doi:10.11959/j.issn.1000-436x.2017032

基于离散粒子群优化的多目标服务路径构建算法

马丁[1,2]，庄雷[1]，兰巨龙[3]
（1. 郑州大学信息工程学院，河南 郑州 450001；2. 河南工业大学信息科学与工程学院，河南 郑州 450001；
3. 国家数字交换系统工程技术研究中心，河南 郑州 450002）

摘 要：针对当前关于服务路径构建问题的研究主要围绕单一优化目标，构建时延最小、开销最低或负载均衡的服务路径，忽略了服务路径的综合质量，提出了一种基于离散粒子群优化的多目标服务路径构建算法(MOPSO)。为了提高收敛速度，优化算法的性能，进一步研究了候选节点和路径的评价标准，提出一种粒子位置初始化和更新策略(PIFC)。仿真实验表明，与已有算法相比，所提出的算法有效地优化了服务路径的质量，提高了服务路径的构建成功率和长期平均收益。

关键词：网络功能虚拟化；服务链；服务路径；多目标；粒子群优化
中图分类号：TP393 **文献标识码**：A

Discrete particle swarm optimization based multi-objective service path constructing algorithm

MA Ding[1,2], ZHUANG Lei[1], LAN Ju-long[3]
(1. School of Information and Engineering, Zhengzhou University, Zhengzhou 450001, China;
2. College of Information Science and Engineering, Henan University of Technology, Zhengzhou 450001, China;
3. China National Digital Switching System Engineering & Technological Research Center, Zhengzhou 450002, China)

Abstract: Aiming at previous research primarily focused on constructing service paths with a single objective, for example, latency minimization, cost minimization or load balance, which ignored the overall performance of constructed service paths, a multi-objective service path constructing algorithm based on discrete particle swarm optimization (MOPSO) was proposed. To promote the convergence rate and improve constructing performance, the criterions for selecting candidate physical nodes and paths were explored, and a particle position initialization and update strategy (PIFC) was designed. Simulation experiments show that the proposed algorithms can improve the overall quality of service paths and increase the success rate and long-term average revenue.

Key words: network function virtualization, service chain, service path, multi-objective, particle swarm optimization

1 引言

网络功能虚拟化(NFV，network function virtualization)[1]提出“Softwarization”的思路，将网络功能与定制的硬件设备解耦，利用虚拟化技术以软件单元，即虚拟网络功能(VNF，virtual network function)的形式实现在通用的服务器平台上，通过VNF 的部署与组合实现多样化的服务定制。与传统的中间盒(middlebox)方式相比，NFV 能够实现网络功能的灵活开发、部署与迁移，从而有效地降低了设备投资成本。通过与软件定义网络(SDN，software defined network)[2,3]技术相结合，采用控制器上层的管理程序对网络实施管控，进一步提高了可管理性，降低了运维成本。

在 NFV 体系结构中，为了满足多样化的应用需求，实现服务的按需定制，控制器需要执行服务

收稿日期：2016-08-18；**修回日期**：2016-12-23
基金项目：国家重点基础研究发展计划（“973”计划）基金资助项目（No.2012CB315901）；国家自然科学基金资助项目（No.61379079）；河南省国际合作基金资助项目（No.152102410021）
Foundation Items: The National Basic Research Program of China (973 Program) (No.2012CB315901), The National Natural Science Foundation of China (No.61379079), The International Cooperation Program of Henan (No.152102410021)

路由算法，对需求中的每个服务，依序映射至能够承载该服务的节点上，构建一条端到端数据路径，使网络流依次被所需的服务处理。这种穿越整个网络，满足特定功能与性能需求，由节点与链路组成的直连序列被称之为服务路径(SP，service path)，该问题被称之为服务路径构建问题(SPCP，service path constructing problem)，包含在应用请求中的有序服务序列称之为服务链(service chain)。在服务路径的构建过程中，需要重点解决以下问题：1) 服务的执行需要计算与存储资源，数据的传输需要带宽资源，在资源容量有限的条件下，应如何高效地分配资源，以映射更多的服务链；2) 服务提供商根据用户需求构建服务路径需要为租用的资源支付费用，通过服务路径为用户提供定制服务从而获取收益，因此，需要将开销与收益纳入评价体系，需考虑如何有效地租用资源，以降低开销，提高收益；3) 服务实例可以部署在任意网络节点上，其中，服务承载节点的位置影响着服务路径的传输时延与链路资源使用量，性能影响着服务实例的处理时延，可用资源影响着承载服务的能力。因此，需解决如何在诸多候选的服务实例中进行选择，构建最优的服务路径问题。

由于服务路径构建是 NP 难问题[4~6]，已展开的研究工作主要集中在设计启发式的算法以获得近似最优解。Choi 等[7]提出了构造分层图(layered graph)的方法解决服务的执行顺序问题。Huang 等[6]提出了构造服务分步搜索图(service step search graph)的方法，在保证服务执行顺序的同时兼顾了资源的超量使用问题。Raman 等[8]使用可用资源容量的倒数重新设定分层图边的权值，提出了一种负载均衡算法。Bari 等[9]提出一种基于动态规划的启发式算法，通过 VNF 的动态编排实现开销与性能之间的均衡。Sahhaf 等[4,5]提出一种基于服务链分解的映射算法，使基于相同实现技术的 VNF 优先互联并映射到相同的服务器上，从而有效地降低了整体开销。Dietrich 等[10]通过请求拓扑分割，将拓扑片段映射到数据中心的候选服务器上，利用虚拟网关构建服务链子图实现流量汇聚，从而解决跨域的服务链映射问题。段通等[11]对 SDN 体系结构下的网络功能划分、描述和组合机制进行了研究，并提出一了种节点优先算法。Hu 等[12]通过 VNF 的灵活组合，构建安全服务路径，从而提供可定制安全服务。

上述研究工作，从不同的角度进行了探索，但是它们的目标可能是最小化开销、最小化端到端时延或均衡负载。很少有研究能够综合考虑构建一条既优化时延和开销，又均衡负载的服务路径。本文从提高综合构建性能出发，利用加权求和的方法将多个目标转化为单一的服务质量目标，并提出了一种基于粒子群优化的多目标服务路径构建算法(MOPSO)。为降低时延，减少资源使用量，降低资源碎片出现的概率，提高粒子群算法的收敛速度，进一步研究了候选节点和路径的评价标准，提出了一种指导粒子初始化和更新的优化策略 PIFC。基于该策略，PIFC-MOPSO 算法能够在满足资源和时延约束的条件下有效地利用资源、均衡负载，构建一条综合指标近似最优的服务路径，从长期效果看，能够有效地提升服务路径的构建成功率，增加收益。

2　多目标服务路径构建模型

2.1　网络模型

2.1.1　底层网络

底层网络的拓扑可以描述为一个带权无向图，标记为 $G^{\mathrm{S}}=(N^{\mathrm{S}}, L^{\mathrm{S}}, A_{\mathrm{N}}^{\mathrm{S}}, A_{\mathrm{L}}^{\mathrm{S}})$，其中，$N^{\mathrm{S}}$ 为底层节点的集合；L^{S} 为底层链路的集合；$A_{\mathrm{N}}^{\mathrm{S}}$ 为底层节点 $n^{\mathrm{S}}(n^{\mathrm{S}}\in N^{\mathrm{S}})$所拥有属性的集合，包括该节点的可用计算能力(CPU capacity)$C(n^{\mathrm{S}})$，该节点上能够提供的服务集合 $S(n^{\mathrm{S}})=\{S_k|$节点 n^{S} 能够实例化服务 $S_k\}$，该节点上服务 S_k 实例的处理时间为 $d_{S_k}(n^{\mathrm{S}})$，单位计算资源开销(per time unit CPU capacity)为 c^{S}；$A_{\mathrm{L}}^{\mathrm{S}}$ 为底层链路 $l^{\mathrm{S}}(l^{\mathrm{S}}\in L^{\mathrm{S}})$所拥有属性的集合，包括该链路的可用带宽 $B(l^{\mathrm{S}})$，传输时延 $d(l^{\mathrm{S}})$，单位带宽资源开销 b^{S}。底层网络所有无环路径的集合表示为$\prod^{\mathrm{S}}$，任意两节点 n_i^{S}, $n_j^{\mathrm{S}}\in N^{\mathrm{S}}$ 之间的无环路径集合标记为$\prod^{\mathrm{S}}(n_i^{\mathrm{S}}, n_j^{\mathrm{S}})$，$P^{\mathrm{S}}\in\prod^{\mathrm{S}}$ 表示底层网络中的一条无环路径，$H(P^{\mathrm{S}})$表示路径 P^{S} 的长度，即跳数。如图 1 所示，底层网络节点内的 S_1~S_4 表示节点能够实例化的服务，节点内部的数字表示节点的当前可用计算能力，链路上的数字表示可用带宽与传输时延。

2.1.2　服务请求

服务请求(SR, service request)体现了用户特定的功能和资源需求，用逻辑服务路径的形式表示，其拓扑可以描述为一个带权有向图，标记为 $G^{\mathrm{R}}=(N^{\mathrm{R}}, L^{\mathrm{R}}, R_{\mathrm{N}}^{\mathrm{R}}, R_{\mathrm{L}}^{\mathrm{R}})$，边的有向性表示服务实例的执行顺序约束。其中，$N^{\mathrm{R}}=\{n_0^{\mathrm{R}}, n_1^{\mathrm{R}}, \cdots, n_\lambda^{\mathrm{R}}, n_{\lambda+1}^{\mathrm{R}}\}$表示逻辑路径节点的

