研究与开发

基于预测的 5G 网络切片算法

张佳庚[1,2]，祝敏[2]，杜丰[1]，王齐[1]，刘俊[1]，锁志海[1]，王力[2]
（1. 西安交通大学网络信息中心，陕西 西安 710054;
2. 西安交通大学电子与信息学部，陕西 西安 710054）

摘　要：5G 网络实时应用场景对网络切片的建立提出了严格的要求，需要使用预测算法提前隔离资源，降低网络切片的建立时间。提出了基于预测的 5G 网络切片算法，以四阶矩为代价函数，在算法复杂度不高的前提下提供必要的预测精度，根据预测结果在 5G 网络中提前隔离虚拟节点资源和虚拟链路资源，当网络切片请求到达时，直接拉起容器，完成网络切片的动态创建。仿真结果表明，所提算法的预测精度能够达到 90%，在复用原始网络切片资源的条件下，新请求网络切片的创建时间减少 50%。

关键词：预测；四阶矩；微服务；实例化
中图分类号：TP3
文献标识码：A
doi: 10.11959/j.issn.1000−0801.2021225

5G network slicing algorithm based on prediction

ZHANG Jiageng[1,2], ZHU Min[2], DU Feng[1], WANG Qi[1], LIU Jun[1], SUO Zhihai[1], WANG Li[2]
1. Network & Information Center, Xi'an Jiaotong University, Xi'an 710054, China
2. Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710054, China

Abstract: 5G on line scenario needs short build time of network slice, it can use prediction algorithm to isolate network resource in advance to reduce the construction time of network slice. A 5G network slicing algorithm based on prediction was proposed, which used fourth square as cost function to predict the network resource requests with accepted accurate and lower complexity of algorithm. And then based on the prediction result, virtual nodes and links resource were allocated, and container was pulled up when network slice request arrives to network, the dynamic creation of network slices was completed. The simulation results show that the proposed algorithm can get 90% accurate of prediction, and reduce 50% construction time of network slice, and improve the network utilization ration，when new network slice reusing the original network slice.

Key words: prediction, fourth square, micro service, instantiation

1　引言

3GPP 在 5G 网络组网协议中明确了 5G 网络架构，其最大的特点是采用基于服务的架构（service based architecture，SBA）架构，对应不同应用场景由网络切片选择功能（network slice

收稿日期：2021−06−17；修回日期：2021−09−18
基金项目：国家自然科学基金资助项目（No.61973244）
Foundation Item: The National Natural Science Foundation of China (No.61973244)

selection function，NSSF）创建网络切片[1]，如图 1 所示。网络切片中包括虚拟网元和虚拟链路，由虚拟链路负责将虚拟网元按照一定的拓扑连接，映射到实际的物理节点和物理链路上，完成网络切片的实例化，动态创建网络切片[2]，网络切片不仅提升了 5G 网络资源分配的灵活性，同时也为网络的管理、控制提供了更大的空间。

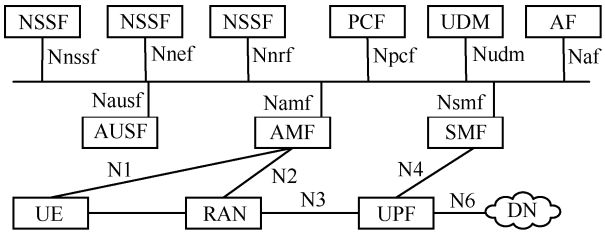

图 1　5G 网络的 SBA 架构

5G 网络切片的建立过程需要经历决策、资源隔离、虚拟网元拉起等过程，在决策阶段以资源分配算法，最优化网络切片到物理网络资源的映射，提升网络切片的接纳成功率和物理网络的资源利用率[3]。对应到 5G 网络的 SBA 架构，决策功能由 NSSF 根据当前实时的资源分布和网络切片的资源需求确定在哪些物理节点和物理链路上承载对应的虚拟网元和虚拟链路，资源隔离则由对应的物理节点和物理链路隔离出给定资源量的容器和虚拟路径，最后将虚拟网元的镜像装载进隔离出的容器内，拉起容器即完成了网络切片的创建。在上述过程中，资源隔离占用的时间最长，对于 5G 网络中要支持的实时应用场景来说，其要求的端到端时延小于 1 ms。根据实时的资源需求动态创建网络切片带来的时延难以满足 5G 中的实时类业务应用场景。因此，基于预测的 5G 网络切片算法被提出，通过在网络中预留资源，提前完成资源隔离来减少动态创建网络切片的时间。

文献[4]提出使用最大相关熵准则（maximum correntropy criterion，MCC）作为代价函数预测微服务请求占用的资源，由于 MCC 需要计算真实值与预测值之间的相关熵，造成其计算复杂度过高。文献[5]提出使用核方法将互相关熵的计算过程从欧几里得空间等变换到再生核希尔伯特空间（reproducing kernel Hilbert space，RHKS），将高维乘法变换为低维加法，其采用的核函数为高斯核函数。核函数的选择将关系到预测算法的精度，而核函数又与实际的问题解空间分布强相关，使得核函数的选择成为预测算法的超参数，降低了算法的适用范围。

采用 MCC 作为代价函数的原因是 MCC 能够刻画数据的高维特征，具有较高的鲁棒性，不受非高斯、非线性噪声的影响。5G 网络在实时场景下的资源请求和其应用场景强相关，在给定的无人驾驶、工业控制的场景下，其噪声虽然有高维特征，但是可以使用高阶矩来滤除，没有必要使用无限高维特征描述。在 5G 网络实时场景的资源请求样本中，几乎不会出现高于 4 阶的噪声，代价函数中包括过高的阶数，不仅对预测精度没有促进作用，反而会增加算法的复杂度。因此，本文提出 5G 网络中基于预测的微服务实例化算法，使用四阶矩作为预测算法的代价函数，消除高阶噪声的影响。以较低的算法复杂度提供可接受的预测精度，提前在网络中隔离对应的资源，降低网络切片实例化的时间。

2　四阶矩代价函数

高阶矩一般被使用在 AFA（adaptive filtering algorithm）[6-7]、回声消除[8-9]、主动噪声抑制[10-11]等领域。在高阶矩中，典型的有归一化 NLMS（normalized least mean square）、归一化 LMF（normalized least mean fourth）。相对于归一化 LMS，归一化 LMF 能够处理非高斯噪声[12-13]，因此，各种变种的归一化 LMF 被提出，如稀疏 LMF[14-15]、扩散 LMF[16-17]等。近年来，比例归一化 LMF（proportionate normalized LMF，PNLMF）在文献[18-19]中被提出，用于在迭代训练的过程中成比例地修改训练误差，独立更新系

