内嵌式数据处理单元的服务功能链编排机制

邢若鹏[1]，郭少勇[1]，邱雪松[1]，郝佳恺[2]，李宇婷[2]
（1.北京邮电大学网络与交换技术国家重点实验室，北京 100876；2.国网北京市电力公司，北京 100031）

摘　要：针对云–边–端场景下大规模、多元化业务应用对高灵活性和高性能网络服务的需求，提出一种内嵌式数据处理单元的服务功能链编排机制。首先，提出了一种内嵌式数据处理单元的服务功能链架构，以解决传统服务功能链无法兼顾灵活性和性能的问题。其次，针对该架构下服务功能需求异构资源的特点，提出综合考虑资源占用率、硬件加速器利用率、端到端时延和部署响应时间的服务功能链部署问题。为解决该问题，设计了一种基于遗传粒子群优化算法的服务功能链部署方法，同时提出子链划分方法和相似请求合并方法，分别优化方法的计算时间和部署方法的平均响应时间。仿真结果表明，相较于对比方法，所提方法可以得出更优的服务功能链部署策略，且具备较快的决策速度。

关键词：服务功能链；服务功能链部署；云–边–端网络；数据处理单元；遗传粒子群优化算法
中图分类号：TP309
文献标志码：A
DOI：10.11959/j.issn.1000–436x.2026079

Service function chain orchestration mechanism on embedded DPU

Xing Ruopeng[1], Guo Shaoyong[1], Qiu Xuesong[1], Hao Jiakai[2], Li Yuting[2]
1. State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
2. State Grid Beijing Electric Power Company, Beijing 100031, China

Abstract: In response to the demand for high flexibility and high-performance network services for large-scale and diversified business applications in cloud-edge-end scenarios, a service function chain orchestration mechanism on embedded data processing units was proposed. Firstly, an architecture of service function chain on embedded data processing units was proposed to address the issue of traditional service function chains being unable to balance high flexibility and high-speed processing. Secondly, in response to the heterogeneous resources requirement of service functions under this architecture, a service function chain deployment problem was formulated by comprehensively incorporating resource occupancy, hardware accelerator utilization, end-to-end latency, and deployment response time. To solve this problem, a service function chain deployment method based on the genetic particle swarm optimization algorithm was developed, and the process of dividing sub chains and merging similar requests was designed to improve the speed of the proposed method. Simulation results show that, compared with other methods, the proposed method provides better service function chain deployment decisions and has a faster decision-making speed.

Keywords: service function chain, service function chain deployment, cloud-edge-end network, data processing unit, genetic particle swarm optimization algorithm

收稿日期：2025–12–30；**修回日期：**2026–03–18
通信作者：邱雪松，xsqiu@bupt.edu.cn
基金项目：国家重点研发计划基金资助项目（No.2024YFB3108204）
Foundation Item：The National Key Research and Development Program of China (No.2024YFB3108204)

0 引言

云–边–端网络架构在隐私保护、带宽压力、业务时延、资源利用率等方面相比传统云计算体系具备优势，已经越来越多地应用于智慧城市、工业控制、智能电网等领域[1]。在云–边–端架构网络中，业务类型趋向多元化和智能化，跨主体间交互愈加频繁，基础设施对外开放程度不断提升，接入网络的智能化终端数量激增，网络系统的复杂性和动态性急剧提升[2]，这对网络服务提供的灵活性和网络服务处理的高效性提出了更高的要求。

为灵活提供网络服务，服务功能链（service function chain，SFC）是一种可行的解决方案[3]。该架构利用虚拟化技术统一纳管服务器上的计算、存储资源池，将网络服务以可打包容器形式在服务器上快速部署和启动，同时利用软件定义网络技术引导业务流量依次通过所需的网络功能[4]，能够为不断变化的网络环境和业务按需提供网络服务，提升服务效率和降低服务成本。在云–边–端网络中，为进一步降低网络服务的时延，SFC可能被更多地部署于边缘侧[5]。但是，摩尔定律逐渐失效，通用处理器的性能提升无法赶上云–边–端网络流量的增长速度，难以匹配低时延线速服务功能处理的需求[6]。利用软硬件协同计算加速网络服务处理可以解决该问题[7]，但是多种多样的网络服务需要的硬件加速器种类繁多，将其在各个网络边缘广泛全量部署的成本极高，不具备现实可行性。

数据处理单元（data processing unit，DPU）是一种集成了CPU、内存、现场可编程门阵列（field programmable gate array，FPGA）等芯片资源的新型可编程专用处理器[8]，具有PCIe接口，可以内嵌于服务器、边缘网关、交换机等设备，提供模块化的处理能力扩展[9]。FPGA的远程烧录和动态部分重配置[10]技术，可以实现硬件加速器在有限FPGA资源下的按需部署和不停机重调整。将可内嵌的DPU作为网络SFC的算力基座，并在其上部署网络服务容器和硬件加速器，可以兼顾云–边–端网络服务提供的灵活性和服务处理的高效性，具备极大的潜力，但是目前尚未有研究提出相关机制。同时，传统的CPU、图形处理单元（graphics processing unit，GPU）等通用芯片的SFC部署可直接考虑计算节点剩余可用的计算资源量，但DPU的SFC服务功能不能直接利用FPGA资源进行计算，而是需要先消耗FPGA资源部署对应功能的可被调用的加速器，再将功能部署在已部署的对应加速器上。在这种情形下，一个DPU节点可以部署数个和数种加速器，同时一个加速器提供的处理能力支持多个服务功能的并行处理，形成“功能→加速器”“加速器→节点”的级联映射，这样的SFC编排部署模型与传统的“功能→节点”单级映射形成了差异。直接使用传统方法，将忽略多个SFC在部署时复用同一加速器的可能，造成SFC在资源占用、服务质量、能耗等性能指标劣化。因此，有必要构建一种能准确表征DPU的SFC部署级联映射和多维约束的模型，以支持其部署。

1 相关工作

1.1 现有工作

目前，围绕软硬件协同的SFC，已有研究工作通过可编程硬件设备卸载服务器负荷的方式加速SFC的处理[11-12]，但未考虑将可编程硬件设备本身作为SFC的运行载体。围绕SFC编排部署，Farkiani等[13]先用启发式算法寻找接近最优解集，然后利用精确算法求解其中的最优解，获得有限资源条件下SFC部署的整体利润最优策略。Huang等[14]将SFC部署问题建模为马尔可夫链，同时提出一种基于后向维特比的启发式算法求解兼顾资源消耗和服务质量的部署结果。Xiong等[15]针对子域信息不可见的环境，将SFC划分为部署到多个子域的片段，同时通过图匹配网络评估优化SFC分配策略，降低了SFC部署的资源消耗。Zhang等[16]在考虑存在虚拟网络功能依赖的条件下，提出启发式算法在多域网络部署SFC，降低了端到端通信时延和资源利用成本。上述工作均为基于纯CPU和网络功能虚拟化的SFC架构，无法直接迁移至DPU的SFC部署场景，并且当大量业务请求并发时平均响应时间较长，严重影响现实可用性。在异构计算资源SFC部署方面，Hu等[17]研究了软硬件协同异构资源的SFC部署，讨论了不同卸载比例的收益，但其工作仅围绕基于GPU的硬件加速展开。Huang等[18]研究在边缘异构设备上进行统一的SFC部署问题，但是该场景下的异构设备不涉及异构资源。

1.2 本文工作

针对上述问题，本文提出一种内嵌式DPU的

SFC 架构。该架构是将可编程硬件设备作为 SFC 运行载体的一种技术路线，其优势是通过硬件加速器的重配置和软硬件协同计算兼顾网络服务的灵活部署和高效处理。基于该架构，本文建立了内嵌式 DPU 的 SFC 部署、资源占用率、硬件加速器利用率、端到端时延和部署响应时间的模型，并提出在最小化平均资源利用率和端到端时延的同时，最大化部署成功率和硬件加速器利用率的优化目标。针对该优化问题，本文设计了一种基于遗传粒子群优化算法（particle swarm optimization & genetic algorithm，PSO-GA）的 SFC 部署策略优化算法，并引入子链划分和相似请求合并两个过程，以优化该算法的平均响应时间。仿真结果表明，本文算法相比基准算法，SFC 部署成功率提升 3.8%，平均端到端时延降低 4.96%，部署的硬件加速器平均利用率提升 9.32%，部署 SFC 产生的总功耗降低 10.47%。本文两项优化过程可以在不降低 SFC 部署成功率的同时，将 SFC 部署平均响应时间缩短 42.1%。

2　系统模型

2.1　内嵌式 DPU 的 SFC 架构

本文内嵌式 DPU 的 SFC 架构如图 1 所示，面向云–边–端网络，以网络安全服务场景的 SFC 为例，其工作过程包括感知、提取、生成、部署等步骤。当业务请求发生时，系统首先从网络环境中感知发出安全服务请求的业务场景；然后提取该业务场景的上下文，包括源设备类型、目的设备类型等足以描述场景特征的信息；接下来根据安全管理员配置的安全服务策略、可部署的安全服务功能和场景上下文，生成符合需求的安全 SFC；最后根据生成的 SFC 启动 SFC 实例，完成计算资源、存储资源和网络资源的映射，将链上的网络服务功能部署至网络环境中合适的 DPU 节点上，并引导业务流量按 SFC 次序依次通过链上网络服务功能处理，实现根据不同业务场景的安全需求提供个性化的网络安全服务。

云–边–端架构下的物理网络可以由无向图 $\mathcal{G}=(\mathcal{V},\mathcal{L})$ 表示，其中，$\mathcal{V}$ 代表设备节点，$\mathcal{L}$ 代表物理链

