使得原先无法进行匹配的单列元素可以进行匹配处理。

(2)最佳匹配对象的选取:首先使用延迟接受算法进行初步的匹配,此算法的过程是若集合中存在 $\varphi_i$ 没有匹配,则选择其为主动元素,向列表矩阵中最靠前的 $\varphi_i'$ 发起配对请求,被请求元素 $\varphi_i'$ 会将其与已有的匹配元素进行比较,并在资源的约束下,保留 $\varphi_i$ 请求或者拒绝其请求,若 $\varphi_i$ 被拒绝则将 $\varphi_i'$ 从最优列表矩阵中删除,并向列表中次优的元素继续发起请求,若被接受则形成暂时匹配。接下来对已有匹配集合 $Q$ 进行调整,首先删除由镜像产生的重复匹配组合项得到集合 $Q'$,将现有的组合,根据双方的干扰程度之和,从大到小进行排序。选择双方干扰度最大的组合为目标组合,将其拆分,加入到尾部组合,判断在满足资源约束下,是否存在已有匹配对与重组后元素间干扰的累加和小于未拆分前干扰度之和,不存在则保留原组合,若存在则拆分重组,重组后返回继续判定,最终得到集合 $\xi$。

(3)节点映射:由于模型的设定,物理机类型统一,故将 VNF 组合向网络拓扑中心靠拢,方便后续的服务请求调度,故以延时 De 为权值,使用 Dijkstra 算法计算给定节点 $c$ 到网络拓扑中其他节点延时的大小,得到权值集合 $\tau$。将得到的组合集合 $\xi$ 中的元素映射到权值较小的集合 $\tau$ 中的节点元素,最终输出最佳组合放置集合 $L'$。

算法 2:TAA 算法。

输入:服务请求的虚拟网络功能集合 $\Omega$,网络拓扑 $G(E,V)$,组合放置集合 $L'$

输出:服务请求调度集合 $S$

1. while $\Omega \neq \varnothing$ do
2. 读取所需 VNF 信息构建初步顺序
3. if 无依赖关系的 VNF 个数 ≠ 0 do
4. if　NVF ratio($m$) < 1　do
5. 　序列头部升序排列　end if
6. if　NVF $ratio(m)$ > 1 do
7. 　　序列尾部升序排列 end if
8. end if　return VNF order
9. 得到必经点及顺序约束
10. Dijkstra 算法
11. if $\sum De(u,v)$ < $M$De　do 加入服务请求调度集合 $S$
12. else 丢弃此请求
13. 输出最终服务请求调度集合 $S$
14. 结束

算法 2:

(1)顺序建立:请求集合 $\Omega$ 不为空,对每条服务请求中的 VNF 的流修改因子 ratio($m$) 进行读取,判断服务请求中的 VNF 是否存在依赖关系的 VNF,确立具有依赖关系的 VNF 的流经顺序。然后判断服务请求中的 VNF 是否存在没有依赖关系的 VNF,若有,且 VNF 的流修改因子小于 1,则将其在调度列表中头部进行升序排列,若 VNF 的流修改因子大于 1,则将其在调度列表中尾部进行升序排列,从而确立了此条服务请求中 VNF 的所有流经顺序。

(2)路径选择:对于每条服务请求读取其源节点以及目的节点,并将其值分别赋予 $k_1$, $k_2$ 根据服务请求所需的 VNF 所在的物理节点,建立必经点的路径约束,再根据上述处理提供的 VNF 流经顺序,建立顺序约束。最后通过双约束的 Dijkstra 算法以链路延时为权值,进行路径选择,若总传播时延超过给定的时延阈值则丢弃此条服务请求,若满足约束则加入到最优路径集,最终为请求集合中的每条服务规划出一条路径,得到服务请求调度集合 $S$。

### 3.2 算法复杂度分析

用 $\omega$ 表示 VNF 的个数,在算法 1 中构建干扰度矩阵所需循环次数为 $O(\omega^2)$,构建优先列表矩阵所需循环次数为 $O(\omega^2)$,匹配过程中最大循环次数为 $O(\omega^2)$,重组调整过程中的最大循环次数 $O(\omega^2 - 2\omega)$。Dijkstra 算法时间复杂度为 $O(v^2)$,$v$ 表示物理节点个数,故算法 1 的整体复杂度为 $O(v^2 + \omega^2)$。