统的各个参数[20]。

以上文献中提出的以四阶矩为代价函数的算法要求训练样本的噪声服从同一分布。但 5G 网络实时场景下的资源请求训练样本中的噪声不服从同一分布，使得上述算法性能大大下降。因此，基于非偏估计的残差补偿 AFA 算法被提出，如基于残差补偿的归一化 LMS（bias-compensated NLMS）[21-22]、仿射投影算法（affine projection algorithm，APA）[23]、基于非偏估计的 APA 算法[24]、残差补偿归一化子带滤波算法[25]、基于残差补偿的归一化 LMF（bias-compensated NLMF）算法[26]等，文献[27]在文献[21-22]的基础上，在代价函数中加入 l_0 正则化项，提升预测算法的泛化能力。

2.1 预测问题的输入和输出

由于5G网络实时场景下的资源请求与人的行为强相关，而人的行为存在非常大的不确定性，不能使用上述文献提出的算法进行处理，因此本文将残差补偿和非偏估计用于 PNLMF 算法中，解决 5G 网络实时场景下资源请求的预测问题。

将每个网络切片请求的资源作为回归预测的值。此时，每个网络切片请求的资源定义如下：

$$r=\alpha\sum_{i=1}^{I}r_i^{\mathrm{cpu}}+\beta\sum_{i=1}^{I}r_i^{\mathrm{mem}}+\gamma\sum_{j=1}^{J}r_j^{\mathrm{bw}} \tag{1}$$

其中，α、β、γ 分别表示 CPU、内存、带宽资源的等效权重，I 表示网络切片请求中的虚拟网元个数、J 表示请求中的链路条数。此时，网络切片请求的等效资源就是回归问题的输出。

对于任意一个网络切片请求 i，根据其包含的 L 个特征，文献[5]中将回归问题的输入定义如下：

$$u(i)=[u_1(i),u_2(i),\cdots,u_L(i)]^{\mathrm{T}} \tag{2}$$

$d(i)$ 作为输出值，定义为网络切片请求中包括的总资源量。表示如下：

$$d(i)=u^{\mathrm{T}}(i)w^{o}+v(i) \tag{3}$$

其中，$\boldsymbol{w}^{o}=\left[w_1^{o},w_2^{o},\cdots,w_L^{0}\right]^{\mathrm{T}}$ 表示为系统建立的模型中包括的参数。$\boldsymbol{v}(i)$为系统噪声。

此时，系统的预测误差为：

$$e(i)=d(i)-\boldsymbol{u}^{\mathrm{T}}(i)\boldsymbol{w}(i) \tag{4}$$

$\boldsymbol{w}(i)=[w_1(i),w_2(i),\cdots,w_L(i)]^{\mathrm{T}}$ 表示对应任意一个网络切片的资源请求样本，训练完回归模型后对应的模型参数。

回归模型的更新过程如下：

$$\boldsymbol{w}(i+1)=\boldsymbol{w}(i)+\frac{\mu\boldsymbol{G}(i)e^{3}(i)u(i)}{u^{\mathrm{T}}(i)\boldsymbol{G}(i)\boldsymbol{u}(i)+\mu} \tag{5}$$

其中，μ 表示步长；$\eta>0$ 为正则化参数；$\boldsymbol{G}(i)=\mathrm{diag}(g_1(i),g_2(i),\cdots,g_L(i))$ 为每次更新过程中更新步长过程中使用的对角阵。在上述对角阵中 $g_l(i)=\gamma_l(i)/\sum_{i=1}^{L}\gamma_i(i),1\leqslant l\leqslant L$，$\gamma_l(i)=\max\left[\rho\max\{\xi,|w_1(i)|,\cdots,|w_L(i)|\},|w_l(i)|\right]$。

ρ 和 ξ 的典型取值为 $\rho=5/L$，$\xi=0.01$，且都大于 0。ξ 用于 $w_l(i)$ 的初始化，保证其能够得到快速更新，ρ 用于保证任意 $w_l(i)$ 在进化时如果其值远远小于 $w_l(i)$ 的最大值时，能够得到更新。

不失一般性，系统的输出信号包括实际输出信号和噪声，定义如下：

$$\hat{\boldsymbol{u}}(i)=\boldsymbol{u}(i)+n(i) \tag{6}$$

$\boldsymbol{n}(i)=\left[n_1(i),\ n_2(i),\cdots,\ n_L(i)\right]^{\mathrm{T}}$ 为实际输出信号，其中 $n_l(i)(l\in[1,L])$ 是均值为 0、方差为 σ_n^2 的高斯噪声。此时，将输出信号记为：

$$\begin{aligned}&\hat{e}(i)=d(i)-\hat{\boldsymbol{u}}^{\mathrm{T}}\boldsymbol{w}(i)=\\&d(i)-\left(\boldsymbol{u}(i)+\boldsymbol{n}(i)\right)^{\mathrm{T}}\boldsymbol{w}(i)=e(i)-\boldsymbol{n}(i)^{\mathrm{T}}\boldsymbol{w}(i)\end{aligned} \tag{7}$$

结合文献[6]和上述计算式，对系统的输出做有偏估计，并将式（3）定义为 $\boldsymbol{B}(i)$，使用 $\hat{\boldsymbol{u}}(i)$ 和 $\hat{e}(i)$ 更新 $\boldsymbol{u}(i)$ 和 $e(i)$ 并代入式（7），得到：

$$\boldsymbol{w}(i+1)=\boldsymbol{w}(i)+\frac{\mu\boldsymbol{G}(i)\hat{e}^{3}(i)\hat{\boldsymbol{u}}(i)}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}+\boldsymbol{B}(i) \tag{8}$$

在此基础上，定义权重误差为：

$$\tilde{\boldsymbol{w}}(i)=\boldsymbol{w}(i)-\boldsymbol{w}^{o} \tag{9}$$

为了保证 $\boldsymbol{B}(i)$ 可解，提出以下假设。

假设 1　误差信号 $\boldsymbol{n}(i)$ 和输入信号 $\boldsymbol{u}(i)$ 相互独立。

假设 2　$\boldsymbol{n}(i)$ 和 $\boldsymbol{u}(i)$ 不相关。

假设 3　权重序列 $\tilde{\boldsymbol{w}}(i)$ 中的权重相互之间独立。

假设 4　$\tilde{\boldsymbol{w}}(i)$ 和 $\hat{\boldsymbol{u}}(i)$ 相互独立。

代入式（7）、式（8），得到式（10）：

$$\tilde{\boldsymbol{w}}(i+1)=\tilde{\boldsymbol{w}}(i)+\frac{\mu\boldsymbol{G}(i)\tilde{e}^{3}(i)\hat{\boldsymbol{u}}(i)}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}+\boldsymbol{B}(i) \tag{10}$$