集合，n_0^R为源节点，$n_{\lambda+1}^R$为目的节点，λ 表示所需服务的数量，$L^R=\{<n_0^R,n_1^R>,<n_1^R,n_2^R>,\cdots,<n_\lambda^R,n_{\lambda+1}^R>\}$表示逻辑链路的集合。$R_N^R$表示节点 $n_i^R \in N^R\ (1\leqslant i\leqslant\lambda)$需求属性的集合，由服务需求 $S(n_i^R)$、计算能力需求 $\mu(S(n_i^R))$、租用单位计算能力所需支付的费用 c^R 组成。R_L^R表示链路 $l^R\in L^R$ 需求属性的集合，由带宽需求 $\mu(l^R)$、租用单位带宽所需支付的费用 b^R 组成。每个服务请求可用三元组 $SR(G^R, t_a, t_d)$表示，其中，t_a 表示请求的到达时间，t_d 表示请求的持续时间。如图 1 所示，对于服务请求 SR_1 和 SR_2，s 和 d 分别表示源节点和目的节点，逻辑节点 a、b、c、d、e、f 内的 S_1~S_4 表示需求的服务，节点内部的数字表示计算能力需求，A~F 表示底层网络节点，逻辑链路上的数字表示数据传输所需的带宽，此外，图 1 中虚线箭头表示服务的映射；s、d 之间的虚线表示节点的对应关系不存在映射。

2.2　形式化描述

2.2.1　问题分析

服务请求中所需的服务链表示为 $SC = \{S(n_1^R), S(n_2^R), \cdots, S(n_\lambda^R)\}$，除源节点和目的节点之外，剩余每个逻辑节点表示一个所需的服务。服务路径完整构建过程可以描述为在接收到用户发送的服务请求之后，控制器依次将 SC 中的服务映射到能够提供该服务且计算能力满足需求的底层节点上，同时，在服务的承载节点之间建立一条满足相应带宽需求的最优底层路径，确保该路径上的数据流依次经过 SC 中约定的服务实例，最终构建一条满足特定功能、性能与时延需求的端到端服务路径。若上述路径不存在，则拒绝该请求。

值得注意的是，在 SPC 问题中引入收益的概念后，从服务与链路映射、资源分配的角度来看，SPC 问题与经典的虚拟网映射(VNE，virtual network embedding)问题[13~16]和虚拟数据中心映射(VDCE，virtual date center embedding)问题[17,18]变得有些相近，但是仍然存在主要的差异，概括如下：1) 虚拟网拓扑可以为任意结构，服务链一般为线性或分支结构；2) 在 VNE/VDCE 问题中，虚拟节点的映射没有特定的顺序约束；而在 SPC 问题中，服务链中服务的映射必须满足有序性约束；3) 在 VNE/VDCE 问题中，虚拟节点与物理节点之间是一对一映射，而在 SPC 问题中，一个服务链中的多个服务可以映射到相同的服务节点上；4) 在 VNE 问题中，物理节点设备类型单一（路由或交换设备）；在 VDCE 问题中，物理节点设备类型进一步多样化（如服务器、交换机、存储设备等）；而在 SPC 问题中，大量不同类型的服务采用不同的实现技术，分布在不同类型的网络设备上；5) VNE/VDCE 问题未考虑端到端时延。

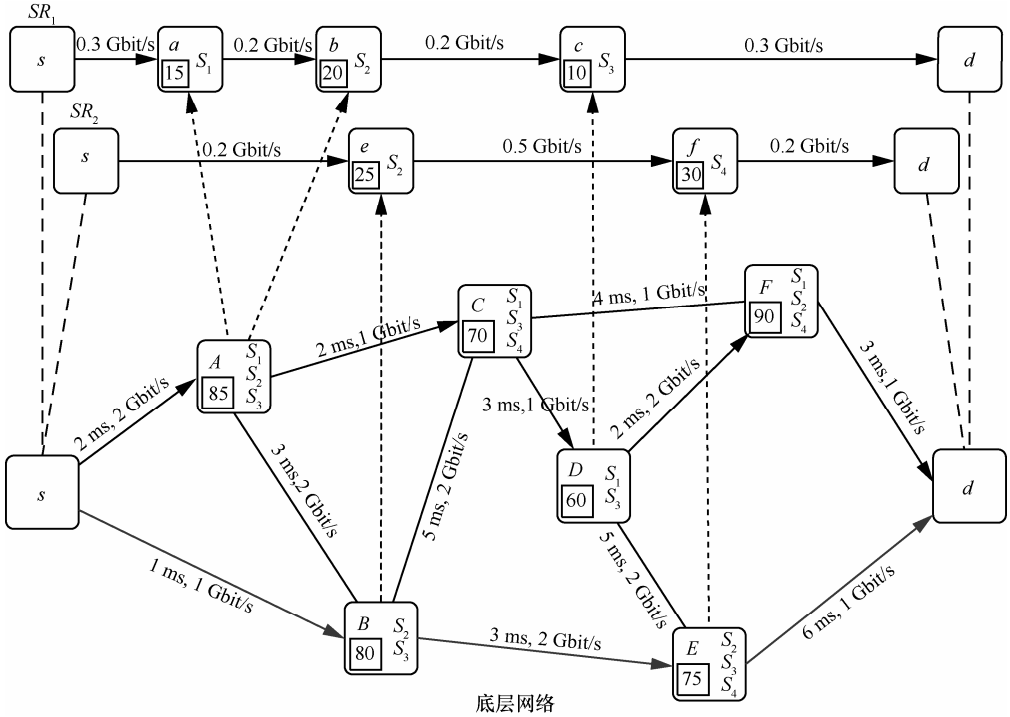

图 1　底层网络与服务路径示例

2.2.2 问题描述

根据前一节的分析，服务路径的构建包括 2 个主要过程。

1) 服务到底层节点的映射，定义为

$$M_{\mathrm{S}}: SQ \mapsto \bigcup_{i=1}^{\lambda} CN_i,\ \forall S(n_i^{\mathrm{R}}), S(n_j^{\mathrm{R}}) \in SQ,$$

$$M_{\mathrm{S}}(S(n_i^{\mathrm{R}})) \in CN_i$$

如果 $M_{\mathrm{S}}(S(n_i^{\mathrm{R}})) = M_{\mathrm{S}}(S(n_j^{\mathrm{R}}))$，$j$ 和 i 不一定相等，表明允许同一请求的多个服务映射在相同的底层节点上。CN_i 表示 SC 中第 i 个服务 $S(n_i^{\mathrm{R}})$的候选底层节点集合，定义为

$$CN_i = \{n^{\mathrm{S}} \mid S(n_i^{\mathrm{R}}) \in S(n^{\mathrm{S}}), C(n^{\mathrm{S}}) \geqslant \mu(S(n_i^{\mathrm{R}})), n^{\mathrm{S}} \in N^{\mathrm{S}}\}$$

服务到候选节点的映射示例如图 1 所示，服务请求 SR_1 的服务映射结果为：$\{S_1 \mapsto A,\ S_2 \mapsto A,\ S_3 \mapsto D\}$。服务请求 SR_2 的服务映射结果为：$\{S_2 \mapsto B,\ S_4 \mapsto E\}$。

2) 逻辑链路到底层路径的映射，定义为

$$M_{\mathrm{L}}: L^{\mathrm{R}} \mapsto P^{\mathrm{S}}, \forall l^{\mathrm{R}} = (n_i^{\mathrm{R}}, n_j^{\mathrm{R}}) \in L^{\mathrm{R}},$$

$$M_{\mathrm{L}}(n_i^{\mathrm{R}}, n_j^{\mathrm{R}}) \subseteq P^{\mathrm{S}}(M_{\mathrm{S}}(S(n_i^{\mathrm{R}})), M_{\mathrm{S}}(S(n_j^{\mathrm{R}})))$$

同时满足带宽约束

$$B(P^{\mathrm{S}}) \geqslant \mu(l^{\mathrm{R}}), \forall P^{\mathrm{S}} \in M_{\mathrm{L}}(l^{\mathrm{R}})$$

逻辑链路映射的示例如图 1 所示，服务请求 SR_1 的逻辑链路映射结果为：$\{(s,a) \mapsto (s,A),\ (a,b) \mapsto (A, A),\ (b,c) \mapsto \{(A,C),\ (C,D)\},\ (c,d) \mapsto \{(D,F),\ (F,d)\}\}$，其中，逻辑链路$(a, b)$映射到节点 A 的内部，节点内部带宽为无限大，因此，带宽需求确定能够满足，而且不会产生链路资源的使用开销。

服务路径标记为 SP，是由底层节点和链路组成的直连序列，定义如下

$$\begin{aligned} SP &= \{N^{SP}, L^{SP}, M^{SP}\} \\ &= \{(n_{i_1}^{\mathrm{S}}, M_{i_1}), l_{i_1}, (n_{i_2}^{\mathrm{S}}, M_{i_2}), l_{i_2}, \cdots, l_{i_{m-1}}, (n_{i_m}^{\mathrm{S}}, M_{i_m})\} \end{aligned} \quad (1)$$

$$M_{i_k} = \{S(n^{\mathrm{R}}) \mapsto n_{i_k}^{\mathrm{S}} \mid M_{\mathrm{S}}(S(n^{\mathrm{R}})) = n_{i_k}^{\mathrm{S}}\} \quad (2)$$

s.t.