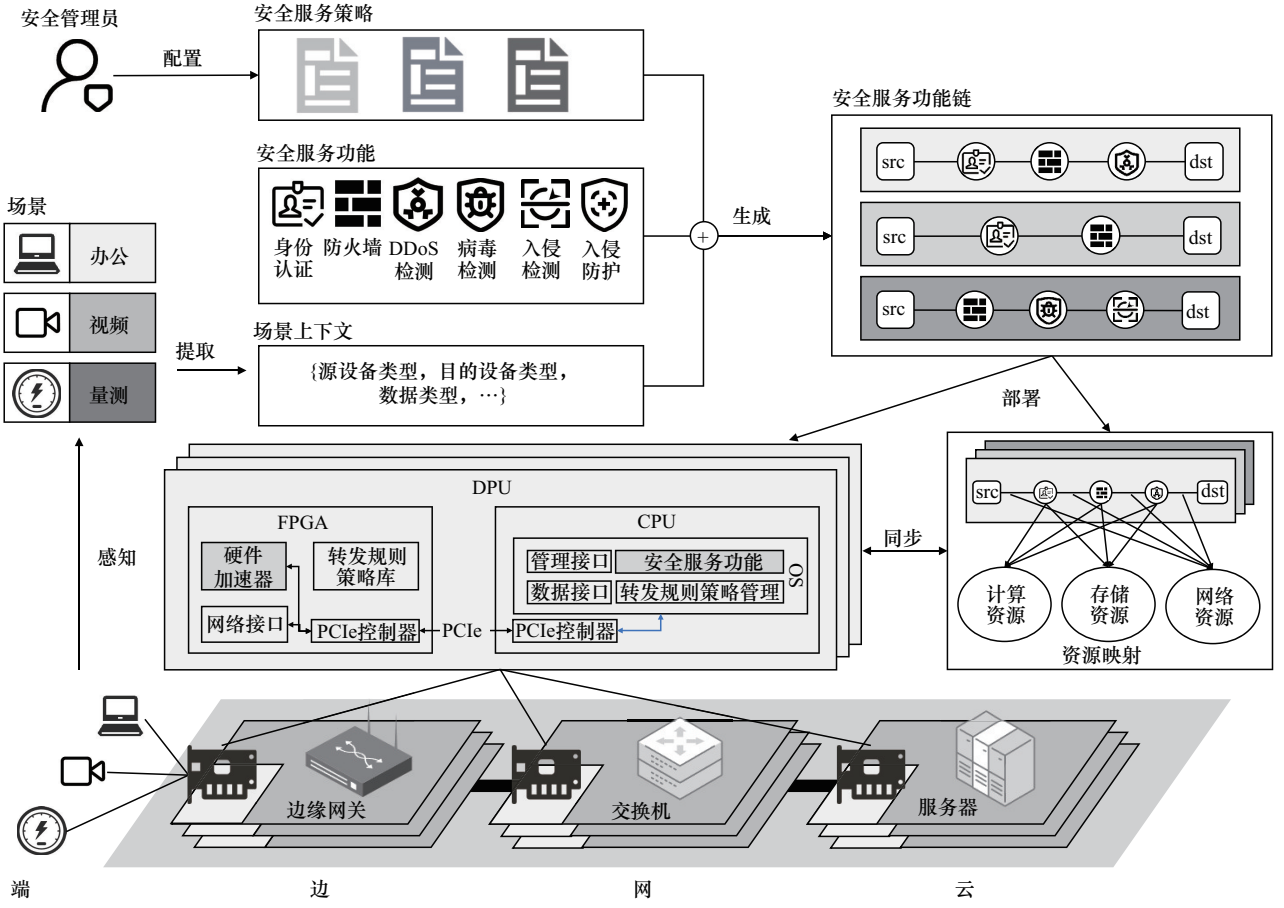


图 1　内嵌式 DPU 的 SFC 架构

路。设备节点包括两类：1)内嵌DPU的节点，这些节点可以部署并运行网络服务功能，其集合为$\mathcal{V}^{\text{dpu}}$；2)未内嵌DPU的节点，这些节点只具备流量转发能力，其集合为$\mathcal{V}^{\text{switch}}$。内嵌DPU的节点$v$包含3项属性，分别为CPU资源$\text{cpu}_v^{\text{dpu}}$、内存资源$\text{mem}_v^{\text{dpu}}$和FPGA资源$\text{fpga}_v^{\text{dpu}}$。$l_{u,v} \in \mathcal{L}$代表节点$u$和节点$v$之间的物理链路，所有物理链路的最大数据传输带宽定义为矩阵$\boldsymbol{B} = \left(b_{u,v}\right) \in \mathbb{R}^{\mathcal{V}\times\mathcal{V}}$，$t$时刻的链路时延定义为矩阵$\boldsymbol{D} = \left(d_{u,v}^t\right) \in \mathbb{R}^{\mathcal{V}\times\mathcal{V}}$，若节点$u$和节点$v$之间不存在物理链路，则$b_{u,v} = 0$，且$d_{u,v}^t = \infty$。

2.2 SFC部署

内嵌式DPU的SFC部署涉及的对象包括网络服务功能、硬件加速器、SFC请求，其部署结果包含DPU节点、网络服务功能、硬件加速器，以及虚拟链路和物理链路之间的映射关系。

网络中可编排的N种网络服务功能定义为集合$F = \left\{f_1, f_2, \cdots, f_N\right\}$。其中，网络服务功能$f_n$的硬件加速器所需的FPGA资源量为$\text{fpga}_n^{\text{acc}}$，该硬件加速器的处理能力定义为$p_n$。在网络中，已部署的硬件加速器定义为集合$\text{Acc} = \left\{a_1, a_2, \cdots, a_M\right\}, M = |\text{Acc}|$，同时定义运算$\text{TypeA}\left(a_m\right) \in F$为第$m$个硬件加速器对应的网络服务功能。

网络中动态到达的SFC请求定义为集合$R = \left\{r_1, r_2, \cdots, r_S\right\}$，令$\text{src}_s$和$\text{des}_s$分别为请求$r_s$的源节点和目的节点。请求$r_s$的有序网络服务功能表示为$\text{FN}_s = \left\{f_1^s, f_2^s, \cdots, f_I^s\right\}$，同时定义运算$\text{TypeF}\left(f_i^s\right) \in F$为获取网络功能$f_i^s$的种类。请求$r_s$的每个网络服务功能的CPU资源需求表示为$\text{CPU}_s^{\text{fn}} = \left\{\text{cpu}_{s,1}^{\text{fn}}, \text{cpu}_{s,2}^{\text{fn}}, \cdots, \text{cpu}_{s,I}^{\text{fn}}\right\}$，内存资源需求表示为$\text{MEM}_s^{\text{fn}} = \left\{\text{mem}_{s,1}^{\text{fn}}, \text{mem}_{s,2}^{\text{fn}}, \cdots, \text{mem}_{s,I}^{\text{fn}}\right\}$，FPGA资源需求表示为$\text{FPGA}_s^{\text{fn}} = \left\{\text{fpga}_{s,1}^{\text{fn}}, \text{fpga}_{s,2}^{\text{fn}}, \cdots, \text{fpga}_{s,I}^{\text{fn}}\right\}$，对硬件加速器的处理能力占用表示为$P_s^{\text{fn}} = \left\{p_{s,1}^{\text{fn}}, p_{s,2}^{\text{fn}}, \cdots, p_{s,I}^{\text{fn}}\right\}$，网络服务功能的处理时延为$\text{DL}_s = \left\{\text{dl}_{s,1}, \text{dl}_{s,2}, \cdots, \text{dl}_{s,I}\right\}$，请求的到达时刻为$t_s$，链路带宽需求为$\text{bw}_s$，请求能容忍的最高时延阈值为$\text{tl}_s$，SFC在网络中的生命周期为$\text{lf}_s$。

为表示SFC部署过程中多个对象之间的映射关系，本文分别定义以下4个二进制变量。

1) $\text{FV}_{i,v}^{s,t}$：表示网络服务功能与DPU节点之间的映射关系。当请求r_s的SFC中的网络服务功能f_i^s在时刻t被映射至节点v时，$\text{FV}_{i,v}^{s,t} = 1$，否则$\text{FV}_{i,v}^{s,t} = 0$。

2) $\text{FM}_{i,m}^{s,t}$：表示网络服务功能与硬件加速器之间的映射关系。当请求r_s的网络服务功能f_i^s在时刻t被映射至硬件加速器a_m时，$\text{FM}_{i,m}^{s,t} = 1$，否则$\text{FM}_{i,m}^{s,t} = 0$。

3) $\text{AV}_{m,v}^{s,t}$：表示硬件加速器与DPU节点之间的映射关系。当硬件加速器a_m在时刻t被映射至节点v时，$\text{AV}_{m,v}^{s,t} = 1$，否则$\text{AV}_{m,v}^{s,t} = 0$。

4) $T_{i,j,u,v}^{s,t}$：表示虚拟链路与物理链路之间的映射关系。当请求r_s的相邻服务功能f_i^s和f_j^s之间的虚拟链路在时刻t被映射至物理链路$l_{u,v}$时，$T_{i,j,u,v}^{s,t} = 1$，否则$T_{i,j,u,v}^{s,t} = 0$。

2.3 资源占用率

SFC部署的资源占用包含节点资源和链路资源两方面。在节点资源方面，考虑DPU具备的CPU、内存和FPGA资源3个方面。t时刻部署请求r_s的SFC后，DPU节点v的资源占用率为

$$U(v,t) = \omega_c \frac{\sum_{n=1}^{S}\sum_{i=1}^{|\text{FN}_s|}\text{FV}_{i,v}^{n,t}\text{cpu}_{s,n}^{\text{fn}}}{\text{cpu}_v^{\text{dpu}}} + \omega_m \frac{\sum_{n=1}^{S}\sum_{i=1}^{|\text{FN}_s|}\text{FV}_{i,v}^{n,t}\text{mem}_{s,n}^{\text{fn}}}{\text{mem}_v^{\text{dpu}}} + \omega_f \frac{\sum_{m=1}^{|\text{Acc}|}\text{AV}_{m,v}^{s,t}\text{fpga}_{\text{TypeA}\left(a_m\right)}^{\text{acc}}}{\text{fpga}_v^{\text{dpu}}} \tag{1}$$

其中，ω_c、ω_m和ω_f是处理器资源、内存资源和FPGA资源的归一化权重系数[14]，该系数应预先根据资源的重要性配置，且$\omega_c + \omega_m + \omega_f = 1$。由此可得$t$时刻所有DPU节点的平均资源占用率为

$$\varepsilon^{\mathcal{V}}(t) = \frac{\sum_{v\in\mathcal{V}^{\text{dpu}}}U(v,t)}{\left|\mathcal{V}^{\text{dpu}}\right|} \tag{2}$$

t时刻部署请求r_s后，链路$l_{u,v}$的资源占用率为

$$L(u,v,t) = \frac{\sum_{n=1}^{S}\sum_{u,v\in\mathcal{V}}\sum_{i,j\in\text{FN}_s}\text{bw}_s T_{i,j,u,v}^{s,t}}{b_{u,v}} \tag{3}$$

t时刻所有物理链路的平均资源占用率为

$$\varepsilon^{\mathcal{L}}(t)=\frac{\sum_{u,v\in\mathcal{V}}L(u,v,t)}{|\mathcal{L}|} \tag{4}$$

在t时刻部署r_s后，节点和链路的总体占用率为

$$\varepsilon(t)=\omega_{\mathcal{V}}\varepsilon^{\mathcal{V}}(t)+\omega_{\mathcal{L}}\varepsilon^{\mathcal{L}}(t) \tag{5}$$

其中，$\omega_{\mathcal{V}}$和$\omega_{\mathcal{L}}$是节点和链路资源占用率的归一化权重系数[14]，且$\omega_{\mathcal{V}}+\omega_{\mathcal{L}}=1$。