为了计算 $\boldsymbol{B}(i)$，根据文献[23]，当 $E(\tilde{\boldsymbol{w}}(i)\,|\,\hat{\boldsymbol{u}}(i))=0$ 时，得到所提代价函数的无偏估计，如下：

$$E\left(\tilde{\boldsymbol{w}}(i+1)\,|\,\hat{\boldsymbol{u}}(i)\right)=0 \tag{11}$$

在给定 $\hat{u}(i)$ 的条件下，将式(11)代入式(10)，得到如下结果：

$$\begin{aligned}&E\left(\tilde{\boldsymbol{w}}(i+1)\,|\,\hat{\boldsymbol{u}}(i)\right)=E\left(\tilde{\boldsymbol{w}}(i)\,|\,\hat{\boldsymbol{u}}(i)\right)+\\&\mu E\left(\frac{\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)\tilde{e}^{3}(i)}{\hat{\boldsymbol{u}}^{\mathrm{T}}\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\,|\,\hat{\boldsymbol{u}}(i)\right)+E\left[\boldsymbol{B}(i)\,|\,\hat{\boldsymbol{u}}(i)\right]\end{aligned} \tag{12}$$

根据式（11）和式（12），得到如下结果：

$$E\left[\boldsymbol{B}(i)\,|\,\hat{\boldsymbol{u}}(i)\right]=-\mu E\left(\frac{\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)\tilde{e}^{3}(i)}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\,|\,\hat{\boldsymbol{u}}(i)\right) \tag{13}$$

将式（6）和式（7）代入式（13），得到如下结果：

$$\begin{aligned}&E\left(\frac{\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)\tilde{e}^{3}(i)}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\varepsilon}\Big|\hat{\boldsymbol{u}}(i)\right)=E\left\{\frac{\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)\left(e(i)-\boldsymbol{n}(i)^{\mathrm{T}}\boldsymbol{w}(i)\right)^{3}}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\,|\,\hat{\boldsymbol{u}}(i)\right\}=\\&E\left\{\frac{\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)\left(e^{3}(i)-3e(i)^{2}\boldsymbol{n}(i)^{\mathrm{T}}\boldsymbol{w}(i)+3e(i)\left(\boldsymbol{n}(i)^{\mathrm{T}}\boldsymbol{w}(i)\right)^{2}-\left(\boldsymbol{n}(i)^{\mathrm{T}}\boldsymbol{w}(i)\right)^{3}\right)}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\,|\,\hat{\boldsymbol{u}}(i)\right\}=\\&\left\{E\left(\frac{\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)e^{3}(i)}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\,|\,\hat{\boldsymbol{u}}(i)\right)-E\left(\frac{\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)e(i)\left(\boldsymbol{n}(i)^{\mathrm{T}}\boldsymbol{w}(i)\right)^{2}}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\,|\,\hat{\boldsymbol{u}}(i)\right)-E\left(\frac{\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)\left(\boldsymbol{n}(i)^{\mathrm{T}}\boldsymbol{w}(i)\right)^{3}}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\,|\,\hat{\boldsymbol{u}}(i)\right)\right\}\end{aligned} \tag{14}$$

在假设 1 和假设 2 的条件下，修改式（13），得到：

$$\begin{aligned}&E\left(\frac{\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)e^{3}(i)}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\,|\,\hat{\boldsymbol{u}}(i)\right)=E\left(\frac{\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)\left(\boldsymbol{u}^{\mathrm{T}}(i)w^{o}+v(i)-\boldsymbol{u}^{\mathrm{T}}(i)\boldsymbol{w}(i)\right)^{3}}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\,|\,\hat{\boldsymbol{u}}(i)\right)=\\&E\left(\frac{\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)\left(\boldsymbol{u}^{\mathrm{T}}(i)\tilde{\boldsymbol{w}}(i)+v(i)\right)^{3}}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\,|\,\hat{\boldsymbol{u}}(i)\right)=\\&E\left(\frac{\boldsymbol{G}(i)\left[\left(\boldsymbol{u}^{\mathrm{T}}(i)\tilde{\boldsymbol{w}}(i)\right)^{3}+3\left(\boldsymbol{u}^{\mathrm{T}}(i)\tilde{\boldsymbol{w}}(i)\right)^{2}v(i)+3\boldsymbol{u}^{\mathrm{T}}(i)\tilde{\boldsymbol{w}}(i)v^{2}(i)+v^{3}(i)\right]}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\,|\,\hat{\boldsymbol{u}}(i)\right)=0\end{aligned} \tag{15}$$

$$E\left(\frac{\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)\left(\boldsymbol{n}(i)^{\mathrm{T}}\boldsymbol{w}(i)\right)^{2}}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\,|\,\hat{\boldsymbol{u}}(i)\right)=0 \tag{16}$$

$$E\left(\frac{\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)\left(n(i)^{\mathrm{T}}\boldsymbol{w}(i)\right)^{3}}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}|\hat{\boldsymbol{u}}(i)\right)=$$

$$E\left(\frac{\boldsymbol{G}(i)\left(\hat{\boldsymbol{u}}(i)+\boldsymbol{n}(i)\right)v_{\mathrm{in}}(i)^{\mathrm{T}}\boldsymbol{w}(i)^{\mathrm{T}}\boldsymbol{n}(i)\boldsymbol{n}(i)^{\mathrm{T}}\boldsymbol{w}(i)}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}|\hat{\boldsymbol{u}}(i)\right)=\sigma_n^4E\left(\frac{\boldsymbol{G}(i)\boldsymbol{w}(i)\boldsymbol{w}(i)^{\mathrm{T}}\boldsymbol{w}(i)}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}|\hat{\boldsymbol{u}}(i)\right)\tag{17}$$

由于采用了四阶矩作为代价函数，文献[12-13]中使用归一化 LMF 处理高斯噪声的方式，定义噪声的误差为 σ_n^2，得到其平方 σ_n^4，并将其代入式（15）～式（17），得到如下结果：

$$E\left(\boldsymbol{B}(i)|\hat{\boldsymbol{u}}(i)\right)=\boldsymbol{\mu}\sigma_n^4E\left(\frac{G(i)\boldsymbol{w}(i)\boldsymbol{w}(i)^{\mathrm{T}}\boldsymbol{w}(i)}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}|\hat{\boldsymbol{u}}(i)\right)\tag{18}$$

利用文献[23]中的统计近似方法，得到：

$$\boldsymbol{B}(i)=\mu\sigma_n^4\frac{\boldsymbol{G}(i)\boldsymbol{w}(i)\boldsymbol{w}(i)^{\mathrm{T}}\boldsymbol{w}(i)}{\hat{\boldsymbol{u}}^{T}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\tag{19}$$

将式（19）代入式（10），得到将四阶矩作为代价函数时的权重更新公式如下：

$$\boldsymbol{w}(i+1)=\left(i+\mu\sigma_n^4\frac{\boldsymbol{G}(i)\boldsymbol{w}(i)\boldsymbol{w}(i)^{\mathrm{T}}}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\right)\boldsymbol{w}(i)+\frac{\mu\boldsymbol{G}(i)\hat{e}^3(i)\hat{\boldsymbol{u}}(i)}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\tag{20}$$