$$C(n_{i_k}^{\mathrm{S}}) \geqslant \sum_{S(n^{\mathrm{R}}) \mapsto n_{i_k}^{\mathrm{S}} \in M_{i_k}} \mu(S(n^{\mathrm{R}})), \forall n_{i_k}^{\mathrm{S}} \in N^{SP}, M_{i_k} \neq \varnothing \quad (3)$$

$$B(l_{i_k}) \geqslant \sum_{P^{\mathrm{S}} \in M_{\mathrm{L}}(L^{\mathrm{R}}), l_{i_k} \in P^{\mathrm{S}}} \mu(l^{\mathrm{R}}), \forall l_{i_k} \in L^{SP} \quad (4)$$

$$D(SP) \leqslant D_{\max} \quad (5)$$

其中，N^{SP} 和 L^{SP} 分别表示服务路径的节点和链路集合，m 表示节点数量，M^{SP} 表示服务路径中服务到底层节点映射的集合。$n_{i_1}^{\mathrm{S}}$ 为源节点，$n_{i_m}^{\mathrm{S}}$ 为目的节点，l_{i_k} 表示连接节点 $n_{i_k}^{\mathrm{S}}$ 和 $n_{i_{k+1}}^{\mathrm{S}}$ 的链路，$M_{i_k} \in M^{SP}$ 表示节点 $n_{i_k}^{\mathrm{S}}$ 上服务映射的集合，如果 $M_{i_k} = \varnothing$，则表明 $n_{i_k}^{\mathrm{S}}$ 未执行任何服务实例。式(3)的约束表示节点的可用计算能力能够执行其承载的服务实例。式(4)的约束表示链路的可用带宽能够支持其承载的逻辑链路进行数据传输。式(5)的约束表示服务路径的时延必须小于最大允许的端到端时延 $D_{\max}$。如图 1 所示，为 SR_1 建立的服务路径为：$\{(s,\ \varnothing), (s, A), (A, \{S_1 \mapsto A,\ S_2 \mapsto A\}),\ (A,C), (C,\ \varnothing),\ (C,\ D),\ (D, \{S_3 \mapsto D\}), (D, F), (F,\ \varnothing), (F, d), (d,\ \varnothing)\}$。

2.3 评价指标

为一个服务请求构建的单条服务路径，其质量可从以下方面进行评价。

1) 收益与开销

与文献[5,13]类似，服务提供商成功为一个服务请求构建服务路径获得的收益等于用户租用计算资源和带宽资源支付的费用之和，定义如下

$$R(SP) = \sum_{n^{\mathrm{R}} \in N^{\mathrm{R}}} \mu(S(n^{\mathrm{R}}))c^{\mathrm{R}} + \sum_{l^{\mathrm{R}} \in L^{\mathrm{R}}} \mu(l^{\mathrm{R}})b^{\mathrm{R}} \quad (6)$$

服务提供商成功为一个用户构建服务路径付出的开销等于使用底层节点的计算资源和底层链路的带宽资源支出的费用之和，定义如下

$$C(SP) = \sum_{n^{\mathrm{R}} \in N^{\mathrm{R}}} \mu(S(n^{\mathrm{R}}))c^{\mathrm{S}} + \sum_{l^{\mathrm{R}} \in L^{\mathrm{R}}} \sum_{P^{\mathrm{S}} \in M_{\mathrm{L}}(l^{\mathrm{R}}), l^{\mathrm{S}} \in P^{\mathrm{S}}} \mu(l^{\mathrm{R}})b^{\mathrm{S}} \quad (7)$$

2) 时延

服务路径的端到端时延表示数据流从源节点发出到目的节点接收之间所用的时间，由服务路径上服务实例的执行时延与通信链路的传输时延组成，定义如下

$$D(SP) = \sum_{n^{\mathrm{R}} \in N^{\mathrm{R}}} \sum_{n^{\mathrm{S}} \in M_{\mathrm{S}}(S(n^{\mathrm{R}}))} d_{S(n^{\mathrm{R}})}(n^{\mathrm{S}}) + \sum_{l^{\mathrm{R}} \in L^{\mathrm{R}}} \sum_{P^{\mathrm{S}} \in M_{\mathrm{L}}(l^{\mathrm{R}}), l^{\mathrm{S}} \in P^{\mathrm{S}}} d(l^{\mathrm{S}}) \quad (8)$$

3) 负载强度

在构建服务路径时，除了要考虑服务的功能需求之外，还需要考虑服务承载节点及其相邻链路的负载情况，将服务和逻辑链路尽量映射到资源丰富的底层节点和链路上。负载强度(LD，load degree)用来衡量服务路径的资源用量和负载程度。服务路

径中底层节点 $n_{i_k}^{\mathrm{S}}$ 的负载强度定义如下

$$LD_{n_{i_k}^{\mathrm{S}}}=\sum_{S(n^{\mathrm{R}})\mapsto n_{i_k}^{\mathrm{S}}\in M_{i_k}}\frac{\mu(S(n^{\mathrm{R}}))}{C(n_{i_k}^{\mathrm{S}})},\forall n_{i_k}^{\mathrm{S}}\in N^{SP},M_{i_k}\neq\varnothing \quad (9)$$

节点的负载强度综合考虑了节点的可用计算能力以及服务的计算能力需求。根据式(3)，其取值范围为 0~1，而且值越小，将服务映射到该节点的可能性越大，越有利于底层节点的负载均衡。

基于节点负载强度的思路，服务路径中链路 $l_{i_k}^{\mathrm{S}}$ 的负载强度定义如下

$$LD_{l_{i_k}^{\mathrm{S}}}=\sum_{P^{\mathrm{S}}\in M_{\mathrm{L}}(l^{\mathrm{R}}),l_{i_k}^{\mathrm{S}}\in P^{\mathrm{S}}}\frac{\mu(l^{\mathrm{R}})}{B(l_{i_k}^{\mathrm{S}})},\forall l_{i_k}^{\mathrm{S}}\in L^{SP} \quad (10)$$

服务路径 *SP* 的负载强度定义为

$$LD(SP)=\omega_{\mathrm{N}}\sum_{n_{i_k}^{\mathrm{S}}\in N^{SP}}LD_{n_{i_k}^{\mathrm{S}}}+\omega_{\mathrm{L}}\sum_{l_{i_k}^{\mathrm{S}}\in L^{SP}}LD_{l_{i_k}^{\mathrm{S}}} \quad (11)$$

其中，ω_{N}、ω_{L} 是权重参数，用来调整节点和链路负载强度的侧重程度。*LD*(*SP*)体现了服务路径节点资源和链路资源的整体利用情况。

4) 服务路径综合质量

上述定义的开销、时延和负载强度都是衡量服务路径构建质量的重要指标，为了同时对这 3 个指标进行优化，采用加权求和的方法，首先定义服务路径 *SP* 的综合评价函数如下

$$W(SP)=\alpha\frac{C(SP)}{C_{\max}(SP)}m+\beta\frac{D(SP)}{D_{\max}(SP)}m+\gamma LD(SP) \quad (12)$$

服务路径 *SP* 的综合质量 *Q*(*SP*)定义为

$$Q(SP)=\frac{1}{W(SP)} \quad (13)$$

其中，式(12)已经进行了无量纲处理，α、β、γ 是权重参数，用来调整优化的侧重点，*Q*(*SP*)值越大，服务路径的质量越高。

随着请求的到达和离开、服务路径的构建与拆除，衡量服务路径长期综合构建质量的评价指标包括长期平均收益、平均开销、收益开销比、平均负载强度和平均构建成功率等。

2.4 模型构建

本节以提高服务路径综合质量为目标，建立 SPC 问题的整数线性规划模型。首先，在此模型中引入二进制变量 $x_i^{S(u)}$和 y_{ij}^{uv}，如果逻辑节点 *u* 所需的服务 *S*(*u*)被映射到底层节点 *i* 上，则 $x_i^{S(u)}$值为 1，否则值为 0；如果底层链路 $l_{i,j}$ 承载了逻辑链路 $l_{u,v}$，则 y_{ij}^{uv} 值为 1；否则值为 0。

在此模型中，服务路径 *SP* 的开销表示为

$$C(SP)=\sum_{u\in N^{\mathrm{R}}}\mu(S(u))c^{\mathrm{S}}+\sum_{l_{u,v}\in L^{\mathrm{R}}}\sum_{l_{i,j}\in L^{SP}}y_{ij}^{uv}\mu(l_{u,v})b^{\mathrm{S}}$$

时延表示为

$$D(SP)=\sum_{u\in N^{\mathrm{R}}}\sum_{i\in N^{SP}}x_i^{S(u)}d_{S(u)}(i)+\sum_{l_{u,v}\in L^{\mathrm{R}}}\sum_{l_{i,j}\in L^{SP}}y_{ij}^{uv}d(l_{i,j})$$

