研究与开发

基于边缘计算的数据密集型服务部署

高永梅[1]，程冠杰[2]

（1. 杭州职业技术学院信息工程学院，浙江 杭州 310018;
2. 浙江大学计算机科学与技术学院，浙江 杭州 310027）

摘　要：日益增长的数据量对数据处理的要求越来越高，于是出现了数据密集型服务。在解决复杂问题时，多个数据密集型服务通常会形成一个服务组合。由于服务组件之间存在大量的数据传输，巨大的传输时延会对系统的整体性能造成影响。在边缘计算环境中，基于否定选择算法，为降低服务组合中的数据传输时间提出了一种优化部署策略。首先，给出了此类数据密集型服务组件部署问题的定义，并为该部署问题构建优化模型；然后，设计了一种否定选择算法来获取最佳的部署方案；为了评估该算法的适用性和收敛性，使用遗传算法和模拟退火算法与其对比，结果显示，提出的算法在这种数据密集型服务组件的部署问题中表现得更为出色。

关键词：服务组合；服务部署；边缘计算；云计算

中图分类号：TP302

文献标识码：A

doi: 10.11959/j.issn.1000−0801.2019170

Data-intensive service deployment based on edge computing

GAO Yongmei[1], CHENG Guanjie[2]

1. Information Engineering Institute, Hangzhou Vocational & Technical College, Hangzhou 310018, China
2. College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China

Abstract: The demand is getting higher and higher for data processing due to big data volume, thus, data-intensive service shave emerged. When solving complex problems, multiple data-intensive services are often united as a service portfolio. Due to the huge data transmission between service components, a great transmission delay will affect the overall performance of the system. In the edge computing environment, an optimized deployment strategy based on the negative selection algorithm was proposed to reduce the data transmission time in the service composition. Firstly, the definition of such a data-intensive service component deployment problem was given, and the deployment problem was modeled as an optimization model; then, a negative selection algorithm was designed to obtain the best deployment solution. In order to evaluate the applicability and convergence of the algorithm, it was compared with genetic algorithm and simulated annealing algorithm. The results show that proposed algorithm outperforms other schemes in this data-intensive service deployment problem.

Key words: service composition, service deployment, edge computing, cloud computing

收稿日期：2019−01−10；**修回日期**：2019−06−24

基金项目：浙江省教育厅科研项目（No.Y201635225）

Foundation Item: Education Department Research Project of Zhejiang Province (No.Y201635225)

1 引言

随着面向服务计算（service oriented computing）[1-2]的发展，网络服务技术近年来吸引了来自工业界和学术界的广泛关注并且取得了非常显著的成果。与此同时，数据密集型服务应运而生，在服务计算中担任着越来越重要的角色。

云计算环境中的服务部署是一个非常热门的问题，有很多相关工作。当前的云市场由许多不同的公共云提供商组成，在接口、定价方案、虚拟机提供和增值功能等方面都高度分散。在这种情况下，云代理可以提供中介和聚合功能，使用户能够跨云实现虚拟基础设施部署。然而，当前的大多数云代理并不提供高级的服务管理功能来进行自动决策。Lucas-Simarro 等人[3]提出了一种适合于多云环境的新型云代理架构，可以在多云环境中不同的服务调度策略下工作。Yang 等人[4]在对环境中的用户偏好、优化目标和各种场景约束的描述和分析的基础上，为面向用户偏好的服务选择和部署提出了一种分类方法。Zhang 等人[5]提出了一种基于控制模型和博弈理论的动态服务放置问题的框架，进一步考虑了多个服务提供者以动态方式争夺资源的情况，并提出了一种协调机制。Kang 等人[6]指出了在云计算中多种服务协同部署的问题，他们的研究目标是为多个相关的服务进行最优部署，而不是服务组合，服务之间的相互影响使得服务组合的结构更为复杂。Yuan 等人[7]探索了云工作流的独特特性，提出了一个群聚数据放置策略，可以根据数据之间的相关性在数据中心自动分配数据。

云计算环境下的服务部署研究大多旨在基于某种优化准则（如成本优化或性能优化）或者不同的环境条件（如动态和静态、实例类型、服务工作负载等）解决多云条件下单一的服务调度问题[8]，很少有工作集中在服务组件的最优化部署问题上。另外，大多数有关服务部署的研究是云计算环境中的部署问题，很少考虑边缘计算环境中的服务部署问题。但是随着计算和存储能力逐渐下降到网络边缘，与终端用户越来越近，服务在边缘环境中的部署问题越发关键。

边缘计算（edge computing，EC）[9]，是在靠近物或数据源头的网络边缘侧，融合网络、计算、存储、应用核心能力的分布式开放平台，就近提供边缘智能服务，满足行业数字化在敏捷联接、实施业务、数据优化、应用智能、安全与隐私保护等方面的关键需求。移动端的边缘技术即移动边缘计算（mobile edge computing，MEC）是一项引领 5G 通信时代的新兴技术，它能够在移动网络的边缘或靠近用户的无线接入网侧提供信息技术服务环境和云端计算功能[10]。

本文研究的基于边缘计算的数据密集型服务组件部署问题更贴近现实，并且具有更大的挑战和难度。使用某些固定在云端的云服务提供商提供的服务，各个服务组件之间存在相互影响，如何同时在云计算和边缘计算环境下进行服务组件的最优化部署是一个崭新的课题，本文将优化数据传输时延作为目标。