2.2 噪声信号的均方误差估计

文献[21-23,25-27]提出了很多针对噪声信号的均方误差评估方法，而在解决实际问题时，需要根据样本的特征选择不同的噪声估计方法，保证较高的噪声估计精度。如果噪声误差估计过高，会减弱小于请求资源量期望的样本的作用；反之，如果噪声误差估计过低，则预测模型的精度受噪声的影响较大。

由于 5G 网络实时应用场景的资源请求中噪声与用户的行为强相关，表现为用户的某些随机申请 5G 网络资源的行为，这会影响网络切片请求的资源量的统计规律，无法进行准确的描述。

文献[27]中的噪声信号均方误差评估方法考虑了输入输出噪声的相关性，在应用于解决本节所提持久型微服务请求资源量的预测问题时，可以抵消噪声误差在输入和输出上的作用；同时，遗忘因子的引入，也给调整噪声估计结果提供了超参，通过设置合适的遗忘因子，缩小噪声信号均方误差的评估结果。因此，本节采用文献[27]提出的方法对噪声的均方误差进行估计。

文献[27]中定义的噪声均方误差如下：

$$\sigma_n^2(i)=\frac{\sigma_{\hat{e}}^2(i)}{L\sigma_w^2(i)+\kappa+\dfrac{\sigma_{\hat{e}}^2(i)L}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\hat{\boldsymbol{u}}(i)}}\tag{21}$$

$$\sigma_{\hat{e}}^2(i)=\delta\sigma_{\hat{e}}^2(i-1)+(1-\delta)\hat{e}^2(i)\tag{22}$$

$$\sigma_w^2(i)=\delta\sigma_w^2(i-1)+(1-\delta)\frac{1}{L}\boldsymbol{w}^{\mathrm{T}}(i)\boldsymbol{w}(i)\tag{23}$$

在式（20）和式（21）中，参数 δ 为遗忘因子，κ 为输入输出噪声相关参数。此时，式（20）可以改写为：

$$\boldsymbol{w}(i+1)=\left(i+\mu\sigma_n^4(i)\frac{\boldsymbol{G}(i)\boldsymbol{w}(i)\boldsymbol{w}(i)^{\mathrm{T}}}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{u}(i)+\varepsilon}\right)\boldsymbol{w}(i)+\frac{\mu\boldsymbol{G}(i)e(i)\hat{\boldsymbol{u}}(i)}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\varepsilon}\tag{24}$$

此时，基于四阶矩的代价函数进行权重估计时，采用算法 1 进行更新。

算法 1

初始化 $\boldsymbol{w}(0)=0$

定义的参数包括：$\mu,\varepsilon,\rho,\xi,\delta,\kappa$

针对任意一次迭代 i：

步骤 1 $\hat{e}(i)=d(i)-\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{w}(i)$

步骤 2　$\boldsymbol{G}(i)=\mathrm{diag}\left(g_1(i),g_2(i),\cdots,g_L(i)\right)$

步骤 3　$g_l(i)=\dfrac{\gamma_l(i)}{\sum_{i=1}^{M}\gamma_i(i)}$，$1\leqslant l\leqslant L$

步骤 4　$\gamma_l(i)=\max\left[\rho\max\left\{\xi,|w_1(i)|,\cdots,|w_M(i)|\right\},|w_l(i)|\right]$

步骤 5　$\sigma_n^2(i)=\dfrac{\sigma_{\tilde{e}}^2(i)}{L\sigma_{\boldsymbol{w}}^2(i)+\kappa+\dfrac{\sigma_{\tilde{e}}^2(i)L}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\hat{\boldsymbol{u}}(i)}}$

步骤 6　$\boldsymbol{w}(i+1)=\left(i+\mu\sigma_n^4(i)\dfrac{\boldsymbol{G}(i)\boldsymbol{w}(i)\boldsymbol{w}(i)^{\mathrm{T}}}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\eta}\right)\boldsymbol{w}(i)+\dfrac{\mu\boldsymbol{G}(i)e(i)\hat{\boldsymbol{u}}(i)}{\hat{\boldsymbol{u}}^{\mathrm{T}}(i)\boldsymbol{G}(i)\hat{\boldsymbol{u}}(i)+\boldsymbol{\eta}}$

3　基于四阶矩预测的网络切片算法

3.1　基于预测的网络切片算法

基于预测的网络切片算法流程如图 2 所示。

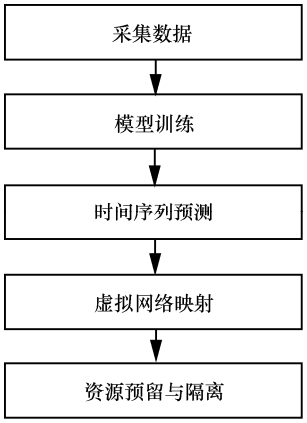

图 2　基于预测的网络切片算法流程

步骤 1　采集历史样本，包括一个星期内所有 5G 实时应用场景下的网络切片请求资源量。

步骤 2　根据历史样本训练，其代价函数为本文定义的四阶矩，训练算法采用 BP 神经网络，训练后得到预测模型。

步骤 3　将 5G 网络实时应用场景下的请求资源量看作时间序列，预测下一时刻的网络切片请求资源量。

步骤 4　使用文献[28]提出的虚拟网络映射算法，确定承载下一时刻网络切片请求资源量的物理节点和物理链路。

步骤 5　在确定的物理节点和物理链路上预留给定的资源，完成虚拟节点和虚拟链路隔离。

3.2　算法复杂度分析

由于本文使用 BP 神经网络训练预测模型，预测算法的时间复杂度与每次训练的样本个数、计算梯度时的复杂度、迭代次数相关。在预测算法使用了 mini-batch 方法进行训练，每次 mini-batch 训练时使用的样本个数为样本总数的一部分。

使用 BP 神经网络进行训练，每次更新进行权重更新时的复杂度分析如下：

假设神经网络有 d 个输入神经元，l 个输出神经元，J 个隐藏层，每个隐藏层中包括 q_j 个神经元，第 j 个隐藏层中第 h 个神经元的阈值为 γ_{jh}。其中，输出层中第 j 个神经元的阈值为 θ_j，输入层第 i 个神经元与第一个隐藏层中第 h 个神经元的连接权重为 v_{ih}，最后一个隐藏层中第 h 个神经元与输出层第 j 个神经元之间的连接权重为 w_{hj}。