负载强度表示为

$$LD(SP)=\omega_{\mathrm{N}}\sum_{i\in N^{SP}}\frac{\sum_{u\in N^{\mathrm{R}}}x_i^{S(u)}\mu(S(u))}{C(i)}+\omega_{\mathrm{L}}\sum_{l_{i,j}\in L^{SP}}\frac{\sum_{l_{u,v}\in L^{\mathrm{R}}}y_{ij}^{uv}\mu(l_{u,v})}{B(l_{i,j})}$$

根据式(12)和式(13)，该整数线性规划模型的目标函数定义如下

$$\max Q(SP)=\frac{1}{\alpha\frac{C(SP)}{C_{\max}(SP)}m+\beta\frac{D(SP)}{D_{\max}(SP)}m+\gamma LD(SP)} \quad (14)$$

s.t.

$$\forall i\in N^{SP},\sum_{u\in N^{\mathrm{R}}}x_i^{S(u)}\mu(S(u))\leqslant C(i) \quad (15)$$

$$\forall l_{i,j}\in L^{SP},\sum_{l_{u,v}\in L^{\mathrm{R}}}y_{ij}^{uv}\mu(l_{u,v})\leqslant B(l_{i,j}) \quad (16)$$

$$\forall u\in N^{\mathrm{R}},\sum_{i\in N^{S}}x_i^{S(u)}=1 \quad (17)$$

$$\forall i\in N^{SP},\sum_{u\in N^{\mathrm{R}}}x_i^{S(u)}\leqslant\lambda \quad (18)$$

$$\forall i\in N^{SP},\forall l_{u,v}\in L^{\mathrm{R}},\sum_{l_{i,j}\in L^{SP}}y_{ij}^{uv}-\sum_{l_{j,i}\in L^{SP}}y_{ji}^{uv}=x_i^{S(u)}-x_i^{S(v)} \quad (19)$$

$$\sum_{u\in N^{\mathrm{R}}}\sum_{i\in N^{SP}}x_i^{S(u)}d_{S(u)}(i)+\sum_{l_{u,v}\in L^{\mathrm{R}}}\sum_{l_{i,j}\in L^{SP}}y_{ij}^{uv}d(l_{i,j})\leqslant D_{\max} \quad (20)$$

$$\forall i\in N^{\mathrm{S}},\forall u\in N^{\mathrm{R}},x_i^{S(u)}\in\{0,1\} \quad (21)$$

$$\forall l_{i,j}\in L^{\mathrm{S}},\forall l_{u,v}\in L^{\mathrm{R}},y_{ij}^{uv}\in\{0,1\} \quad (22)$$

其中，式(15)为计算资源约束，确保底层节点的可用资源能够满足执行服务实例的需求。式(16)为带宽资源约束，确保底层链路的可用带宽足以承载映射至其上的逻辑链路。式(17)为服务到底层节点的映射约束，确保同一个服务请求中的服务只能映射

到唯一的底层节点上。式(18)为底层节点承载服务的约束，对于同一个服务请求，一个底层节点可以承载其中多个服务。式(19)为逻辑链路到所映射底层网络路径的连接性约束。式(20)为端到端时延约束，确保服务路径的端到端时延满足 SLA(service level agreement)中的约定。式(21)和式(22)为变量 $x_i^{S(u)}$和 y_{ij}^{uv} 的取值范围约束。

3　离散粒子群优化算法

粒子群优化（PSO，particle swarm optimization）是一种全局优化技术，由 Kennedy 和 Eberhart[19]在 1995 年首次提出。其中，个体被称为“粒子”，个体构成的集体被称为“种群”，算法思想概括如下。首先根据种群的规模随机初始化一群粒子，每个粒子根据自身的速度穿越 d 维的搜索空间，向问题可行解的方向飞行；在飞行的过程中，每个粒子根据自身经过的最优位置和整个种群经过的最优位置，不断动态地调整飞行速度，以获得最优解。PSO 算法兼具并行计算和群体优化的特性，且具有实现简单、收敛速度快、需要调整的参数较少等优点，因此，采用 PSO 解决多目标约束下的服务路径构建问题，能够扩大搜索范围，提高有效路径被发现的概率，在较短时间内发现近似的最优解。

3.1　离散模型构建

粒子群算法的基本模型[20]通常用于解决连续空间的优化问题。而服务路径构建问题，主要由服务映射和逻辑链路映射组成，显然是一个离散分配问题。因此，粒子的位置、速度和操作需要重新进行定义，使粒子在整数空间内进行飞行和搜索，参考文献[15,21,22]，具体定义如下。

1) 粒子的位置

服务链的每一个服务都需要映射到唯一的候选节点上，因此，用 λ 表示算法搜索空间的维度。相应地，第 i 个粒子在第 t 次迭代时的位置表示为 λ 维向量 $\boldsymbol{p}_i(t)=(p_{i,1}, p_{i,2}, \cdots, p_{i,j}, \cdots, p_{i,\lambda})$，代表第 i 个可能的服务映射方案，其中，$p_{i,j}$ 取值为正整数，表示第 j 个服务所映射的候选节点的索引值。

2) 粒子的速度

第 i 个粒子在第 t 次迭代时的速度表示为 λ 维向量 $\boldsymbol{v}_i(t)=(v_{i,1}, v_{i,2}, \cdots, v_{i,j}, \cdots, v_{i,\lambda})$，用来引导粒子向更优解方向飞行。其中，$v_{i,j}$ 为二进制变量，当 $v_{i,j}$ 取值为 1，表示第 j 个服务所映射的底层节点需要从 CN_j 中重新选择。

3) 适应函数

根据 2.4 节的整数线性规划模型，将其目标函数作为适应函数，标记为 $f(\boldsymbol{p})$。

4) 粒子的更新

根据粒子位置和速度的定义，粒子群优化的速度和位置更新公式定义如下

$$\boldsymbol{v}_i(t+1)=w\boldsymbol{v}_i(t)+cr_1(\boldsymbol{pbest}_i(t)-\boldsymbol{p}_i(t))+cr_2(\boldsymbol{gbest}(t)-\boldsymbol{p}_i(t)) \tag{23}$$

$$\boldsymbol{p}_i(t+1)=\boldsymbol{p}_i(t)+\boldsymbol{v}_i(t+1) \tag{24}$$

在算法的迭代进化过程中，第 i 个粒子会根据当前速度向量 $\boldsymbol{v}_i(t)$、自身的局部最优解 $\boldsymbol{pbest}_i(t)$和整个群体的全局最优解 $\boldsymbol{gbest}(t)$动态更新映射调整决策 $\boldsymbol{v}_i(t+1)$，进而更新现有的映射方案 $\boldsymbol{p}_i(t)$。其中，$\boldsymbol{pbest}_i(t)$和 $\boldsymbol{gbest}(t)$均为 λ 维向量：$\boldsymbol{pbest}_i(t)=(pb_{i,1}, pb_{i,2}, \cdots, pb_{i,j}, \cdots, pb_{i,\lambda})$，$\boldsymbol{gbest}(t)=(gb_1, gb_2, \cdots, gb_j, \cdots, gb_\lambda)$。$w$ 为惯性权重，表示粒子保持当前速度的概率，参数 cr_1 和 cr_2 为加速因子，分别表示粒子参照局部最优解和全局最优解进行调整的概率，w、cr_1 和 cr_2 均为正常量，且满足 $w+cr_1+cr_2=1$。

5) 操作定义

在式(23)和式(24)中，根据服务路径的构建规则，“+”“−”操作的含义重新定义如下。

$\boldsymbol{pbest}_i(t)-\boldsymbol{p}_i(t)$表示当前服务映射方案与局部最优映射方案之间的差异，$\boldsymbol{gbest}(t)-\boldsymbol{p}_i(t)$表示当前服务映射方案与全局最优映射方案之间的差异，若位于“−”两端的向量在同一维上的值相同，则该维差值结果等于 0，否则等于 1。如 $\boldsymbol{p}_i(t)=(1,3,4,6)$, $\boldsymbol{pbest}_i(t)=(2,3,5,6)$, $\boldsymbol{pbest}_i(t)-\boldsymbol{p}_i(t)=(1,0,1,0)$。

$w\boldsymbol{v}_i(t)+cr_1(\boldsymbol{pbest}_i(t)-\boldsymbol{p}_i(t))+cr_2(\boldsymbol{gbest}(t)-\boldsymbol{p}_i(t))$表示以 w 的概率维持当前的映射方案，以 cr_1 的概率参照局部最优映射方案进行调整，以 cr_2 的概率参照全局最优映射方案进行调整。如 $w=0.3$, $cr_1=0.7$, $\boldsymbol{v}_i(t)=(1,1,0,0)$,$\boldsymbol{pbest}_i(t)-\boldsymbol{p}_i(t)=(1,0,1,0)$,$w\boldsymbol{v}_i(t)+cr_1(\boldsymbol{pbest}_i(t)-\boldsymbol{p}_i(t))=(1, \#, \#, 0)$，“#”表示该维的值取 0 或 1，由参数 w 和 cr_1 代表的概率确定，上述例子中第一个“#”以 0.3 的概率取 1，以 0.7 的概率取 0。

$\boldsymbol{p}_i(t)+\boldsymbol{v}_i(t+1)$表示当前服务映射方案参照更新后的速度向量进行调整。如 $\boldsymbol{p}_i(t)=(1,3,4,6)$, $\boldsymbol{v}_i(t+1)=(1, 0, 0, 0)$, $\boldsymbol{p}_i(t)+\boldsymbol{v}_i(t+1)=(*, 3, 4, 6)$, “*”表示第一个服务映射的底层节点需要重新选择。