2.4　硬件加速器利用率

t时刻硬件加速器a_m的利用率为

$$A(m,t)=\frac{\sum_{n=1}^{S}\sum_{i=1}^{|\mathrm{FN}_s|}\mathrm{FM}_{i,m}^{s,t}p_{n,i}^{\mathrm{fn}}}{p_{\mathrm{TypeA}(a_m)}} \tag{6}$$

t时刻所有硬件加速器的平均利用率为

$$\eta(t)=\frac{\sum_{m=1}^{|\mathrm{Acc}|}A(m,t)}{|\mathrm{Acc}|} \tag{7}$$

2.5　端到端时延

SFC的端到端时延包含两个部分，即SFC经过的所有链路的时延和所有节点上网络服务处理的时延。因此，SFC的端到端时延可以表示为

$$\delta(r_s,t)=\sum_{u,v\in\mathcal{V}}\sum_{i,j\in\mathrm{FN}_s}d_{u,v}^{t}T_{i,j,u,v}^{s,t}+\sum_{i=1}^{I}\mathrm{dl}_{s,i} \tag{8}$$

2.6　部署响应时间

当大量服务请求并发到达时，设此时进入排队的服务请求为有序集合$Q=\{r_1,r_2,\cdots,r_K\}$，则部署r_k的真实响应时间为r_k的算法时间与r_k前所有请求的算法时间之和。设最终记录得到上述并发请求的发生时刻为t^s，算法完成时刻为$T=\{t_1^e,t_2^e,\cdots,t_K^e\}$，则$r_k$的部署响应时间为$t_k^e-t^s$。因此所有请求的平均部署响应时间为

$$\tau=\frac{\sum_{n=1}^{K}(t_n^e-t^s)}{|Q|} \tag{9}$$

2.7　优化问题描述

为了保证实际部署的可用性，内嵌式DPU的SFC部署需要满足一些约束条件，即

$$\sum_{n=1}^{s}\sum_{i=1}^{|\mathrm{FN}_s|}\mathrm{FV}_{i,v}^{s,t}\mathrm{cpu}_{s,n}^{\mathrm{fn}}\leqslant\mathrm{cpu}_{v}^{\mathrm{dpu}},\forall v\in\mathcal{V}^{\mathrm{dpu}} \tag{10}$$

$$\sum_{n=1}^{s}\sum_{i=1}^{|\mathrm{FN}_s|}\mathrm{FV}_{i,v}^{s,t}\mathrm{mem}_{s,n}^{\mathrm{fn}}\leqslant\mathrm{mem}_{v}^{\mathrm{dpu}},\forall v\in\mathcal{V}^{\mathrm{dpu}} \tag{11}$$

$$\sum_{m=1}^{|\mathrm{Acc}|}\mathrm{AV}_{m,v}^{s,t}\mathrm{fpga}_{\mathrm{TypeA}(a_m)}^{\mathrm{acc}}\leqslant\mathrm{fpga}_{v}^{\mathrm{dpu}},\forall v\in\mathcal{V}^{\mathrm{dpu}} \tag{12}$$

$$\sum_{n=1}^{s}\sum_{u,v\in\mathcal{V}}\sum_{i,j\in\mathrm{FN}_s}\mathrm{bw}_s T_{i,j,u,v}^{s,t}\leqslant b_{u,v},\forall u,v\in\mathcal{V} \tag{13}$$

$$\sum_{u,v\in\mathcal{V}}T_{i,j,u,v}^{s,t}-\sum_{u,v\in\mathcal{V}}T_{i,j,v,u}^{s,t}=\begin{cases}-1, i\in\mathrm{des}_s\\ 1, i\in\mathrm{src}_s\\ 0,\text{其他}\end{cases} \tag{14}$$

$$\sum_{s=1}^{|\mathrm{SFC}|}\sum_{i=1}^{|vnf_s|}\mathrm{FM}_{i,m}^{s,t}p_s^i\leqslant p_m,\forall m\in[1,|\mathrm{Acc}|] \tag{15}$$

其中，式(10)、式(11)和式(12)分别表示节点的CPU、内存和FPGA资源占用量必须低于节点所拥有的资源量；式(13)表示链路带宽的占用量必须低于链路的最大带宽；在流量守恒约束方面，式(14)表示每个节点进入的流量和流出的流量数量相等，除非该节点是SFC的起始节点或目的节点；在硬件加速器的利用率方面，式(15)表示每个硬件加速器被利用的处理能力不能超出其具备的处理能力。

为保障服务质量，SFC的时延必须小于时延阈值tl_s，所以服务质量约束为

$$\delta(r_s)\leqslant\mathrm{tl}_s \tag{16}$$

本文的优化目标是找到最优的SFC部署决策，在最小化平均资源利用率和端到端时延的同时，最大化硬件加速器利用率，其数学表达式为

$$\begin{aligned}&\min\{\omega_d\delta(r_s,t)+\omega_l\varepsilon(t)-\omega_a\eta(t)\}\\&\text{s.t. 式(10)~式(16)}\\&\quad\mathrm{FV}_{i,v}^{s,t},\mathrm{FM}_{i,m}^{s,t},\mathrm{AV}_{m,v}^{s,t},T_{i,j,u,v}^{s,t}\in\{0,1\},\forall m,i,j,u,v\end{aligned} \tag{17}$$

其中，ω_d、ω_l、ω_a为权重。

2.8　问题复杂度分析

由式(17)定义的内嵌式DPU的SFC部署问题是NP困难问题，该问题可以被简化为一种被称为“多准则的装箱问题”的变体[16]，其中每个待部署的网络功能可视为待装载的物品，DPU节点则可被视为箱子。若仅考虑式(10)~式(14)、式(16)约束，部署问题可以被视为经典装箱问题的简化形式。同时，内嵌式DPU的SFC部署问题还考虑了DPU内FPGA实现的硬件加速器对SFC部署的影响，即式(15)，其形成的分团问题是内嵌式DPU的SFC部署问题的子集，因此附加了式(15)约束的部署问题比装箱问题更复杂，据此可以推断，内嵌式

DPU的SFC部署问题同样是NP困难问题。

3　基于遗传粒子群的SFC优化部署方法

粒子群优化算法作为一种基于群智能理论的优化算法，具有运算速度快、全局搜索能力好等特点，非常适合在动态、多目标场景下寻优[19]。但是传统的粒子群优化算法一般应用于连续问题，难以直接应用于离散问题。本文采用遗传粒子群混合算法，用遗传算法的交叉和变异操作替换粒子群优化算法中的位置更新，用交叉和变异发生的概率替换粒子群优化算法中的速度。基于遗传粒子群的SFC优化部署流程如图2所示。

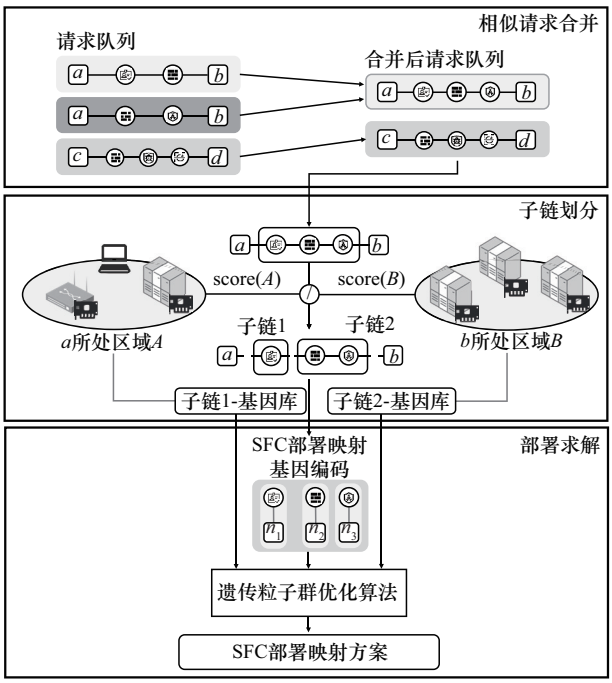


图2　基于遗传粒子群的SFC优化部署流程

流程分为3个主要阶段：第一阶段，将时间和空间维度上相似的多个SFC请求合并为同一请求，减少请求队列中待部署SFC请求的数量，减少部署求解次数，加快计算速度；第二阶段，遍历合并后的请求队列，将处理每个请求时的SFC部署问题按区域分割为较小规模的子问题，压缩动作空间，加快计算效率和收敛速度，形成划分后的两段子链及其对应的基因库；第三阶段，根据两段子链对应的基因库，编码生成一批代表部署映射策略的粒子，通过遗传粒子群优化算法迭代更新粒子位置，探索SFC部署动作空间，求得最优SFC部署映射方案。

3.1　遗传粒子群优化算法

本文将SFC上的有序服务功能部署的物理节点的索引序列视作粒子的当前位置，即遗传算法中的基因，设定SFC优化部署算法的变异操作和交叉操作的具体内容如下。

1) 变异操作：选取索引序列中的第i个索引，将其替换为候选区域内除自身外的随机DPU节点。

2) 交叉操作：随机选取索引序列中第i至j个索引形成子序列，将其与另一个序列的相同位置的子序列互换。

为表达离散问题下粒子的搜索速度，规定以下粒子属性，用以描述粒子自身的探索方向，维持探索方向的程度，以及学习自身极值和全局极值位置的程度。

1) 变异系数c_0、自身极值交叉系数c_1、全局极值交叉系数c_2分别代表粒子迭代更新自身位置时执行变异、与自身极值交叉、与全局极值交叉3种操作的发生概率，且$c_0+c_1+c_2=1$，$0\leqslant c_0,c_1,c_2\leqslant 1$。

2) 变异方向$\boldsymbol{v}$是一个每个维度均为正数的单位向量$\{v_1,v_2,\cdots,v_n\}$，其中维数n为索引序列的长度，即SFC的长度，v_i代表变异操作选取第i个索引执行替换的概率，且$v_1^2+v_2^2+\cdots+v_n^2=1$。

3) 极值学习系数wl表示执行交叉操作时，选取索引子序列的长度。交叉操作时选取的子序列越长，粒子更新位置时向自身极值或全局极值靠拢的速度越快。

为避免遗传粒子群优化算法“早熟收敛”，本文采用自适应的变异/交叉系数和学习系数调节策略，使算法在初期进行高效全局搜索，在中期专注于局部搜索，在末期确保收敛。变异和交叉系数的计算式为

$$\begin{cases} c_0(t)=1-\left(\dfrac{t}{T}\right)^p \\ c_1(t)=\left(\dfrac{t}{T}\right)^p\left[1-\left(\dfrac{t}{T}\right)^q\right] \\ c_2(t)=\left(\dfrac{t}{T}\right)^p\left(\dfrac{t}{T}\right)^q \end{cases} \tag{18}$$

其中，T为总迭代次数；t为当前迭代次数；p、q为幂指数，分别控制变异概率和自身极值交叉概率的下降速度。基于改进遗传粒子群的SFC优化部署算法的步骤如下所示。

步骤 1　参数设置和粒子初始化。设置粒子数num，最大迭代次数it，停止窗口W，概率系数调节参数p、q；随机产生N个SFC长度的索引序列作为初始粒子$X_i^0(i=1,2,\cdots,N)$；为每个粒子随机生成其变异方向$\boldsymbol{v}$、学习系数wl。