2 服务组合部署问题定义

2.1 服务部署限制因素

实际应用场景中对于服务部署有很多的限制因素，如网络带宽、数据中心负载均衡、I/O 能力、计算能力等，本文实验场景中主要考虑以下限制因素。

（1）数据传输

对于一个数据密集型服务组合，数据中心之间巨大的数据传输是不可避免的。因此，只有综合考虑数据中心之间的网络带宽和传输的数据量，才能有效降低服务组合的总体传输时延。

（2）服务依赖关系

服务组件之间存在两种复杂的依赖关系，即数据依赖和逻辑依赖。数据依赖是指执行某些服

务的输出数据恰好是执行另一些服务所需的输入数据，这就形成了彼此之间的数据相关性；逻辑依赖是指各个服务组件的执行逻辑：即执行一次完整的服务组合的过程中各个服务组件的执行路径，由于服务组件之间存在与、或、非等多种逻辑关系，所以执行路径可能有很多种。显然，有依赖关系的服务组件应该部署得越近越好，这样可以有效降低数据传输的开销。

（3）服务供应商约束

运行一次应用需要多个服务协同执行，其中可能会使用到某个云服务提供商提供的云服务，而云服务是固定在云端的，只能在边缘端自由部署服务组件，但是这些服务组件与云服务同样存在依赖关系，因此需要将云服务单独考虑，又不能忽略它对其他组件的影响。

（4）数据中心的存储容量

执行每个服务组件之前，需要将执行该服务所需的输入数据存储到服务所部署的数据中心，这些数据总量不能超过数据中心的存储容量。

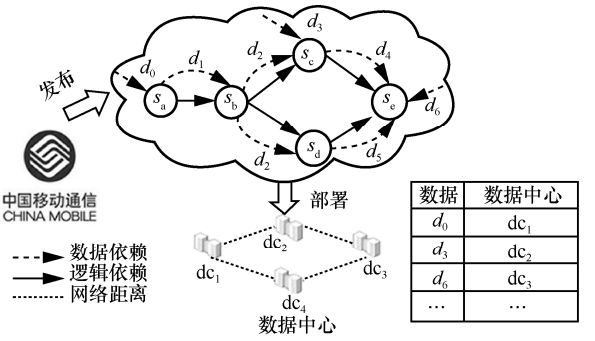
2.2　数据密集型服务组件部署场景假设

本文以中国移动为例来阐述数据密集型服务组件的部署问题[11]。中国移动在全国各地建立多个数据中心来存储大规模数据，并提供各种移动互联网服务。这些服务是通过存储在不同数据中心的服务组件，根据一定的执行逻辑组合实现的。另外，构成服务组合的服务组件也需要使用来自不同数据中心的数据作为输入。

假设中国移动想部署一个由 s_a 到 s_e 5 个服务组件组成的服务组合 S，并且需要处理来自 dc_1 到 dc_4 4 个数据中心存储的数据，如图 1 所示。部署服务组合 S 也就是将每个服务组件部署到一个最合适的数据中心。

数据中心分布在不同省份，因此它们之间的网络距离不同。此外，服务组件之间存在大量的数据传输，并且传输的数据量不同。如服务组件 s_c，它处理来自 dc_2 的数据 d_3 和执行 s_b 输出的数据 d_2，同时又输出数据 d_4 给服务组件 s_e。所以，将某个服务组件部署在哪个数据中心，需要综合考虑数据传输开销，而数据传输开销同时受传输数据量和传输距离的影响。另外，s_c 的部署还需考虑 s_b 和 s_e，因为它与 s_b 和 s_e 之间存在相互影响。因此，为确保系统高效运行，需要为服务组合制定一个全局的最优部署策略。

目前，类似图 1 中的数据中心大都位于国内各个省份的云服务器上，彼此之间的距离很远。本研究是在图 1 的基础上加入边缘端的考虑，将数据中心部署在边缘服务器中，为终端用户直接提供服务。这种边缘部署大大降低了用户请求的响应时延，提升了用户体验，更符合现在服务计算的发展方向。所以本文研究云和边缘的协同环境，考虑到现实中云服务的固定性，本文将着重研究边缘端的服务组件部署问题。



数据	数据中心
d_0	dc_1
d_3	dc_2
d_6	dc_3
…	…

（a）中国移动服务部署　（b）数据中心存储数据

图 1　移动公司云计算应用场景

2.3　问题定义和建模

本文形式化定义一些在数据密集型服务组合及其部署问题中需要使用的关键概念。

定义 1　（数据密集型服务组合）使用一个有向图表示一个数据密集型服务组合，用一个三元组表示，即 DSC=< S, L, D >，其中：

- S 是服务组合中服务组件的集合。在图 1 中，S 由 s_a 到 s_e 5 个服务组件组合而成。
- $L=\{ldep_{ij}|s_i s_j \in S\}$ 是服务组件之间逻辑依赖的集合。一个逻辑依赖定义了服务的执行

顺序，服务组件之间可以存在多种逻辑，这些逻辑影响了服务组件的执行路径，不同的服务组件执行路径会产生不同的数据传输时间。

- $D=\{ddep_{ij}|s_i s_j \in S\}$是服务组件之间数据依赖的集合，服务组件之间存在大量的数据传输，这些传输数据即数据依赖的表现形式。

此外需要注意的是，执行服务需要的输入数据可能存储在某个数据中心中，也可能是执行完某个服务的输出数据。

定义 2 （逻辑依赖）逻辑依赖定义了服务组件之间的执行顺序，映射了一条执行路径。主要考虑下述逻辑依赖类型。

- AND-split：与分散节点，把一个单独的进程划分为多个可以并行的过程，这样多个服务组件就可以同时执行。图 2 中，在 AND 节点之后的 s_5 和 s_7 就可以被同时执行。
- AND-join：与汇聚节点，多个并行的进程汇聚至该点。
- XOR-split：异或分散节点，把一个单独的进程划分为多个可以并行的进程，其中只有一个进程可以被选择执行。注意 OR-split 是一个相似的逻辑，但不一样的是，OR-split 可以选择的进程范围是 0 到总的进程数。
- XOR-join：异或汇聚节点，其中有两个或多个可选分支，在没有同步的情况下聚集在一起。