3.2　算法设计

在资源有限的约束条件下，如果采用随机的方

式从候选节点集合初始化或更新粒子的位置向量，会导致资源利用率下降、资源碎片增多、底层网络负载不均，开销和时延增大，严重时会使式(16)、式(19)和式(20)中的带宽、连接性和时延约束得不到满足，导致逻辑链路映射失败。由于失败后需要重新初始化粒子的位置向量，导致算法的收敛速度下降。为了解决上述问题，需要建立候选节点和路径的评价标准，制定粒子初始化和更新策略，并确保算法在第一次迭代过程得到的映射方案是有效可行的。

3.2.1 候选节点和路径评价标准

1) 候选节点评价标准

在为第 j 个服务选择映射目标时，需要建立候选节点 $n^{\mathrm{S}}\in CN_j$ 的评价标准，度量其承载第 j 个服务的概率。根据服务路径的构建过程和优化目标，在映射第 j 个服务时，不仅要考虑候选节点本身的属性，还需要考虑第 $j-1$ 个服务的承载节点与候选节点之间的路径 $P^{\mathrm{S}}_{j-1,n^{\mathrm{S}}}$ 的服务质量，首先从路径长度的角度评价候选节点 n^{S}。

$$H(j,n^{\mathrm{S}})=H(P^{\mathrm{S}}_{j-1,n^{\mathrm{S}}})$$

由于 $P^{\mathrm{S}}_{j-1,n^{\mathrm{S}}}$ 所承载逻辑链路的带宽需求相同，因此，该段路径的开销取决于路径长度。其次，从路径时延角度评价候选节点 n^{S}。

$$D(j,n^{\mathrm{S}})=D(P^{\mathrm{S}}_{j-1,n^{\mathrm{S}}})+d_j(n^{\mathrm{S}})$$

其中，$d_j(n^{\mathrm{S}})$表示候选节点 n^{S} 实例化的第 j 个服务的处理时间。然后从路径负载强度的角度评价候选节点 n^{S}。

$$LD(j,n^{\mathrm{S}})=\omega_{\mathrm{N}}\frac{\mu(j)}{C(n^{\mathrm{S}})}+\omega_{\mathrm{L}}\sum_{l^{\mathrm{S}}\in P^{\mathrm{S}}_{j-1,n^{\mathrm{S}}}}\frac{\mu(l^{\mathrm{R}}_{j-1,j})}{B(l^{\mathrm{S}})}$$

最后候选节点 n^{S} 的综合评价标准定义为

$$\psi(j,n^{\mathrm{S}})=\frac{1}{\alpha\dfrac{H(j,n^{\mathrm{S}})}{H_{\max}(j,n^{\mathrm{S}})}+\beta\dfrac{D(j,n^{\mathrm{S}})}{D_{\max}(j,n^{\mathrm{S}})}+\gamma\dfrac{LD(j,n^{\mathrm{S}})}{H(j,n^{\mathrm{S}})+1}} \tag{25}$$

候选节点的 ψ 值越大，从第 $j-1$ 个服务到第 j 个服务的子服务路径质量越高，被映射的概率越高。在初始化粒子位置的过程中，从源节点开始，服务和相应逻辑链路的映射依次参照候选节点的评价标准在一个阶段内完成，当所有服务映射完成之后，第 λ 个服务与目的节点之间逻辑链路的映射需要依据候选路径的评价标准来完成。

2) 候选路径评价标准

与候选节点的评价标准类似，第 $j-1$ 与第 j 个服务之间的逻辑链路 $l^{\mathrm{R}}_{j-1,j}$ 的候选路径 $P^{\mathrm{S}}_{j-1,j}$ 的评价标准也根据优化目标综合得到，其中，路径长度、时延、负载强度的评价标准依次为

$$H(l^{\mathrm{R}}_{j-1,j},P^{\mathrm{S}}_{j-1,j})=H(P^{\mathrm{S}}_{j-1,j})$$

$$D(l^{\mathrm{R}}_{j-1,j},P^{\mathrm{S}}_{j-1,j})=D(P^{\mathrm{S}}_{j-1,j})$$

$$LD(l^{\mathrm{R}}_{j-1,j},P^{\mathrm{S}}_{j-1,j})=\sum_{l^{\mathrm{S}}\in P^{\mathrm{S}}_{j-1,j}}\frac{\mu(l^{\mathrm{R}}_{j-1,j})}{B(l^{\mathrm{S}})}$$

候选路径 $P^{\mathrm{S}}_{j-1,j}$ 的综合评价标准定义为

$$\varphi(l^{\mathrm{R}}_{j-1,j},P^{\mathrm{S}}_{j-1,j})=\frac{1}{\alpha\dfrac{H(l^{\mathrm{R}}_{j-1,j},P^{\mathrm{S}}_{j-1,j})}{H_{\max}(l^{\mathrm{R}}_{j-1,j},P^{\mathrm{S}}_{j-1,j})}+\beta\dfrac{D(l^{\mathrm{R}}_{j-1,j},P^{\mathrm{S}}_{j-1,j})}{D_{\max}(l^{\mathrm{R}}_{j-1,j},P^{\mathrm{S}}_{j-1,j})}+\gamma\dfrac{LD(l^{\mathrm{R}}_{j-1,j},P^{\mathrm{S}}_{j-1,j})}{H(l^{\mathrm{R}}_{j-1,j},P^{\mathrm{S}}_{j-1,j})}} \tag{26}$$

候选节点的 φ 值越大，从第 $j-1$ 个服务的承载节点到第 j 个服务的承载节点之间的底层路径的质量越高，被映射的概率越高。与 ψ 相比，φ 只关注候选路径上底层链路的属性，忽略了路径终端节点的属性，需要注意的是，逻辑链路 $l^{\mathrm{R}}_{\lambda,\lambda+1}$ 候选路径的 φ 与 ψ 值相同。当粒子位置向量的第 j 维需要调整更新时，可先依据 ψ 值重新映射第 j 个服务和逻辑链路 $l^{\mathrm{R}}_{j-1,j}$，再依据 φ 映射逻辑链路 $l^{\mathrm{R}}_{j,j+1}$。

3.2.2 粒子初始可行解生成算法

单个粒子初始可行解的生成过程对应一条服务路径的构建过程。在可行解的生成策略中，首先将整条服务路径按序分解为 $\lambda-1$ 条子服务路径，然后依据候选节点和路径的评价标准逐步进行服务和逻辑链路的映射，详细过程如算法 1 所示。

算法 1 粒子初始化与可行解检测算法（PIFC）

输入：底层网络拓扑 $G^{\mathrm{S}}=(N^{\mathrm{S}}, L^{\mathrm{S}}, A^{\mathrm{S}}_{\mathrm{N}}, A^{\mathrm{S}}_{\mathrm{L}})$，服务请求 $SR(G^{\mathrm{R}}, t_{\mathrm{a}}, t_{\mathrm{d}})$；

输出：服务路径 $SP=(N^{SP}, L^{SP}, M^{SP})$；

1) $f(\boldsymbol{p}_i)$表示第 i 个粒子的适应函数；

2) for j=1 to λ+1 do

3) 构建第 j 个服务的候选底层节点集合 CN_j；

4) for all $n^{\mathrm{S}}\in CN_j$ do

5)　利用 K 最短路径算法求从第 j−1 个服务的承载节点到 n^{S} 之间满足带宽约束的前 K 条最短路径，计算每条路径的 ψ 和 φ 的值，用 $P^{\mathrm{S}}_{j-1,n^{\mathrm{S}}}$ 表示 ψ 值最小的路径。如果上述路径不存在，则 ψ=φ=0；

6)　end for

7)　if 路径 $P^{\mathrm{S}}_{j-1,n^{\mathrm{S}}}$ 的数量>0 then

8)　计算第 j 个服务的候选节点 $n^{\mathrm{S}}_k \in CN_j(1 \leqslant k \leqslant m)$被映射的概率：$P_k = \dfrac{\psi(j,n^{\mathrm{S}}_k)}{\sum_{z=1}^{m}\psi(j,n^{\mathrm{S}}_z)}$，$m$ 为集合 CN_j 中节点的个数，将第 j 个服务按此概率映射到节点 n^{S}_k，然后将逻辑链路 $l^{\mathrm{R}}_{j-1,j}$ 映射至路径 $P^{\mathrm{S}}_{j-1,n^{\mathrm{S}}_k}$；

9)　else

10)　不能发现有效的服务路径，$f(\boldsymbol{p}_i)$=0，初始化可行解失败，算法结束；

11)　end if

12) end for

13) 初始化可行解成功，计算 $f(\boldsymbol{p}_i)$的值，算法结束。

一般情况下，在算法的初始阶段，计算资源和带宽资源比较充裕，不会遇到初始化可行解失败的情况。对于特殊的情况，可对 PIFC 算法进行适当的改进，如引入回溯机制，当步骤 4)~步骤 6)不能发现有效的路径时，将前一个服务映射到其他节点，然后继续映射该服务。

3.2.3　基于 PIFC-PSO 的构建算法

算法首先调用 PIFC 算法获得种群内所有粒子的初始化可行解；然后在粒子群算法的迭代进化过程中，不断地更新粒子位置（参照 PIFC 算法步骤 4)~步骤 8)），重新映射服务，以获得近似最优解，详细过程如算法 2 所示。