步骤 2　节点间链路映射。利用Dijkstra算法计算获得每个粒子的前后索引节点之间的最短路由路径作为链路映射的结果。

步骤 3　适应度函数值计算。评价当前每个粒子所代表映射方案的适应度值，即SFC部署的优化目标函数。

步骤 4　个体极值和群体极值更新。更新当前个体的极值$\text{Pbest}_i(i=1,2,\cdots,N)$、当前个体的最好位置$\text{Xbest}_i$、全局极值Gbest和全局最好位置Xbest。

步骤 5　粒子群位置更新。通过公式计算变异和交叉的概率系数，按照概率执行变异或交叉操作，更新每个微粒的位置，同时重新计算更新后的节点间链路映射结果。

步骤 6　未达到结束条件，转到步骤3。

步骤 7　达到结束条件，输出全局极值Gbest和全局最好位置Xbest。

当$t\to T$时，根据式(18)可知$c_0(t)\to 0$，$c_1(t)\to 0$，$c_2(t)\to 1$，因此可认为本文算法在迭代计算的末期更新粒子位置时只执行与全局极值的交叉操作。此时将粒子当前位置向量与全局最优位置向量视为等长度字符串并计算汉明距离$L^h(t)$，则有$0\leqslant L^h(t+1)\leqslant L^h(t)$。因此当$T$足够大时，所有粒子的位置向量与全局最优位置的汉明距离将减少为0，本文遗传粒子群优化算法必然收敛。但是根据粒子群优化算法的特性[20]，其收敛于局部与全局最优的迭代上界与粒子数量和粒子速度相关。3.2节的实验结果展现了在本文场景下算法参数对算法收敛性的影响。

本文遗传粒子群优化算法的复杂度受DPU节点数量$|\mathcal{V}|$、链路数量$|\mathcal{L}|$、可部署服务功能数量$|F|$、SFC长度$|D|$、粒子数量$|N|$和最大迭代次数$|T|$影响。步骤1时间复杂度为$O(|D|\times|N|)$，步骤2时间复杂度为$O(|\mathcal{V}|^2)+O(|D|\times(|N|-1))$，步骤3时间复杂度为$O(|N|\times|D|\times(|\mathcal{V}|+|\mathcal{L}|+|F|+|\mathcal{L}|))$，步骤4时间复杂度为$O(N)$，步骤5时间复杂度为$O(|N|\times|D|)$，单轮迭代的时间复杂度为上述复杂度之和$O(|N|\times|D|\times(|\mathcal{V}|+|\mathcal{L}|+|F|))$，则算法的总体时间复杂度为$O(|T|\times|N|\times|D|\times(|\mathcal{V}|+|\mathcal{L}|+|F|))$。算法包含的主要数据结构及其空间占用为粒子的位置矩阵（$O(|N|\times|D|)$）、粒子的速度矩阵（$O(|N|\times|D|)$）、个体最优位置矩阵（$O(|N|\times|D|)$）、个体最优适应度数组（$O(|N|)$）、全局最优位置向量（$O(|D|)$）、全局最优适应度（$O(1)$）、节点资源状态数组（$O(|\mathcal{V}|)$）和链路资源状态数组（$O(|\mathcal{L}|)$），将上述空间复杂度求和，可得总体空间复杂度为$O(|N|\times|D|)$。

3.2　子链划分

在云–边–端架构网络中，云网络、边缘网络中的设备物理位置相对集中，天然形成内部链路时延较低的云网络和边缘网络区域；跨云、跨边、云边之间的网络链路则由于物理距离的增加而时延较高。因此，为降低SFC的端到端时延，应尽量选择网络服务请求源节点和目的节点所处网络区域内的节点部署网络服务功能，且避免多次经过跨网络区域的高时延链路。基于上述认识，本文提出子链划分方法，先确定SFC将被部署的网络区域，然后根据区域内节点剩余资源和链路剩余资源的综合评分及SFC的资源需求，将SFC划分为映射到不同区域的子链，从而限制每个子链上服务功能所能映射的物理节点的范围，排除次优动作空间，提升遗传粒子群优化算法求解的速度。

云网络或边缘网络等网络区域可以表示为云–边–端整体网络拓扑的子图$\mathcal{G}^{\text{sub}}=\{V^{\text{sub}},L^{\text{sub}}\}\subset\mathcal{G}$，$t$时刻节点$v$的剩余资源加权和为

$$\begin{aligned}R(v,t)=\omega_c\Big(\text{cpu}_v^{\text{dpu}}-\sum_{n=1}^{S}\sum_{i=1}^{|\text{FN}_s|}\text{FV}_{i,v}^{n,t}\text{cpu}_{s,n}^{\text{fn}}\Big)+\\ \omega_m\Big(\text{mem}_v^{\text{dpu}}-\sum_{n=1}^{S}\sum_{i=1}^{|\text{FN}_s|}\text{FV}_{i,v}^{n,t}\text{mem}_{s,n}^{\text{fn}}\Big)+\\ \omega_f\Big(\text{fpga}_v^{\text{dpu}}-\sum_{m=1}^{|\text{Acc}|}\text{AV}_{m,v}^{s,t}\text{fpga}_{\text{TypeA}(a_m)}^{\text{acc}}\Big)\end{aligned}\tag{19}$$

网络区域内的总剩余资源加权和为

$$\text{SV}(t)=\sum_{v\in V^{\text{sub}}}R(v,t)\tag{20}$$

t时刻网络区域内链路的剩余资源为

$$E(u,v,t)=b_{u,v}-\sum_{n=1}^{S}\sum_{u,v\in\mathcal{V}}\sum_{i,j\in \mathrm{FN}_s}\mathrm{bw}_s T_{i,j,u,v}^{s,t} \tag{21}$$

网络区域内链路的平均资源剩余为

$$\mathrm{SL}(t)=\frac{\sum_{u\in V^{\mathrm{sub}}}E(u,v,t)}{\left|L^{\mathrm{sub}}\right|} \tag{22}$$

定义二进制变量 $F_{i,m}^{s}$，当 $\mathrm{TypeAcc}(a_m)=\mathrm{TypeF}(f_i^s)$ 时，$F_{i,m}^{s}=1$，否则 $F_{i,m}^{s}=0$。网络区域内硬件加速器的资源剩余为

$$\mathrm{SA}(t)=\sum_{v\in V^{\mathrm{sub}}}\sum_{m=1}^{|\mathrm{Acc}|}\sum_{i=1}^{I}\mathrm{AV}_{m,v}^{s,t}F_{i,m}^{s}\left(p_m-\sum_{s=1}^{S}\sum_{i=1}^{|\mathrm{fn}_s|}\mathrm{FM}_{i,m}^{s,t}\mathrm{bw}_s\right) \tag{23}$$

由此，本文定义网络区域的剩余资源加权综合评分为

$$\mathrm{Score}\left(\mathcal{G}^{\mathrm{sub}}\right)=s^{V}\mathrm{SV}(t)+s^{L}\mathrm{SL}(t)+s^{A}\mathrm{SA}(t) \tag{24}$$

其中，s^V、s^L 和 s^A 分别为节点、链路和硬件加速器资源的评分加权系数，可根据需求调整。同时，定义 r_s 所请求 SFC 的资源需求加权评分为

$$G(r_s)=\omega_V\left(\omega_c\sum_{i=1}^{I}\mathrm{cpu}_{s,i}^{\mathrm{fn}}+\omega_m\sum_{i=1}^{I}\mathrm{mem}_{s,i}^{\mathrm{fn}}+\omega_f\sum_{i=1}^{I}\mathrm{fpga}_{s,i}^{\mathrm{fn}}\right)+\omega_L\mathrm{bw}_s \tag{25}$$

子链划分的过程如算法 1 所示。

算法 1 子链划分算法

输入 长度为 n 的 SFC 请求 $r=\{f_1,f_2,\cdots,f_n\}$，该请求的源节点网络区域 $\mathcal{G}_1^{\mathrm{sub}}$ 和目的节点网络区域 $\mathcal{G}_2^{\mathrm{sub}}$ 及其节点和链路资源数据，网络中已部署的 SFC 集合 s

输出 该请求 SFC 的子链 r_1 和 r_2

1) 根据式(24)计算 $\mathcal{G}_1^{\mathrm{sub}}$ 和 $\mathcal{G}_2^{\mathrm{sub}}$ 的剩余资源加权综合评分 $\mathrm{Score}\left(\mathcal{G}_1^{\mathrm{sub}}\right)$ 和 $\mathrm{Score}\left(\mathcal{G}_2^{\mathrm{sub}}\right)$
2) $r_1\leftarrow\varnothing$，$r_2\leftarrow r$，$l\leftarrow\infty$
3) for $i=1,2,\cdots,n$ do
4) 根据式(25)分别计算 r_1 和 r_2 的资源需求加权评分 $G(r_1)$ 和 $G(r_2)$
5) if $\left|\frac{\mathrm{Score}\left(\mathcal{G}_1^{\mathrm{sub}}\right)}{\mathrm{Score}\left(\mathcal{G}_2^{\mathrm{sub}}\right)}-\frac{G(r_1)}{G(r_2)}\right|<l$ then
6) $l\leftarrow\left|\frac{\mathrm{Score}\left(\mathcal{G}_1^{\mathrm{sub}}\right)}{\mathrm{Score}\left(\mathcal{G}_2^{\mathrm{sub}}\right)}-\frac{G(r_1)}{G(r_2)}\right|$
7) else
8) exit
9) end if
10) r_1 添加 f_i，r_2 删除 f_i
11) end for

3.3 相似请求合并

云–边–端网络架构多为工业或能源互联网采用，在工业或能源互联网中极易出现周期性并发相似业务的情景，例如新能源消纳中边缘物联代理设备周期性采集太阳能终端的负荷数据[21]。这些周期性并发的相似业务产生的 SFC 需求在发生时间、源/目的节点位置和所需功能等维度上相近，可将它们合并为一个 SFC 请求并由一次遗传粒子群混合算法求解部署策略具备可行性。因此，本文提出相似请求合并方法，根据部署区域内资源的负载情况将当前待部署请求队列中的多个相似请求合并为同一请求进行处理，减少 SFC 部署求解次数，缩短 SFC 请求并发到达时每个请求在队列中的排队时间，从而缩短 SFC 部署的平均响应时间。相似请求合并算法如算法 2 所示。

算法 2 相似请求合并算法

输入 节点资源值 R^{cpu}，R^{mem}，R^{fpga}，合并阈值 w^{com}，相似 SFC 请求列表 $S=[s_1,s_2,\cdots,s_N]$

输出 合并后的 SFC 请求子集的列表 $S^{\mathrm{result}}=[S_1^{\mathrm{com}},S_2^{\mathrm{com}},\cdots,S_I^{\mathrm{com}}]$