在算法 2 中 VNF 的顺序处理的复杂度取决于服务链的长度,以及无顺序约束的 VNF 个数,定义平均服务链长度为 $\varepsilon$ 最终顺序建立的平均复杂度为 $O((\varepsilon/2)^2)$。Dijkstra 算法的时间复杂度为 $O(n^2)$,其中 $n$ 表示节点跳数,由于约束条件的存在,算法中使用的 Dijkstra 算法的复杂度为 $O(\varepsilon^2)$,故其整体复杂度为 $O(\varepsilon^2 + n^2)$。

### 3.3 算法近似比分析

CAPA 算法所要解决的是一个 NP-难问题,故对此进行近似比分析。将物理机假设为单位容量为 $C$ 的箱子,VNF 假设为大小为 $s$ 的物品,设 $s_i < 1, i = 1,2,\cdots,n$。设 OPT($I$) 为最优解所用箱子数目的一个上界,CAPA($I$) 为 CAPA 算法的解。由算法描述中 CAPA 算法的拆分重组规则可知,所有箱子中至多有一个非空箱子,所装的物体体积小于 1/2。

现在设 $\varepsilon_i$ 为第 $i$ 个箱子中的空余容量,$\delta_i$ 为物品占用第 $i$ 个箱子的容量 $\varepsilon_i + \delta_i = C$。那么有:

$\sum_{i=1}^{k} \delta_i = \sum_{i=1}^{n} s_i$,其中 $k$ 表示占用的箱子个数。

对于第 $k$ 个箱子有两种情况:

(1) $\varepsilon_k < \delta_k$;

(2) $\varepsilon_k > \delta_k$。

对于情况 2,有:$\varepsilon_{k-1} < \delta_k$,$\varepsilon_k < \delta_{k-1}$,所以:

$\varepsilon_{k-1} + \varepsilon_k < \delta_{k-1} + \delta_k$。

故对于两种情况都有 $\sum_{i=1}^{k} \varepsilon_i < \sum_{i=1}^{k} \delta_i = \sum_{i=1}^{n} s_i$,故:

$$\mathrm{CAPA}(I) = \frac{\sum_{i=1}^{k} \varepsilon_i + \sum_{i=1}^{k} \delta_i}{C} < \frac{2\sum_{i=1}^{n} s_i}{C}$$

最优解的一个上界为所有箱子恰好装入全部物体,即:

$$\mathrm{OPT}(I) = \frac{\sum_{i=1}^{n} s_i}{C}$$

故 CAPA 算法近似比为:

$$\rho\mathrm{CAPA}(I) = \frac{\mathrm{CAPA}(I)}{\mathrm{OPT}(I)} < 2。$$

## 4 仿真实验结果及分析

### 4.1 仿真实验环境参数

硬件环境为 Inter Core i5-7200U CPU 2.71 GHz RAM 4 GB 的 Windos10 家庭版操作系统;该文选择的仿真平台是 matlab2017a。实验采用的网络拓扑为真实的网络拓扑 Internet2(Internet OS3E)[17]。实验设定 VNF 个数 $Z = 32$,实验考虑多种不同资源 $R(u)(r_1, r_2, \cdots, r_n)$,每个 VNF 对多种不同资源都有不同的权 $\lambda(\lambda_1, \lambda_2, \cdots, \lambda_n)$,每个 VNF 都有一个固有的流修改因子 ratio($m$)。

### 4.2 性能分析

#### 4.2.1 CAPA 算法的性能分析

图 4 表示不同算法情况下,网络中分别部署 4,8,16,32 个 VNF 时,网络的整体标准化吞吐量变化情况(利用式(5)计算,并做标准化处理,标准化处理最终为两个结果的比值)。从实验结果图中发现,在 VNF 数量较少的情况下,CAPA 算法与 First-fit 算法[18]差异不是特别明显。这是因为当 VNF 数量较少时,组合的方式很少,当 VNF 数量增加使差异不断增加是因为 First-fit 算法在寻找组合对象时只考虑选择对象对自己的干扰程度,没有考虑自身对选择对象的干扰程度,使得单方面最优,达不到双方最优。图中 random 表示不考虑干扰因素以随机的方式组合,其代表的柱型起伏不定的原因是由于 VNF 数量少时偶然性因素较大。