图 2 是数据密集型服务组合的样例。

定义 3 （数据依赖）定义在两个服务组件之间的数据依赖 $ddep_{ij}=<s_i, s_j, data_{ij}>$表示服务组件 s_i 的输出数据作为 s_j 的输入，s_i 和 s_j 之间的数据传输量为 $data_{ij}$。数据依赖关系是一个数据密集型服务组合的基本特征。

定义 4 （数据中心）部署服务组件的数据中心集可以表示为 $DC=\{dc_i| i=1, 2, \cdots\}$。一个数据中心表示为 $dc_i=<sc_i, st_i>$，其中 sc_i 表示 dc_i 的存储容量，st_i 是 dc_i 的负荷阈值分数。每组数据中心间的网络带宽用一个矩阵 BW 表示，BW 的元素 BW_{ij} 代表 dc_i 和 dc_j 之间的带宽，BW 有两个特点：当 $i=j$ 时，$BW_{ij}=0$；$BW_{ij}=BW_{ji}$。

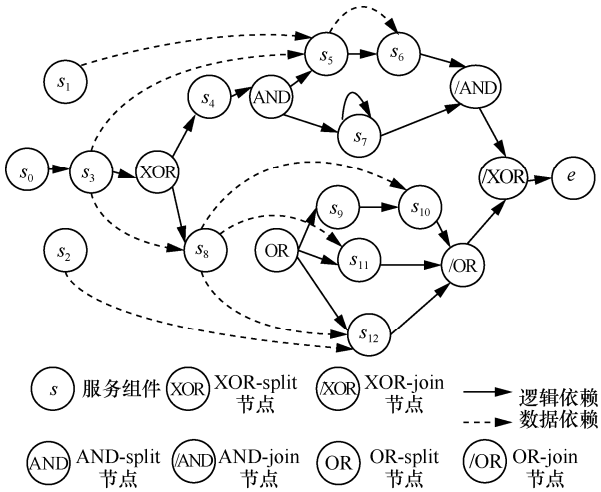


图 2 数据密集型服务组合示例

定义 5 （云服务组件）为了使研究更贴近实际场景，将服务组合中的部分组件固定在云端，如图 3 所示，s_1、s_2、s_3 定义为云服务组件。即为了完整地执行一次应用，需要同时使用某些云服务提供商提供的云服务和边缘环境中可自由部署的服务。这就意味着不需要管理固定在云端的服务组件，而是在边缘端合理部署剩下的服务组件。因为位于同一个云服务器中的各个云服务之间的数据传输速率是非常快的，本文认为云服务组件之间的数据传输时间可以忽略不计。

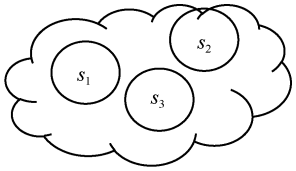


图 3 云服务组件

定义 6 （部署策略）给定一个数据密集型服务组合 DSC 和一个数据中心集 DC，将一种部署策略表示为：

$$\mathrm{ds}=\bigcup_{i=\{1,2,\cdots|\mathrm{dsc}.S|\}}\{s_i\rightarrow \mathrm{dc}_j \mid \mathrm{dc}_j\in \mathrm{DC}\} \quad (1)$$

除了固定在云端的部分服务组件，对于 DSC 中的服务组件，在 DC 中找一个数据中心来部署。不同的服务组件可以部署在同一个数据中心上，但是执行这些服务组件所需要的输入数据总量不能超过所部署数据中心的存储容量。允许有空余的数据中心，本文认为一种部署策略中存在空余的数据中心说明该部署策略具有很大的可扩展性。

定义 7（时延）本文不考虑移动用户端，服务组合的响应时延是从接受到服务请求到部署服务组件并执行完服务准备返回结果的总时间：

$$\mathrm{Latency}=T_{\mathrm{data}}+T_{\mathrm{exec}}+T_{\mathrm{other}} \quad (2)$$

其中，T_{data} 是服务组件之间的数据传输时间。如果存在数据依赖的几个服务组件部署在同一个数据中心，那么忽略它们之间的数据传输时间。T_{exec} 是服务执行时间。T_{other} 是其他时间（如服务组件部署时间），这部分时间应该由管理程序在微秒数量级内自动实现，与服务组件之间庞大的数据传输时间相比可以忽略不计。本文认为当一个服务组件所需的输入数据准备完毕后，服务器就可以迅速执行。由于服务执行时间与服务器计算力成反比，本文认为边缘服务器拥有充分的计算资源，所以服务执行时间可安全地省略。那么：

$$\mathrm{Latency}\approx T_{\mathrm{data}} \quad (3)$$

本文研究的目的是从服务组件间的数据传输成本着手，找出最优的服务部署策略。这实际上就转化为一个优化问题，优化目标是获取一个最小时延的部署策略。服务组件之间的逻辑关系决定了可能存在多种服务执行路径，需要针对所有路径找出一种最优部署策略。由于部署在相同数据中心的服务组件之间具有更高的数据传输速率，理论上服务组件部署得越近越好，但是数据中心存在容量限制。另外，云服务组件与边缘端服务组件的相互影响、云服务器与边缘端各数据中心的带宽差异，都增大了服务组件部署问题的复杂性。

3　算法和实验过程

3.1　基于 NSA 的服务组件部署算法概述

否定选择算法（negative selection algorithm，NSA）来源于人工免疫系统（artificial immune system，AIS）领域，可以将 AIS 定义为一种基于理论免疫学和观察到的免疫功能、原理和模型的计算智能系统，AIS 已经成功地在很多领域解决了一些复杂的问题[12]。目前主要受关注的人工免疫系统算法有克隆选择算法、否定选择算法、人工免疫算法、树突细胞算法，本文选择的是第二种算法。