算法 2　基于 PIFC-PSO 的多目标服务路径构建算法（PIFC-MOPSO）

输入：底层网络拓扑 G^{S}=(N^{S}，L^{S}，$A^{\mathrm{S}}_{\mathrm{N}}$，$A^{\mathrm{S}}_{\mathrm{L}}$)，服务请求 SR(G^{R}，t_{a}，t_{d})；

输出：服务路径 SP=(N^{SP}，L^{SP}，M^{SP})；

1) 对每一个粒子，调用 PIFC 算法生成初始可行解，计算每个粒子的适应值 $f(\boldsymbol{p}_i)$，保证粒子互不相同；

2) 使用粒子 i 的初始位置向量初始化 $\boldsymbol{pbest}_i$，使用适应值最大粒子的位置向量初始化 $\boldsymbol{gbest}$；

3) repeat

4)　for 每一个粒子 i do

5)　利用式(23)和式(24)更新第 i 个粒子的速度向量与位置向量。对需要重新映射的第 j 个服务，使用 PIFC 中的方法，先依据 ψ 值重新映射该服务和逻辑链路 $l^{\mathrm{R}}_{j-1,j}$，再依据 φ 值映射逻辑链路 $l^{\mathrm{R}}_{j,j+1}$；

6)　end for

7)　重新计算 $f(\boldsymbol{p}_i)$的值，如果 $f(\boldsymbol{p}_i) > f(\boldsymbol{pbest}_i)$，则设置 $\boldsymbol{pbest}_i$ 等于 $\boldsymbol{p}_i$，如果 $f(\boldsymbol{pbest}_i) > f(\boldsymbol{gbest})$ 则设置 $\boldsymbol{gbest}$ 等于 $\boldsymbol{pbest}_i$；

8) until 满足终止条件

9) 如果 $f(\boldsymbol{gbest})$>0，则输出相应的服务路径 SP，即为求得的近似最优解，算法结束；否则，如果 $f(\boldsymbol{gbest})$ == 0，则表明服务路径构建失败，未找到可行解，算法结束。

4　实验仿真与分析

4.1　实验环境设置

底层网络拓扑使用 GT-ITM 工具[23]随机生成，包含 50 个节点，约 130 条链路。底层节点的计算能力与底层链路的带宽服从[50,100]的均匀分布，单位计算资源开销 c^{S} 和带宽资源开销 b^{S} 均为 1。底层网络支持的服务类型数量设定为 10，每个底层节点随机提供其中的 1~5 种。依据文献[5,24]，服务实例的处理时间取决于该服务的类型和网络节点的处理能力，底层链路的传输时延与链路两端点之间的欧式距离成正比，两者依据上述原则进行设定，取值范围介于 1~10 个时间单位。

服务请求的到达过程服从泊松分布，平均 100 个时间单位内到达 4 个请求，每个服务请求的持续时间服从平均 1 000 个时间单位的指数分布。服务链由 4 种服务组成，服务类型随机且不重复，每个服务所需的计算能力在[1, 50]均匀分布，逻辑链路所需的带宽在[1, 50]均匀分布，单位计算能力和单位带宽需支付的费用 c^{R} 和 b^{R} 均为 1，端到端时延允许的最大值 $D_{\max}$ 设定为 100 个时间单位。每次模拟实验的时间约为 50 000 个时间单位，从第 2 000 个时间单位起，每隔 4 000 个时间单位记录一次数据，每组设置进行 10 次模拟实验，实验结果取平均值。

对于 PIFC-MOPSO 算法，设定种群规模数为 5，迭代次数上限为 10，式(23)中的 w、cr_1 和 cr_2 分别设定为 0.1、0.2 和 0.7。式(11)中负载强度的权重参数 ω_{N} 和 ω_{L} 设置为 1。

4.2　仿真实验结果分析

在模拟实验中，对底层网络构建服务路径的长期平均收益、平均构建成功率、平均开销、收益开

销比、平均负载强度与端到端时延等主要评价指标进行了测量和记录，展示了它们随时间变化的情况，从而比较不同构建策略的长期运行效果。

为了评估不同目标对服务路径构建效果的影响，获得权重参数 α、β 和 γ 的调节启发，逐一对 PIFC-MOPSO 算法仅考虑服务路径开销、时延或负载强度的场景进行模拟实验，结果如图 2~图 7 所示，其中，横坐标为时间单位的个数。

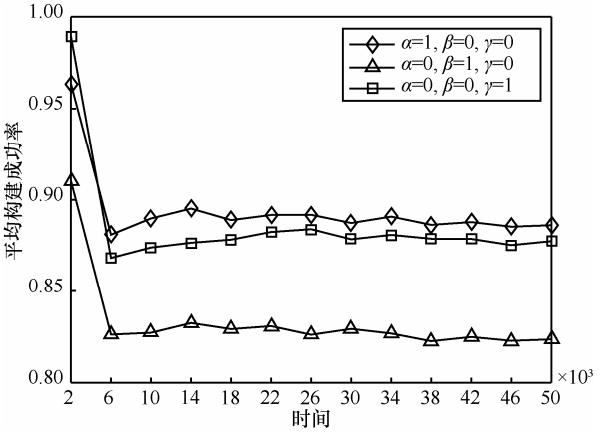

图 2　PIFC-MOPSO 算法在不同目标下的平均构建成功率对比

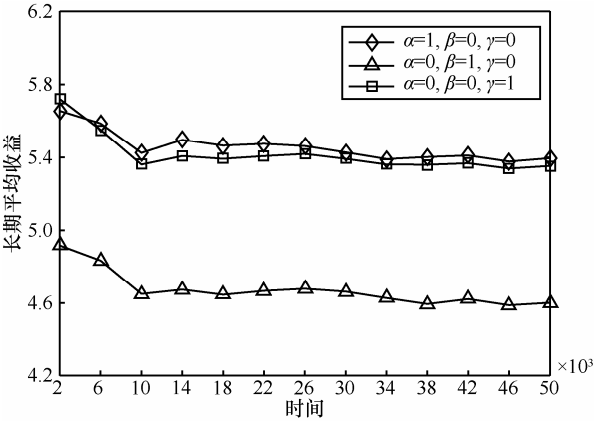

图 3　PIFC-MOPSO 算法在不同目标下的长期平均收益对比

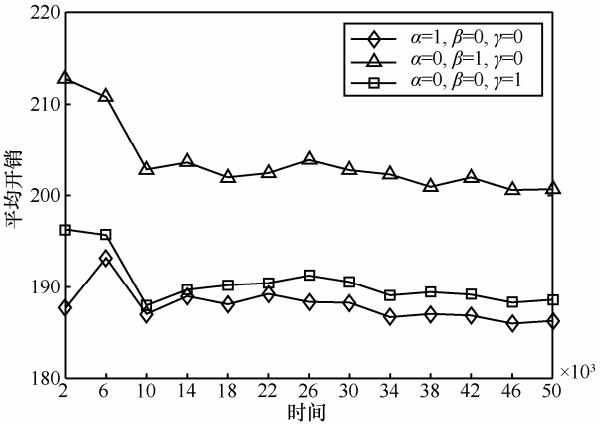

图 4　PIFC-MOPSO 算法在不同目标下的平均开销对比

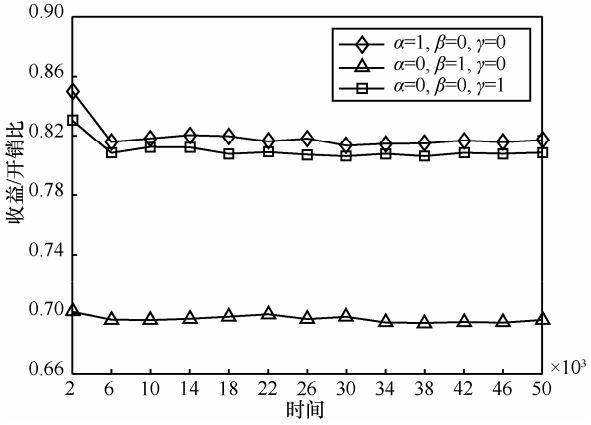

图 5　PIFC-MOPSO 算法在不同目标下的收益/开销比对比

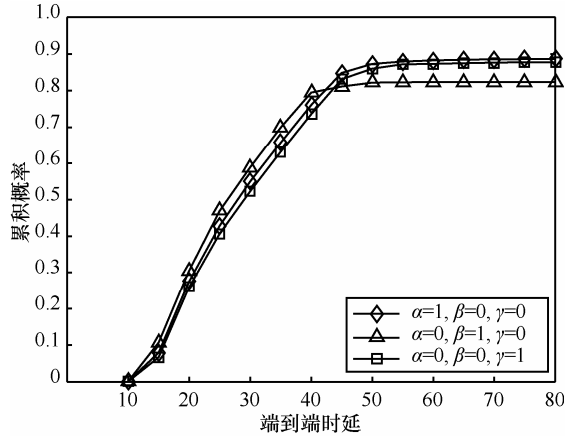

图 6　PIFC-MOPSO 算法在不同目标下的端到端时延累积概率分布对比

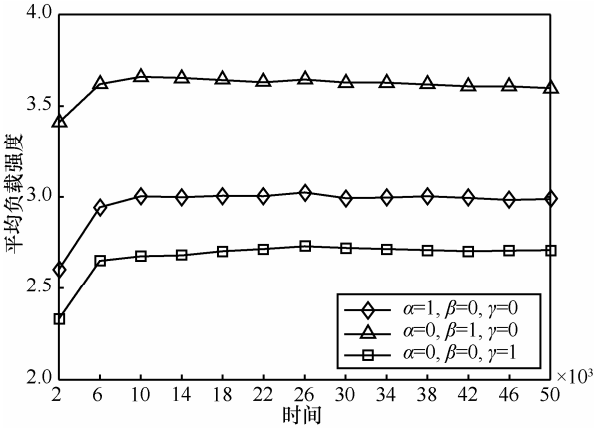

图 7　PIFC-MOPSO 算法在不同目标下的平均负载强度对比

图 2~图 4 表明，在仅以路径开销(α=1)或负载强度(γ=1)为目标的场景下，平均构建成功率、长期平均收益和平均开销的值较为接近，远优于仅考虑路径时延(β=1)的场景。主要原因在于当仅以路径时延为目标时，忽略了底层网络资源的有效利用及负载均衡，难以避免服务路径长度过长和资源碎片的产生，导致资源消耗量过大、负载