1)while $S\neq\varnothing$ do
2) $s^{\mathrm{com}}\leftarrow s_1$，$S$ 移除第一个元素
3) for i=1,2,⋯,|S| do
4) if $S=\varnothing$ then exit
5) if s_i 与 s^{com} 的起始节点区域不同或终末节点区域不同 then continue
6) $f\leftarrow \mathrm{FN}_{s_i}\cap\mathrm{FN}_{s^{\mathrm{com}}}$
7) if $f=\varnothing$ or f 所包含功能在 s_i 和 s^{com} 中的先后顺序不一致 then continue
8) 计算 s^{com} 和 s_i 合并后所需的每一种网络服务功能的资源占用 $\{r_1^{\mathrm{cpu}},r_2^{\mathrm{cpu}},\cdots,r_P^{\mathrm{cpu}}\}$，$\{r_1^{\mathrm{mem}},r_2^{\mathrm{mem}},\cdots,r_P^{\mathrm{mem}}\}$，$\{r_1^{\mathrm{fpga}},r_2^{\mathrm{fpga}},\cdots,r_P^{\mathrm{fpga}}\}$
9) if $\forall 1\leqslant p\leqslant P, r_p^{\mathrm{cpu}}\leqslant R^{\mathrm{cpu}}w^{\mathrm{com}}$ 且 $r_p^{\mathrm{mem}}\leqslant R^{\mathrm{mem}}w^{\mathrm{com}}$ 且 $r_p^{\mathrm{fpga}}\leqslant R^{\mathrm{fpga}}w^{\mathrm{com}}$ then
10) 将 s_i 合并至 s^{com}，S 移除 s_i
11) end if

12)　end for

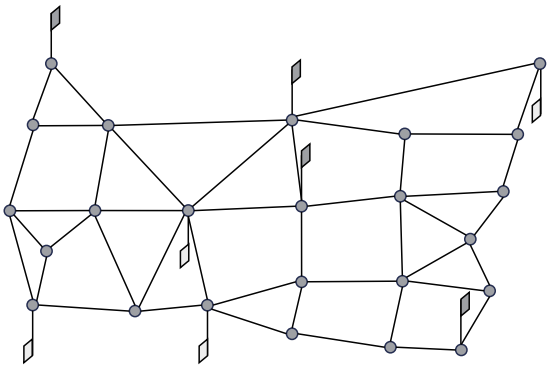
13)end while

合并阈值 w^{com} 由用户根据需求设置，且 $0 < w^{com} \leqslant 1$。当 w^{com} 较大时，合并后的SFC请求数量较少，部署求解次数少，但更容易出现资源碎片，使后续请求SFC部署的失败概率升高。当 w^{com} 较小时，合并后的请求数量较多，但是不易出现资源碎片。

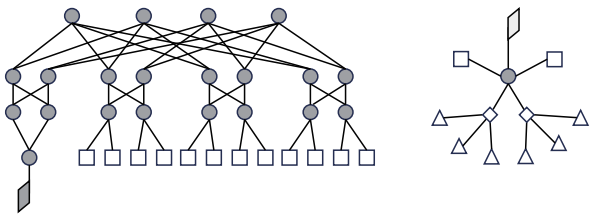
4　仿真分析

4.1　实验设置

本文仿真实验的网络环境以US-NET[22]拓扑为骨干网络，并在该拓扑的基础上扩展云网络和边缘网络部分，从而模拟完整的云–边–端网络环境。云–边–端网络仿真拓扑如图3所示，其中，圆形代表交换机，白色正方形代表云服务器或边缘服务器，菱形代表边缘网关，三角形代表终端，浅色和深色旗帜分别代表云网络、边缘网络与骨干网络的连接位置。



(a) US-NET骨干网络拓扑及云网络、边缘网络的接入位置



(b) 云网络拓扑、边缘网络拓扑及其接入骨干网络的位置

图3　云–边–端网络仿真拓扑

对于云网络拓扑部分，本文选择4-port胖树拓扑作为其网络结构，其中一个Pod作为云网络的进出口[23]。对于边缘网络拓扑部分，本文选择星形拓扑作为其网络结构，其中数个边缘服务器和设备接入网关与一个边缘核心交换机相连，每个设备接入网关又与数个终端设备相连。本文在US-NET骨干网络的基础上添加了4个云网络和4个边缘网络，其中，4个边缘网络分别具备4、6、8、10个终端设备，1、1、2、2个边缘网关，0、1、2、3个边缘服务器。上述云网络和边缘网络接入骨干网络的节点为随机选取。最终网络拓扑中包含112个未内嵌DPU的节点和60个内嵌DPU的节点，只具备流量转发能力的节点占比为65.1%。

在上述网络拓扑中，骨干网络物理链路的时延根据其交换机节点之间的距离计算得到，骨干网络物理链路的带宽设置为25~40 Gbit/s。云网络物理链路的时延遵循2~5 ms的均匀分布随机生成，云网络物理链路的带宽设置为10~25 Gbit/s。边缘网络物理链路的时延遵循2~10 ms的均匀分布随机生成，边缘网络物理链路的带宽设置为10 Gbit/s [13]。本文假设内嵌式DPU只能部署于云服务器、边缘服务器和边缘网关，因此将网络拓扑中上述节点的处理器资源设置为6核，内存资源设置为8 GB，FPGA逻辑资源量设置为500 000单位[24]。

本文设置的6种网络中可使用的网络服务功能资源需求实验参数设置如表1[25]所示。在SFC请求数据集方面，本文分别为两种典型场景生成SFC请求数据集。针对静态压力场景SFC请求数据集，在同一时刻随机生成100个SFC请求。针对动态仿真场景SFC请求数据集，在0~120 000 ms时刻内随机生成200个SFC请求。上述数据集中网络服务请求的源节点和目的节点均从云服务器、边缘服务器和终端设备中随机选取，每个SFC上的有序网络功能不重复随机选取，且每个网络功能的处理器需求、内存需求、硬件加速器的FPGA资源需求和硬件加速器的处理能力占用按表1所示的比例随机生成，每个请求的最大容忍时延阈值设置为始末节点间预测的最短链路的传输时延随机增加30~100 ms，流量的带宽需求设置为100~500 Mbit/s。除此以外，本文在生成SFC请求数据集时，会按照0、25%和50%的比例生成并发的SFC请求，用以模拟不同并发请求比例的情况。

在算法参数设置方面，本文选取不同参数组合进行多次实验，选择出表现最优的参数组合用于对比实验，遗传粒子群混合算法的仿真实验参数设置如表2所示。

表1 网络服务功能资源需求实验参数设置

网络服务功能	CPU消耗/核	内存消耗/GB	吞吐量/(Mbit·s^{-1})	逻辑资源消耗	处理能力/(Mbit·s^{-1})	处理时延/μs
防火墙	1	2	1 000	50 000	1 000	120
接入认证	2	4	1 000	75 000	600	160
DDoS检测	2	3	500	150 000	500	83
病毒检测	1	4	1 000	100 000	1 000	200
IDS	2	4	500	125 000	800	648
IPS	3	4	500	200 000	700	200

表2 仿真实验参数设置

参数	值
相似请求合并阈值w^{com}	0.5
资源评分系数s^V, s^L, s^A	0.75, 0.2, 0.05
最大迭代次数it	100
停止窗口W	20
粒子数num	100
极值学习系数wl	1
节点资源占用率权重系数$\omega_c,\omega_m,\omega_f$	0.3, 0.3, 0.4
总体资源占用率权重系数ω_V,ω_L	0.5, 0.5
节点每单位CPU核、内存功耗系数$P_{\mathrm{cpu}},P_{\mathrm{mem}}$	6, 1
加速器每单位处理带宽和待机功耗系数$P_{\mathrm{acc}},P_{\mathrm{base}}$	0.01, 3
链路每单位带宽占用功耗系数P_{band}	0.01
优化目标函数权重系数$\omega_d,\omega_l,\omega_a$	1, 40 000, 100

本文实验设置两类典型场景：1)静态压力场景，所有SFC请求在同一时刻到达，主要验证SFC部署算法在资源有限且SFC请求压力较高的情况下的性能；2)动态仿真场景，所有SFC请求在不同时刻按顺序到达，并且随机生成SFC的生命周期，在每个部署的SFC结束生命周期之前，其存在于网络拓扑中并占用相应的资源，只有当SFC结束生命周期，才离开网络并释放相应资源。

在静态压力场景下，本文选择4种对比算法。

1) 随机算法（Random）。首先获取SFC请求的源节点和目的节点所处的网络区域，然后将所请求的SFC上的每个功能随机部署至上述网络区域的DPU节点中。

2) 贪婪算法（Greedy）。首先利用Dijkstra算法获取SFC请求的源节点到目的节点的最短路径，然后在该路径的节点中枚举所有可行解，挑选端到端时延最小的解。

3) 资源与服务质量感知的部署算法（RQAP）[14]。首先，排序所有SFC请求，并将其与其他SFC的源节点、目的节点和链上功能相似性高的SFC部署，方便后续SFC部署在相同功能的位置。然后，根据资源情况为每个SFC构建马尔可夫链状态转移概率矩阵，通过反向维特比启发式算法逐步确定每个功能的部署位置。最后，为每两个功能间的虚拟链路映射物理链路。

4) 软件依赖感知的部署算法（SD-VDSM）[16]。首先，根据节点的拓扑复制生成SFC的待部署功能数量层的图并连接。然后，通过层次分析法计算CPU资源、内存资源、硬盘资源、软件依赖、处理时延、带宽资源、链路时延的加权和，为多层图赋值。最后，根据加权和迭代确定功能的放置节点和通信链路。由于本文场景不考虑软件依赖，本文在对比实验中将该算法软件依赖部分修改为服务功能与其对应硬件加速器的依赖。

所有对比算法的运行环境均为配备了AMD 9700X处理器和48 GB内存的计算机，RQAP和SD-VDSM算法都针对本文场景进行了调优，选择其最优实验结果进行对比。

本文在静态压力场景下对比以下5项指标。

1) 部署成功率。该指标用于对比不同算法在同一时刻向网络中部署越来越多SFC时的成功率。在相同数据集下，处理相同SFC请求数量时该指标越高，则表明算法越能够为更多的SFC寻求可行的部署策略。计算方法为

$$\text{部署成功率}=\frac{\text{当前成功部署SFC数量}}{\text{当前处理的SFC请求数量}}$$

2) 平均端到端时延。该指标用于对比不同算法成功部署的SFC的平均端到端时延。在相同数据集下，若部署的SFC的平均端到端时延越低，则表

明算法越能为SFC寻求低时延的部署策略。计算方法参考式(8)。

3) 资源占用率。该指标用于对比不同算法部署大量SFC时所使用网络内的CPU、内存和FPGA资源的加权占用率。在相同数据集下，若处理相同SFC请求数量时该指标越低，则表明算法越能寻求资源占用低的部署策略。计算方法参考式(2)。

4) 硬件加速器利用率。该指标用于对比不同算法部署SFC时，当前网络中部署的硬件加速器的利用情况。若处理相同SFC请求数量时该指标越高，则表明算法越能寻求硬件加速器利用率高的部署策略。计算方法参考式(7)。