否定选择算法采取了一种生物选择思想，其思想灵感来源于胸腺中 T 细胞成熟过程中出现的否定和确定的选择过程。NSA 的基本操作如图 4 所示。

（1）初始化基因库；

（2）通过基因重组获取不成熟的免疫细胞或者抗体；

（3）通过否定选择算法去除不成熟的免疫细胞中的不好的成分，获取到成熟的细胞；

（4）根据一个适应度函数评估获得抗体；

（5）如果成熟的抗体可以很好地辨认抗原，等位基因的一致性将在基因库管理中自适应地增加。

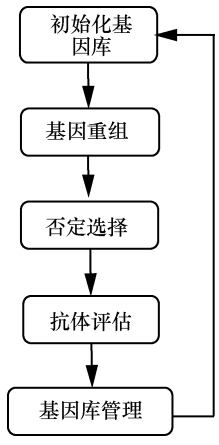


图 4　NSA 的基本操作

重复步骤（2）~步骤（5），随着否定选择算法的每次选择评估更新，越来越多特征较差的解

决方案会被剔除，好的基因会以越来越高的概率得到保留，有效地加速了算法的收敛过程。相对于一般的基于迭代的优化算法，否定选择算法会以一定的概率跳出当前最优解，避免结果陷入局部最优情况。

将否定选择算法对应到本文数据密集型服务组件的部署问题的实验场景中：要解决的问题被视为 NSA 的抗原，也就是数据密集型服务组合部署的优化问题与抗原相对应；每种服务组件部署方案对应于否定选择算法的一个抗体；抗体是由许多基因组成的，即某种部署策略中为每个服务组件所部署的数据中心对应于该抗体中的每个基因。

为了对比不同部署方案的性能，在每轮迭代中取一个传输时延最小的部署策略作为本轮迭代的最佳部署方案，然后算出每轮迭代中最佳部署方案的目标函数，观察算法最终收敛的结果。其算法流程如图 5 所示。

3.2 实验过程

3.2.1 初始化操作

首先，否定选择算法通过产生大量的抗体来工作，每一个都代表了数据密集型服务组合部署问题的一个可能的部署方案。因此，使用整数数组来表示所选的数据中心，并将一个解决方案向量映射到一个特定的数据密集型服务组件。例如，抗体向量（2,3,1,2）意味着在数据密集型服务组合中有 4 个组件服务，第一个组件服务部署在数据中心 2 中，第二个组件服务部署在数据中心 3 中，以此类推。

在初始化操作中，将部署问题中选定的数据中心作为基因片段，初始化所有可能的基因片段的值，并将它们放入基因库。所有用于基因库的复合阵列构成了数据密集型服务组件部署问题的解决方案空间。然后，确定抗体的种群大小，这决定了为每个迭代生成多少解向量，即生成多少数据密集型服务组件的部署策略。并且设置最大的迭代次数来决定何时终止该算法。

此外，本文在初始化阶段定义了两个二维数组，即 Consistence 矩阵和 SELF 矩阵。

- Consistence 矩阵：用于在基因重组操作中为每个服务组件选取合适的数据中心。一致性矩阵（即 Consistence）包含 $p \times q$ 个元素，其中 p 是矩阵的行，是服务组件的数量，而 q 是矩阵的列，是数据中心的数量。在初始化阶段，Consistence 矩阵的所有元

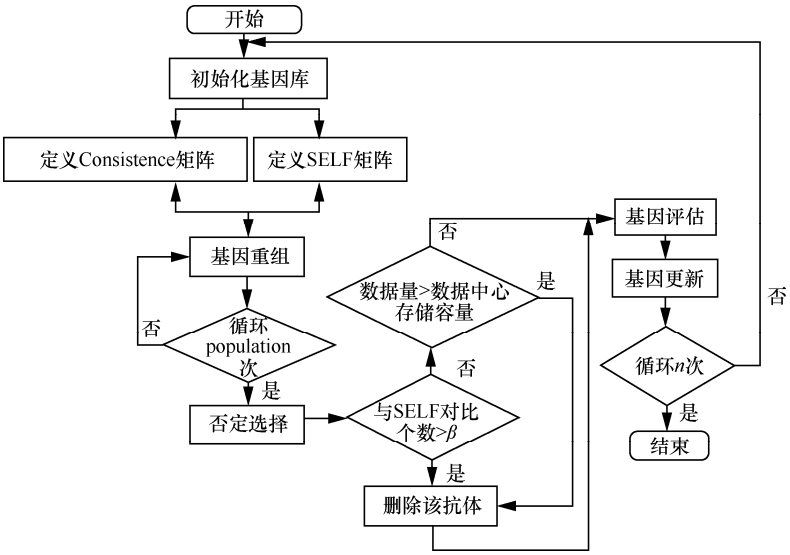


图 5 基于 NSA 的数据密集型服务组件部署算法流程

素都被初始化为 1，矩阵中的值将在基因库更新操作中进行更新。

- SELF 矩阵：可以理解为一个简单的集合，该集合用来记录质量不好的基因，形式化的理解就是将第 p 个服务组件部署到第 q 个数据中心中，但是这种部署策略的性能很低，所以将 SELF 矩阵中的第 p 行、第 q 列对应的元素设置为 1。因此，SELF 矩阵与 Consistence 矩阵一致，也包含 $p\times q$ 个元素。SELF 集合被初始化为空集，在基因库更新操作时动态更新。