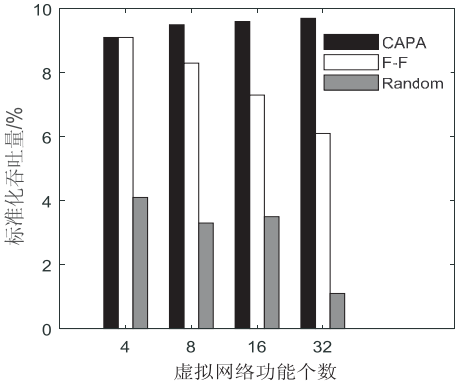


图 4 不同 VNF 个数下的网络标准化吞吐量情况

#### 4.2.2 TAA 算法的性能分析

图 5 表示在 Internet OS3E 网络中,设定服务链总长度为 10,当平均无依赖关系的 VNF 占服务请求链的百分比增加时,两种算法情况的对比。纵坐标的标准化处网络吞吐量,可由式(16)经过标准化处理得

到。从图中可以看出,当服务链请求中的无依赖关系的 VNF 占总体服务链请求所需的 VNF 比重越小,TAA 算法性能会越接近于 Dijkstra 算法[19],这是因为当具有先后依赖关系的 VNF 占服务链请求所需的 VNF 比重越大,TAA 算法所能调整的范围就会越小,在服务链请求中的 VNF 全部相互依赖的情况下,TAA 接近普通算法。在实际情况中服务链请求中的 VNF 是有相当一部分可以调整的。

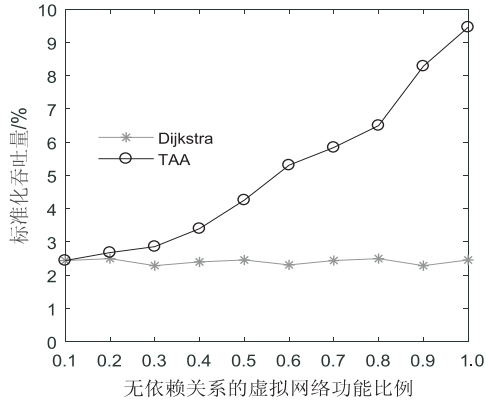


图 5　无依赖关系的 VNF 比例与性能间关系

4.2.3　标准化传播时延分析

图 6 表示在不同的无依赖关系的 VNF 比例下,两种 VNF 放置方法与网络标准化传播时延的关系,其中服务链总长度设定为 10,用 TAA 算法进行调度。图中可以看到随着流经顺序可调的 VNF 比例的增加,两种放置方法对应的网络中标准化传播延时都在减小,这是因为服务请求调度的顺序约束减少了。而中心化放置方式对应的网络标准化传播时延,始终较小的原因是由于随机放置,会使得某些 VNF 被放置在网络边缘,流服务请求必须达到网络边缘获取相应的处理,仿真结果中随机放置情况下 TAA 算法所得平均端到端延时为 0.55 ms。而中心化放置使得 VNF 集中于网络拓扑中心附近,服务请求在网络拓扑中心附近便可获取所需的处理,实验分析表明 CAPA 算法中将 VNF 放置在网络拓扑中心附近,能够在一定程度上减小服务请求调度过程中的网络传播时延。