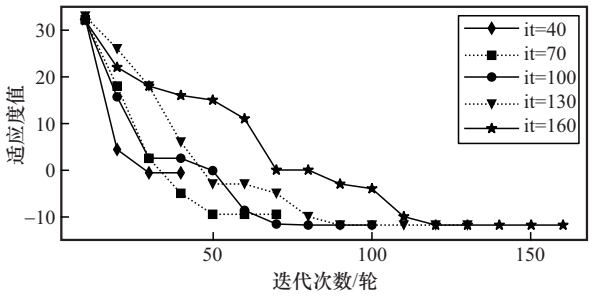
5) 总功耗。该指标用于对比不同算法部署大量SFC时，当前网络为支持所部署的SFC而发生的总功耗。若处理相同SFC请求数量时该指标越低，则表明算法越能寻求功耗发生低的部署策略，更加节能，服务成本更低。本文设计的总功耗指标包含节点功耗和链路功耗，节点功耗则包含CPU功耗、内存功耗和硬件加速器功耗[26]。忽略CPU、内存和通信链路的待机功耗，则CPU功耗为 $W_{\text{cpu}} = P_{\text{cpu}}x_{\text{cpu}}$，$W_{\text{mem}} = P_{\text{mem}}x_{\text{mem}}$，$W_l = P_{\text{band}}x_l$，其中，$P_{\text{cpu}}$、$P_{\text{mem}}$和$P_{\text{band}}$为每单位CPU、内存和带宽的功耗系数，$x_{\text{cpu}}$、$x_{\text{mem}}$和$x_l$为节点已使用的CPU、内存数量和链路已占用的带宽。在硬件加速器功耗部分，由于FPGA的功耗特性[27]，当FPGA不烧录处理逻辑时，其功耗为零；当FPGA烧录硬件加速器后，因为一定比例的电路会跟随时钟进行电平翻转，硬件加速器天然会消耗一定的功率，同时会根据处理的带宽增加而提升工作频率，进而提升功耗，因此硬件加速器功耗表示为 $W_{\text{acc}} = P_{\text{acc}}x_{\text{acc}} + P_{\text{base}}$，其中，$P_{\text{acc}}$为每单位处理带宽的功耗系数，$P_{\text{base}}$为基础功耗，$x_{\text{acc}}$为硬件加速器已占用的处理带宽。总功耗指标等于所有DPU节点的CPU功耗、内存功耗、网络中所有已部署硬件加速器的功耗及所有链路功耗之和。所有功耗系数的取值如表2所示。

在动态仿真场景下，本文进行消融实验验证两种优化方法对基于遗传粒子群的SFC部署算法性能的影响，对比算法包括基础遗传粒子群混合算法（PSO-GA-base）、子链划分增强的遗传粒子群混合算法（PSO-GA-sub）、相似请求合并增强的遗传粒子群混合算法（PSO-GA-com）和采用两项增强的遗传粒子群混合算法（PSO-GA-all）。设置SFC长度为5并选择不同比例的并发请求数据集，对比SFC部署的成功率、SFC端到端时延和每个请求的平均部署响应时间。

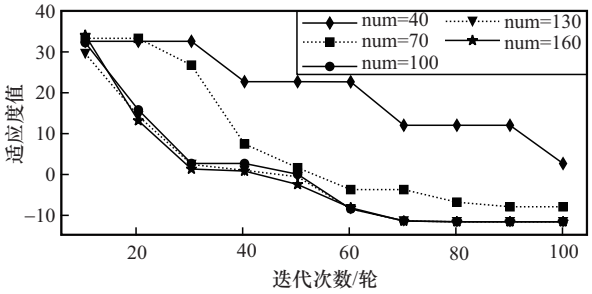
由于本文遗传粒子群混合算法和SFC请求数据集生成都包含随机过程，以下每轮实验的结果均采用相同参数设置生成5个SFC请求数据集并重复进行5次实验后取平均值，以避免随机误差。

4.2　实验结果与分析

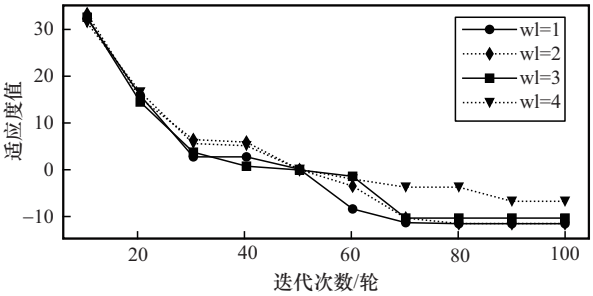
为探究PSO-GA参数对算法收敛性能的影响，使用不同参数组合进行实验，PSO-GA的收敛性能如图4所示。每轮对比实验仅调整一项参数值，其他参数按表2设置。



(a) 不同最大迭代次数时适应度值随迭代次数的变化



(b) 不同粒子数量时适应度值随迭代次数的变化



(c) 不同学习系数时适应度值随迭代次数的变化

图4　不同参数组合下PSO-GA的收敛性能

由图4(a)可知，当最大迭代次数过少时，全局搜索阶段过短，PSO-GA无法充分探索动作空间，快速收敛于局部极值或非极值；当最大迭代次数过大时，算法能够进行充分探索，但是进入全局收敛

阶段的速度较慢。由图4(b)可知，当粒子数量过少时，算法无法充分探索动作空间，最终收敛于局部极值或非极值；当粒子数量过多时，算法可以收敛于全局极值，但是粒子数的增加在收敛效果上存在边际效应，这是因为会出现大量相同的粒子，造成计算的浪费。由图4(c)可知，当极值学习系数过大时，算法在粒子探索局部时速度过快，容易飞跃局部极值。因此，本文最大迭代次数和粒子数量应根据问题规模和实际运行情况取合适的值，极值学习系数需要设定较小的值，从而使算法在保证收敛性的同时具有较快的收敛速度。

将本文算法与前述4种算法进行对比，在SFC长度固定为5的条件下，对SFC部署的成功率、平均端到端时延、资源占用率、硬件加速器利用率和总功耗进行测量统计，实验结果如图5~图9所示。

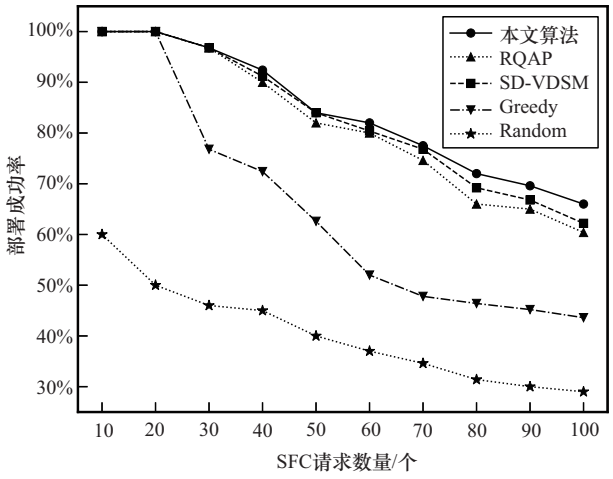


图5　5种算法的部署成功率随SFC请求数量增多的变化

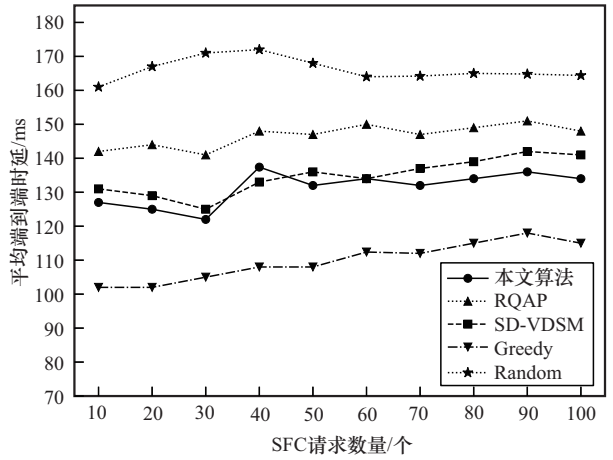


图6　5种算法的平均端到端时延随SFC请求数量增多的变化

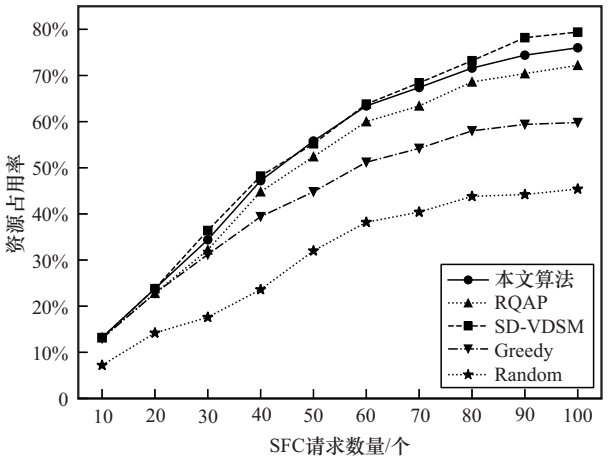


图7　5种算法的资源占用率随SFC请求数量增多的变化

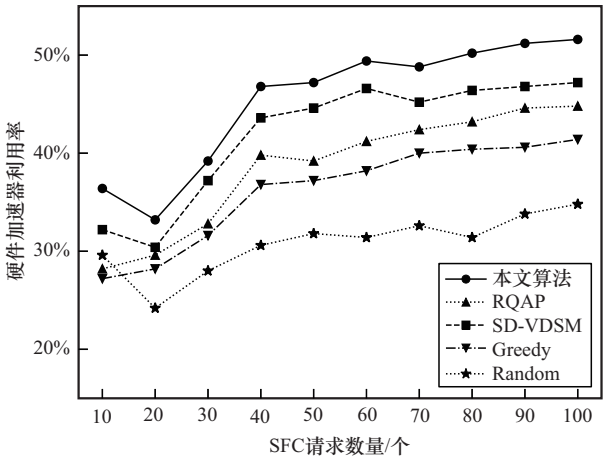


图8　5种算法的硬件加速器利用率随SFC请求数量增多的变化

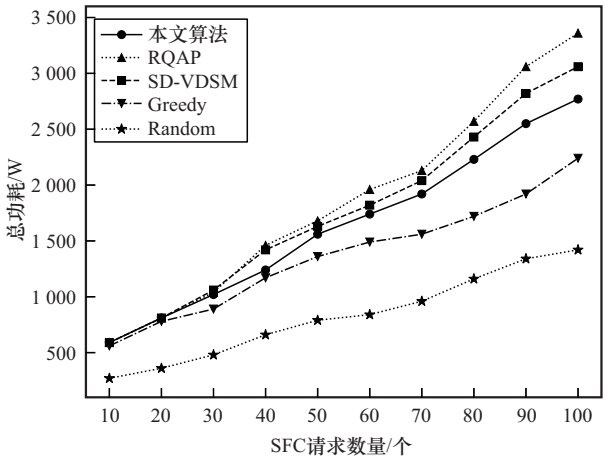


图9　5种算法的总功耗随SFC请求数量增多的变化

由图5~图9可知，Random的性能最差，其部署成功率未超过60%。虽然其资源占用率、硬件加速器利用率和总功耗最低，但这是因为成功部署的SFC数量过少，无法说明Random在这些指