为了营造一个简单可扩展且易于理解的实验研究场景，本文在边缘端设置了 10 个数据中心，服务组合是由 0~8 共 9 种存在复杂的相互依存关系的服务组件组成，其中 0~7 服务组件按照一定的部署策略部署在位于边缘端的数据中心里，服务组件 8 是位于云端的云服务，不具备管理权限，所以不需要考虑其部署问题。另外，在边缘端的数据中心中只有服务 7 与云服务存在数据传输，即运行一次应用需要按某种路径在边缘端执行完 0~7 共 8 个服务组件，最后执行云端的云服务。研究目的是寻求一种服务组件的部署策略使得按这些服务组件的某种执行路径执行完一次应用所花费的数据传输时延是最小的。

3.2.2　基因重组操作

在这一步中，研究的目标是产生大量的抗体。在初始化操作中设置了种群大小 population，即要生成的数据密集型服务组件的部署策略数目为 population。每个抗体由 p 个基因组成，其中 p 是数据密集型服务组合中组件服务的数量。对于一个抗体 v 中的每个服务组件 i，根据 select（v,i）为其选择一个合适的数据中心。循环 population 次，即将所有的数据密集型服务组件的部署策略中的服务组件都部署到边缘端相应的数据中心中。为第 i 个服务组件选择第 j 个数据中心是基于一个动态概率：

$$\text{probability}(i,j)=\frac{C_{i,j}\times\text{localfitness}(i,j)}{\sum_{k=1}^{q}C_{i,k}\times\text{localfitness}(i,k)} \tag{4}$$

其中，$C_{i,j}$ 是 Consistence 矩阵中的元素，q 是边缘端数据中心的数目。$\text{localfitness}(i,j)$ 用来评估将第 i 个服务组件部署在第 j 个数据中心是否合适。$\text{localfitness}(i,j)$ 的计算式为：

$$\text{localfitness}(i,j)=\frac{\text{dc}_j.\text{sc}-\text{data}_i-\text{data}_{\text{init}}}{\text{dc}_j.\text{sc}} \tag{5}$$

其中，data_i 是指执行服务组件 i 所需的数据，dc_j.sc 是第 j 个数据中心总的存储容量，$\text{data}_{\text{init}}$ 是指第 j 个数据中心中本身存储的数据量。数据中心剩余的存储容量越多，说明该数据中心的本地适应度值越高，意味着具有更高的可扩展性。

根据每个数据中心的本地适应度值和 Consistence 矩阵中的元素计算出第 i 个服务组件部署在各个数据中心中的概率值，然后选出一个概率最大的数据中心，即第 i 个服务组件选择的数据中心。在现实情况中，某些服务只能被部署在某个特定的数据中心，这些服务组件所在的数据中心的选择概率设为 1。

在这一步操作中，为部署策略中的所有服务组件选择了对应的数据中心，并且根据动态概率进行了首轮筛选。

3.2.3　否定选择操作

否定选择操作是否定选择算法的基础，可以有效地减少服务组件部署策略的搜索空间。在基因重组过程中产生了确定数目的抗体，但是这些抗体的性能良莠不齐，甚至有些抗体可能是不合法的。否定选择操作负责去除质量差的抗体，保留比较好的服务组件部署方案。

分两种情况将性能较差的数据密集型服务组件部署策略从总体部署方案空间中移除：对于每个通过基因重组操作新获得的抗体，将其中所有的基因对与 SELF 中值为 1 的元素组成的基因对集合作对比，如果相同的基因对数目大于一个设

定的阈值 β，则该抗体被认为是不合格的，从总数中删除；假设一个抗体本身就是不合法的，如存储在该数据中心的服务组件所需的输入数据大小超出了该数据中心的存储容量，那么这个部署策略将会被移除。

通过这两种删除策略大大减少了性能较差的部署策略的数目，使得在之后每次迭代中所获得的服务组件部署策略越来越好。

3.2.4 基因评估操作

在基因评估操作中，计算出每个抗体的适应度来评估该抗体的质量好坏。在数据密集型服务组件的部署问题中，主要考虑的影响因素是服务组件之间的数据传输时延，然后根据数据传输时延计算目标函数：

$$\text{fitness}(v)=\frac{\omega}{\text{latency}} \tag{6}$$

其中，latency 是每种数据密集型服务组件部署策略的数据传输时延，ω 是一个方便输出适应度的可调系数。可以看出输出的 fitness 值越大，该部署策略的数据传输时延越小；反之，则数据传输时延越大，这是分析实验输出的有力依据。

为了计算每种部署策略的数据传输时延，本文设计了一个 getLatency()方法。根据某种部署策略，在得出执行一次应用时所有可能的执行路径之后，根据：

$$\text{数据传输时间}=\frac{\text{传输数据量}}{\text{数据中心之间的带宽}} \tag{7}$$

可以计算出任意两个服务组件之间的数据传输时间，然后进行叠加求和得出该服务部署策略的总传输时延。

评估一个最佳服务部署策略的思路是：服务组件之间存在比较复杂的逻辑依赖，针对某个服务组件部署策略，对于所有服务组件可能的执行路径，分别利用上述方案求出其数据传输时间，然后选取其中数据传输时间最大的路径，将这个最大的数据传输时间作为该种数据密集型服务组合部署策略所对应的 latency 值。

最后将所有数据密集型服务组合部署策略的 latency 值计算出来并作比较，得出数据传输时间最小的一种服务组合部署策略，即要求的最佳策略。

3.2.5 基因库更新操作

基因库更新操作是本文算法的核心部分，更新的目的是在下一轮迭代中的基因重组操作中获得更好的抗体，即通过改变 Consistence 矩阵中的值降低获得性能较差的抗体的可能性，增大获得性能较好的抗体的可能性，然后在基因重组操作中的 select 操作中，为数据密集型服务组件部署策略的每个服务组件获取更合适的数据中心。