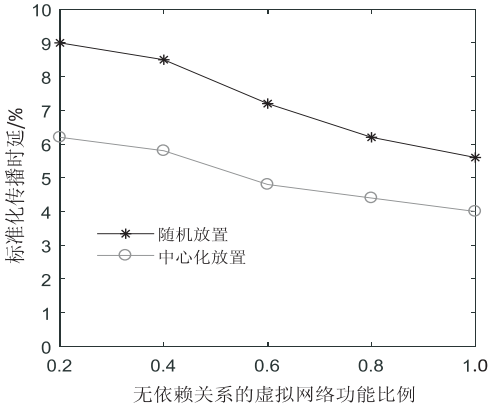


图 6　标准化传输时延无依赖关系 VNF 比例间关系

4.2.4　算法整体性能分析

图 7 中,考虑干扰方案表示在 Internet2(Internet OS3E)网络中,将 VNF 通过算法 1 进行组合放置在网络拓扑中,设定服务链总长度为 10,服务链请求中具有依赖关系 VNF 的比例为 60%,并使用 TAA 算法进行服务请求调度。未考虑干扰方案表示在 Internet2(Internet OS3E)网络中,通过 First-fit 算法进行组合放置,并使用 Dijkstra 算法进行请求调度。从实验生成图中可以看出,当数据流不断增加时,两种方案下的标准化网络吞吐量都有所下降,但是考虑干扰方案下的网络吞吐量始终比未考虑干扰方案下的标准化网络吞吐量优。

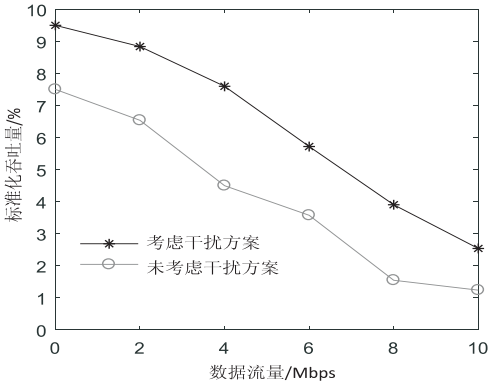


图 7　标准化处理能力随数据流大小变化示意图

## 5　结束语

提出了一种考虑 VNF 间性能干扰的组合放置问题,并根据 VNF 的修改数据流大小的特性设计了服务链请求调度算法,从而减小了 VNF 间性能干扰,提高了网络吞吐量。理论分析以及仿真结果表明,该策略较忽视干扰因素的一般策略可将网络吞吐量提升 1.3 倍。今后将研究在干扰情况下的动态部署 VNF 以及如何权衡成本开销与网络性能间的关系问题,并将在真实环境中进行实验以提高结果的准确性。

**参考文献:**

[1] 谷允捷,胡宇翔,丁悦航,等. 基于流量演化感知的服务功能链在线弹性编排策略[J]. 电子学报,2019,47(10):2192-2201.

[2] 周伟林,杨　芫,徐明伟. 网络功能虚拟化技术研究综述[J]. 计算机研究与发展,2018,55(4):675-688.

[3] 魏　亮,黄　韬,张　娇,等. 基于强化学习的服务链映射算法[J]. 通信学报,2018,39(1):90-100.

[4] ZHANG Q, XIAO Y, LIU F, et al. Joint optimization of chain placement and request scheduling for network function virtualization[C]//IEEE international conference on distributed computing systems. Atlanta, GA, USA: IEEE, 2017: 731-741.

[5] ERAMO V, MIUCCI E, AMMAR M, et al. An approach for

service function chain routing and virtual function network instance migration in network function virtualization architectures[J]. IEEE/ACM Transactions on Networking,2017,25(4):2008-2025.

[6] LI D,HONG P, XUE K ,et al. Virtual network function placement considering resource optimization and SFC requests in cloud datacenter[J]. IEEE Transactions on Parallel and Distributed Systems,2018,29(7):1664-1677.

[7] LIN J W, CHEN C H. Interference-aware virtual machine placement in cloud computing system [C]//International conference on computer & information science. Kuala Lumpeu,Malaysia:[s. n. ],2012:598-603.

[8] 古英汉,伊 鹏,韩伟涛,等. 基于干扰估计的虚拟功能服务链创建及部署方法[J]. 计算机应用研究,2016,33(10):3123-3127.

[9] ZENG C,LIU F,CHEN S,et al. Demystifying the performance interference of co-located virtual network functions [C]//2018 IEEE conference on computer communications (INFOCOM). Honolulu,HI,USA:IEEE,2018:765-773.