记最后一个隐藏层中第 h 个神经元接收到的输入为 $\alpha_h=\sum_{i=1}^{h}v_{ih}\times x_i$，输出层第 j 个神经元接收到的输入为 $\beta_j=\sum_{h=1}^{q}w_{hj}\times b_h$，并且记第 i 个隐藏层的第 h 个神经元与第 $i+1$ 个隐藏层中的第 j 个神经元之间的连接权重为 w_{hj}^{i}。

在每次迭代过程中，需要更新的权重个数为：$d\times q_1+\sum_{j=1}^{J-1}q_j\times q_{j+1}+q_J\times l$，其中，$d\times q_1$ 为输出层到第一个隐藏层的权重个数，$q_J\times l$ 为最后一个隐藏层到输入层的权重个数，$\sum_{j=1}^{J-1}q_j\times q_{j+1}$ 为隐藏层之间的权重。

除了上述权重外，还需要确定每个隐藏层和输出层的阈值，其个数为：$l+\sum_{j=1}^{J}q_j$。因此，总的参数个数为：$q_1+q_J+q_2\times(d+q_1)+l\times(q_J+1)+$

$\sum_{j=2}^{J-1} q_j \times \left(1+q_{j+1}\right)$。

在计算每个需要迭代计算的权重的时间复杂度时，记每个训练样本 $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ 的输出为 $\boldsymbol{y}_k = \left\{\hat{y}_1^k, \hat{y}_2^k, \cdots, \hat{y}_l^k\right\}$，则四阶矩代价函数为：$E_k = \frac{1}{4} \times \sum_{j=1}^{l} \left(\hat{y}_j^k - y_j^k\right)^4$。使用梯度下降法进行权杖更新，以 η 为学习率，则：$\Delta w_{hj} = -\eta \times \frac{\partial E_k}{\partial w_{hj}}$。

使用反向递推，得到 $\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \times \frac{\partial \hat{y}_j^k}{\partial \beta_j} \times \frac{\partial \beta_j}{\partial w_{hj}} \times \prod_{i=J-2}^{1} \frac{\partial \beta_i}{\partial w_{hj}^{i+1}}$，其中，$\frac{\partial \beta_j}{\partial w_{hj}} = b_h$。

由于激活函数 sigmoid 有以下性质：$f'(x) = f(x) \times \left(1 - f(x)\right)$，可得到：$g_j = \frac{\partial E_k}{\partial \hat{y}_j^k} \times \frac{\partial \hat{y}_j^k}{\partial \beta_j} = -\left(\hat{y}_j^k - y_j^k\right) \times f'\left(\beta_j - \theta_j\right) = \hat{y}_j^k \times \left(1 - \hat{y}_j^k\right) \times \left(y_j^k - \hat{y}_j^k\right)$。

得到 BP 算法中权重更新的计算式为：$\Delta w_{hj} = \eta \times g_j \times b_h$，$\Delta \theta_{hj} = -\eta \times g_j$，$\Delta v_{hj} = \eta \times e_h \times x_i$，$\Delta \gamma_{jh} = \eta \times e_h$。

此时，$e_h = -\frac{\partial E_k}{\partial b_h} \times \frac{\partial b_h}{\partial \alpha_h} = -\sum_{j=1}^{l} \frac{\partial E_k}{\partial b_h} \times \frac{\partial \beta_j}{\partial b_h} \times f'\left(\alpha_h - \gamma_h\right) = b_h \times \left(1 - b_h\right) \times \sum_{j=1}^{l} w_{hj} \times g_j$。

将每次权重更新时的代价定义为 C，$C = O\left(2+2l\right) + O\left(2+2d\right) + O\left(\sum_{j=1}^{J} 2 + 2 \times q_j\right)$。

其中，l 表示输出层包括的神经元个数；d 表示输入层包括的神经元个数；q_j 表示第 j 个隐藏层中包括的神经元个数。

每次权重更新时的代价定义为 C 可以改写为：$C = O\left(d + l + \sum_{j=1}^{J} q_j\right)$，即整个网络中包括的神经元的个数。

整个预测算法的时间复杂度与迭代次数、训练时使用的样本个数、使用梯度下降法更新权重时的计算复杂度相关，因此，基于四阶矩的预测算法的时间复杂度为 $O\left(I \times n \times C\right)$，其中，$I$ 表示神经网络训练时的最大迭代次数；n 表示每次输入神经网络中进行 mini-batch 训练时的样本个数；C 表示使用梯度下降法进行权重更新时的计算复杂度。

由于在训练过程中每次迭代后都需要保存样本、权重，因此预测算法的空间复杂度为：$O\left(K \times N + \sum_i 2 \times n_i\right) = O\left(K \times N + 2 \times \hat{n} \times l\right)$。其中，$K$ 表示样本的维度，为自变量的维度和因变量的维度之和；N 表示样本的个数；n_i 表示每层神经网络的节点个数，由于每个节点需要保存 ω 和 b 两个参数，则每个节点的空间复杂度为 $2 \times n_i$；$\hat{n}$ 则是每层神经网络的平均节点个数；l 为神经网络的层数。

4 实验结果与分析

4.1 预测算法的性能分析

为了分析所提算法的预测准确度和稳定性，以 30 天的 5G 网络实时应用场景下资源请求数据作为训练样本，并且在所提算法和对比算法中，设置规则化参数 $\eta = 0.0001$、$\xi = 0.01$、$\kappa = 0.0001$、$\delta = 0.7$。为了分析所提算法的性能，将其与对比算法进行性能分析，对比算法包括：文献[10]提出的 NLMS 算法、文献[11]提出的 NLMF 算法、文献[18]提出的 PNLMS 算法、文献[19]提出的 PNLMF 算法、文献[26]提出的 BCNLMF 算法，本文所提算法被标记为 BCPNLMF 算法，即使用残差补偿和非偏估计的 PNLMF 算法。

在实验过程中，将所采集的样本中 70%作为训练集，其余 30%作为测试集。在测试集中对所提算法和对比算法的效果进行分析和比较。

4.1.1 MSD 分析

为了分析所提算法，使用均方误差（mean square deviation，MSD）对所提算法与对比算法在

给定的样本集下进行训练并分析其性能。所提算法和对比算法具有相同的四阶矩代价函数，对比算法包括 NLMS、PNLMS、NLMF、PNLMF 和 BCNLMF。文献[25]中对 MSD 的定义如下：

$$\mathrm{MSD}(i)=\frac{\left\|\boldsymbol{w}(i)-\boldsymbol{w}^{0}\right\|^{2}}{\left\|\boldsymbol{w}^{0}\right\|^{2}} \tag{25}$$

训练后得到的系统模型的稀疏度定义如下：

$$\mathrm{SR}=\frac{N_{\text{non-zero}}}{L} \tag{26}$$

其中，$N_{\text{non-zero}}$ 是训练所得系统模型中非零的系数，仿真过程中，将其设置为 4/50。

本文所提算法与对比算法在相同收敛速度下的 MSD 分析如图 3 所示。从图中可以看出，所提算法与对比算法相比，其 MSD 最小，在预测时相比较于其他算法误差最小。