首先将全部抗体的适应度函数均值算出来，然后将所有抗体的适应度值与之作比较。将部署策略分为两个部分：将具有较高适应度函数值抗体放到好的一组，较低的抗体放到差的一组。对于好的一组中的抗体，将其中的所有基因对应在 Consistence 矩阵的元素值增大，使得在下一轮迭代中获取这些基因的可能性增大。给定一个抗体 v，如果其中第 i 个基因选择了第 j 个数据中心，那么 $C_{i,j}$ 将根据式（8）被更新：

$$C_{i,j}'=\rho\times C_{i,j}+\frac{\text{fitness}(v)}{\text{fitness}(\max)} \tag{8}$$

其中，ρ 是与算法收敛性相关的衰减系数，fitness(max)是所有抗体中最大的适应度值。

对于适应度值较低的抗体，将其中比较差的基因放入 SELF 集合中。针对某种部署策略，计算出其中所有服务组件所部署的数据中心的 localfitness 值。在此设置一个 SELF 阈值 α，如果数据中心的 localfitness 值小于 α，则被认为是性能比较差的基因，将其放入 SELF 集合，这使得在下一次迭代的否定选择操作中将性能不好的服务组件部署策略移除的概率增大。

执行了基因库更新操作之后，较好性能的抗

体的一致性就会增加，而性能较低的抗体的一致性就会相对降低，这就使得在下一轮迭代中的基因重组阶段产生更好的服务组件部署策略，从而加快实验收敛速度。

4　实验评估

4.1　参数选择及性能分析

为了评估实验步骤中 α、β 和 ρ 3 个重要参数的影响，本实验在 1 个中等规模的数据密集型服务组合中进行，其中包括 9 个组件服务和 10 个用于调度的服务器，给定服务组件之间的数据依赖关系和逻辑依赖关系。此外，由于没有标准数据集，为了避免相对偏差，预先给定了每个数据中心的存储容量和它们之间的带宽以及服务组件之间的数据传输量。在实验中，抗体的大小设置为 50，即每轮迭代生成 50 组部署策略，最大的迭代次数是 500，算法独立运行 50 次以获得无偏见的统计结果。

参数 α 是一个基因的本地适应度阈值，即 localfitness，用来决定性能较差的抗体中的某个基因是否要被放入 SELF 集合中，当 α 增大时，本地适应度值小于 α 的基因变多，即被放入 SELF 集合中的元素增多。参数 β 表示抗体在 SELF 集合中出现基因的总数阈值，用来决定在否定选择操作中哪些抗体被移除。即当某个抗体中包含 SELF 基因的个数多于 β 时，该抗体就会被移除。当 β 增大时，抗体被移除的概率就会降低。衰减系数 ρ 用来衡量一致性矩阵的惯性。为了评估这 3 个系数对实验的影响，本研究根据表 1 设置了参数值。

表 1　设定测试参数值

参数	α	β	ρ
实验 1	0.1，0.3，0.5	1	1
实验 2	0.1	1，5，10	1
实验 3	0.1	1	0.5，0.9，1

图 6~图 8 分别显示改变对应参数时每轮迭代平均最佳适应度。从图 6 中可以看出，α=0.1 时的收敛结果微大于 α=0.3 和 0.5 时的收敛结果。由理论分析可知，当 α 变得更大时，在基因库更新操作中会将更多的基因放到 SELF 集合中，导致抗体中的基因和 SELF 中的元素相同的概率变大，在否定选择操作时将会抛弃更多的抗体，致使剩下的抗体搜索空间大幅缩减，显然不能使 α 过高。从图 7 可以看出，参数 β 对解决方案的精确度和计算效率的影响很小。