[10] WANG L,LU Z,WEN X,et al. Joint optimization of service function chaining and resource allocation in network function virtualization[J]. IEEE Access,2016,4:8084-8094.

[11] 史久根,张 径,徐 皓,等. 一种面向运营成本优化的虚拟网络功能部署和路由分配策略[J]. 电子与信息学报,2019,41(4):973-979.

[12] HYODO N,SATO T,SHINKUMA R,et al. Virtual network function placement model for service chaining to relax visit order and routing constraints[C]//IEEE international conference on cloud networking. Tokyo,Japan:IEEE,2018:1-3.

[13] ALLYBOKUS Z, PERROT N, LEGUAY J, et al. Virtual function placement for service chaining with partial orders and anti-affinity rules[J]. Networks,2018,71(2):97-106.

[14] VIZARRETA P,TRIVEDI K,HELVIK B,et al. Assessing the maturity of SDN controllers with software reliability growth models[J]. IEEE Transactions on Network and Service Management,2018,15(3):1090-1104.

[15] HUANG M,LIANG W,MA Y,et al. Throughput maximization of delay-sensitive request admissions via virtualized network function placements and migrations[C]//IEEE international conference on communications. Kansas City,MO,USA:IEEE,2018:1-7.

[16] CHEN Z,ZHANG S,WANG C,et al. A novel algorithm for NFV chain placement in edge computing environments [C]//IEEE global communications conference. Abu Dhabi,United Arab Emirates:IEEE,2018:1-6.

[17] 史久根,郑 伟,贾坤荥,等. 软件定义网络中基于负载均衡的多控制器部署算法[J]. 电子与信息学报,2018,40(2):455-461.

[18] NTENE N,VAN VUUREN J H. A survey and comparison of heuristics for the 2D oriented on-line strip packing problem [J]. ORiON,2008,24(2):157-183.

[19] 王树西,李安渝. Dijkstra 算法中的多邻接点与多条最短路径问题[J]. 计算机科学,2014,41(6):217-224.

(上接第 141 页)

[17] ZHANG Y. The unified image encryption algorithm based on chaos and cubic S-Box[J]. Information Sciences,2018,540(6):361-377.

[18] SILVA V M,FLORES R,RENTERÍA C,et al. Substitution box generation using chaos: An image encryption application [J]. Applied Mathematics and Computation,2018,332:123-135.

[19] LIU H,ZHAO B,HUANG L. A novel quantum image encryption algorithm based on crossover operation and mutation operation[J]. Multimedia Tools and Applications,2019,78:20465-20483.

[20] NOROUZI B,SEYEDZADEH S M,MIRZAKUCHAKI S. A novel image encryption based on row-column, masking and main diffusion processes with hyper chaos[J]. Multimedia Tools and Applications,2015,74(3):781-811.

[21] LAMINE S M,IBTISSEM B. A pseudo-random numbers generator based on a novel 3D chaotic map with an application to color image encryption [J]. Nonlinear Dynamics,2018,94(1):723-744.

[22] YU C,LI J,LI X. Four-image encryption scheme based on quaternion Fresnel transform, chaos and computer generated hologram[J]. Multimedia Tools and Applications,2018,77(4):4585-4608.

[23] PARVAZ R,ZAREBNIA M. A combination chaotic system and application in color image encryption[J]. Optics and Laser Technology,2018,101:30-41.

[24] HUANG L,CAI S,XIAO M,et al. A simple chaotic map-based image encryption system using both plaintext related permutation and diffusion[J]. Entropy,2018,20(7):535-554.

[25] PATRO K,ACHARYA B. Secure multi-level permutation operation based multiple colour image encryption[J]. Journal of Information Security and Applications,2018,40(2):111-133.

[26] HE Y,ZHANG Y Q,WANG X Y. A new image encryption algorithm based on two-dimensional spatiotemporal chaotic system[J]. Neural Computing and Applications,2018,8:1-14.

[27] WU X J,WANG K S,WANG X Y,et al. Lossless chaotic color image cryptosystem based on DNA encryption and entropy[J]. Nonlinear Dynamics,2017,90(2):855-875.