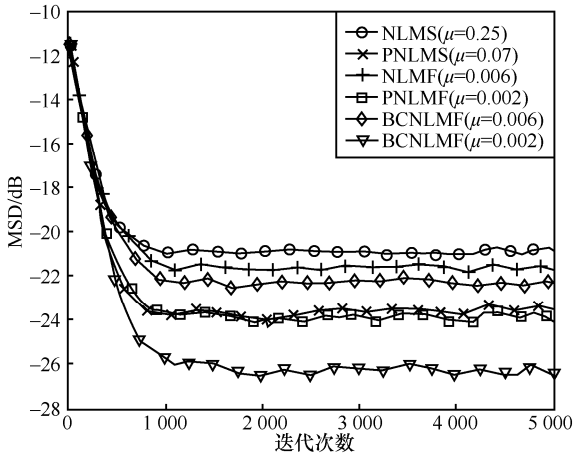

图 3　相同收敛速度下本文所提算法与对比算法的 MSD 分析

在获得给定 MSD 条件下，本文所提算法与对比算法的收敛速度分析如图 4 所示。从图 4 中可以看出，所提算法的收敛速度更快，这是因为在训练过程中使用得分矩阵进行了参数更新。

4.1.2　SR 对 MSD 的影响

在所提算法和对比算法中对 SR 进行调整，随着算法迭代将其从 2/50 调整到 5/50，观察 SR 随着迭代变化时对算法 MSD 参数的影响。图 5 所示为不同 SR 值随迭代变化对 MSD 的影响。

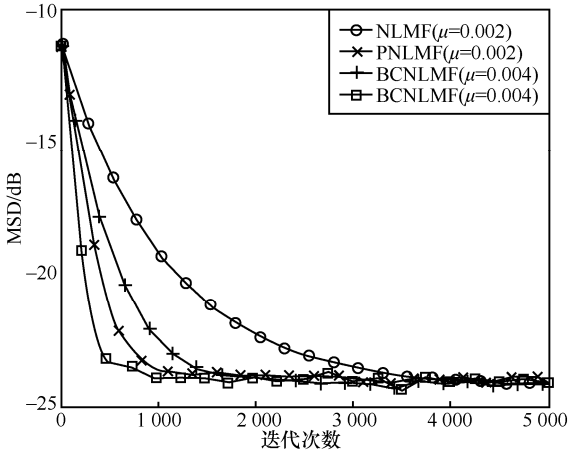

图 4　获得相同 MSD 时本文所提算法与对比算法的收敛速度分析

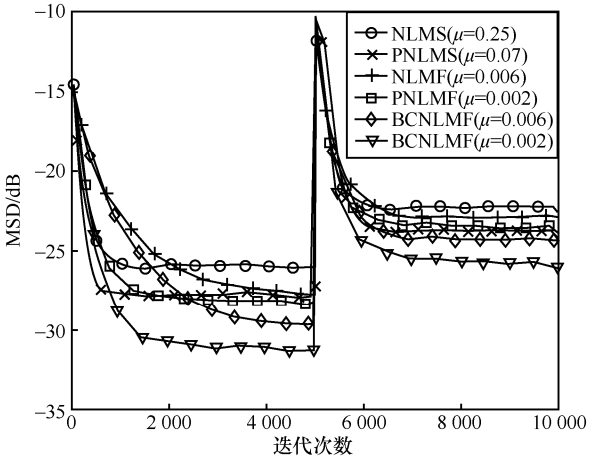

图 5　不同 SR 值随迭代变化时对 MSD 的影响

从图 5 中可以看出，SR 随着迭代次数进行变化时，本文所提算法与对比算法相比，其 MSD 仍然是最小的，说明所提算法更加稳定，能够获得更小的预测误差。

通过上述实验结果分析，本文所提算法与对比算法相比，能够获得更好的预测精度，保证预测结果与测试集的对应样本的误差的标准方差最小，并且本文所提算法能够获得最快的收敛速度和最佳的稳定性。

4.2　基于预测的 5G 网络切片算法分析

4.2.1　网络环境设计

物理节点属性包括 CPU 核数、内存大小，链路属性包括带宽。仿真实验中的物理节点数量服从（100，1 000）之间的均匀分布，每个物理节点上的 CPU 核数为集合（16，32，64）上随机选择

的数值，选择的每个数值的概率服从均匀分布。每个物理节点上的内存为集合（128，256，512）上随机选择的数值，选择每个数值的概率服从均匀分布，内存的单位是 MB。物理链路均为 10 GE 带宽，提供 10 Gbit/s 的传输能力，物理节点之间是否连接的概率服从参数为 0.1 的负指数分布。

网络切片的资源请求包括虚拟网元个数、每个虚拟网元包括的 CPU 核数和内存请求，连接每个网元的传输带宽，其表现形式为物理承载环境的一个子网。其中，虚拟网元的个数服从（5，10）之间的均匀分布，每个网元请求的 CPU 核数服从（2，5）之间的均匀分布，每个网元请求的内存数从集合（8，16，32）中选择，选择的每个数值的概率服从均匀分布，单位为 MB。微服务请求中每个网元之间连接的概率服从参数为 0.1 的负指数分布，且请求的带宽服从（10，100）之间的均匀分布，单位为 Mbit/s。

网络切片请求的时间间隔服从参数为 μ 的均匀分布，如果网络切片被成功接纳并实例化，其存在的时间服从参数为 β 的负指数分布；如果被拒绝，则直接丢弃不排队。此时，物理承载环境中的负载近似为 β/μ。

在文献[28]的基础上使用本文所提预测算法，完成网络切片的资源预留，考查所提算法与文献[28]的阻塞率、剩余资源分布、收益和网络切片的实例化时间。文献记为“粒子群算法”，本文所提预测算法与文献[28]中所提网络切片实例化算法记为“粒子群+预测”。

4.2.2 实验结果分析

（1）阻塞率分析

本文所提算法与对比算法的阻塞率分析如图 6 所示，在增加了所提的基于预测的网络切片算法后，在给定负载条件下，本文所提算法与对比算法的效果大致相当，这是因为本文所提算法只是提前预留一部分资源，相当于提前为一部分网络切片请求分配资源。随着负载的增加，本文所提算法与粒子群优化算法都出现了阻塞率升高的情况，并且升高的程度大致相当。