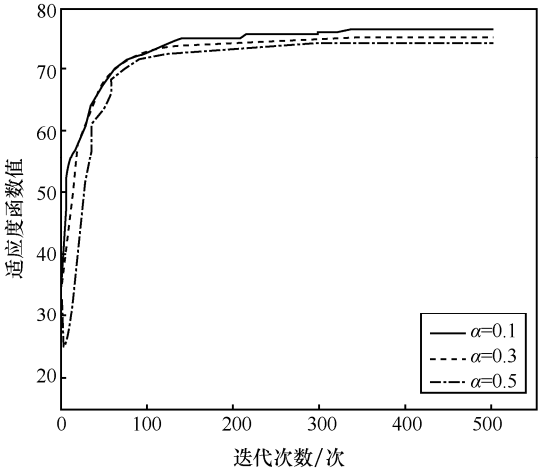


图 6　不同 α 参数收敛结果

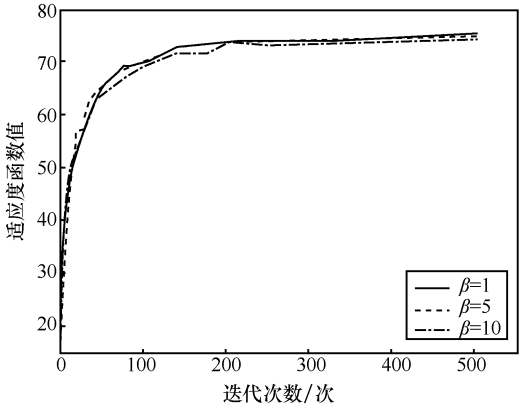


图 7　不同 β 参数收敛结果

从图 8 中可以看出，ρ=1 时的收敛情况比其他两种更好。当 ρ 相对较小时，一致性矩阵的惯量将会降低，导致算法性能不稳定，并且容易陷入局部最优。

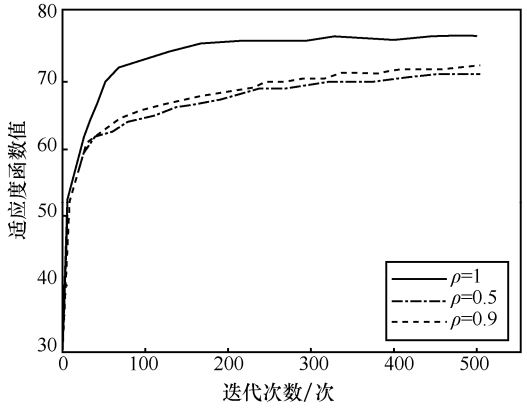


图 8　不同 ρ 参数收敛结果

综上所述，α=0.1、β=1 以及 ρ=1 时的方案会产生更好的性能。

4.2　算法对比实验

本文选择遗传算法和模拟退火算法两种性能高效的启发式算法来求解数据密集型服务组件部署问题。以下是算法对比实验的实现思路和结果分析。

4.2.1　遗传算法（GA）实现

遗传算法中的个体指的是数据密集型服务组件的一种部署策略，群体即给定数目的服务组件部署空间。使用轮盘赌策略根据每个个体的适应度值计算出相对适应值大小，然后每轮选择出适应度值较大的两种服务组件部署策略进行交叉操作，交换两个父体的某个服务组件所部署的数据中心。最后执行遗传操作中的变异部分，随机选取两个父体中的某个服务组件，将其随机部署到某个数据中心，即产生了两种新的服务组件部署策略，判断合法后放入新的部署策略集合中，直到新的部署策略达到指定种群数目。表 2 是遗传算法的参数设置。

表 2　遗传算法参数

算法参数	设定值
种群规模	50
演化代数	1 000
交叉概率	0.8
变异概率	0.05

4.2.2　模拟退火算法（SA）实现

模拟退火算法从某一较高的温度出发，伴随着温度参数的不断下降，算法中的解趋于稳定，但是这个解可能是一个局部最优解。在初始化阶段，随机生成初始解，即为服务组件部署策略中的服务组件随机选取数据中心。然后，为了避免局部最优，进行扰动产生新解，即为某个服务组件选择新的数据中心，从而产生新的部署策略。将新产生的部署策略的目标函数值与原方案的值做差，判断是直接接受新解还是按照 Metropolis 准则接受新解。算法的参数设置见表 3。

表 3　模拟退火算法参数

算法参数	设定值
初始温度	100
迭代次数	1 000
温度下界	1.00×10^{-8}
温度下降率	0.98

4.2.3　实验结果分析

观察 3 种算法的输出结果，从 3 个角度来对比实验性能：每轮迭代输出的 fitness；获得最大 fitness 的收敛速度；每轮迭代输出的部署策略。

3 种算法在初始化阶段均先随机为每个服务组件部署数据中心，之后在每轮迭代中不断优化。在每次执行 3 种算法的输出结果中，每轮迭代对应的最佳部署方案和其适应度值都会有所差异。但 3 种算法的每轮执行结果呈现出一致的收敛规律，所以本文选取其中 1 次的执行结果来分析 3 种算法展现出来的性能差异。

首先，对比 3 种算法每轮迭代输出结果中的 fitness，fitness 越大，说明最佳部署策略的数据传输时延越低。NSA 的 fitness 结果见表 4，否定选择算法在第 45 轮迭代后收敛到最高适应度值 87.777 8；GA 的 fitness 结果见表 5，GA 在 843 轮迭代后收敛到最高适应度值 45.703 8；SA 的 fitness 结果见表 6，SA 在第 80 轮迭代后收敛到最

高适应度值 70.422 54。可以直观地发现否定选择算法获得的最终适应度值是最高的，因此使用否定选择算法收敛到的最终部署策略所对应的数据传输时间是最短的，并且与使用遗传算法和模拟退火算法差距非常明显。

表 4　NSA 的 fitness 结果

迭代次数	fitness
40	60.942 02
41	65.152 41
42	76.162 43
43	76.162 43
44	76.162 43
45	87.777 8
46	87.777 8
47	87.777 8
48	87.777 8

表 5　GA 的 fitness 结果

迭代次数	fitness
840	43.649 03
841	43.649 03
842	43.649 03
843	45.703 84
844	45.703 84
845	45.703 84
846	45.703 84
847	45.703 84
848	45.703 84

表 6　SA 的 fitness 结果

迭代次数	fitness
76	47.869 793
77	47.869 793
78	47.869 793
79	47.869 793
80	70.422 54
81	70.422 54
82	70.422 54
83	70.422 54
84	70.422 54

然后，关注获得最大 fitness 的收敛速度，忽略局部最优情况的收敛范围，直接关注收敛到最终结果时的迭代次数。如表 4 所示，否定选择算法经历 40 多次迭代就收敛到最小数据传输时间，而遗传算法和模拟退火算法分别需要 800 多次和 70 多次。因此，否定选择算法具有更快的收敛性。

最后，观察每轮迭代输出的部署策略，本文输出的部署策略表现形式是一个序列。例如，序列 0，0，0，0，0，0，2，0 表示前 6 个服务组件和第 8 个服务组件全部部署在第 1 个数据中心里，第 7 个服务组件部署在第 3 个数据中心里。

NSA 的部署策略见表 7，在否定选择算法中，当收敛到最短数据传输时间后，部署策略变化仅在最后第 7 个服务组件中发生，并且仅集中在几种固定的部署策略中。这是因为边缘端服务组件之间存在一些逻辑联系，不同的服务执行路径可能会产生相同的数据传输时延。