不均、易产生资源瓶颈、服务路径构建成功率和长期收益大大降低。与仅以负载强度为优化目标的场景相比，仅以路径开销为优化目标的场景只考虑逻辑链路所映射底层网络路径的长度，由于算法允许多个服务映射到同一底层节点上，因此，单个底层节点承载多个服务的概率提高，使路径长度缩短，链路资源的消耗减少，节省的资源为构建更多的服务路径提供了可能，因此，服务路径构建成功率略高。但是，由于前者并未考虑节点与链路的负载情况，产生资源碎片的可能性增加，接受资源需求量较大的服务请求的概率降低，因此，两者长期平均收益比较接近。从图 4、图 6 和图 7 可以看出，当算法仅考虑单一优化目标时，在相应的评价指标上均取得了最佳的效果，仅考虑路径开销和负载强度时，两者取得了接近的性能。图 5 展示了平均收益与平均开销的比值，该指标能够体现算法的资源使用效率，仅以路径开销为目标时，算法取得了最低开销和最高收益，因此，收益/开销比最高，能够最有效地利用资源，而仅以时延为目标时，开销最高，收益最低，资源利用率最差。

通过上述比较和分析，如果在构建服务路径时只考虑单一的优化目标，难以取得低开销、低时延、高收益、高接收率的映射效果，因此，需要综合考虑 3 个因素。根据各目标对性能的影响程度，服务路径开销和负载强度两目标设定为具有相同的优先级(α=γ=0.4)，而端到端时延目标优先级最低(β=0.2)。

在接下来的模拟实验中，为了评估 PIFC-MOPSO 算法的优越性，选择经典的 Layered Graph 算法[7]和 Step Search 算法[6]作为比较对象，对性能进行全方面的对比。同时，为了评估 PIFC 策略的必要性和优越性，引入 Rand-MOPSO 算法作为比较对象，该算法采用随机的方式初始化和更新粒子的位置向量，参数设置与 PIFC-MOPSO 保持一致，仿真结果如图 8~图 13 所示，其中，横坐标为时间单位的个数。

图 8 和图 9 表明，与 Layered Graph 和 Step Search 算法相比，PIFC-MOPSO 算法将服务路径平均构建成功率分别提高了 23.4%和 13.1%，将长期平均收益分别提高了 70%和 33.9%。从 Layered Graph 和 Step Search 算法的角度分析，原因有 2 点：1）两者均是以最小化端到端时延为优化目标，仅考虑避免资源的过量使用，未考虑高效的资源分配及均衡负载；2）虽然两者构建的候选图不同（分层图和分步搜索图），但均是在候选图上应用 Dijkstra 最短路径算法构建服务路径，在最短路径生成树的构建过程中进行资源的预留。由于服务路径构建问题的 NP 复杂性和 Dijkstra 算法处理资源预留问题的不足，两者均存在有效服务路径未被发现并构建的问题，导致构建成功率和收益较低。如在分层图中，为了最小化时延，Dijkstra 算法在映射服务时，并未考虑服务承载节点此时的资源利用情况，而是始终选择权值最小的垂直边，使对应节点同时承载相同请求中的多个服务，资源消耗过快，形成资源瓶颈，影响后续的服务路径构建。从 PIFC-MOPSO 算法的角度分析，原因在于：1）算法利用粒子的群体进化效应，扩大了搜索的范围，能够有效发现存在的服务路径；2）算法综合考虑了服务路径开销与负载强度 2 个因素，减少了链路资源消耗，均衡了网络负载，能够有效地提高收益，构建更多的服务路径。据此分析，如图 10、图 11 和图 13 所示，与 Layered Graph 和 Step Search 算法相比，PIFC-MOPSO 算法的优势在于：1）显著降低了平均开销（分别降低 17.4%和 11.9%）；2）能够更有效地使用资源，收益开销比分别提高了 26.7%和 17.9%；3）使网络负载更加均衡，负载强度分别降低了 31.9%和 24.2%。图 12 表明随着服务请求数量的增加、可用资源量的减少，端到端时延在逐渐增加，随着资源瓶颈的出现，服务路径构建失败的次数增多，当曲线趋于平缓直至收敛时，表明无新的服务路径被构建。其中，Layered Graph 算法最早达到收敛状态，然后是 Step Search 算法，最后是 PIFC-MOPSO 算法，也进一步表明 PIFC-MOPSO 算法能够构建最多的服务路径，但平均端到端时延略高于其他算法。

图 8 表明，Rand-MOPSO 算法的平均构建成功率与 PIFC-MOPSO 算法相近。但是从图 9、图 10、图 11 和图 13 中可以看出，其长期平均收益和收益开销比明显低于 PIFC-MOPSO 算法（分别降低 4%和 5.7%），其平均开销和负载强度明显高于 PIFC-MOPSO 算法（分别高出 3%和 5.2%）。主要原因在于 Rand-MOPSO 算法在初始化和更新粒子位置时均未考虑底层网络路径长度和负载状况，导致链路资源消耗量大，资源碎片较多。图 12 表明 Rand-MOPSO 算法的平均端到端时延总体高于

PIFC-MOPSO 算法，这是因为 PIFC 方法在粒子位置向量初始化和更新时均考虑了候选服务的处理时延和候选路径的传输时延。综上所述，利用 PIFC 方法进行粒子的初始化和更新，有效地避免了随机方式带来的不确定性，为粒子的飞行方向提供了指导，在相同的迭代次数下，提高了服务路径的构建质量。