标上具有优势。此外，Random 的平均端到端时延也远超过其他算法。Random 的实验结果可以初步说明本文内嵌 DPU 的 SFC 部署存在问题，需要按照特定的规则生成部署策略，才能满足部署需求。

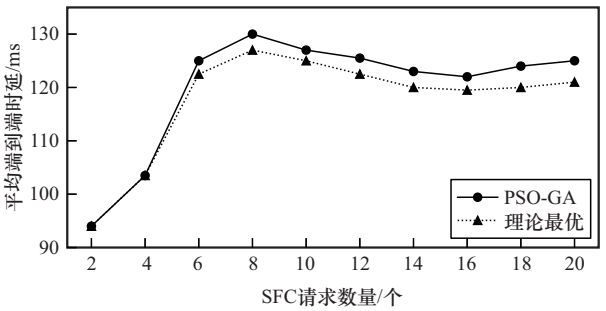
Greedy 相比其他算法更早且更多遇到无法部署的 SFC 请求，因为 Greedy 倾向于将 SFC 部署在最短路径中的少数节点上，这会导致部分节点更快耗尽资源，从而使后续到达的 SFC 请求可以部署的节点减少，大幅降低后续请求的部署成功率。但是由图 6 可以看出，Greedy 的平均端到端时延是最低的，这在请求的服务质量要求较高时具备一定的优势。

图 5 显示，RQAP、SD-VDSM 和本文算法的部署成功率处于同一梯队，且本文算法的部署成功率略高于前两种算法，相比次优的 SD-VDSM 提升 3.8%，说明本文算法在本文场景下更具优势，可以在资源一定的情况下部署更多的 SFC。这是因为本文算法在部署时不仅考虑 SFC 的端到端时延和资源消耗的平衡性，还考虑提升硬件加速器的利用率。因为在 FPGA 资源受限的条件下，每个节点能部署的硬件加速器的数量和种类是有限的。如果只考虑资源消耗的平衡性，则在 SFC 部署时不能有效利用已部署并启动的硬件加速器。频繁部署并启动新的硬件加速器会导致网络内的 FPGA 资源更快耗尽，使后续到达的请求可供选择的节点减少，并且还会降低硬件加速器的利用率，造成能耗的浪费。

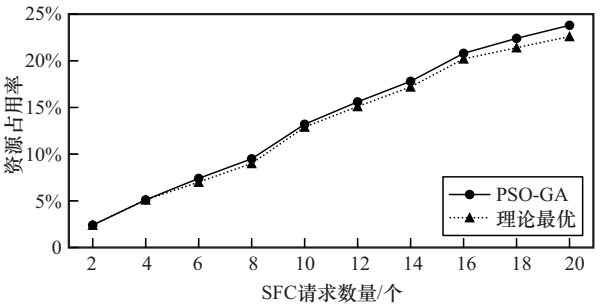
从图 6~图 8 可以看出，在平均端到端时延方面，当 SFC 请求增多时，本文算法优于 RQAP 和 SD-VDSM，相比次优的 SD-VDSM 降低 4.96%；在资源占用率方面，本文算法仅略差于 RQAP；在硬件加速器利用率方面，本文算法高于次优的 SD-VDSM，较之提升 9.32%。这说明本文算法有效地权衡了 SFC 部署时的端到端时延、资源利用率和硬件加速器利用率，且性能优于其他新算法。

由图 9 可得，在总功耗方面，本文算法低于次优的 SD-VDSM，较之降低 10.47%。这是因为本文算法倾向于将服务功能部署于已部署的硬件加速器上，降低了网络内部署的硬件加速器的数量，使 FPGA 产生的基础功耗下降，从而明显地降低了总功耗。因此，本文算法在部署 SFC 时，产生的总功耗更低，可以令服务提供商在更低的功率消耗下提供服务，降低服务成本。

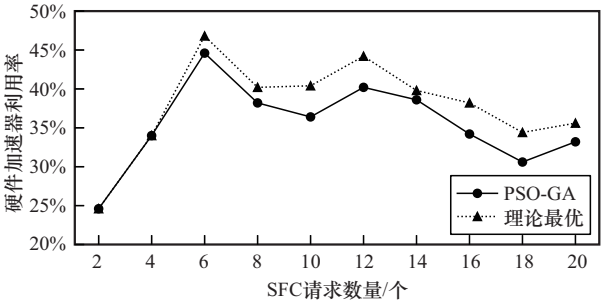
PSO-GA 作为启发式算法，能在有限时间内求得次优解。为分析 PSO-GA 所求次优解与理论最优解的性能差异，本文分析了上述实验的前 20 个 SFC 请求，利用枚举法获取每个请求的所有部署策略，挑选优化目标函数值最小的策略作为理论最优解与本文算法进行对比，PSO-GA 所求次优解与理论最优解的性能对比如图 10 所示。由图 10 可知，随着部署的 SFC 请求数量增多，PSO-GA 所求次优解相比理论最优解，在平均端到端时延、资源占用率和硬件加速器利用率指标上均略差于理论最优解，鉴于本文问题场景下最优解极长的求解时间，次优解相比最优解的差距处于可接受范围内。



(a) 平均端到端时延随SFC请求数量增多的变化



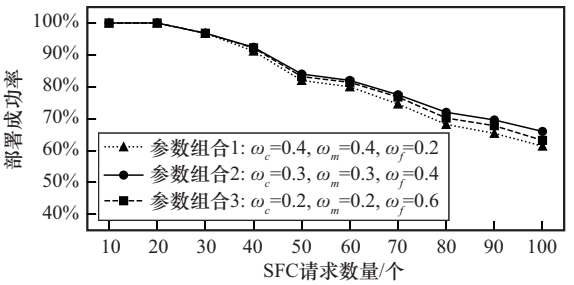
(b) 资源占用率随SFC请求数量增多的变化



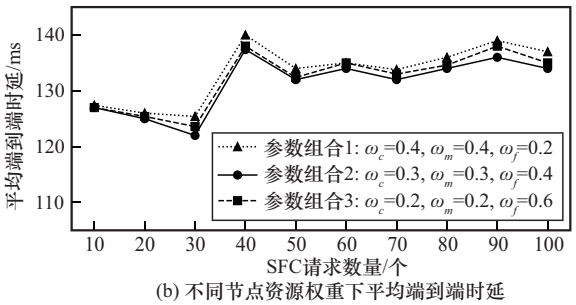
(c) 硬件加速器利用率随SFC请求数量增多的变化

图 10　PSO-GA 所求次优解与理论最优解的性能对比

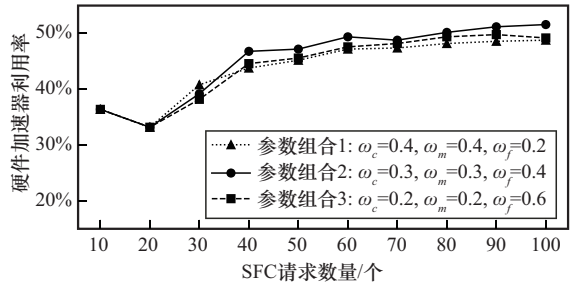
本文还在不同的节点资源占用率权重参数和总体资源占用率权重参数组合下进行了对比实验，结果如图11和图12所示，该实验中除了图中注明的参数组合取值，其他取值均与表2相同。从图11可以看出，在节点资源占用率权重参数方面，部署成功率、平均端到端时延和硬件加速器利用率均在参数组合2下表现最好，参数组合1表现最差。这是因为节点资源占用率权重参数决定了优化目标函数中CPU、内存和FPGA资源占用率的影响程度，在本文问题场景下，3种资源均应满足约束，所以应选择适当的参数组合，任何一种资源的权重过高或过低都会降低优化结果的性能表现。从图12可以看出，在总体资源占用权重参数方面，参数组合5在部署成功率、平均端到端时延和硬件加速器利用率上均表现最好。因此在设置总体资源占用权重时，也应选择合适的参数组合，以达到更好的算法性能。



(a) 不同节点资源权重下部署成功率随SFC请求数量增多的变化

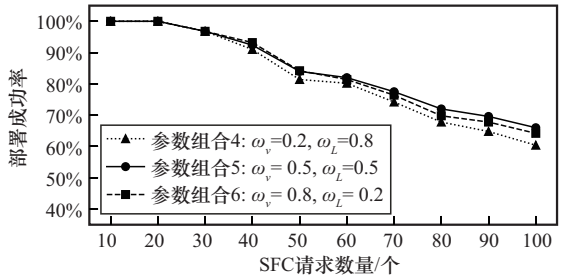


(b) 不同节点资源权重下平均端到端时延随SFC请求数量增多的变化

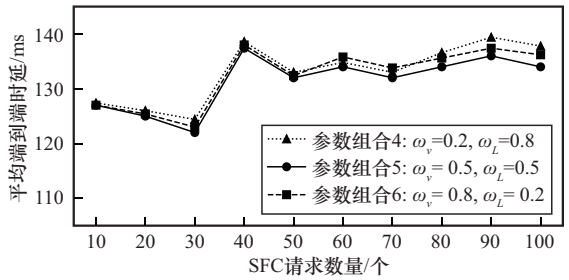


(c) 不同节点资源权重下硬件加速器利用率随SFC请求数量增多的变化

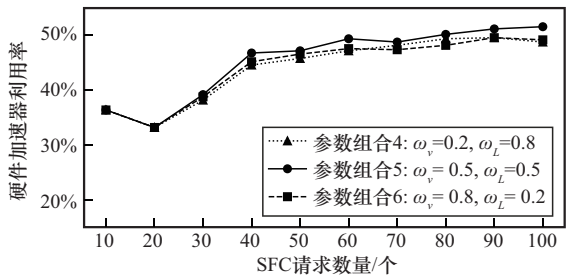
图11　不同节点资源占用权重参数组合下算法的性能对比



(a) 不同总体资源权重下部署成功率随SFC请求数量增多的变化



(b) 不同总体资源权重下平均端到端时延随SFC请求数量增多的变化

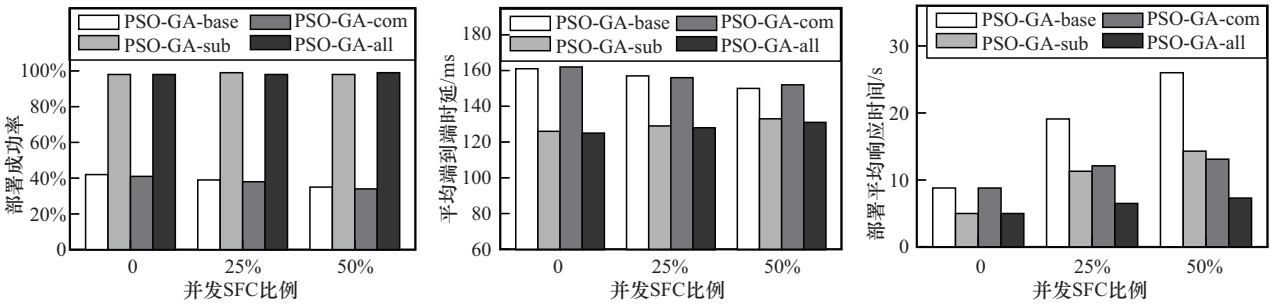


(c) 不同总体资源权重下硬件加速器利用率随SFC请求数量增多的变化

图12　不同总体资源占用权重参数组合下算法的性能对比

除此以外，为分析两项优化步骤对遗传粒子群混合算法总体性能的影响程度，本文在请求SFC长度设定为5的条件下，利用3种不同比例并发请求的数据集进行仿真消融实验，实验结果如图13所示。