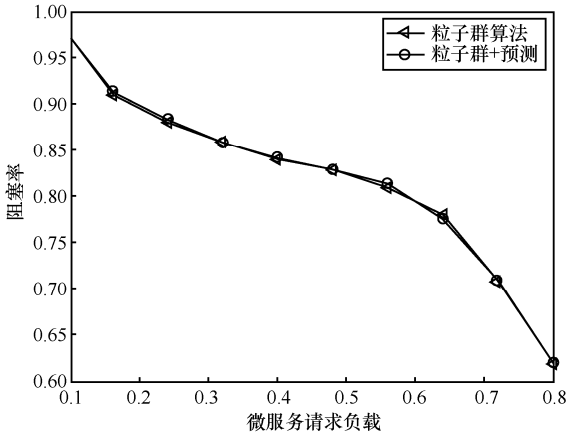

图 6 基于预测的微服务实例化算法的阻塞率分析

随着预留资源的网络切片个数的增加，对于本文所提基于预测的网络切片算法，其阻塞率基本没有变化。

（2）剩余资源分布分析

本文所提算法与对比算法的剩余资源分布分析如图 7 所示，本文所提算法与对比算法相比，其剩余资源分布差别不大，这是因为所提算法只是使用了对比算法提前为一部分网络切片请求分配资源，还是以网络切片请求被实例化后物理承载环境中剩余资源分布更均匀为优化目标。并且随着负载的增加，本文所提算法与对比算法都会使阻塞率增加。

随着预留资源的网络切片请求的个数的增加，对于本文所提算法，物理承载环境中剩余资源分布基本没有变化。

（3）收益分析

本文所提算法与对比算法的收益分析如图 8 所示，与阻塞率和物理承载环境的剩余资源分布情况类似，本文所提算法与对比算法相比，收益基本相当。并且随着负载的增加，本文所提算法的收益相比于与对比算法，其增加的趋势和幅度基本一致。并且随着预留资源的网络切片的个数的增加，对于本文所提算法，其收益也基本没有变化。

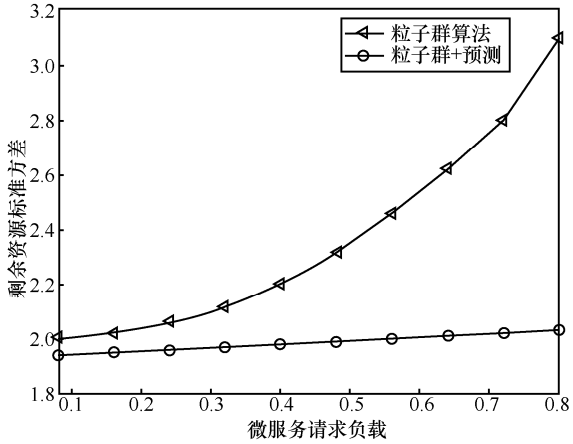

图 7　基于预测的微服务实例化算法的剩余资源分布分析

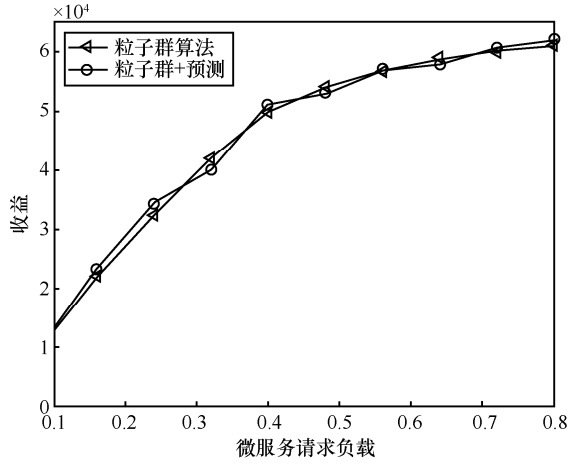

图 8　基于预测的微服务实例化算法的收益分析

（4）实例化时间分析

网络切片实例化的时间包括算法执行时间、资源准备时间、资源分配时间，具体定义如下：

$$\mathrm{Time}=T_{\mathrm{com}}+T_{\mathrm{pre}}+T_{\mathrm{allo}} \tag{27}$$

其中，算法执行时间远远小于资源分配时间，相比可以忽略不计。而对于网络切片请求来说，如果包括的节点和链路越多，每个节点和链路上请求的资源越多，其资源准备时间越长，并且资源分配时间与资源准备时间呈正向线性相关关系，即资源准备时间越长，资源分配时间越长。表 1 为所提算法与对比算法的实例化时间对比。

从表 1 中可以看出，本文所提算法与对比算法相比，能够显著降低网络切片的创建时间，根本原因是新创建的网络切片复用了原始网络切片的资源，即原来的网络切片结束服务后只释放了资源，而没有销毁容器，新创建的网络切片直接复用原始网络切片释放的容器，省去了创建容器的时间。而创建容器的时间在网络切片实例化过程中所占比重最大。

因此，本文所提算法对于减少网络切片请求的实例化时间贡献最大，从仿真结果上看，本文所提算法能够将网络切片的实例化时间减少 50%。并且随着预留资源的网络切片请求个数的增加，其建立时间降低得更多。这是因为预留资源的网络切片请求数越多，越能够节省更多的网络切片实例化时分配资源的时间。

此外，如图 9 所示，网络切片请求中的虚拟网元个数越多，其平均资源准备时间越长，这是因为在承载原虚拟网元请求的容器上需要回收垃圾，重新在容器中装载新的虚拟网元请求，才能实例化新的网络切片。而回收垃圾和装载新虚拟网元的时间与虚拟网元请求的个数成正比。

5　结束语

本文提出一种 5G 网络中基于预测的微服务实例化算法。将四阶矩作为代价函数，使用 BP 神经网络训练得到预测模型。根据预测模型的输出在 5G 网络中提前隔离资源，当网络切片请求到达时，直接拉起容器、隔离虚拟链路，完成网络切片的动态创建，可以将网络切片的创建时间减

表 1　网络切片的实例化时间分析（平均微服务请求数：30）

算法	算法执行时间/ms	资源准备时间/ms	资源分配时间/ms	微服务实例化时间/ms
对比算法	151	3 233	22	3 406
本文算法	127	1 632	22	1 772

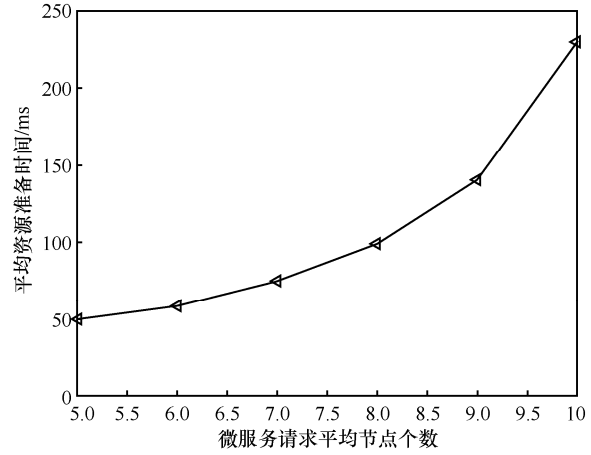

图 9　网络切片实例化时间网络切片请求平均节点个数的变化曲线

少 50%。对 5G 网络中实时应用场景下的网络切片创建方法具有一定的参考价值。

参考文献：

[1] LIU Y Q, PENG M G, SHOU G C, et al. Toward edge intelligence: multiaccess edge computing for 5G and Internet of things[J]. IEEE Internet of Things Journal, 2020, 7(8): 6722-6747.