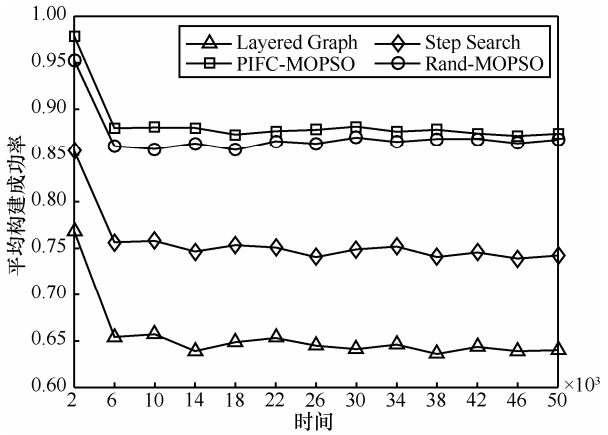

图 8 PIFC-MOPSO 算法与相关算法的平均构建成功率对比

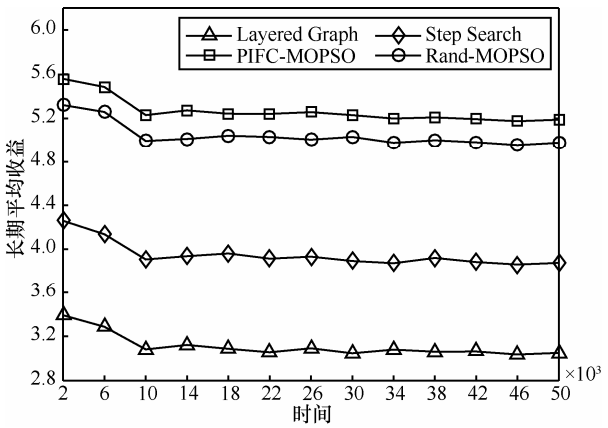

图 9 PIFC-MOPSO 算法与相关算法的长期平均收益对比

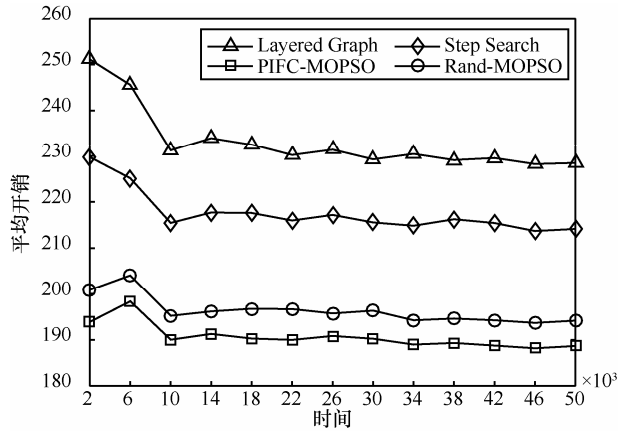

图 10 PIFC-MOPSO 算法与相关算法的平均开销对比

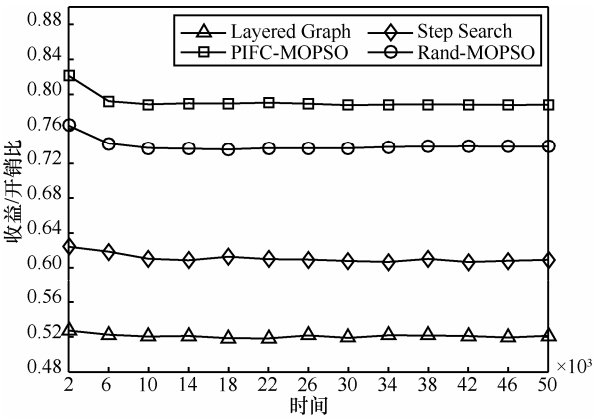

图 11 PIFC-MOPSO 算法与相关算法的收益/开销比对比

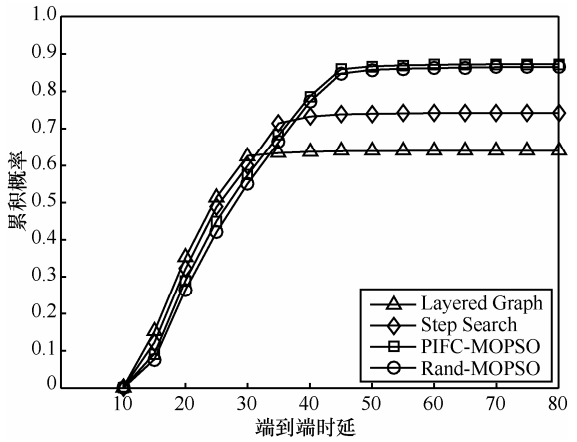

图 12 PIFC-MOPSO 算法与相关算法的端到端时延的累积概率分布对比

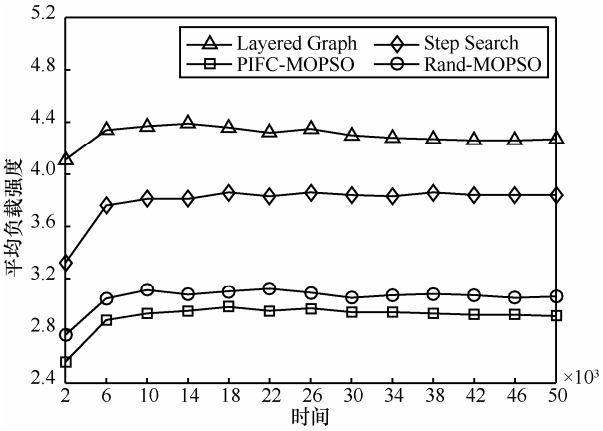

图 13 PIFC-MOPSO 算法与相关算法的平均负载强度对比

5 结束语

网络功能虚拟化将网络层与传输层的网络功能以软件单元的形式实现在核心网络的路由器或服务器上，利用控制器对网络功能进行编排与组合，构建端到端的服务路径，从而支持多样化的服务定制，解决了现阶段中间盒部署方式的灵活性与

可扩展性缺陷问题。针对上述服务提供机制中关键的服务路径构建问题，本文从降低时延、减少开销、均衡负载的角度出发，通过加权求和的方法将多个指标转化为单一的服务路径质量评价指标，建立了服务路径构建问题的整数线性规划模型。然后根据粒子群并行搜索的特征建立了服务路径构建的离散粒子群模型，并设计了适合服务路径构建的粒子群优化算法 MOPSO。为进一步减少资源消耗量，降低资源碎片出现的概率，提高粒子群优化的收敛速度，提出了一种指导粒子位置初始化和更新的优化策略 PIFC，与 MOPSO 结合得到 PIFC-MOPSO 算法。仿真实验结果表明，与现有服务路径构建算法相比，MOPSO 算法有效地优化了服务路径的综合质量，提高了服务路径的构建成功率和长期平均收益。与粒子位置初始化和更新采用随机方式的 Rand-MOPSO 算法相比，PIFC-MOPSO 算法制定了候选节点和路径的评价标准，为粒子的飞行提供了有效的指导，从而进一步优化了算法的性能。

参考文献：

[1] CHIOSI M, DON C, PETER W, et al. Network functions virtualisation: an introduction, benefits, enablers, challenges and call for action[C]//SDN and OpenFlow World Congress Darmstadt, Germany: ESTI, 2012:22-24.

[2] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow: enabling innovation in campus networks[J]. ACM Sigcomm Computer Communication Review, 2008, 38(2): 69-74.

[3] XIA W, WEN Y, FOH C H, et al. A survey on software-defined networking[J]. IEEE Communications Surveys & Tutorials, 2015, 17(1): 27-51.

[4] SAHHAF S, TAVERNIER W, COLLE D, et al. Network service chaining with efficient network function mapping based on service decompositions[C]//2015 1st IEEE Conference on Network Softwarization (NetSoft 2015). 2015:1-5.

[5] SAHHAF S, TAVERNIER W, ROST M, et al. Network service chaining with optimized network function embedding supporting service decompositions[J]. Computer Networks, 2015, 93(P3): 492-505.

[6] HUANG X, GANAPATHY S, WOLF T. Evaluating algorithms for composable service placement in computer networks[C]//International Conference on Communications (ICCC 2009). 2009: 1-6.

[7] CHOI S, TURNER J, WOLF T. Configuring sessions in programmable networks[J]. Computer Networks, 2003, 41(2): 269-284.

[8] RAMAN B, KATZ R H. Load balancing and stability issues in algorithms for service composition[C]//22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM 2003). 2003: 1477-1487.

[9] BARI M F, CHOWDHURY S R, AHMED R, et al. On orchestrating virtual network functions[C]//11th International Conference on Network and Service Management (CNSM 2015). 2015: 50-56.

[10] DIETRICH D, ABUJODA A, PAPADIMITRIOU P. Network service embedding across multiple providers with nestor[C]//IFIP Networking Conference (Network-ing 2015). 2015:1-9.

[11] 段通, 兰巨龙, 程国振,等. 基于元能力的 SDN 功能组合机制[J]. 通信学报, 2015, 36(5):156-166.
DUAN T, LAN J L, CHENG G Z, et al. Functional composition in software-defined network based on atomic capacity[J]. Journal on Communications, 2015, 36(5):156-166.

[12] HU Y X, LI Y F, XING C Q, et al. Providing customized security based on network function composition and reconfiguration[J]. China Communications, 2016, 13(z1): 177-189.

[13] YU M L, YI Y, REXFORD J, et al. Rethinking virtual network embedding: substrate support for path splitting and migration[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 17-29.

[14] CHOWDHURY N M K, RAHMAN M R, BOUTABA R. Vineyard: virtual network embedding algorithms with coordinated node and link mapping[J]. IEEE/ACM Transactions on Networking, 2012, 20(1): 206-219.

[15] CHENG X, SU S, ZHANG Z, et al. Virtual network embedding through topology awareness and optimization[J]. Computer Networks, 2012, 56(6):1797-1813.

[16] FISCHER A, BOTERO J F, BECK M T, et al. Virtual network embedding: a survey[J]. IEEE Communications Surveys & Tutorials, 2013, 15(4): 1888-1906.

[17] YANG Y, CHANG X L, LIU J, et al. Towards robust green virtual cloud data center provisioning[J]. IEEE Transactions on Cloud Computing, 2015: 1-14.

[18] BARI M F, BOUTABA R, ESTEVES R, et al. Data center network virtualization: a survey[J]. IEEE Communications Surveys & Tutorials, 2013, 15(2):909-928.

[19] KENNEDY J, EBERHART R C. Particle swarm optimization[C]// International Conference on Neural Networks. 1995: 1942-1948.

[20] EBERHART R C, SHI Y. Particle swarm optimization: Development, applications and resources[C]//The 2001 Congress on Evolutionary Computation. 2001: 81-86.

[21] TAO F, ZHAO D, HU Y, et al. Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system[J]. IEEE Transactions on Industrial Informatics, 2008, 4(4): 315-327.

[22] ZHAO X, SONG B, HUANG P, et al. An improved discrete immune optimization algorithm based on PSO for QoS-driven web service composition[J]. Applied Soft Computing, 2012, 12(8):2208-2216.

[23] ZEGURA E W, KENNETH L C, SAMRAT B. How to model an internetwork[C]//15th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 1996). 1996: 594-602.

[24] DOBRESCU M, ARGYRAKI K, RATNASAMY S. Toward predictable performance in software packet-processing platforms[C]//USENIX Symposium on Networked Systems Design and Implementation (NSDI 2012). 2012:141-154.

作者简介：

马丁（1978-），男，回族，河南夏邑人，郑州大学博士生，主要研究方向为新型网络体系结构、路由与交换技术等。

庄雷（1963-），女，山东日照人，郑州大学教授、博士生导师，主要研究方向为未来互联网体系结构、网络虚拟化等。

兰巨龙（1962-），男，河北张北人，国家数字交换系统工程技术研究中心总工程师、教授、博士生导师，主要研究方向为新一代信息网络关键理论与技术。