由图13(a)可以看出，在不同并发SFC请求比例情况下，PSO-GA-sub和PSO-GA-all的成功率均接近99%，PSO-GA-base和PSO-GA-com的成功率远低于PSO-GA-sub和PSO-GA-all。这是因为未进行子链划分时，遗传粒子群优化算法的动作空间过大，包含极不可能将网络服务功能部署到的区域节点。因此，在当前遗传粒子群优化算法参数设置下，算法易陷入无法满足部署条件的局部最优解，导致求解的部署方案被拒绝，成功率大幅降低。同时，随着并发SFC比例增高，PSO-GA-sub和PSO-GA-all的成功率没有显著变化，PSO-GA-base和PSO-GA-com的成功率进一步降低。这表明本文子链划分优化方法在不同并发SFC请求比例下均具有良好的表现。



(a) 部署成功率随并发SFC比例上升的变化　(b) 平均端到端时延随并发SFC比例上升的变化　(c) 部署平均响应时间随并发SFC比例上升的变化

图 13　两项优化方法对 PSO-GA 性能影响的消融实验结果

由图 13(b)可以看出，PSO-GA-sub 和 PSO-GA-all 的平均时延大幅低于 PSO-GA-base 和 PSO-GA-com。这是因为子链划分优化过程限定了遗传粒子群优化算法探索最优解区域，使其更容易探索获得全局最优解，也就是时延更低的解。

由图 13(c)可以看出，性能表现最好的是 PSO-GA-all，不同并发 SFC 比例下平均响应时间为 5.05~7.28 s；性能表现最差的是 PSO-GA-base，不同并发 SFC 比例下平均响应时间为 8.71~25.83 s。当并发 SFC 比例为 0 时，PSO-GA-sub 优于只采用相似合并优化的算法，并且与 PSO-GA-all 近似。当并发 SFC 比例增加到 50% 时，PSO-GA-com 的平均响应时间逐渐优于 PSO-GA-sub。由此可见，相似合并优化显著缩短了并发 SFC 请求的排队时间。因此，本文子链划分和相似合并两项优化均可以有效提升内嵌式 DPU 的 SFC 部署的性能。

5　结束语

本文面向云–边–端架构网络，研究了网络服务提供和服务的优化部署问题。首先，提出了一种内嵌式 DPU 的 SFC 架构，通过硬件加速器的重配置和软硬协同计算兼顾网络服务的灵活部署和高效处理。然后，建立了内嵌式 DPU 的 SFC 部署、资源占用率、硬件加速器利用率、端到端时延和部署响应时间的模型，并提出在最小化平均资源利用率和端到端时延的同时，最大化部署成功率和硬件加速器利用率的优化目标。最后，设计了一种基于遗传粒子群优化算法的 SFC 部署策略优化算法，并引入子链划分和相似请求合并两个过程优化该部署算法的平均响应时间。仿真结果表明，本文算法在 SFC 部署成功率、平均端到端时延、硬件加速器利用率、总功耗和平均部署响应时间指标上优于对比算法。在未来的工作中，将考虑 FPGA 的工作特性，进一步研究硬件加速器部署时间超出时延阈值情况下 SFC 部署与硬件加速器迁移的协同优化，以及多类型异构加速器（如专用集成电路、GPU 等）并存情况下的 SFC 部署。

参考文献：

[1] 董裕民, 张静, 谢昌佐, 等. 云边端架构下边缘智能计算关键问题综述: 计算优化与计算卸载[J]. 电子与信息学报, 2024, 46(3): 765-776.
Dong Y M, Zhang J, Xie C Z, et al. A survey of key issues in edge intelligent computing under cloud-edge-terminal architecture: computing optimization and computing offloading[J]. Journal of Electronics & Information Technology, 2024, 46(3): 765-776.

[2] 郭少勇, 刘岩, 邵苏杰, 等. 新型电力系统数据跨域流通泛安全边界防护技术[J]. 电力系统自动化, 2024, 48(6): 96-111.
Guo S Y, Liu Y, Shao S J, et al. Ubiquitous security boundary protection technology for cross-domain data circulation in new power system[J]. Automation of Electric Power Systems, 2024, 48(6): 96-111.

[3] Zhang P Y, Zhang Y, Kumar N, et al. Energy-aware positioning service provisioning for cloud-edge-vehicle collaborative network based on DRL and service function chain[J]. IEEE Transactions on Mobile Computing, 2024, 23(4): 3424-3435.

[4] Lin R P, He L, Luo S, et al. Energy-aware service function chaining embedding in NFV networks[J]. IEEE Transactions on Services Computing, 2023, 16(2): 1158-1171.

[5] Li H, Li X, Qian Z Z, et al. Resource-aware service function chain deployment in cloud-edge environment[C]//Proceedings of the IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). Piscataway: IEEE Press, 2021: 1-6.

[6] Li B J, Tan K, Luo L L, et al. ClickNP: highly flexible and high performance network processing with reconfigurable hardware[C]//Proceedings of the 2016 ACM SIGCOMM Conference. New York: ACM Press, 2016: 1-14.

[7] Li L Y, Pan H, Wan X C, et al. Harmonia: a unified framework for heterogeneous FPGA acceleration in the cloud[C]//Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2. New York: ACM Press, 2025: 498-514.

[8] Intel Corporation. Infrastructure processing units (IPUs)[EB]. (2025-10-08) [2025-12-30].

[9] 刘忠沛, 吕高锋, 王继昌, 等. 专用数据处理器综述[J]. 计算机工程与科学, 2023, 45(2): 215-227.

Liu Z P, Lyu G F, Wang J C, et al. Review on data processing unit[J]. Computer Engineering & Science, 2023, 45(2): 215-227.
[10] Kornaros G, Leivadaros S, Kolimbianakis F. Flexible updating of Internet of Things computing functions through optimizing dynamic partial reconfiguration[J]. ACM Transactions on Embedded Computing Systems, 2024, 23(2): 1-25.
[11] Katsikas G P, Barbette T, Kostic D, et al. Metron: NFV service chains at the true speed of the underlying hardware[C]//15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18). Berkeley: USENIX Association, 2018: 171-186.
[12] Xue Y H, Zhu Z Q. On the upgrade of service function chains with heterogeneous NFV platforms[J]. IEEE Transactions on Network and Service Management, 2021, 18(4): 4311-4323.
[13] Farkiani B, Bakhshi B, Ali MirHassani S, et al. Prioritized deployment of dynamic service function chains[J]. IEEE/ACM Transactions on Networking, 2021, 29(3): 979-993.
[14] Huang H J, Tian J L, Yin H, et al. RQAP: resource and QoS aware placement of service function chains in NFV-enabled networks[J]. IEEE Transactions on Services Computing, 2023, 16(6): 4526-4539.
[15] Xiong Q L, Liao Z F, Tang X Y. A SFC placement strategy in low-visibility multi-domains networks for low resource usage cost[C]//Proceedings of the 2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA). Piscataway: IEEE Press, 2024: 1827-1832.
[16] Zhang Y H, Wang R, Hao J, et al. Service function chain deployment with VNF-dependent software migration in multi-domain networks[J]. IEEE Transactions on Mobile Computing, 2025, 24(5): 3685-3702.
[17] Hu Y, Li T. Enabling efficient network service function chain deployment on heterogeneous server platform[C]//Proceedings of the 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA). Piscataway: IEEE Press, 2018: 27-39.
[18] Huang Y D, Yao T T, Lin Z L, et al. Efficient service function chain placement over heterogeneous devices in deviceless edge computing environments[J]. IEEE Transactions on Computers, 2025, 74(1): 222-236.
[19] Zhang K X, Hu S H, Qu Z H, et al. Container-aware service function chains placement and optimization in vehicular edge computing[C]//Proceedings of the 2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA). Piscataway: IEEE Press, 2024: 623-630.
[20] Wang D S, Tan D P, Liu L. Particle swarm optimization algorithm: an overview[J]. Soft Computing, 2018, 22(2): 387-408.
[21] 吕顺利, 丁杰, 张海滨, 等. 新型电力系统智慧物联感知技术标准体系研究及思考[J]. 电力信息与通信技术, 2021, 19(8): 39-46.
Lyu S L, Ding J, Zhang H B, et al. Research and thinking on the standard system of new power system smart IoT perception technology[J]. Electric Power Information and Communication Technology, 2021, 19(8): 39-46.
[22] Zheng D Y, Peng C Z, Liao X T, et al. Towards latency optimization in hybrid service function chain composition and embedding[C]//Proceedings of the IEEE INFOCOM 2020 - IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2020: 1539-1548.
[23] Singh A, Ong J, Agarwal A, et al. Jupiter rising: a decade of clos topologies and centralized control in Google's datacenter network[J]. ACM SIGCOMM Computer Communication Review, 2015, 45(4): 183-197.
[24] AMD Zynq UltraScale+ MPSoC ZU7EV[EB]. (2025-10-07) [2025-12-30].
[25] Chen C, Nagel L, Cui L, et al. Distributed federated service chaining: a scalable and cost-aware approach for multi-domain networks[J]. Computer Networks, 2022, 212: 109044.
[26] 王昌达, 李柔, 张治平. 服务功能链部署的能耗与负载均衡联合优化[J]. 计算机应用研究, 2025, 42(5): 1500-1506.
Wang C D, Li R, Zhang Z P. Joint optimization of energy consumption and load balancing in service function chain deployment[J]. Application Research of Computers, 2025, 42(5): 1500-1506.
[27] Meidanis D, Georgopoulos K, Papaefstathiou I. FPGA power consumption measurements and estimations under different implementation parameters[C]//Proceedings of the 2011 International Conference on Field-Programmable Technology. Piscataway: IEEE Press, 2011: 1-6.

[作者简介]

邢若鹏（1998– ），男，河南洛阳人，北京邮电大学博士生，主要研究方向为软件定义安全、DPU。

郭少勇（1985– ），男，河北邢台人，博士，北京邮电大学教授、博士生导师，主要研究方向为DPU、区块链应用、边缘智能等。

邱雪松（1973– ），男，江西上饶人，博士，北京邮电大学教授、博士生导师，主要研究方向为网络与业务管理、物联网与区块链。

郝佳恺（1972– ），男，山东济南人，国网北京市电力公司正高级工程师、硕士生导师，主要研究方向为网络安全、先进通信技术等。

李宇婷（1985– ），女，山西长治人，博士，国网北京市电力公司高级工程师，主要研究方向为网络安全、先进通信技术等。