表 7　NSA 的部署策略

迭代次数	服务组件部署方案
991	0,0,0,0,0,0,2,0
992	0,0,0,0,0,0,2,0
993	0,0,0,0,0,0,6,0
994	0,0,0,0,0,0,2,0
995	0,0,0,0,0,0,6,0
996	0,0,0,0,0,0,5,0
997	0,0,0,0,0,0,5,0
998	0,0,0,0,0,0,5,0
999	0,0,0,0,0,0,5,0

GA 的部署策略见表 8，在收敛到最终目标时间时，每个服务组件的部署策略依旧在变化，这是不断执行遗传操作的结果。另外，在程序中设定的服务组合中组件服务的数目只有 9 个，因此结果应该收敛在某几种部署策略，不可能得到一个唯一的部署状态。

表 8 GA 的部署策略

迭代次数	服务组件部署方案
991	2,7,8,5,1,0,8,9
992	2,7,8,5,1,0,8,2
993	2,9,8,1,7,8,7,9
994	2,9,8,4,7,8,7,9
995	3,5,3,4,7,4,5,5
996	4,2,3,4,7,4,5,0
997	4,1,3,4,7,4,5,5
998	4,1,3,4,6,4,9,5
999	4,1,3,4,6,4,9,5

SA 的部署策略见表 9，SA 与 GA 状态相同，稳定在某个数据传输时间后又不断进行细微的扰动，使得部署策略不断变化。

表 9 SA 的部署策略

迭代次数	服务组件部署方案
991	8,7,7,3,0,6,1,5
992	8,7,7,3,9,2,1,5
993	3,8,7,9,3,2,1,0
994	3,8,7,8,3,2,1,0
995	3,8,7,8,3,2,1,0
996	0,8,7,8,3,2,5,0
997	7,3,7,8,5,7,5,6
998	7,3,7,8,5,7,5,6
999	7,3,7,8,5,7,5,6

因此从部署方案的收敛性角度来看，否定选择算法是一个更佳的选择。

5 结束语

本文提出了数据密集型服务组件的部署问题，给出了一个形式化的、易于理解的模型和问题描述，将位于云端的服务部署问题迁移到移动边缘环境中。即在使用云服务的基础上，在边缘环境中部署服务组件，并以数据传输时延为优化目标，将求解部署策略问题转化为一个目标优化问题。此外，本文实现了一个适用于研究服务组件部署问题的优化算法：否定选择算法。在后续研究中，可将用户的移动性作为研究场景的约束，进一步考虑服务组件的部署问题，并依托真实的数据集作为实验输入，根据实验反馈进一步改进算法、调整性能。

参考文献：

[1] WANG L J, SHEN J. Data-intensive service provision based on partical swarm optimization[J]. International Journal of Computational Intelligence Systems, 2018, 2(1): 330-339.

[2] PAPAZOGLOU M P, TRAVERSO P, DUSTDAR S, et al. Service-oriented computing: a research roadmap[J]. International Journal of Cooperative Information Systems, 2010, 17(2): 223-255.

[3] LUCAS-SIMARRO J L, MORENO-VOZMEDIANO R, MONTERO R S, et al. Scheduling strategies for optimal service deployment across multiple clouds[J]. Future Generation Computer Systems, 2013, 29(6): 1431-1441.

[4] YANG L T, LIU L, FAN Q. A survey of user preferences oriented service selection and deployment in multi-cloud environment[C]//2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), December 18-20, 2017, Taipei, China. [S.l.:s.n.], 2017.

[5] ZHANG Q, ZHU Q Y, ZHANI M F, et al. Dynamic service placement in geographically distributed clouds[J]. IEEE Journal on Selected Areas in Communications, 2018, 31(12): 762-772.

[6] KANG Y, ZHENG Z, LYU M R. A latency-aware co-deployment mechanism for cloud-based services[C]//The 2012 IEEE 5th International Conference on Cloud Computing, June 24-29, 2012, Honolulu, HI, USA. Piscataway: IEEE Press, 2012: 630-637.

[7] YUAN D, YANG Y, LIU X, et al. A data placement strategy in scientific cloud workflows[J]. Future Generation Computer Systems, 2010, 26(8): 1211-1214.

[8] MORENO-VOZMEDIANO R, MONTERO R S, HUEDO E, et al. Orchestrating the development of high availability services on multi-zone and multi-cloud scenarios[J]. Journal of Grid Computing, 2018, 16(1): 39-53.

[9] 朱丹, 王晓冬. 移动网络边缘计算与缓存技术研究[J]. 铁路计算机应用, 2017, 26(8): 51-54.
ZHU D, WANG X D. Mobile network edge computing and

caching technology[J]. Railway Computer Application, 2017, 26(8): 51-54.

[10] 张建敏, 谢伟良, 杨峰义, 等. 移动边缘计算技术及其本地分流方案[J]. 电信科学, 2016, 32(7): 132-139.
ZHANG J M, XIE W L,YANG F Y, et al. Mobile edge computing and application in traffic offloading [J]. Telecommunications Science, 2016, 32(7): 132-139.

[11] 刘宜明, 李曦, 纪红. 面向 5G 超密集场景下的网络自组织关键技术[J]. 电信科学, 2016, 32(6): 44-51.
LIU Y M, LI X, JI H. Key technology of network self- organization in 5G ultra-dense scenario [J]. Telecommunications Science, 2016, 32(6): 44-51.

[12] DASGUPTA D, YU S, NINO F. Recent advances in artificial immune systems: models and applications[J]. Applied Soft Computing, 2011, 11(2): 1574-1587.

[作者简介]

高永梅（1975− ），女，杭州职业技术学院信息工程学院副教授，主要研究方向为数据挖掘、边缘计算。

程冠杰（1996− ），男，浙江大学计算机科学与技术学院博士生，主要研究方向为边缘计算。