[2] BARAKABITZE A A, BARMAN N, AHMAD A, et al. QoE management of multimedia streaming services in future networks: a tutorial and survey[J]. IEEE Communications Surveys & Tutorials, 2020, 22(1): 526-565.

[3] ZHANG X L, WANG D, YAN K, et al. Design of network slicing system based on SDN/NFV[J]. Frontiers in Artificial Intelligence and Applications, 2019,10(2): 969-974.

[4] ZHU M, QU H, ZHAO J H. Instance expansion algorithm for micro-service with prediction[J]. Electronics Letters, 2018, 54(6): 356-357.

[5] ZHAO J, ZHANG H B, WANG G. Projected kernel recursive maximum correntropy[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2018, 65(7): 963-967.

[6] SHARIATI N, BJÖRNSON E, BENGTSSON M, et al. Low-complexity polynomial channel estimation in large-scale MIMO with arbitrary statistics[J]. IEEE Journal of Selected Topics in Signal Processing, 2014, 8(5): 815-830.

[7] MA W T, DUAN J D, CHEN B D, et al. Recursive generalized maximum correntropy criterion algorithm with sparse penalty constraints for system identification[J]. Asian Journal of Control, 2017, 19(3): 1164-1172.

[8] TSAO Y, FANG S H, SHIAO Y. Acoustic echo cancellation using a vector-space-based adaptive filtering algorithm[J]. IEEE Signal Processing Letters, 2015, 22(3): 351-355.

[9] PADHI T, CHANDRA M, KAR A. Adaptive proportionate normalized least mean squares channel equalizer for MIMO-OFDM systems[C]//2015 Annual IEEE India Conference (INDICON). Piscataway: IEEE Press, 2015: 1-4.

[10] GHASEMI S, KAMIL R, MARHABAN M H. Nonlinear thf-fxlms algorithm for active noise control with loudspeaker nonlinearity[J]. Asian Journal of Control, 2016, 18(2): 502-513.

[11] SRAZHIDINOV R, KAMIL R, MOHD NOOR S B. NLFXLMS and THF-NLFXLMS algorithms for Wiener-Hammerstein nonlinear active noise control[J]. Asian Journal of Control, 2017, 19(5): 1791-1801.

[12] EWEDA E. Dependence of the stability of the least mean fourth algorithm on target weights non-stationarity[J]. IEEE Transactions on Signal Processing, 2014, 62(7): 1634-1643.

[13] EWEDA E, BERSHAD N J, BERMUDEZ J C M. Stochastic analysis of the least mean fourth algorithm for non-stationary white Gaussian inputs[J]. Signal, Image and Video Processing, 2014, 8(1): 133-142.

[14] GUI G, ADACHI F. Sparse least mean fourth algorithm for adaptive channel estimation in low signal-to-noise ratio region[J]. International Journal of Communication Systems, 2014, 27(11): 3147-3157.

[15] LI Y S, WANG Y Y, JIANG T. Norm-adaption penalized least mean square/fourth algorithm for sparse channel estimation[J]. Signal Processing, 2016, 128: 243-251.

[16] WANG W Y, ZHAO H Q. Performance analysis of diffusion least mean fourth algorithm over network[J]. Signal Processing, 2017, 141: 32-47.

[17] NI J G, YANG J. Variable step-size diffusion least mean fourth algorithm for distributed estimation[J]. Signal Processing, 2016, 122: 93-97.

[18] YILMAZ Y, KOZAT S S. An extended version of the NLMF algorithm based on proportionate Krylov subspace projections[C]//2009 International Conference on Machine Learning and Applications. Piscataway: IEEE Press, 2009: 404-408.

[19] SAYIN M O, YILMAZ Y, DEMIR A, et al. The Krylov-proportionate normalized least mean fourth approach: formulation and performance analysis[J]. Signal Processing, 2015, 109: 1-13.

[20] BENESTY J, PALEOLOGU C, CIOCHINĂ S. Proportionate adaptive filters from a basis pursuit perspective[J]. IEEE Signal Processing Letters, 2010, 17(12): 985-988.

[21] KANG B, YOO J, PARK P. Bias-compensated normalised LMS algorithm with noisy input[J]. Electronics Letters, 2013, 49(8): 538-539.

[22] JUNG S M, PARK P. Normalised least-mean-square algorithm for adaptive filtering of impulsive measurement noises and noisy inputs[J]. Electronics Letters, 2013, 49(20): 1270-1272.

[23] ZHI Y F, ZHENG X, LI R. On the convergence behavior of affine projection algorithm with direction error[J]. Asian Journal of Control, 2014, 16(2): 530-538.

[24] ZHAO H Q, ZHENG Z S. Bias-compensated affine-projection-like algorithms with noisy input[J]. Electronics Letters, 2016, 52(9): 712-714.

[25] ZHENG Z S, ZHAO H Q. Bias-compensated normalized subband adaptive filter algorithm[J]. IEEE Signal Processing Letters, 2016, 23(6): 809-813.

[26] ZHENG Z S, LIU Z G, ZHAO H Q. Bias-compensated normalized least-mean fourth algorithm for noisy input[J]. Circuits, Systems, and Signal Processing, 2017, 36(9): 3864-3873.

[27] YOO J, SHIN J, PARK P. An improved NLMS algorithm in sparse systems against noisy input signals[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2015, 62(3): 271-275.

[28] WANG L, QU H, ZHAO J H, et al. Virtual network embedding with discrete particle swarm optimisation[J]. Electronics Letters, 2014, 50(4): 285-286.

[作者简介]

张佳庚（1990− ），男，西安交通大学网络信息中心、西安交通大学电子与信息学部工程师，主要研究方向为下一代互联网、5G 网络、网络舆情监测。

祝敏（1982− ），女，博士，西安交通大学电子与信息学部副研究员，主要研究方向为下一代互联网。

杜丰（1982−），男，西安交通大学网络信息中心工程师，主要研究方向为下一代互联网。

王齐（1982− ），女，现就职于西安交通大学网络信息中心，主要研究方向为高校信息化。

刘俊（1984− ），男，西安交通大学网络信息中心副主任，主要研究方向为网络舆情监测。

锁志海（1971− ），男，西安交通大学网络信息中心主任，主要研究方向为高校信息化、网络舆情监测等。

王力（1986− ），男，博士，西安交通大学电子与信息学部高级工程师，主要研究方向为无线通信技术、5G 网络等。